HOCHSCHULE
MITTWEIDA
UNIVERSITY OF
APPLIED SCIENCES

MASTERARBEIT

Herr
Christian Thormann, B.Sc.

Entwicklung des Energie-Managements für ein Elektrorennfahrzeug (FSE) und Visualisierung der energetischen Zustandsgrößen zur Motorenprüfstandsüberwachung

2015

Fakultät **Elektro- und Informationstechnik**

MASTERARBEIT

Entwicklung des Energie-Managements für ein Elektrorennfahrzeug (FSE) und Visualisierung der energetischen Zustandsgrößen zur Motorenprüfstandsüberwachung

Autor:
Christian Thormann, B.Sc.

Studiengang:
Elektrotechnik

Seminargruppe:
ET13sA-M

Erstprüfer:
Prof. Dr.-Ing. Lutz Rauchfuß

Zweitprüfer:
Prof. Dr.-Ing. Swen Schmeißer

Mittweida, März 2015

Bibliografische Angaben

Thormann, B.Sc., Christian: Entwicklung des Energie-Managements für ein Elektrorennfahrzeug (FSE) und Visualisierung der energetischen Zustandsgrößen zur Motorenprüfstandsüberwachung, 91 Seiten, Hochschule Mittweida, University of Applied Sciences, Fakultät Elektro- und Informationstechnik

Masterarbeit, 2015

Satz: LaTeX

Referat

Diese Arbeit beschäftigt sich mit der Entwicklung des Energie-Managements für einem Motorenprüfstand als Vorstufe für ein Elektrorennfahrzeug (FSE). Bestandteil ist eine Batteriesimulation eines Hochvoltakkus. Diese dient als Grundlage für die Algorithmen des Energiemanagements. Zudem wurde eine energiesparsame Kühlpumpenregelung entworfen. Ebenso ist die Visualisierung der Zustandsgrößen des Teststandes ein Bestandteil der Arbeit. Die Entwicklung eines Fahrsimulators am Teststand ist ebenfalls ein Bestandteil.

I. Inhaltsverzeichnis

II. Abbildungsverzeichnis

III. Tabellenverzeichnis

IV. Abkürzungsverzeichnis

CAN	Controller Area Network
ETAS	Engineering Tools, Application and Services
FSG	Formula Student Germany
LIN	Local Interconnect Network
NEFZ	Neuer europäischer Fahrzyklus
PID	Proportional-Integral-Differential Regler
PWM	Pulsweitenmodulation
SoC	State of Charge
TMM	Technikum Mittweida Motorsport

1 Einleitung

1.1 Motivation zum Projekt

Die Formula Student ist ein internationaler Motorsport-Konstruktionswettbewerb für studentische Teams. Diese müssen ein Rennfahrzeug selbstständig entwerfen, fertigen, testen und gegeneinander antreten. Ziel ist es hierbei das beste Gesamtprojekt bestehend aus Konstruktion, Rennleistung sowie Finanzierung und Präsentation zu erstellen. Hierfür werden regelmäßig Rennsportevents ausgetragen, bei denen sich die Teams miteinander messen. Seit 2006 werden diese Wettbewerbe in Deutschland unter dem Namen „Formula Student Germany“ (FSG) regelmäßig durchgeführt. Im Jahr 2006 wurde das Formula-Student-Team „Technikum Mittweida Motorsport“ (TMM) gegründet. Seither wurden sechs Rennfahrzeuge entwickelt und aufgebaut. Alle bisher gebauten Fahrzeuge wurden mit einem Verbrennungsmotor angetrieben. Für die Saison 2014/2015 soll nun ein Fahrzeug entwickelt werden, welches ein rein elektrisches Antriebskonzept beinhaltet. Zum Testen des Antriebskonzeptes muss nun ein Teststand entwickelt und aufgebaut werden, welcher es ermöglicht, den Motor sowie den Frequenzumrichter zu testen und zu parametrieren. Zudem soll am Teststand das Steuergerät sowie das Kühlkonzept getestet werden.

1.2 Aufgabenstellung

Diese Arbeit beschäftigt sich mit der Batteriesimulation, der Ansteuerung der Kühlmittelpumpe sowie der Visualisierung der Zustandsgrößen des Teststandes. Die Simulation der Batterie soll dazu dienen, Algorithmen für das Energiemanagment zu entwickeln.

Die Simulation des Akkumulators soll eine Lithium-Ionen-Batterie simulieren, wie sie auch im Fahrzeug eingesetzt werden soll. Die Simulation soll in der Lage sein, die Klemmspannung, die Batterietemperatur und den Ladezustand nachzubilden. Dies soll dazu dienen, das Verhalten des Antriebsstranges unter realen Bedingungen zu testen. Zudem könnte eine solche Simulation dazu genutzt werden, andere Batteriekonzepte, wie beispielsweise andere Zellen oder Verschaltungen, zu testen, bevor diese aufgebaut werden. Zudem ist es notwendig, Handlungsempfehlungen für kritische Batteriezustände zu erstellen, um die Batterie zu schützen. Des Weiteren soll eine Abschätzung des verfügbaren Restweges für alle Ladezustände erfolgen. Eine Begrenzung der Antriebsleistung zum sicheren Erreichen eines gewünschten Fahrzieles soll ebenso implementiert werden. Zudem sollen Algorithmen für das Energiemanagment entwickelt und getestet werden. Die Simulation des Akkumulators wird in Kapitel 4 erläutert.

Ebenfalls Teil dieser Arbeit ist die Realisierung der Ansteuerung der Kühlmittelpum-

pe. Es soll ein Kühlkreislauf für den Teststand aufgebaut und getestet werden. Dieser Kühlkreislauf soll als Vorstufe für die Kühlung im Fahrzeug dienen. Die Leistung der Kühlmittelpumpe soll dabei so geregelt werden, dass eine minimale Energiemenge aus der Batterie entnommen wird, jedoch alle Komponenten unter ihrer jeweiligen Grenztemperatur liegen. Kapitel 5 beschäftigt sich mit der Kühlung des Teststandes und der Fahrzeugkühlung.

Zum Testen ist es notwendig, die momentanen Zustandsgrößen des Motors sowie des Inverters anzuzeigen. Hierfür soll eine Visualisierung entworfen werden. Zusätzlich soll ein Fahrsimulator entwickelt werden. In diesem soll ein Fahrer mithilfe der Pedalerie eine Geschwindigkeitskurve nachfahren. Dieser ist in Kapitel 6 beschrieben.

2 Stand der Technik

2.1 Software

2.1.1 MATLAB Simulink

MATLAB/ Simulink ist ein multi-physics Simulationsprogramm auf Basis von MATLAB. Mittels dieser Simulationsmodelle können physikalische Gegebenheiten nachgebildet und analysiert werden. Zudem ist es möglich, erzeugte Simulationen in Quelltext für andere Programme zu übersetzen. Mit derartig erzeutem Code soll beispielsweise das Steuergerät programmiert werden. Verwendet wurden die MATLAB-Versionen R2013b und R2014a.

2.1.2 Vector CANoe

Die Firma Vector stellt Analyse- und Steuersoftware für CAN-Bus-Netzwerke (Controller Area Network) her. Das Programm CANoe ist ein solches Analyseprogramm. Es ist in der Lage mehrere CAN-Netzwerke zu verwalten und zu steuern. Zudem können MATLAB-Simulationen zur Steuerung eingebunden werden. Am Teststand soll mittels des CAN-Adapters VN1630 das CAN-Netzwerk, bestehend aus Invertern und Nebenaggregaten, gesteuert werden. Genutzt wird das Programm CANoe in der Version 8.2.

2.1.3 ETAS Intecrio

Im Rennfahrzeug soll ein Fahrzeugsteuergerät der Firma ETAS eingesetzt werden. Dieses Steuergerät wird am Teststand zur Motorsteuerung für die Maschine, welche in das Fahrzeug eingebaut werden soll, eingesetzt. Hierbei handelt es sich um ein Steuergerät ES910. Dieses Gerät besitzt insgesamt vier CAN-Schnittstellen und zwei LIN-Schnittstellen (Local Interconnect Network). Die LIN-Schnittstelle ist für die Steuerung der Kühlmittelpumpe vorgesehen. Zu Messzwecken wird zudem ein Messmodul ES930 am Teststand eingesetzt. Dieses Modul besitzt digitale und analoge Ein- und Ausgänge. Zudem besitzt es spezielle Eingänge für den direkten Anschluss von Thermoelementen zur Messung von Temperaturen. Die Programmierung erfolgt modellbasiert (mittels MATLAB Simulink-Modellen) durch das Programm INTECRIO. Dieses Programm ermöglicht es aus verschiedenen Modellen ein Betriebssystem zusammenzustellen und dieses auf das Steuergerät zu laden. Verwendet wurde die Version 4.4.1.1028.

2.2 Kommunikation

2.2.1 CAN

Das Controller Area Network (CAN) ist ein Bussystem [11]. Einer der Hauptanwendungsbereiche ist die Automobilindustrie, da die Kommunikation des Steuergerätes mit Sensoren und Aktoren mittels CAN realisiert wird. Es existiert keine physikalische Teilnehmerkennzeichnung, da es sich um ein nachrichtenorientiertes Bussystem handelt. Eine von einem Teilnehmer gesendete Nachricht kann von allen Teilnehmern gelesen werden. Dadurch können wichtige Daten an viele Teilnehmer gesendet werden (Multicasting) oder eine Synchronisation der Teilnehmer durchgeführt werden. Eine Nachricht besteht aus einem Identifier und bis zu acht Datenbytes. Durch Arbitrierung besitzen niedrige Identifier höhere Priorität. Die Datenübertragung erfolgt differenziell über eine Zweidrahtleitung. Die Kommunikation zwischen den Invertern, dem Steuergerät, dem Steuerrechner und der Gleichstromquelle erfolgt über CAN. Die Symbolrate am Teststand beträgt 500 kBaud.

2.2.2 LIN

Local Interconnect Network ist ein Eindraht-Feldbus zur Kommunikation zwischen Sensoren und Aktoren in der Fahrzeugtechnik [21, Seite 79 ff.]. Er wurde Ende der 90er Jahre entwickelt und stellt eine kostengünstige Alternative zu CAN dar, da es eine geringere Datenrate und weniger Flexibilität als CAN aufweist. Der LIN-Bus wird in der Fahrzeugtechnik an den Stellen eingesetzt, an denen CAN zu aufwändig wäre. Ein LIN-Netzwerk besteht aus einem Master-Knoten und mindestens einem Slave-Knoten. Der Master-Knoten kann Daten an einzelne Slaves senden oder Daten von den Slaves anfordern. Dadurch kann keine Telegrammkollision vorkommen. Eine LIN-Nachricht beinhaltet einen Header sowie bis zu acht Datenbytes. Als Datenrate können 2400, 9600 oder 19200 $\frac{\mathrm{bit}}{\mathrm{s}}$ eingestellt werden.

2.3 Lithium-Ionen-Akkumulator

Eine Lithium-Ionen-Batterie [17] ist ein Akkumulator in welchem die positive Elektrode aus einer Lithium-Metalloxidverbindung und die negative Elektrode aus beispielsweise Grafit, Silizium oder Lithium-Titanat besteht. Durch das Laden des Akkus werden Lithiumionen von der positiven an die negative Elektrode abgegeben. Die Elektroden sind durch ein Elektrolyt getrennt. Dadurch entsteht ein Potenzialunterschied nach der elektrochemischen Spannungsreihe der verwendeten Elektrodenmaterialien. Abbildung 2.1 zeigt ein vereinfachtes Ersatzschaltbild einer Lithium-Ionen-Zelle [6, Seite 26]. Das Spannungsverhalten für den Entladefall ist in Abbildung 2.2 dargestellt. Der Widerstand R_i repräsentiert den Innenwiderstand der Zelle. Dieser bewirkt, dass bei steigender

Stromstärke die Klemmspannung abnimmt, da der ohmische Spannungsabfall höher wird. Durch die unterschiedliche Polarität der Ladungsträger bildet sich eine Doppelschichtkapazität, welche durch R_K und C_K beschrieben wird. Zum Entladen müssen die Ladungsträger durch das Elektolyt diffundieren. Die Diffusionsgeschwindigkeit ist hierbei begrenzt. Die Elemente R_D und C_D bilden die Effekte des Diffusionsprozesses nach. Der Spannungsabfall über diesen Elementen steigt mit der entnommenen Ladung. Der Laderate C gibt den Strom in Bezug auf die Gesamtkapazität wieder.

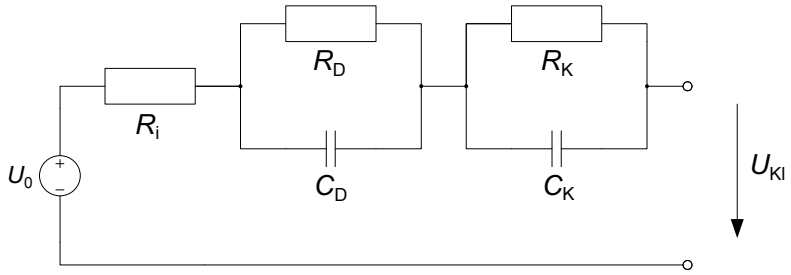


Abbildung 2.1: Ersatzschaltbild Li-Ionen Akkumulator

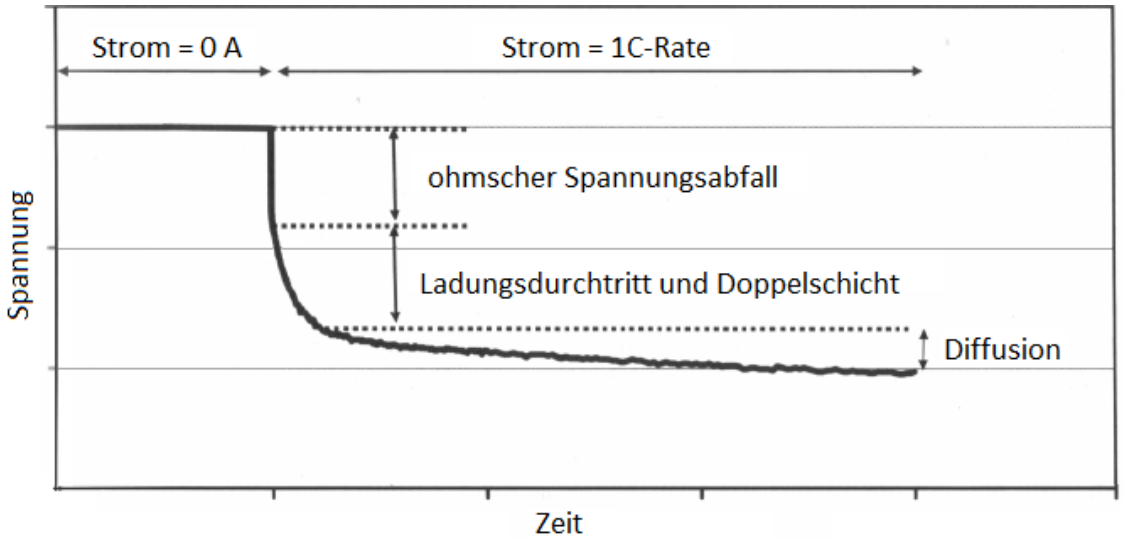


Abbildung 2.2: Klemmspannungs-Zeit-Verlauf Li-Ionen Akkumulator

2.4 Thermisches Verhalten

Um das thermische Erwärmungsverhalten eines Stoffes zu betrachten, ist die Berechnung der Wärmemenge Q notwendig. Diese erechnet sich aus folgender Formel [4, Seite 29]:

$$Q = m \cdot c \cdot \triangle \vartheta \tag{2.1}$$

Das c steht hierbei für die spezifische Wärmekapazität. Diese sagt aus, wieviel Energie notwendig ist, um ein Kilogramm dieses Stoffes um ein Kelvin zu erwärmen. Die Masse

m und die Temperaturänderung $\triangle\vartheta$ werden ebenfalls für die Berechnung benötigt. Die Ableitung der Wärmemenge nach der Zeit ergibt den Wärmestrom:

$$\dot{Q} = \frac{\mathrm{d}Q}{\mathrm{d}t} \tag{2.2}$$

Dieser gibt an, wieviel Wärmeenergie über die Zeit abgegeben oder aufgenommen wurde.

2.5 Mechanik

Um ein Objekt zu bewegen muss eine Arbeit verrichtet werden. Die Arbeit ergibt sich aus der Kraft F sowie dem zurückgelegten Weg s [4, Seite 22]. Die Arbeit kann auch aus einer Spannung U und einer Ladungsmenge Q [4, Seite 34] errechnet werden:

$$W = F \cdot s = U \cdot Q \tag{2.3}$$

Für die Berechnung der notwendigen Arbeit für die gleichmäßige Bewegung in der Ebene wird eine Kraft F_R benötigt, welche die Widerstandskraft (Rollreibung) bricht. Diese setzt sich aus der Reibungszahl μ und der Normalen-Kraft F_N zusammen [4, Seite 23]:

$$F_\mathrm{R} = \mu \cdot F_\mathrm{N} \tag{2.4}$$

Diese ist proportional der Normalen-Kraft F_N, welche wiederum von der Gewichtskraft F_G und dem Winkel zwischen beiden Kräften abhängt. Falls gilt $F_\mathrm{G} \parallel F_\mathrm{N}$, dann gilt [4, Seite 21]:

$$F_\mathrm{N} = F_\mathrm{G} = m \cdot g \tag{2.5}$$

Die Kraft setzt sich aus der Masse m des Objektes und der Fallbeschleunigung g zusammen.

Das bewegte Objekt wird durch das umgebende Fluid einen Widerstand erfahren. Die Widerstandskraft F_W errechnet sich aus dem Strömungskoeffizient c_W, der angeströmten Fläche A, der Dichte des Fluides ρ und der Geschwindigkeit des Objektes v . Diese wird nach folgender Formel errechnet [14, Seite 81]:

$$F_{\mathrm{W}} = c_{\mathrm{W}} \cdot A \cdot \frac{\rho \cdot v^2}{2} \tag{2.6}$$

Die mechanische Leistung P ergibt sich aus der Arbeit W und der Zeit t zu [4, Seite 22]:

$$P = \frac{W}{t} = \frac{F \cdot s}{t} \tag{2.7}$$

Das Weg-Zeitverhältnis $\frac{s}{t}$ ergibt die Geschwindigkeit v. Im Fall einer Drehbewegung wird das Drehmoment M durch die Kraft F und den Radius r bestimmt [4, Seite 22]:

$$M = F \cdot r \tag{2.8}$$

Die Winkelgeschwindigkeit ω ergibt sich aus $\frac{v}{r}$. Für die Leistung ergibt sich:

$$P = F \cdot \frac{s}{t} = F \cdot v = M \cdot \frac{v}{r} = M \cdot \omega \tag{2.9}$$

2.6 Regelungstechnik

2.6.1 Der Regelkreis

Ein Regelkreis dient dazu, eine physikalische Größe auf einem vorgegebenen Sollwert einzustellen. Abbildung 2.3 zeigt einen einfachen Regelkreis. Der Regler hat die Aufgabe, die Regeldifferenz e, welche aus der Differenz zwischen Führungsgröße w und Rückführung gebildet wird, auf null einzustellen. Hierfür errechnet dieser eine Stellgröße y, welche der Regelstrecke zugeführt wird.

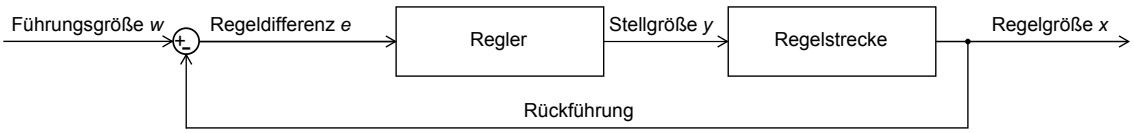


Abbildung 2.3: einfacher Regelkreis

2.6.2 PID-Regler

Ein PID-Regler ist ein stetiger Regler. Ein stetiger Regler ist in der Lage, im Rahmen der technischen Grenzen eine beliebige Stellgröße zu erzeugen. Die Stellgröße y wird

bei diesem Regler anhand der Regelabweichung e sowie einem P-, I- und D-Anteil berechnet. Die Stellgröße wird folgendermaßen berechnet [8, Seite 124]:

$$y(t) = K_\mathrm{p} \cdot e(t) + K_\mathrm{i} \int e(t)dt + K_\mathrm{d} \frac{\mathrm{d}e(t)}{\mathrm{d}t} \tag{2.10}$$

Der PID-Regler in paralleler Form besitzt folgende Übertragungsfunktion:

$$G_\mathrm{R}(\mathbf{s}) = K_\mathrm{p} + \frac{K_\mathrm{p}}{T_\mathrm{n} \cdot \mathbf{s}} + K_\mathrm{p} \cdot T_\mathrm{v} \cdot \mathbf{s} = K_\mathrm{p} + \frac{K_\mathrm{i}}{\mathbf{s}} + K_\mathrm{d} \cdot \mathbf{s} \tag{2.11}$$

Die einzustellenden Parameter K_p, K_i und K_d sind abhängig von der Regelstrecke und dem gewählten Regleroptimierungsverfahren. T_N steht für die Nachstellzeit, T_V für die Vorhaltezeit.

2.6.3 F_{Rt}-Wurzelrekursion

Die Regelung mittels F_{Rt}-Wurzelrekursion soll, im Gegensatz zum PID-Regler, ohne Parametrierung arbeiten [16]. Der Regelalgorithmus besteht aus einer rekursiven Wurzelfunktion, welche sich der Strecke automatisch anpassen soll. Die Bedeutung der Parameter ist in Tabelle 2.1 verzeichnet. Bei einer Regelstrecke ohne Ausgleich wird fogender Regelalgorithmus verwendet [16, Gleichung 3]:

$$y(i+1) = w(w- \mid y(i) \cdot K_\mathrm{S} + h \cdot xd \mid) \tag{2.12}$$

Für eine Strecke mit Ausgleich wird folgender Regelalgorithmus angewandt [16, Gleichung 2]:

$$y(i+1) = \sqrt{w \cdot \mid \frac{y(i)}{K_\mathrm{S}} + h \cdot xd \mid} \tag{2.13}$$

Wert	Bedeutung
w	Sollwert (Führungsgröße)
K_S	Streckengesamtverstärkung
h	Gewichtung
xd	Regeldifferenz (e)

Tabelle 2.1: Wurzelrekursionsparameter

3 Der Teststand

Zum Testen und Parametrieren der Antriebsmaschine und der Nebenaggregate wurde ein Teststand aufgebaut. Dieser besteht aus zwei baugleichen Synchronmaschinen sowie zugehöriger Inverter. Diese sind gegenüberliegend angeordnet und mittels einer Welle verbunden (Abbildung 3.1). Maschine 1 soll später im Rennfahrzeug eingebaut werden. Maschine 2 soll die Trägheit eines Rennfahrzeuges simulieren [13]. Dadurch kann das Verhalten der Antriebsmaschine getestet werden. Als Akkumulatornachbildung wird eine einstellbare Spannungsquelle verwendet.

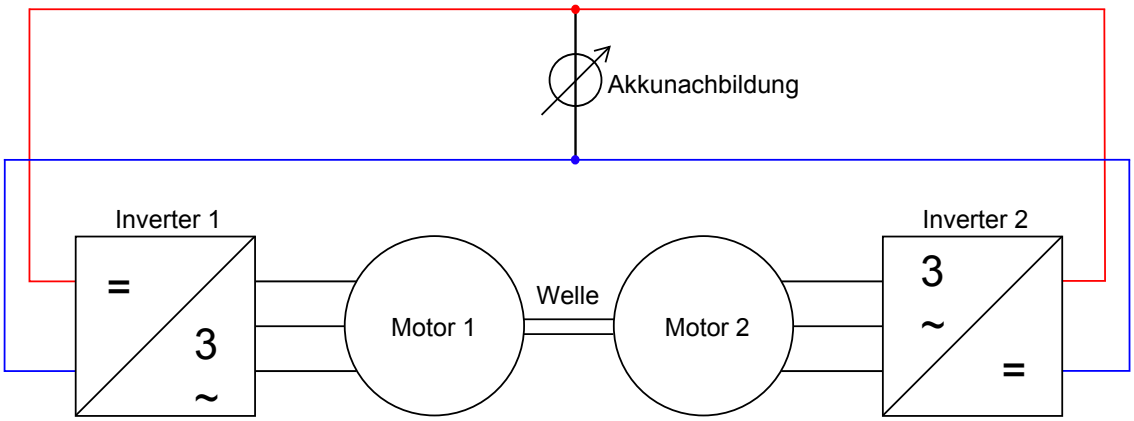


Abbildung 3.1: Aufbau Teststand

3.1 Inverter und Antriebsmaschine

Ein Inverter ist ein Gerät, welches in der Lage ist aus Gleichspannung Wechselspannung zu erzeugen. Die erzeugte Wechselspannung kann hierbei eine beliebige Frequenz und Amplitude im Rahmen der technischen Grenzen besitzen. Der Inverter soll aus der Akkumulatorspannung (Abbildung 3.1, „Akkunachbildung“) eine 3-phasige Wechselspannung für den Motor erzeugen. Im Bremsfall kann der Motor als Generator wirken. Der Inverter soll in diesem Fall die erzeugte Wechselspannung in Gleichspannung wandeln und somit den Akkumulator laden. Dies wird als Rekuperation bezeichnet. Eingesetzt wird ein Inverter der Firma Bosch vom Typ INV-CON.E-2.3 [18]. Der Inverter wird mittels einer CAN-Schnittstelle gesteuert. Zudem beinhaltet der Inverter einen DCDC-Wandler, welcher aus dem Hochvoltakku einen 12-Volt-Akku lädt, welche wiederum Komponenten wie beispielweise Steuergerät, Kühlmittelpumpe oder Display versorgt. Als Antriebsmaschine wird eine Synchronmaschine der Firma Bosch vom Typ SMG-180.1.3 eingesetzt. Die Maximalleistung der Maschine beträgt 90 kW [18, Seite 11]. Am Teststand ist das maximale Drehmoment der Maschinen auf 45 Nm begrenzt.

3.2 Akkunachbildung

Das Elektrorennfahrzeug soll später mit einem Lithium-Ionen Akkumulator betrieben werden. Diese soll aus Zellen der Firma Kokam bestehen [10]. Geplant ist ein Hochvoltakku mit 370 V Klemmspannung. Für den Teststand wird eine einstellbare Spannungsquelle bestehend aus einer Synchronmaschine sowie einem Gleichstromgenerator eingesetzt [5]. Die Steuerung der Gleichstromquelle kann mittels CAN mit dem Teststand kommunizieren.

4 Der Akkumulator

Der Hochvoltakku stellt die Energieversorgung des Umrichters und des Motors bereit. Zu Testzwecken sollte dieser modelliert werden. Die Modellierung des Akkumulators wurde mit MATLAB durchgeführt, da dieses Programm eine Schnittstelle zu CANoe besitzt. Diese ist notwendig, um die Akkunachbildung [5] anzusteuern.

4.1 Modellierung des Akkumulators

Zur Simulation des Akkumulators wurden drei Teilmodelle entworfen, das Ladezustandsmodell, das thermische Modell und das Klemmspannungsmodell. Für das Klemmspannungsmodell wurde zunächst ein Modell des elektrischen Verhaltens einer Kokam-Zelle entworfen.

4.1.1 Kokam-Zelle

Zur Simulation des Akkus wurde zunächst eine Kokam-Zelle anhand des Datenblatts [10, Seite 3] (siehe Abbildung 4.1) modelliert. Nachmodelliert wurden die Entladekurven bei verschiedenen Entladeströmen[1] um die Klemmspannung zu bestimmen. Anhand der Grafik wurden Stützstellen ermittelt. Durch diese Stützstellen wurden mittels Spline-Interpolation die Entladekurven erstellt. Bei einer Lagrange-Interpolation würden bei vielen Stützstellen Oszillationen aufteten. Um dies zu vermeiden, wurde eine kubische Spline-Interpolation gewählt, welche eine stückweise Interpolation durchführt. Dadurch tritt kaum Oszillation auf. Im m-Script „StartUp_Batt“ sowie „StartUp_Batt_no_fig“ werden nun für alle Stützstellen-Matritzen die Spline-Interpolation durchgeführt. Das Script „StartUp_Batt“ wird beim Starten der Simulation einmalig ausgeführt. Die daraus ermittelten Kurven (siehe Abbildung 4.2) werden in eine 2-D Lookup-Table[2] geschrieben. Die Eingangswerte sind die entnommene Ladung (Eingabe in hundertstel-Ah) und der entnommene Strom bezogen auf 1 C (5 A). Anhand dieser Lookup-Tabelle kann nun die Klemmspannung ermittelt werden. Zudem enthält das Modell die konstanten Werte für Leerlaufspannung und den Entladestromwert von 5 A. Die Gesamtkapazität ergibt sich aus der maximal entnehmbaren Ladung anhand der Entladekurven (siehe hierzu Abbildung 4.1). Das Modell simuliert nun das Klemmspannungsverhalten einer Kokam-Zelle (Abbildung 4.3). Anhang A beinhaltet einen Datenblattauszug der Kokam-Zelle. Ein Auszug aus dem Script zur Spline-Interpolation ist in Anhang B eingefügt.

[1] Ein „C“ entspricht bei dieser Zelle 5 A.

[2] Der n-D Lookup-Baustein errechnet anhand einer Interpolation in einer Wertetabelle einen Ausgangswert. Eine 2-D-Tabelle besitzt 2 Eingangsvariablen und eine Ausgangsvariable.

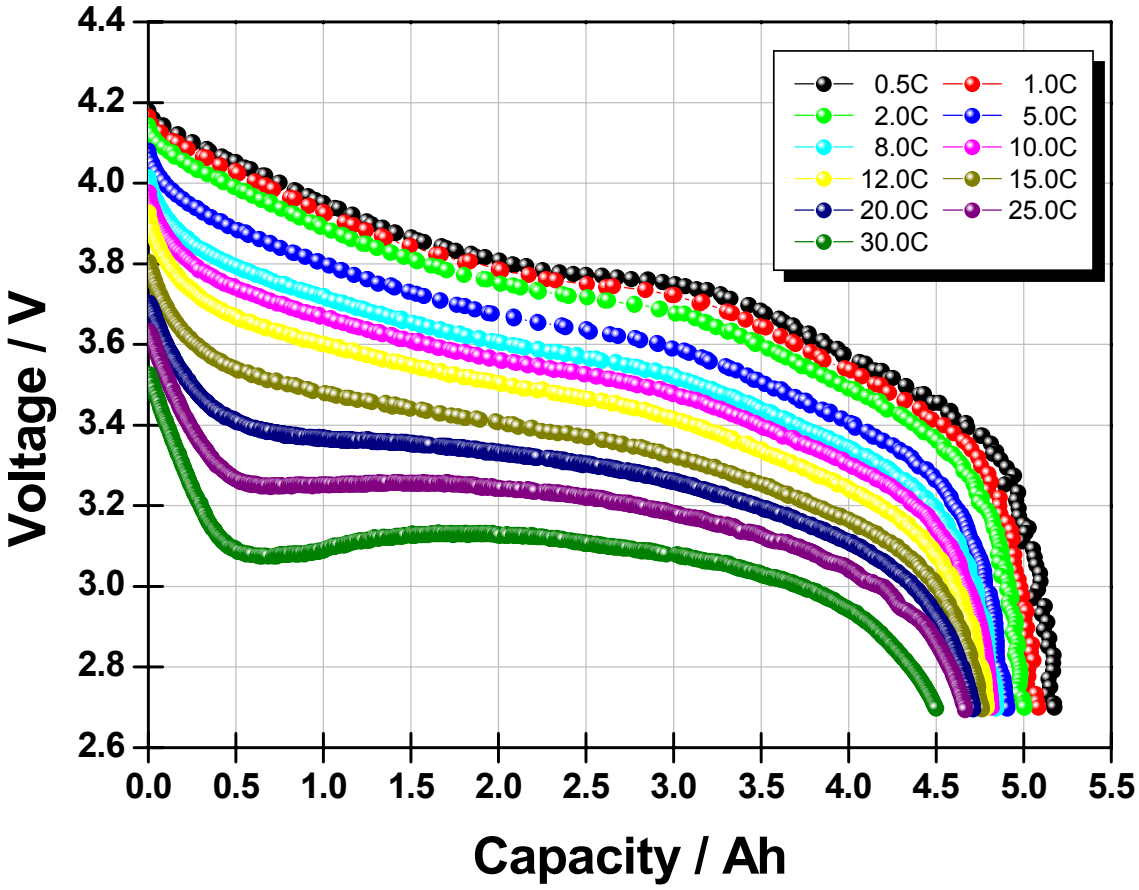


Abbildung 4.1: Kokam Zelle Entladespannung

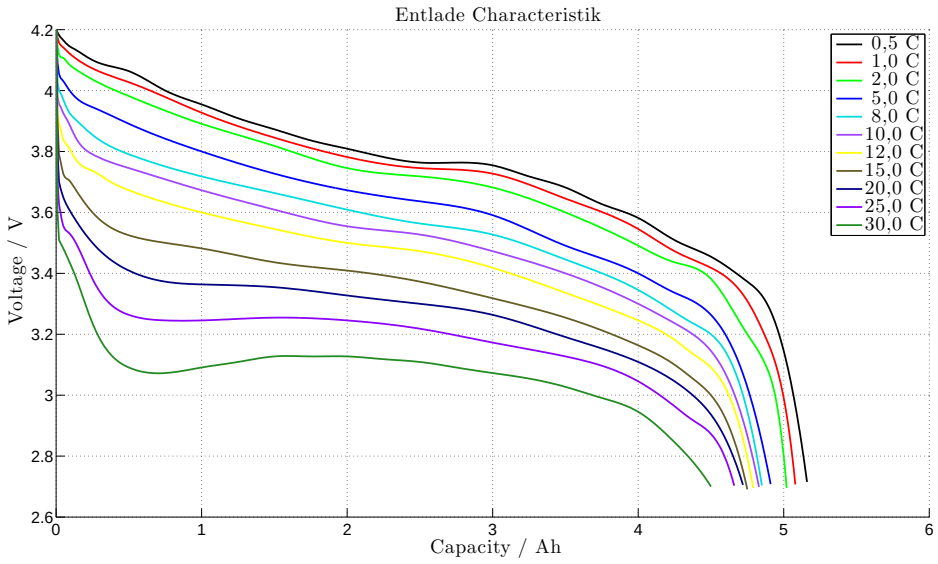


Abbildung 4.2: Ergebnis Spline Interpolation

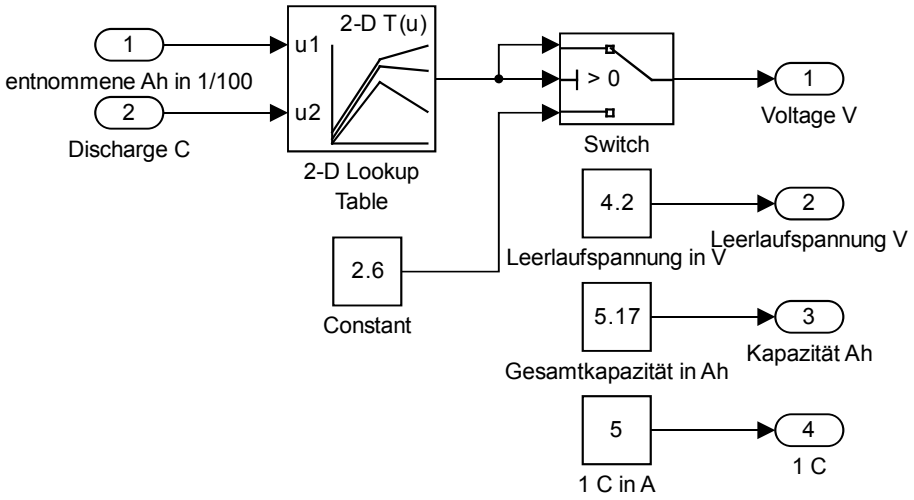


Abbildung 4.3: Modell Kokam Zelle

4.1.2 Klemmspannungsmodell

Das Klemmspannungsmodell berechnet die Klemmspannung des Akkumulators (siehe Abbildung 4.4). Dazu wird das Modell der Kokam-Zelle verwendet. Über den Eingangsport „SOC %“ (Input 1) wird der aktuelle Ladestatus (State of Charge) in Prozent vorgegeben[3]. Durch die Subtraktion von 100 % (Block „Subtract“) wird die entnommene Ladung in Prozent errechnet. Dieser Wert entspricht dem Entladungsgrad (DOD, depth of discharge). Multipliziert mit der Zellkapazität ergibt sich die entnommene Ladung in hundertstel-Ah (die Hundertstel ergeben sich aus der Multiplikation mit dem Entladungsgrad, da dieser nicht auf 1, sondern auf 100 % normiert ist). Der Eingangsport „I / mA“ (Input 3) gibt den aktuell entnommenen Strom in Milliampere wieder. Durch die Division durch 1000 (Block „mA in A“) wird der Strom in Ampere umgerechnet. Über den Eingangsport „Parallel“ (Input 4) wird die Anzahl der parallel verschalteteten Zellen eingegeben. Dieser Wert wird mit der Kapazität einer Zelle multipliziert, um die Gesamtkapazität des Akkumulators zu ermitteln (Block „Gesamtkapazität in Ah“). Der aktuelle Strom wird nun durch die Gesamtkapazität geteilt, um den Entladestrom in C zu erhalten (Block „Entladestrom in C“). Dieser Wert wird an das Kokam-Zelle-Modell weitergegeben, falls der SoC-Modus[4] auf 0 (Normaler Akkubetrieb) steht. Ist der aktuelle SoC-Modus jedoch auf 1 (Steuermodus), so wird die Entladekurve des Entladestromes von 1 C (Block „Constant2“) genutzt. Der Block „Switch“ übernimmt die Fallunterscheidung. Die aktuelle Akkuspannung errechnet sich aus der Zellspannung multipliziert mit der Anzahl der in Reihe geschalteten Zellen (Eingangsport „Zellen in Reihe“, Input 2). Am Spannungsausgang befindet sich zudem ein Tiefpassfilter (Block „Tiefpass“), da sich die Klemmspannung nicht sprunghaft ändern kann. Dieser ist auf die Grenzfrequenz von 1 Hz eingestellt[5]. Die Vorgabe der Frequenz erfolgt im Script „StartUp_Batt_no_fig“. Die Transfer-Funktion $\frac{1}{s+1}$ wird mittels der MATLAB-Funktion „tf“ und „c2d“ für die Code-

[3] Dieser Wert wird später durch das Ladezustandsmodell vorgegeben.
[4] SoC-Modus: siehe 4.1.4 auf Seite 15.
[5] Der Wert von 1 Hz berüht auf einer Annhame und muss mit Messungen bestätigt werden.

Erzeugung diskretisiert. Die Ausgangsgrößen des Klemmspannungsmodells sind die Klemmspannung in Volt (Output 1, „U Klemm V“), die Leerlaufspannung einer Zelle in Volt (Output 2, „U Leer V“) sowie die Zellkapazität in Ampere-Stunden (Output 3, „Kap Ah“).

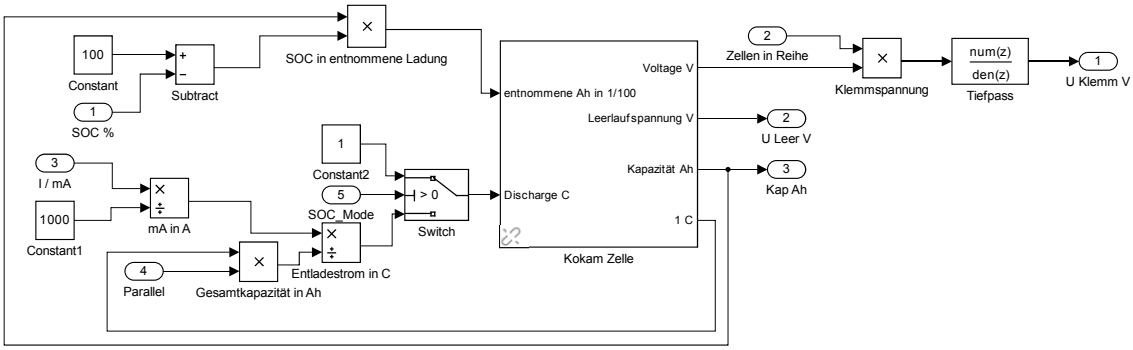


Abbildung 4.4: Klemmspannungsmodell

4.1.3 Thermisches Modell

Um das thermische Verhalten einer Lithium-Ionen-Zelle zu modellieren waren zunächst einige Vorbetrachtungen notwendig. Eine Zelle besteht aus einer Zusammenstellung vieler verschiedener Materialien, welche alle eine spezifische Wärmekapazität besitzen. Zur Vereinfachung wurde eine spezifische Wärmekapazität von c = 2 $\frac{\mathrm{kJ}}{\mathrm{kg \cdot K}}$ für alle Stoffe angenommen, da die Zelle aus Kohlen-Wasserstoffen ($c_{\mathrm{H}} = 14,304 \frac{\mathrm{kJ}}{\mathrm{kg \cdot K}}$, $c_{\mathrm{C}} = 0,709 \frac{\mathrm{kJ}}{\mathrm{kg \cdot K}}$), Grafit, Lithium ($c_{\mathrm{Li}} = 3,582 \frac{\mathrm{kJ}}{\mathrm{kg \cdot K}}$) und verschiedenen Metallen ($c_{\mathrm{Metall}} \sim 1 \frac{\mathrm{kJ}}{\mathrm{kg \cdot K}}$ [2]) besteht. Des Weiteren wurde der Abkühlvorgang der Zelle ausgewertet, um den abgegebenen Wärmestrom zu bestimmen. Die Zelle wurde dafür auf eine Temperatur von 38 $^{\circ}\mathrm{C}$ erwärmt. Die Zelle kühlte sich innerhalb von 35 Minuten auf 28 $^{\circ}\mathrm{C}$ ab. Bei einer Masse von 128 g ergab sich eine abgegebene Wärmemenge von 2,56 kJ (Gleichung 2.1) und ein durchschnittlicher Wärmestrom von 1,2 W (Gleichung 2.2).

Das thermische Modell (Abbildung 4.5) soll dazu dienen das thermische Verhalten der Zellen nachzubilden. In der Zelle entsteht Wärme durch Verlustleistung. Diese errechnet sich aus der Differenz zwischen Leerlaufspannung und Klemmspannung multipliziert mit dem Strom (Block „Pv_mW“). Durch den Integrator (Block „Watt_Joule“) wird diese in Wärmeenergie umgerechnet. Die Division durch die Masse des Akkublocks und die spezifische Wärmekapazität des Systems ergibt sich nach Gleichung 2.1 die Temperaturdifferenz. Diese wird zur Umgebungstemperatur addiert (Input „T Umg °C“) um die Gesamttemperatur zu erhalten. Der abgegebene Wärmestrom muss von der Verlustleistung subtrahiert werden, um den Kühleffekt mit einzubeziehen. Das Setzen des SoC-Modus auf Steuerbetrieb bewirkt ein rücksetzen des Integrators und ein abschalten der thermischen Simulation, da die Temperatur andernfalls unendlich steigen würde.

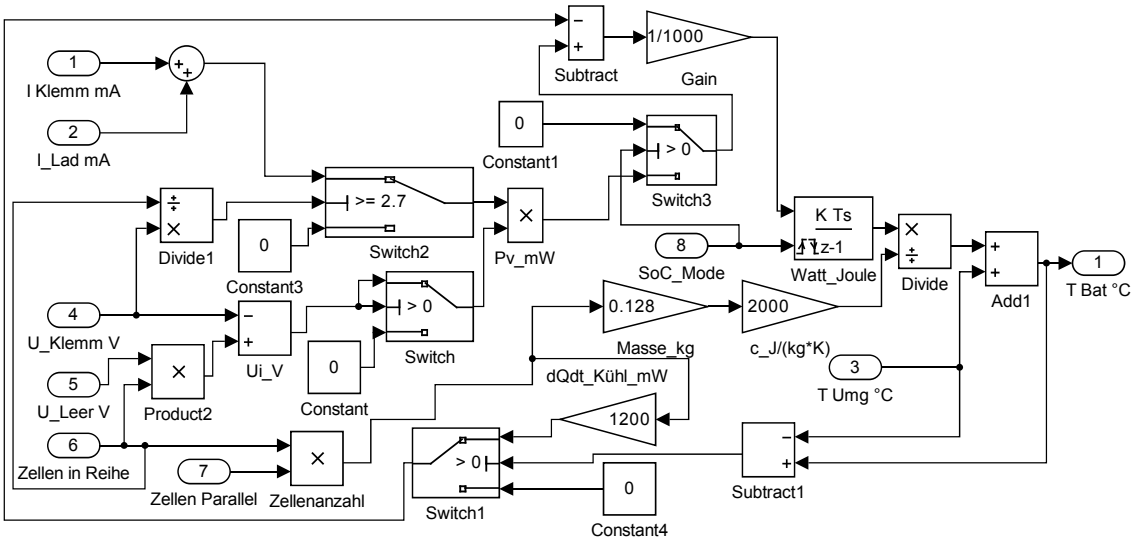


Abbildung 4.5: Thermisches Modell

4.1.4 Ladezustandsmodell

Das Ladezustandsmodell (Abbildung 4.6) soll den aktuellen Ladezustand des Akkumulators (SoC, State of Charge) errechnen. Zudem muss die Möglichkeit gegeben sein, einen festen Ladezustand vorzugeben, um Tests mit einer festen Spannung durchzuführen. Der Ladezustand wird durch die Integration des entnommenen Stromes errechnet. Je mehr Strom entnommen wurde, umso niedriger ist der Ladezustand. Es wurde ein Integrator eingesetzt (Block „Discrete-Time Integrator“), welcher den entnommenen Strom (Input 1, „I mA“ sowie Input 3, „Lade-I mA“) aufsummiert. Der Ladestrom wird hierbei von dem Entladestrom abgezogen. Falls der Ladestrom größer ist als der Entladestrom, wird dadurch ein negativer Wert aufintegriert, wodurch der Ladezustand steigt. Der Ausgangswert des Integrators stellt somit die entnommene Ladung in Milliamperesekunden bereit. Die maximal entnehmbare Ladung berechnet sich aus der Nennkapazität einer Zelle (Input 2, „Kap Ah / Zelle“) multipliziert mit der Anzahl der parallelen Stränge (Input 6, „Stränge Parallel“). Die ideale Gesamtladung liegt somit in Amperestunden vor und muss mit 3600 $\frac{\mathrm{s}}{\mathrm{h}}$ multipliziert werden, um die Stunden in Sekunden umzurechnen und mit 1000 multipliziert werden, um die Ampere in Milliampere umzurechnen. Die entnommene Ladung wird nun mit der Gesamtladung dividiert (Block „entnommene proz“) um das Verhältnis zu bestimmen. Das Ergebnis ist der Entladungegrad. Durch Subtraktion von eins erhält man die noch zur Verfügung stehende Ladung, den SoC. Durch Multiplikation mit 100 erhält man den SoC in ganzzahligen Prozentwerten.

Die Vorgabe eines festen Ladezustands erfolgt über den SoC-Modus (Input 5, „SoC-Mode“). Ist dieser Wert null, so wird der SoC normal errechnet. Falls der SoC-Modus auf eins steht, so wird keine Stromintegration durchgeführt (Block „Switch1“) und es wird permanent der vorgegebene SoC (Input 4, „SollSOC %“) ausgegeben. Wird der SoC-Modus zurück auf null gestellt, so wird die bereits entnommene Ladung, abhängig vom eingestellten Ladezustand, als Startwert für den Integrator berechnet. Hierfür

wird der vorgegebene SoC von 100 subtrahiert, um die entnommenen Prozent zu bestimmen (Block „ent Proz soll“). Dieser Wert wird mit der Maximalladung multipliziert (Block „ent Kap soll“) und mit 100 dividiert (Block „ent Lad soll“), um die entnommene Ladungsmenge zu bestimmen. Diese wird dem Integrator vorgegeben (Integrator-Input „y_0“). Das Umschalten des SoC-Modus bewirkt durch den Reset-Input des Integrators ein Rücksetzen des Integrators und damit die Übernahme des neuen Startwertes.

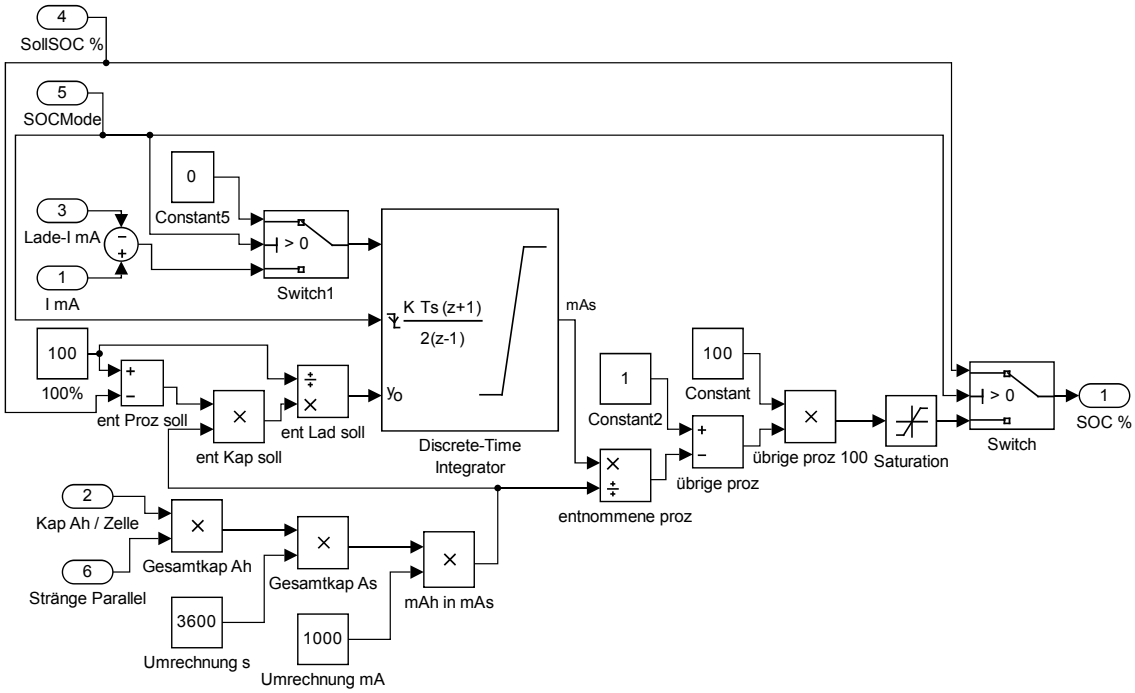


Abbildung 4.6: Ladezustandsmodell

4.1.5 Gesamtmodell

Das Akkumulatormodell (Abbildung 4.7) besteht aus den vorangegangenen Modellen. Diese Form ermöglicht es, einzelne Untermodelle unabhängig von den anderen Modellen zu verändern oder auszutauschen.

4.1.6 Testen des Modells

Zum Testen des Modells (Abbildung 4.8) wurden verschiedene Anwendungsfälle simuliert. Da im Modell Entladestrom und Ladestrom getrennt sind, muss an dieser Stelle eine Fallunterscheidung stattfinden. Ist der Strom positiv, so handelt es sich um einen Entladestrom, andernfalls um einen Ladestrom.

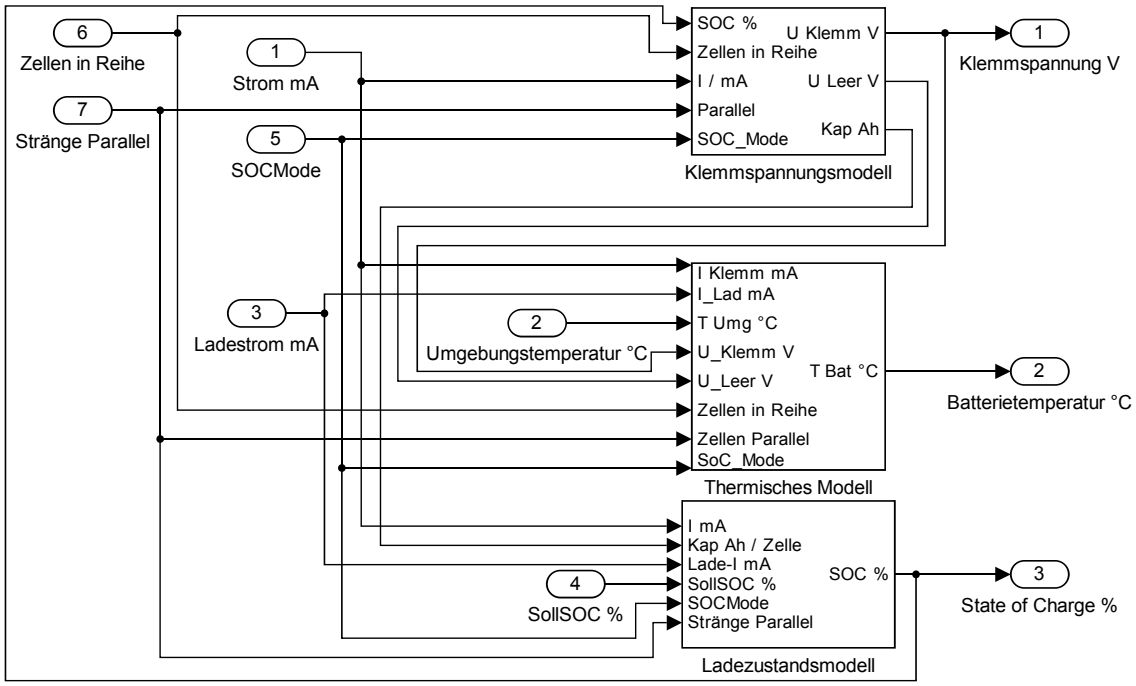


Abbildung 4.7: Akkumulatormodell

4.1.6.1 Szenario 1

Im ersten Anwendungsfall wird eine Kokam-Zelle simuliert, welche mit einem konstanten Entladestrom entladen wird. Folgende Vorgaben wurden eingestellt (Tabelle 4.1):

Größe	Wert
Strom	2,5 A (konstant)
Umgebungstemperatur	20 $\mathrm{°C}$
SoC-Modus	0 (konstant)
Zellen in Reihe	1
Stränge parallel	1

Tabelle 4.1: Vorgabe Szenario 1

Der Entladestrom von 2,5 A entspricht 0,5 C. Daraus ergibt sich eine rechnerische Entladedauer von 2 Stunden, da das Entladen mit 1 C eine Stunde dauert. Der konstante Wert 0 für den SoC-Modus bewirkt, dass eine normale Entladung stattfindet. Ausgewertet wurde der Klemmspannungsverlauf, die Akkutemperatur sowie der Ladezustand.

Der Klemmspannungsverlauf (Abbildung 4.9a) ist ähnlich mit der Entladekurve für 0,5 C (siehe Abbildung 4.1). Zu Beginn ist die Spannung größer als die angegebene Zellspannung ([10, Seite 2]: Nominal Voltage), der Durchschnittswert entspricht jedoch 3,7 V. Zu erkennen ist auch eine Plateauphase, in der die Spannung annähernd konstant ist. Die Klemmspannung unterschreitet nach über 2 Stunden (7330 s = 2,036 h) die Spannung von 2,7 V ([10, Seite 2]: Cut-off Voltage), somit ist die Zelle leer. Das die Entladung länger als 2 Stunden dauert, liegt daran, dass der Akku bei Entladung mit 0,5 C mehr

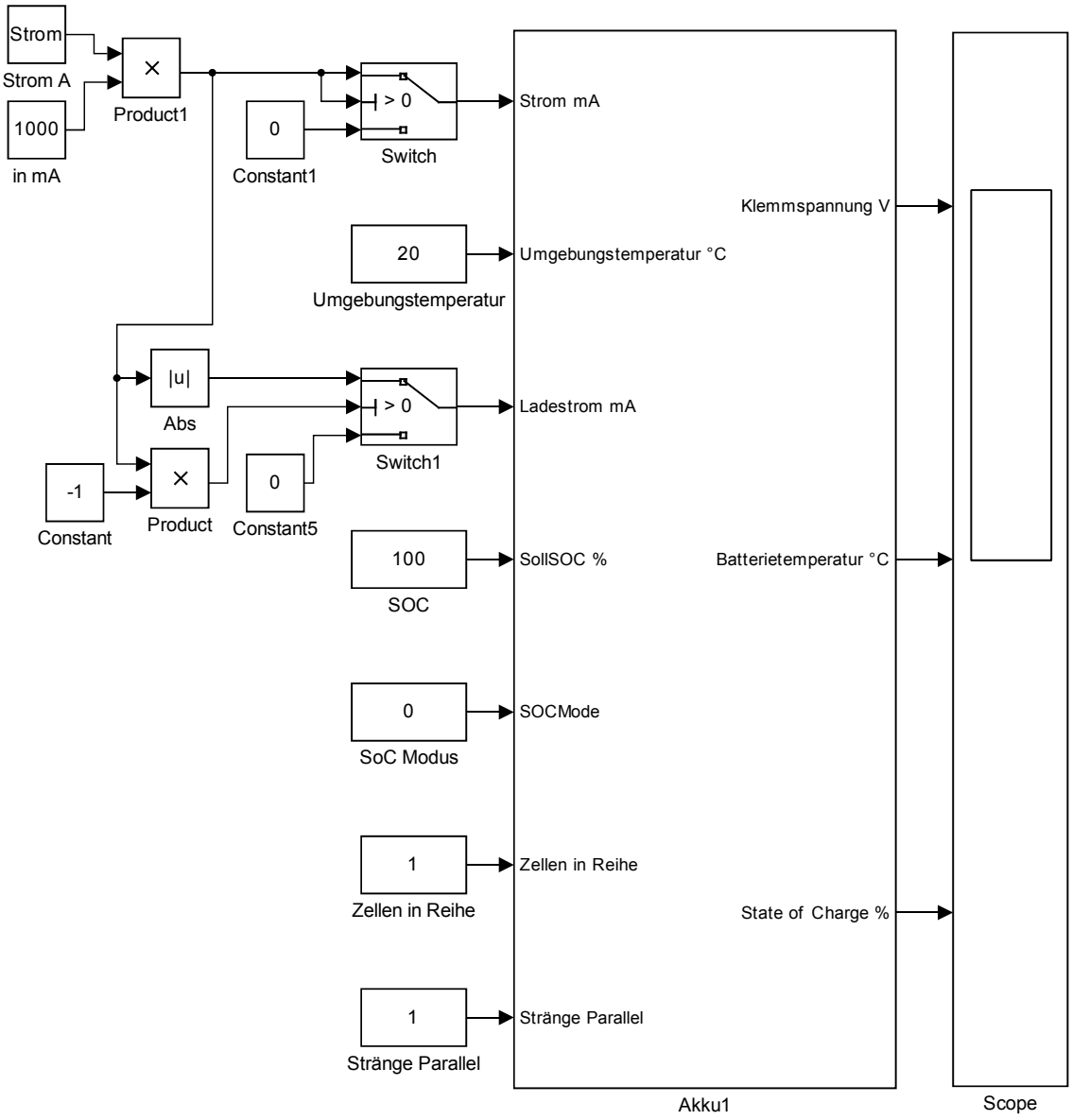
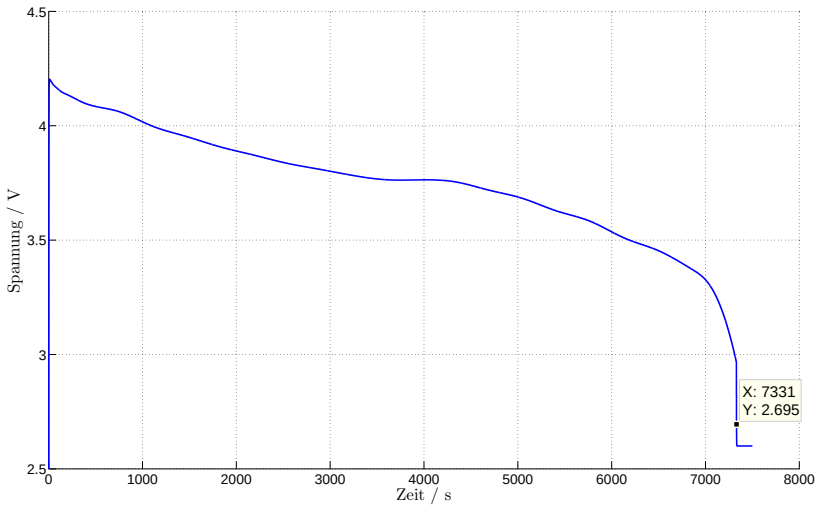


Abbildung 4.8: Testumgebung des Modells

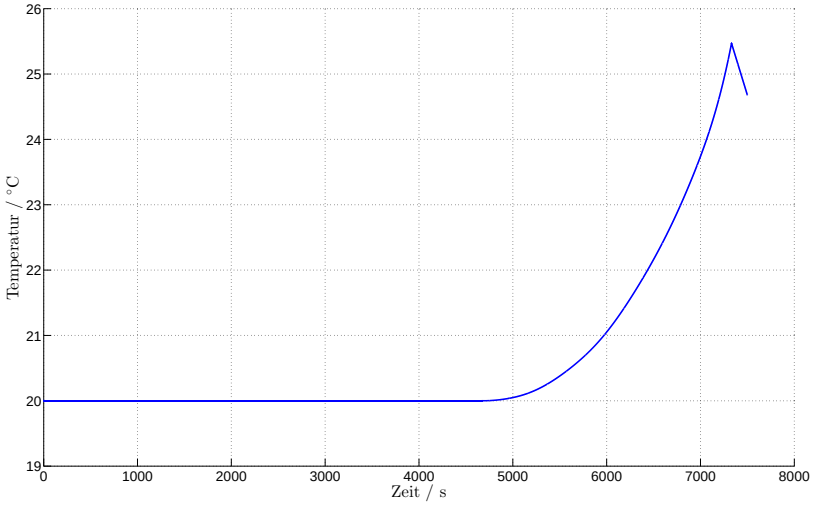
als 5 Ah abgibt. Der Temperaturverlauf (Abbildung 4.9b) zeigt während der Entladung einen Temperaturanstieg bis zur Maximaltemperatur von 25,5 $^{\circ}\mathrm{C}$. Zu erkennen ist auch, dass bis etwa 5000 Sekunden keine Erwärmung stattfindet, da sich die Zelle durch die geringe Leistung schneller abkühlt als erwärmt. Sobald die Zelle leer ist, sinkt die Temperatur ab. Der Ladezustand (Abbildung 4.9c) sinkt konstant ab, da ein konstanter Strom entnommen wurde.

4.1.6.2 Szenario 2

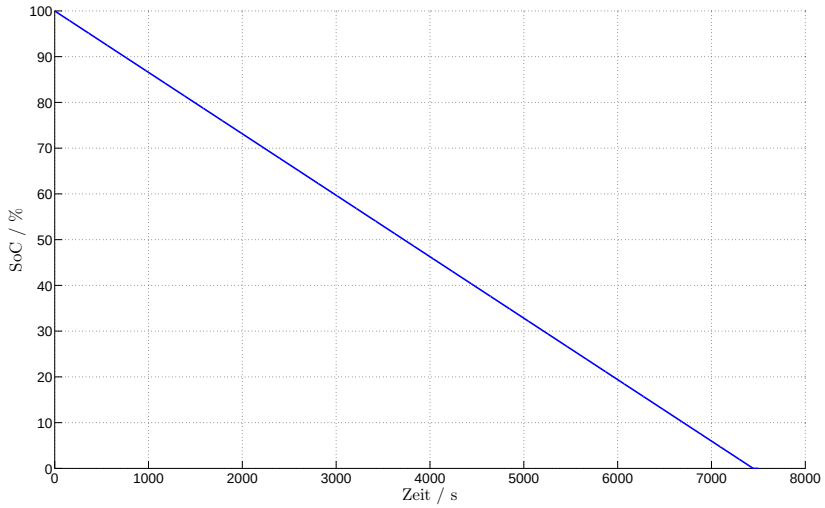
Für den zweiten Anwendungsfall wurde eine Verschaltung gewählt, wie sie auch im Rennfahrzeug angewandt werden könnte (Tabelle 4.2):



(a) Klemmspannungsverlauf



(b) Temperaturverlauf



(c) Ladezustandsverlauf

Abbildung 4.9: Szenario 1

Größe	Wert
Strom	15 A (konstant)
Umgebungstemperatur	20 °C
SoC-Modus	0 (konstant)
Zellen in Reihe	100
Stränge parallel	3

Tabelle 4.2: Vorgabe Szenario 2

Einhundert Zellen in Reihe bewirken, dass die Akku-Klemmspannung das Einhundertfache der Zellspannung beträgt. Durch drei parallele Stränge verdreifacht sich die Ladungsmenge. Dadurch entspricht ein Strom von 15 A der Laderate von 1 C.

Im Klemmspannungsverlauf (Abbildung 4.10a) ist die Durchschnittsspannung von 370 V zu erkennen. Zudem ist zu erkennen, dass der Entladevorgang mit 1 C etwa eine Stunde (3600 s = 1 h) dauert. Der Temperaturverlauf (Abbildung 4.10b) zeigt einen Temperaturanstieg bis etwa 38 °C. Hierbei ist zu beachten, dass die in der Simulation verwendeten Parameter im thermischen Modell (Abbildung 4.5) geschätzt sind und zudem keine Kühlung simuliert wurde. Der Ladezustand (Abbildung 4.10c) sinkt linear ab.

4.1.6.3 Szenario 3

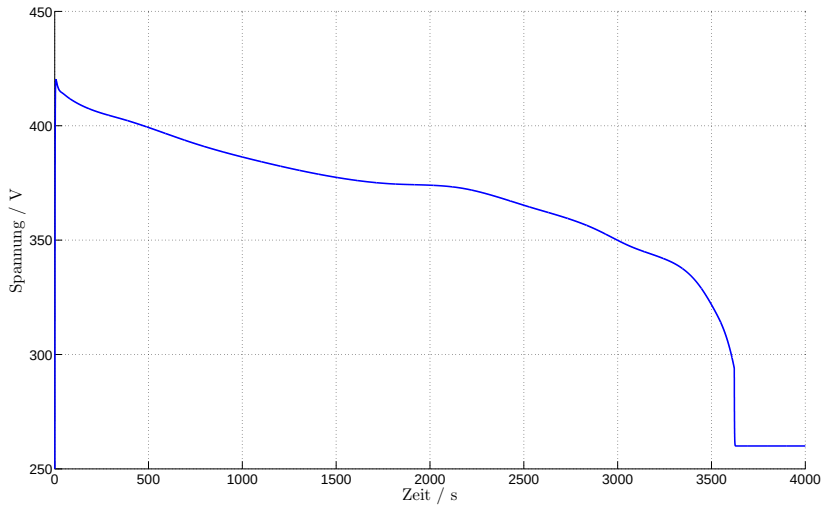
Im dritten Szenario wird ein variabler Strom zur Überprüfung des Modells entnommen (Tabelle 4.3):

Größe	Wert
Strom	variabel (Abbildung 4.11)
Umgebungstemperatur	20 °C
SoC-Modus	0 (konstant)
Zellen in Reihe	100
Stränge parallel	3

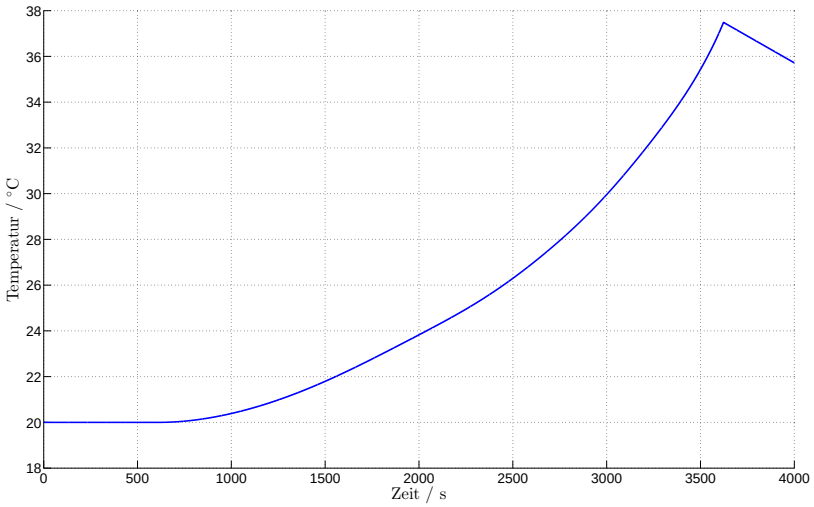
Tabelle 4.3: Vorgabe Szenario 3

Zum Testen wurde ein Strom mit sinusförmigen Entladungsverlauf (Abbildung 4.11) verwendet. Die Amplitude beträgt 15 A. Während der negativen Halbwelle findet eine Ladung des Akkumulators statt. Um mehr zu entladen als zu laden wurde ein Offset von 5 A eingestellt.

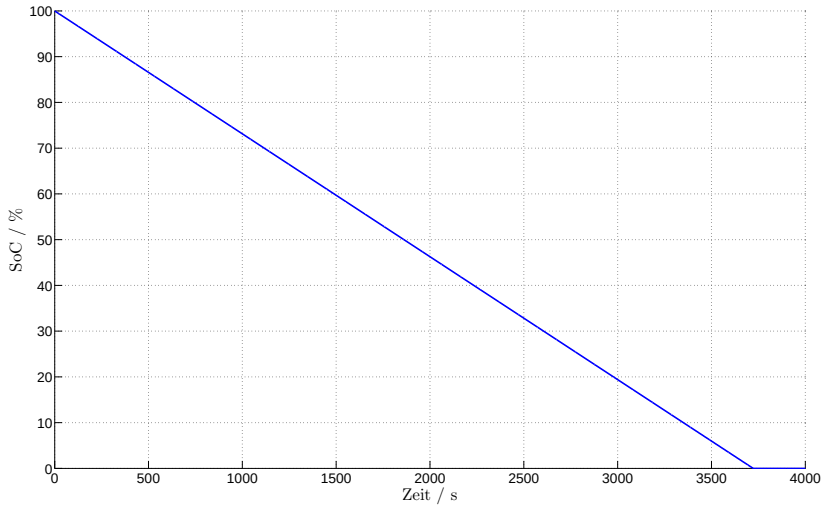
Der Klemmspannungsverlauf (Abbildung 4.12a) ist sehr nichtlinear, weil ein Stromanstieg ein Klemmspannungsabsinken bewirkt und umgekehrt. Zudem steigt die Spannung bei jedem Ladezyklus. Ab etwa 11000 s ist zu erkennen, dass der Akku leer ist, jedoch mit jeder negativen Halbwelle wieder etwas geladen wird und sich anschließend wieder entläd. Anhand des Temperaturverlaufs (Abbildung 4.12b) ist zu erkennen, dass



(a) Klemmspannungsverlauf



(b) Temperaturverlauf



(c) Ladezustandsverlauf

Abbildung 4.10: Szenario 2

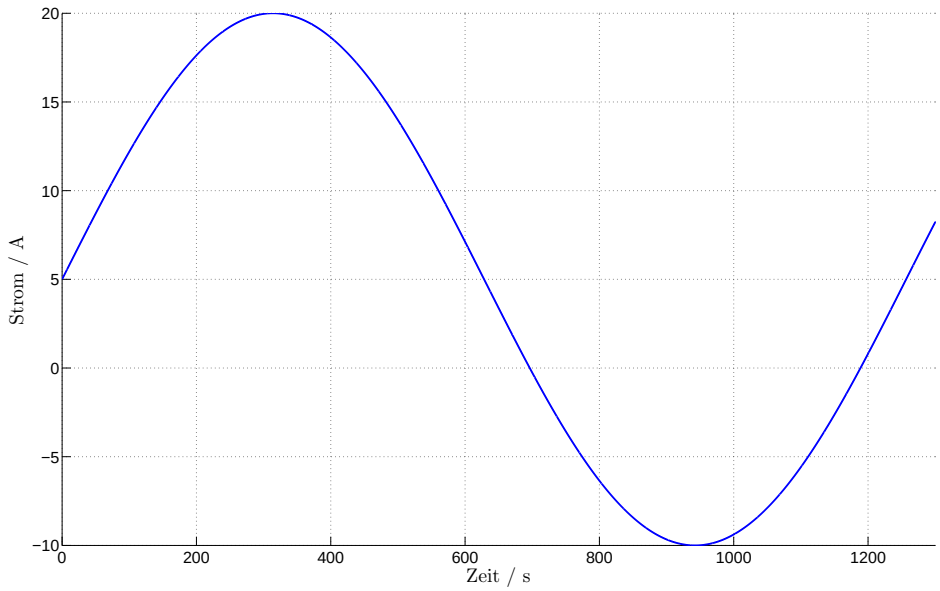


Abbildung 4.11: entnommener Strom Szenario 3

der Akku eine Maximaltemperatur von 41,39 °C erreicht. In den Phasen, in welchen der Akku leer ist, kühlt dieser sich wieder ab. Im Ladezustandsverlauf (Abbildung 4.12c) ist zu erkennen, dass der Akkumulator sinusförmig geladen und entladen wird. Auch hier ist zu erkennen, dass dieser ab etwa 11000 s leer ist und und in der folgendenden Ladephase nachgeladen wird.

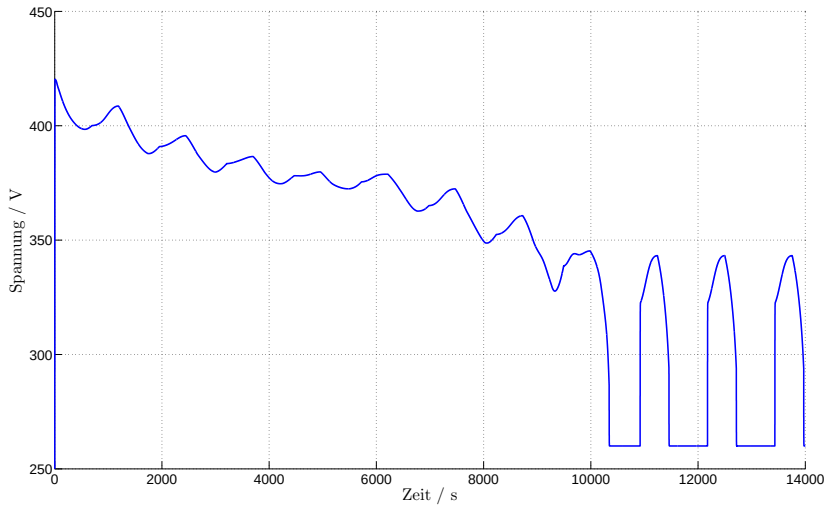
4.1.6.4 Szenario 4

Im vierten Anwendungsfall wird der SoC-Modus getestet (Tabelle 4.4):

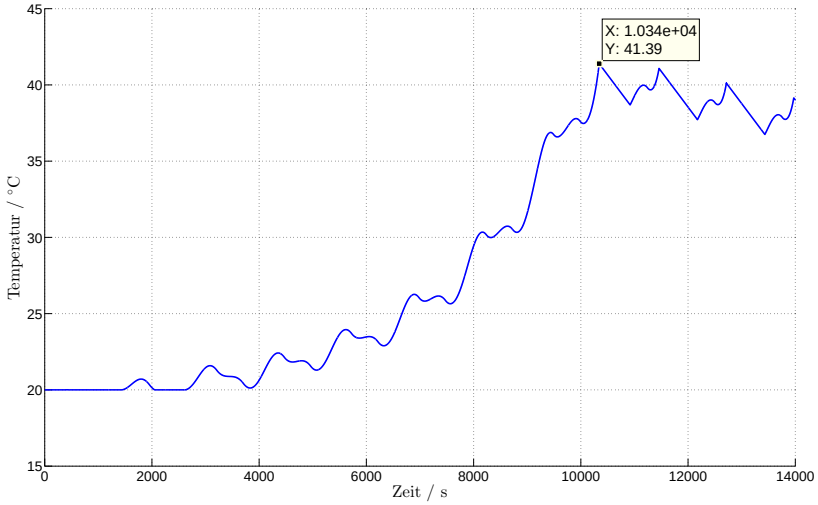
Größe	Wert
Strom	25 A
Umgebungstemperatur	20 °C
SoC-Modus	variabel
SoC Sollwert	80 %
Zellen in Reihe	1
Stränge parallel	1

Tabelle 4.4: Vorgabe Szenario 4

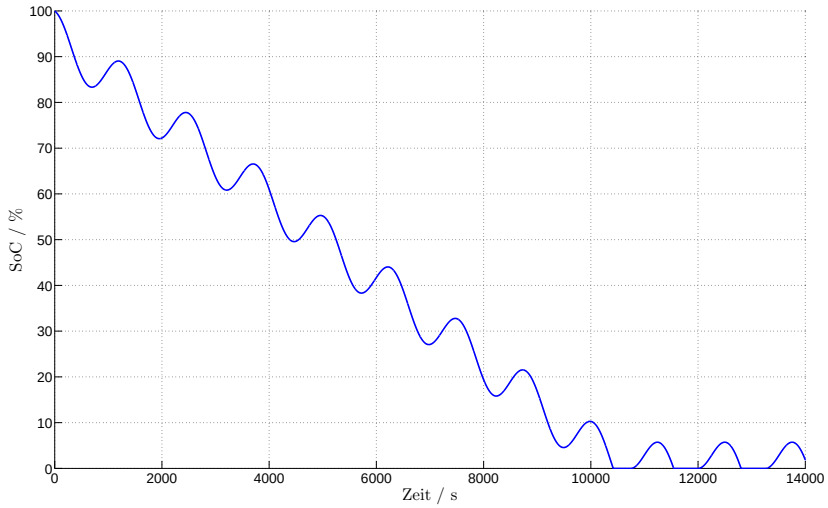
Die Funktionalität des SoC-Modus dient dazu entweder mit einer festen Spannung (einem festen Ladezustand) zu arbeiten (SoC-Modus = 1) oder einen definierten Ladezustand als Startwert vorzugeben (fallende Flanke an SoC-Modus). Für diesen Test wird der SoC-Modus aller 500 Sekunden umgeschaltet. Der Ladezustandssollwert liegt bei 80 %. Der Strom beträgt 25 A.



(a) Klemmspannungsverlauf



(b) Temperaturverlauf



(c) Ladezustandsverlauf

Abbildung 4.12: Szenario 3

Im Klemmspannungsverlauf (Abbildung 4.13a) ist zu sehen, dass zu den Zeiten während denen der SoC-Modus gleich 0 ist, der Akku sich normal verhält. Ist der SoC-Modus gleich 1, so ist die Klemmspannung stabil. Im Umschaltmoment von 1 auf 0 wird der vorgegebene Ladezustand übernommen. Der Temperaturverlauf (Abbildung 4.13b) zeigt in den gesteuerten Phasen eine Temperatur von 20 $^{\circ}\mathrm{C}$ an. In den ungesteuerten Phasen ist ein Temperaturanstieg zu erkennen. Im Ladezustandsverlauf (Abbildung 4.13c) ist zu erkennen, dass dieser konstant sinkt beziehungsweise konstant ist.

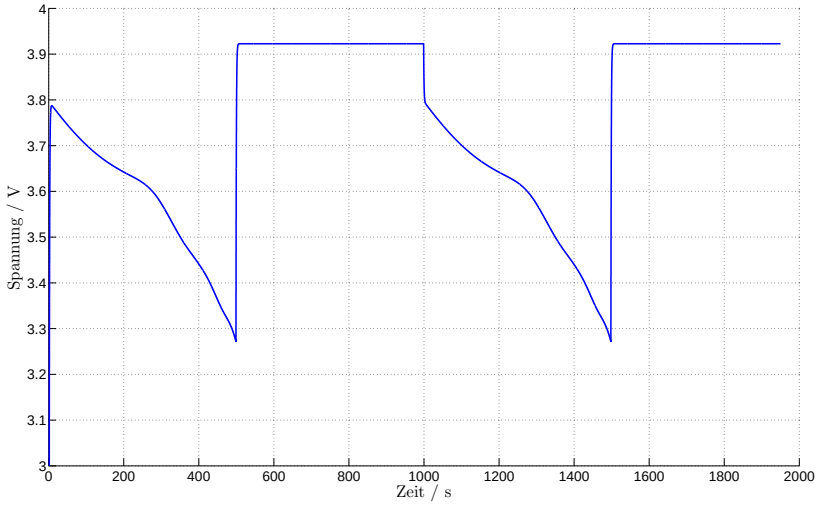
4.1.7 Einbindung in CANoe

Das Akkumulatormodell wurde für CANoe in C-Code übersetzt. Nach der Codeerzeugung kann das Modell in CANoe als dynamische Bibliothek eingebunden werden. Das Modell wird dadurch in CANoe ausgeführt. Zur Kommunikation müssen Signaleingänge und Signalausgänge genutzt werden (Abbildung 4.14). Als Stromeingangswert wird der Wert „CurrentActualHV“ (aktueller Strom Hochvoltseite) von Inverter 1 genutzt. Der Akku-Klemmspannungswert wird zunächst in eine Systemvariable geschrieben und von CANoe in eine CAN-Nachricht eingesetzt. Dieser Wert wird an die Steuerung der Spannungsversorgung (siehe [5]) gesendet.

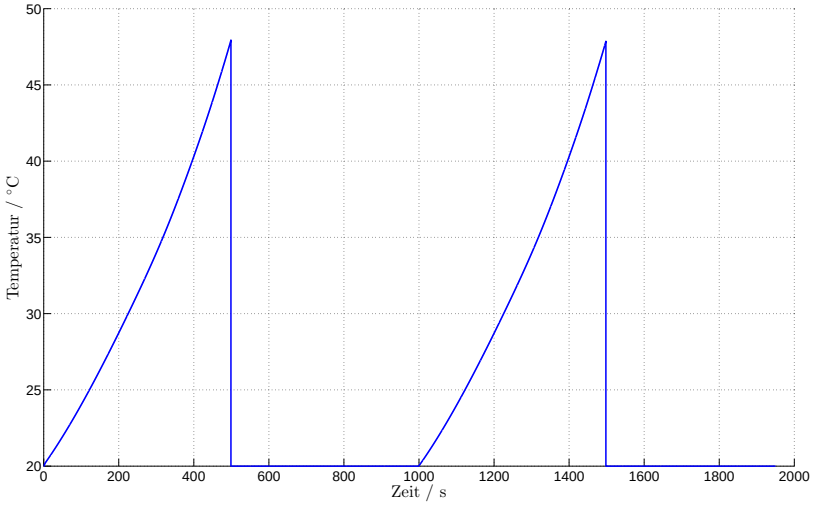
4.2 Handlungsempfehlungen für kritische Akkuzustände

Zum Schutz des Akkus müssen kritische Ladezustände erkannt und entsprechende Handlungen durchgeführt werden. Ein kritischer Zustand ist die Rekuperation bei vollem Akku. Rekuperation ist die Energierückgewinnung, die durch das Bremsen erzeugte Energie wird in den Akku zurückgespeist. Bei einem vollen Akku würde dies zu einer Überladung führen. Daher wird das Laden durch Rekuperation bei einer Akkuspannung über 410 V untersagt, da ab dieser Spannung von einem vollen Akku ausgegangen werden kann. Eine Zelle darf mit maximal 10 A geladen werden (siehe [10]: Charge Condition: Max. Current). Ein Akku mit drei parallelen Strängen darf demnach mit maximal 30 A geladen werden. Zudem ist die Rekuperation laut FSE-Regelwerk erst ab einer Geschwindigkeit größer 5 $\frac{\mathrm{km}}{\mathrm{h}}$ erlaubt [7, Seite 101]. Die Rekuperation des Inverter kann mittels einer CAN-Botschaft begrenzt werden. Diese Vorgaben wurden im ein Modell umgesetzt, welches später auf dem Steuergerät abgearbeitet werden soll. Das Modell ist in Abbildung 4.15 zu sehen. Das Subsystem „U/min in km/h“ rechnet die aktuelle Geschwindigkeit in Kilometer pro Stunde um.

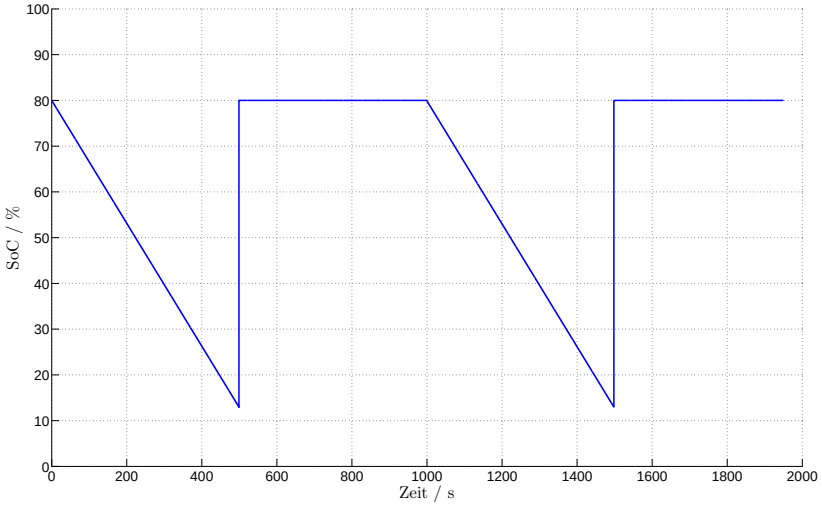
Das Modell wurde mit einem variablen Inverterstrom getestet. Der Test ist in Abbildung 4.16a zu sehen. Abbildung 4.16b zeigt die zugehörige Klemmspannung. Zum Zeitpunkt 10 Sekunden wurde begonnen, einen Strom von 60 A zu entnehmen. Da zu diesem Zeitpunkt die Klemmspannung größer als 410 V ist, wird der maximale Rekuperati-



(a) Klemmspannungsverlauf



(b) Temperaturverlauf



(c) Ladezustandsverlauf

Abbildung 4.13: Szenario 4

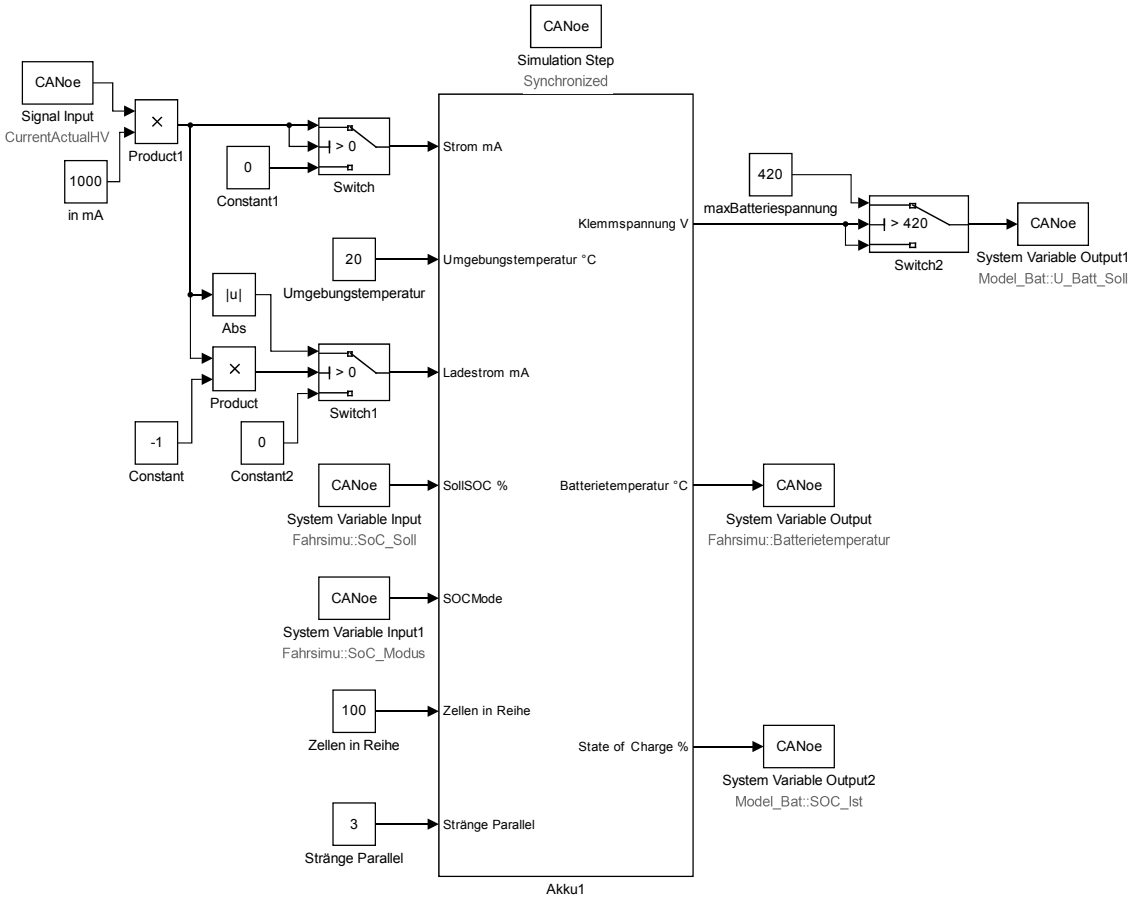


Abbildung 4.14: Akkumulatormodell mit CANoe

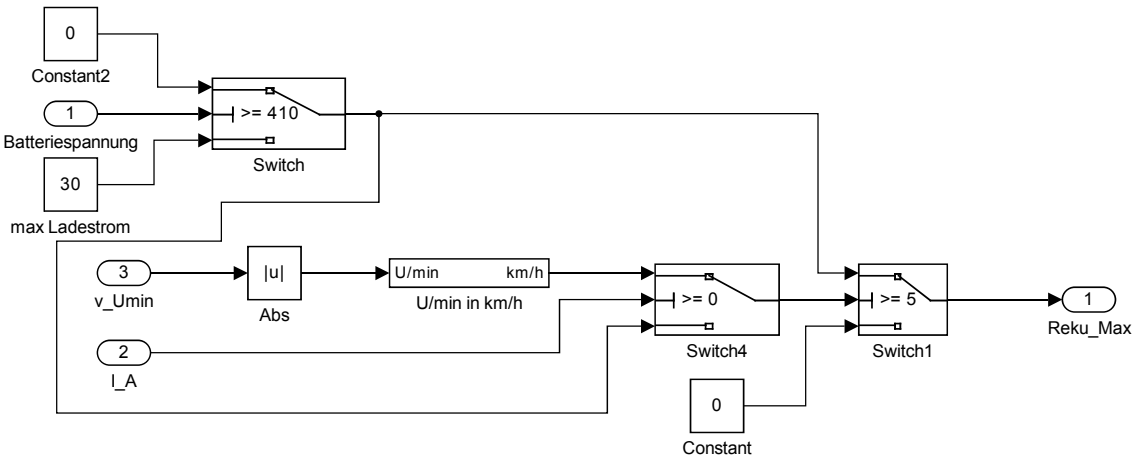


Abbildung 4.15: Modell Rekuperationssteuerung

onsstrom auf 0 A gesetzt. Erst nachdem die Klemmspannung den Grenzwert unterschreitet (15 Sekunden) wird der maximale Strom auf 30 A gesetzt. Für diesen Test wurde die Geschwindigkeit des Fahrzeugs anhand des Stroms ausgewertet. Sinkt die Geschwindigkeit unter 5 $\frac{\mathrm{km}}{\mathrm{h}}$ (0,6 A), so wird der Rekuperationsstrom auf 0 gesetzt (73 bis 80 Sekunden). Steigt die Geschwindigkeit erneut an, so wird wieder der Maximalwert gesetzt. Im Rekuperationsfall (ab 90 Sekunden) steigt die Klemmspannung wieder an, da Energie zurückgespeist wird. Übersteigt die Klemmspannung den Wert 410 V, so wird die Rekuperation wieder deaktiviert (ab 93 Sekunden).

Ein weiterer kritischer Zustand ist die Entladung bei einem niedrigen Ladezustand. Hierbei kann es zu einer Tiefenentladung kommen, welche die Akkuleistungsfähigkeit nachhaltig beeinflussen würde. Unterhalb der „Cut-off Voltage“ [10] darf kein Strom mehr entnommen werden. Bei 100 Zellen in Reihe ergibt sich eine minimale Akkuspannung von 270 V. Die minimale Betriebsspannung für den Inverter beträgt 280 V [18, Seite 71]. Daher wurde ein Modell entworfen, welches den Betriebszustand der Inverter auf „Stand By“ setzten soll, sobald der Akku die Spannung von 280 V unterschreitet. Abbildung 4.17 zeigt das Modell. Der Output „State_INV“ muss in der Fahrzeugsteuerung mit dem Zustandsautomaten, welcher den Inverter steuert [13], verbunden werden, um die Abschaltung einzuleiten.

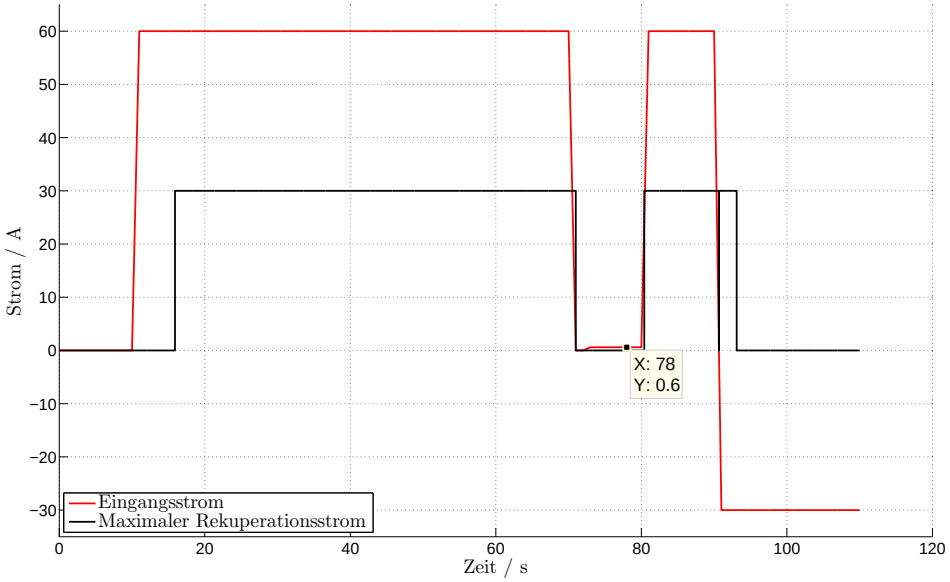
Zudem muss die Temperatur des Akkumulators überwacht werden. Die maximale Entladetemperatur einer Zelle beträgt 60 $^{\circ}\mathrm{C}$. Aus diesem Grund wurde ein Modell entworfen, welches bei einer Akkutemperatur über der Grenztemperatur den Inverterstatus auf „Stand By“ setzen soll, um den Akku zu schützen. Abbildung 4.18 zeigt das Modell. Die Modelle wurden auf das Steuergerät übertragen.

4.3 Restwegberechnung

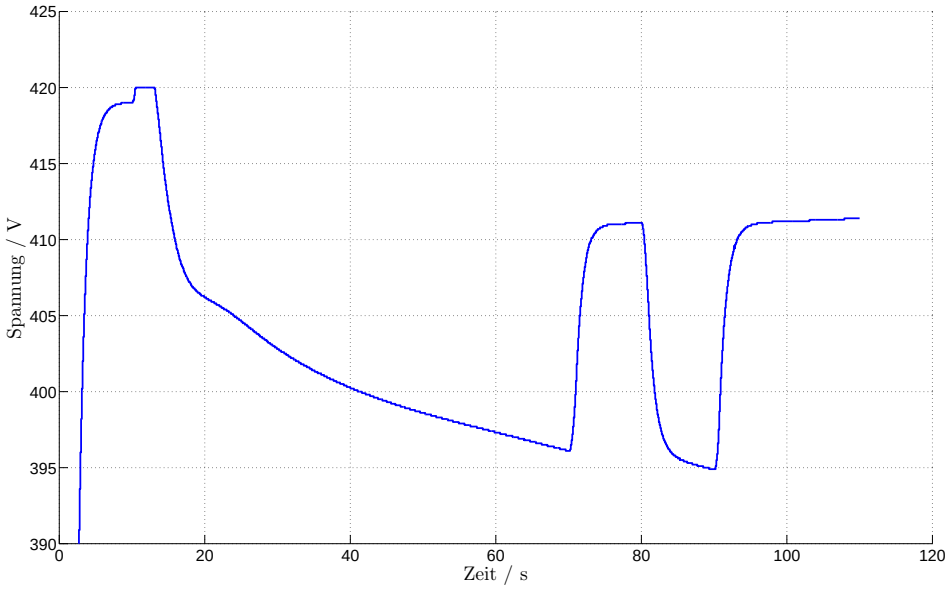
Zur Unterstützung des Fahrers soll anhand des Ladezustands eine Abschätzung getroffen werden, welchen Weg das Fahrzeug noch zurücklegen kann. Dieser Wert soll später im Display angezeigt werden. Hierzu ist es notwendig, zunächst die physikalischen Gegebenheiten zusammenzufassen. Bei dem Rennfahrzeug handelt es sich um einen Objekt, welches mittels mechanischer Arbeit bewegt wird. Die Energiemenge ist durch den Akku begrenzt. Der Akku besteht aus insgesamt 300 Kokam-Zellen[6]. Einhundert Zellen in Reihe ergeben eine Spannung von 370 V. Durch drei parallele Stränge ergibt sich eine Ladung von 15 Ah. Daraus ergibt sich nach Formel 2.3:

$$W_{\mathrm{el}} = U \cdot Q = 370\,\mathrm{V} \cdot 15\,\mathrm{Ah} = 5550\,\mathrm{Wh} = 19980\,\mathrm{kWs} \tag{4.1}$$

[6] Alle in diesem Abschnitt verwendeten Parameter können in dem Script „V_num_StartUp.m“ geändert werden, sobald reale Werte zur Verfügung stehen. Das Script ist in Anhang C beigefügt.



(a) Ströme



(b) Klemmspannung

Abbildung 4.16: Test Rekuperation

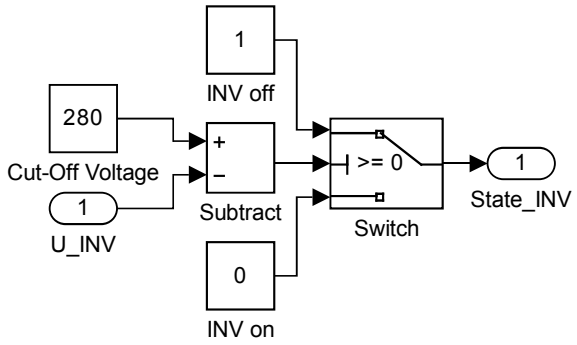


Abbildung 4.17: Tiefenentladungsschutz

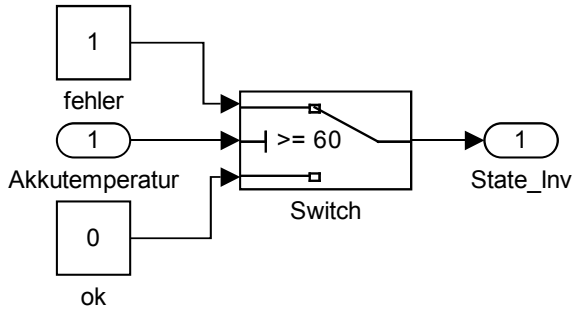


Abbildung 4.18: Akkutemperaturüberwachung

eine Gesamtarbeit von etwa 5,55 kWh bei vollem Akku. Für die Berechnung des Weges ist zudem die Normalkraft des Fahrzeuges zu betrachten. Um die Gewichtskraft zu bestimmen, wird eine Gesamtmasse m von 380 kg angenommen, welche sich aus 320 kg Fahrzeugmasse und 60 kg Fahrergewicht zusammensetzt. Für die Gravitationskonstante g wird 9,81 $\frac{\mathrm{m}}{\mathrm{s}^2}$ eingesetzt. Da die Gewichtskraft senkrecht wirkt, kann diese als Normalkraft angesehen werden. Da $F_\mathrm{G} \parallel F_\mathrm{N}$ gilt, ergibt sich laut Formel 2.5:

$$F_\mathrm{N} = m \cdot g = 380\,\mathrm{kg} \cdot 9,81\,\frac{\mathrm{m}}{\mathrm{s}^2} = 3727,8\,\mathrm{N} \tag{4.2}$$

eine Normalkraft von 3,7 kN um das Objekt zu bewegen. Da das Objekt auf Rädern rollt, wird die Reibungskraft F_R mittels einer Rollreibungszahl μ_R gleich 0,03 (Gummireifen auf Asphalt [4]) einbezogen. Nach Formel 2.4 ergibt sich:

$$F_\mathrm{R} = \mu_\mathrm{R} \cdot F_\mathrm{N} = 3727,8\,\mathrm{N} \cdot 0,03 = 111,834\,\mathrm{N} \tag{4.3}$$

eine Kraft von 112 N für die gleichförmige Bewegung in der Ebene. Durch Formel 2.3 kann nun der Weg s berechnet werden. Es ergibt sich

$$s = \frac{W}{F} = \frac{19980\,\mathrm{kWs}}{111,834\,\mathrm{N}} = \frac{19980\,\mathrm{kNm}}{111,834\,\mathrm{N}} = 178,657\,\mathrm{km} \tag{4.4}$$

ein möglicher Maximalweg von 178,7 km. Bei diesem Idealfall ist die Beschleunigung sowie der Luftwiderstand gleich Null und alle Wirkungsgrade werden als eins angenommen. Daher kann dieser Wert als theoretischer Maximalweg betrachtet werden.

Der Luftwiderstand eines Objektes kann nach Formel 2.6 errechnet werden. Für den Strömungskoeffizient wurde der messtechnisch ermittelte Wert des aktuellen Rennfahrzeuges [20] genutzt ($c_W = 0,53$). Ebenso wurde der Wert für die Referenzfläche aus dieser Quelle genutzt ($A = 0,77\,m^2$). Als Dichte des strömenden Fluides wurde die Luftdichte bei 30 °C [1] eingesetzt ($\rho = 1,164\,\frac{kg}{m^3}$). Durch den Luftwiderstand kann durch Formel 2.8 das Windwiderstandsmoment (M_{Wind}) bestimmt werden. Das Gesamtmoment setzt sich aus dem Reibmoment (M_{Reib}) und dem Windwiderstandsmoment zusammen. Das Reibmoment ergibt sich aus der Reibungskraft F_R und den Radius r des Reifens. Mittels Formel 2.9 kann die mechanische Leistung P_{mech} anhand des Momentes und der Winkelgeschwindigkeit ω bestimmt werden. Die mechanische Leistung ergibt sich aus der elektrischen Leistung P_{el} bei einem angenommenen Antriebswirkungsgrad von 85 % . Dieser Wirkungsgrad fasst dabei die Verluste im Inverter, der Maschine und im Getriebe zusammen. Aus den Formeln 2.4, 2.5, 2.6, 2.8, 2.9 ergibt sich folgende Gleichung:

$$P_{el} \cdot 0,85 = P_{ab} = P_{mech}$$

$$P_{mech} = M \cdot \omega = (M_{Reib} + M_{Wind}) \cdot \frac{v}{r}$$

$$P_{mech} = (F_R \cdot r + F_W \cdot r) \cdot \frac{v}{r}$$

$$P_{mech} = (m \cdot g \cdot \mu_R + c_W \cdot A \cdot \frac{1}{2} \cdot \rho \cdot v^2) \cdot v \tag{4.5}$$

Anhand dieser Gleichung kann nun für eine gegebene Fahrzeuggeschwindigkeit v die abgegebene Leistung P_{mech} des Motors sowie die aufgenommene elektrische Leistung P_{el} aus dem Akkumulator bestimmt werden. In Formel 4.5 wird das dynamische Moment nicht berücksichtigt, da von konstanten Geschwindigkeiten und von einer konstanten Bewegung in der Ebene ausgegangen wird. Benötigt wird allerdings eine Gleichung, in der für eine gegebene Geschwindigkeit die notwendige Leistung und dadurch der Strom errechnet werden kann. Da sich diese Gleichung nur sehr schwierig nach der Geschwindigkeit v umstellen lässt, wurde die Lösung numerisch bestimmt. Hierfür wurden Geschwindigkeitswerte von 1 bis 70 $\frac{m}{s}$ in Hundertstelschritten in die Gleichung 4.5 eingesetzt und dadurch die zugehörigen Leistungen ermittelt. Die Leistungen wurden mittels dem Wirkungsgrad und der zugehörigen Spannung in einen Strom zurückgerechnet. Die zugehörigen Spannungen ergeben sich aus dem Mittelwert der Spannung für die einzelnen Entladeströme. Das Ergebnis ist eine Tabelle (Tabelle 4.5), welche zu den einzelnen Entladeströmen die zugehörige Fahrzeuggeschwindigkeit beinhaltet. Teilt man nun die vorhandene Ladungsmenge im Akku durch den entnommenen Strom, so erhält man die Entladezeit für diesen Stromwert. Die Geschwindigkeit multipliziert

I in A	P in W	t in s	v in $\frac{\mathrm{m}}{\mathrm{s}}$	v in $\frac{\mathrm{km}}{\mathrm{h}}$	s in m	W in Wh
2,5	794,1	21600	5,52	19,872	119230	4764,6
5	1579,2	10800	10,17	33,612	109480	4737,5
10	3129,4	5400	16,2	58,32	87480	4694,1
25	7651,8	2160	25,945	93,402	56041	4591,1
40	12044	1350	31,805	114,5	42937	4516,5
50	14860	1080	34,77	125,17	37552	4457,9
60	17553	900	37,245	134,08	33521	4388,2
75	21372	720	40,295	145,06	29012	4265,4
100	27800	540	44,72	160,99	24149	4170
125	33860	432	48,25	173,7	20844	4063,2
150	39230	360	51,02	183,67	18367	3923

Tabelle 4.5: Strom-Weg-Tabelle

mit der Entladezeit ergibt den zurückgelegten Weg. Anhand der Tabelle ist nun zu sehen das beispielsweise bei einem konstanten Strom von 10 A eine Geschwindigkeit von 16,2 $\frac{\mathrm{m}}{\mathrm{s}}$ (58,32 $\frac{\mathrm{km}}{\mathrm{h}}$) erreicht wird und eine Strecke von 87,48 km in 5400 Sekunden (1,5 Stunden) zurückgelegt wird. Zudem ist zu sehen, dass die verrichtete Arbeit annähernd konstant ist. Die Abweichung entsteht durch die niedrigere Durchschnittsklemmspannungen bei höheren Entladeströmen. Das Script zur Berechnung des Restweges ist in Anhang C beigefügt.

Aus der Tabelle (Tabelle 4.5) werden nun die Strom- und Weg-Werte in eine „Lookup-Table“ geschrieben. Anhand des durchschnittlich entnommenen Stromes kann so der maximale Weg abgeschätzt werden. Dieser wird mit dem Ladezustand multipliziert, um den verfügbaren Restweg zu bestimmen. Abbildung 4.19 zeigt das Modell zur Restwegberechnung. Das Modell wird aller 100 Millisekunden im Steuergerät aufgerufen. Im Modell werden die letzten 100 Werte für den entnommene Strom durch das „Tapped Delay“ gespeichert und anschließend aufsummiert. Dieser Wert wird durch 100 geteilt, um den Mittelwert der letzten 10 Sekunden zu erhalten. Der Wert wird der Lookup-Table zugeführt, um die für diesen Strom maximalen Meter zu erhalten. Der entnommene Wert wird mit dem Ladezustand multipliziert, um die real verfügbaren Restmeter zu errechnen.

4.4 Dynamische Strombegrenzung

Auf einem Rennsportevent muss das Team und das Fahrzeug viele verschiedene Disziplinen absolvieren. Eine der wichtigsten ist hierbei das Endurance-Rennen, ein Ausdauerrennen über 22 Kilometer [7, Seite 164]. Für einen derartigen Fall ist es notwendig, den Akkustrom zu begrenzen, um die gewünschte Distanz zu bewältigen. Andernfalls könnte vor Beendigung der Strecke der Akku leer sein. Der Zusammenhang zwischen Stromentnahme und Reichweite ist unter anderem durch den quadratischen Einfluss

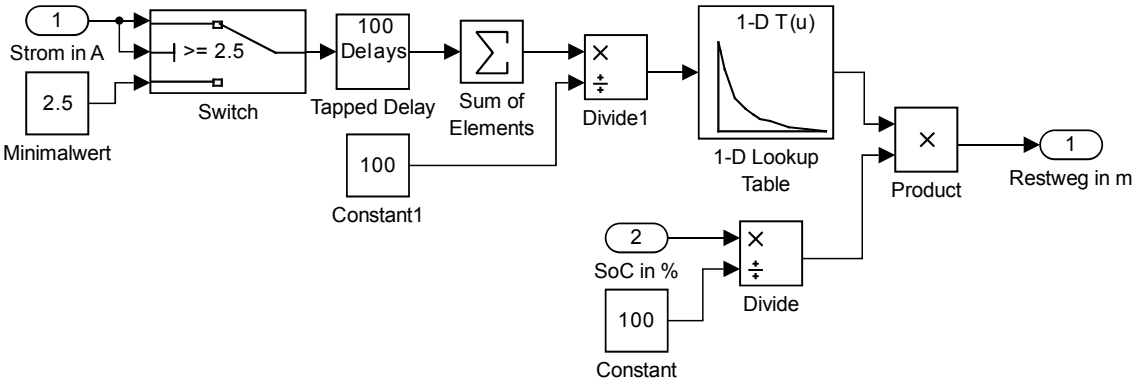


Abbildung 4.19: Modell zur Restwegberechnung

der Geschwindigkeit im Luftwiderstand (Formel 4.5) zu begründen. Für diese Begrenzung wurde die selbe Berechnung wie für den Restweg genutzt. Aus der Tabelle 4.5 wurde für einen gewünschten Weg (s in m) der zugehörige Maximalstrom (I in A) entnommen. Dies wird in MATLAB mittels einer Lookup-Tabelle ausgeführt. Es ist jedoch ineffizient, den Strom auf einen festen Wert zu begrenzen, da während einer Fahrt der entnommene Strom nicht konstant sein wird. Es wird Phasen geben, in denen weniger Strom entnommen wird, beispielsweise wenn vor einer Kurve das Fahrzeug abgebremst wird. Durch solche Fälle kann es dem Fahrer gestattet werden, kurzzeitig mehr Strom aus dem Akku zu entnehmen, als durch den Maximalstrom begrenzt. Hierfür wurde zunächst ein Modell (Abbildung 4.20) entworfen, welches den Mittelwert aus den vergangenen Stromwerten bildet, um abzuschätzen, wie sich der Fahrer verhält. In diesem Modell werden die letzten 1000 Stromwerte des Inverters[7] aufsummiert und durch die Anzahl geteilt. Dieses Modell wird alle 100 Millisekunden im Steuergerät aufgerufen, wodurch die letzten 100 Sekunden[8] erfasst werden. In einem weiteren Modell (Abbildung 4.21) wurde die Lookup-Tabelle eingefügt, welche anhand des gewünschten Weges den zulässigen Dauerstrom ermittelt. Der maximal einstellbare Weg beträgt 119,23 km, eine Vorgabe über diesem Wert wird in den Maxiamlwert umgewandelt. Dieses Modell wird alle 10 Sekunden aufgerufen, wodurch einerseits Rechenleistung eingespart wird und andererseits der Fahrer einen annähernd konstanten Maximalstrom fahren kann. Andernfalls würde nach einer kurzen Stromspitze der dynamische Maximalstrom zu schnell neu berechnet werden. In diesem Modell wird nun das Verhältnis zwischen dem maximalem Strom und Durchschnittsstrom gebildet. Dieses Verhältnis wird anschließend mit dem ermittelten Maximalstrom multipliziert, um den dynamischen Maximalstrom zu erhalten. Im Falle eines sparsamen Fahrers wird das Verhältnis kleiner als 1 sein, so dass der dynamische maximale Strom größer ist als der ermittelte Maximalstrom. Dies erlaubt dem Fahrer temporäre Überschreitungen des Maximalstromes. Im Falle eines Fahrers, welcher immer Maximalstrom aus dem Akku entnimmt („Bleifuss“) wäre das Verhältnis annähernd 1, so dass der dynamische Maximalstrom immer gleich dem ermittelten Maximalstrom ist. Ein solcher Fahrer könnte keine kurzen Stromspitzen

[7] Der Inverter sendet zyklisch aller 10 ms den aktuellen HV-Strom.

[8] Dieser Erfassungszeitraum wurde willkürlich festgelegt. Für den Rennbetrieb muss durch Tests am Fahrzeug ein geeigneter Erfassungszeitraum ermittelt werden.

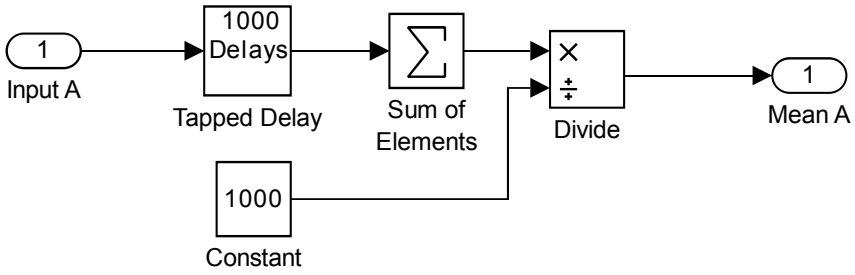


Abbildung 4.20: Modell Mittelwertbildung Strom

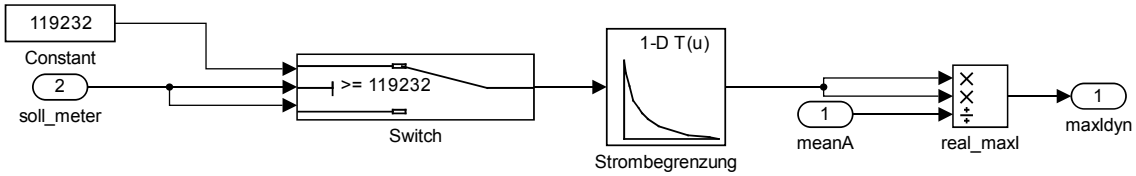


Abbildung 4.21: Modell dynamischer Stromberechnung

aus dem Akku entnehmen. Der Rekuperationsstrom wurde hierbei nicht berücksichtigt, da dieser nicht planbar ist.

Um das Modell zu testen wurde eine Testsimulation entworfen, welche als Simulationszeitbasis 100 Millisekunden nutzt, um die Mittelwertsimulation zeitgerecht aufzurufen. Die dynamische Grenzwertbestimmung wurde in einem getriggertesm Untermodell eingebaut, um die Aufrufzeitbasis auf 10 Sekunden einzustellen. Als Entladestrom wurde eine Sinusfunktion (Amplitude: 10 A, Offset: 15 A, Frequenz: 0,01 $\frac{\mathrm{rad}}{\mathrm{s}}$ = 0,0016 Hz) genutzt, welche mit einem Zufallswert addiert wurde. Als Vorgabewert für die zu fahrenden Distanz wurden 70 km eingegeben. Abbildung 4.22 zeigt den Eingangsstrom (blau). Die rote Kurve zeigt den berechneten Maximalstrom für die verlangten Fahrmeter. Grün dargestellt ist der Mittelwert der letzten 1000 Werte des Eingangsstromes. Die schwarze Kurve zeigt den momentan maximalen Strom, welcher zum Erreichen des gewünschten Fahrzieles bei dem aktuellen Mittelwert dem Akkumulator entnommen werden darf. In dieser Simulation wurden zudem anhand der umgesetzten Energie die zurückgelegten Fahrmeter berechnet. Es wurden rechnerisch 78 km zurückgelegt. Dies übersteigt das gewünschte Fahrziel von 70 km. Es ist zu sehen, dass der aktuelle Strom (blau) den Maximalstrom (rot) kurzzeitig überschreitet. Dies ist jedoch nur nach sparsamen Stromverbrauchsphasen möglich. Fährt ein Fahrer demnach sparsam und bleibt lange unter dem errechneten Maximalstrom, so kann er kurzzeitig Stromspitzen über dem Maximalstrom entnehmen, ohne dabei die gewünschte Fahrtdistanz zu gefährden.

Zur Einbindung in die Firmware für das Steuergerät wurden die Modelle zusammengefasst. Das zusammengefasste Modell wurde übersetzt und in INTECRIO eingebunden. Das Modell wurde in den 100 Millisekunden-Task eingetragen. Der Trigger für das Sromgrenzen-Modell wurde in einen 10 Sekunden-Task eingebunden. Das komplette Modell ist in Abbildung 4.23 dargestellt. Anhang E beinhaltet eine Beschreibung zur Programmierung des Steuergerätes.

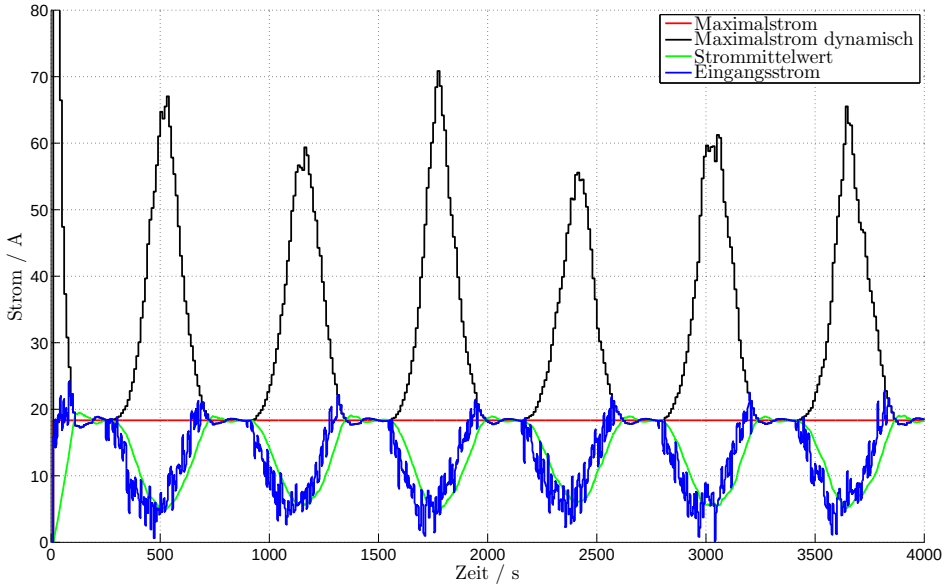


Abbildung 4.22: Ströme der dynamischen Strombegrenzung

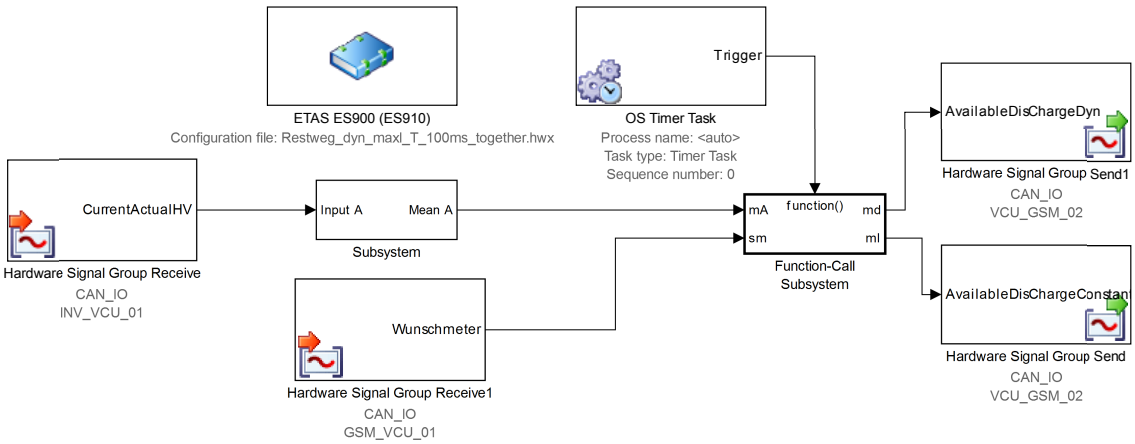


Abbildung 4.23: Stromgrenzen-Modell

5 Kühlung des Teststandes

5.1 Der Kühlkreislauf

Die Maschine und der Inverter erzeugen eine Verlustleistung, welche in Wärme umgesetzt wird. Diese Erwärmung könnte die Geräte zerstören. Um das zu verhindern wurde für den Teststand ein Kühlsystem entworfen, welches zudem als Vorentwicklung für das Kühlsystem im Rennfahrzeug angesehen werden kann. Zunächst wurden die maximal zulässigen Grenztemperaturen der einzelnen Komponenten betrachtet. Diese sind in Tabelle 5.1 verzeichnet. Die kleinste Grenztemperatur bestimmt die Maximaltemperatur des Systems und muss am stärksten gekühlt werden.

Komponente	Maximaltemperatur	Quelle: [18]
Inverter	65℃	Seite 101
DCDC-Wandler	65℃	Seite 101
Motor	100℃	Seite 125

Tabelle 5.1: Maximaltemperaturen

Zur Kühlung werden Kühlmittelpumpen des Typs CWA400 GEN III [15] der Firma Pierburg eingesetzt. Diese besitzt eine maximale Durchflussrate von 9 $\frac{\mathrm{m}^3}{\mathrm{h}}$ [15, Seite 1: Flow rate]. Dies entspricht 150 $\frac{\mathrm{l}}{\mathrm{min}}$. Die empfohlene Durchflussrate für Maschine und Inverter beträgt 8 $\frac{\mathrm{l}}{\mathrm{min}}$ [18, Seite 101]. Die Pumpen sind aus Sicherheitsgründen überdimensioniert. Die Ansteuerung der Pumpen kann mittels Puls-Weiten-Modulation (PWM) oder Local Interconnect Network (LIN) erfolgen. In beiden Fällen kann die Drehzahl der Pumpe vorgegeben werden.

Im Fahrzeug wird später eine Pumpe die Kühlflüssigkeit für ein Inverter und einen Motor pumpen. Am Teststand wurde das selbe Prinzip gewählt. Da nur ein Kühler vorhanden war, wurde ein aufgetrennter Kühlkreislauf mit zwei Pumpen und einem Kühler aufgebaut (Abbildung 5.1). Die Kühlflüssigkeit wird im Kühler durch einen Lüfter abgekühlt. Die Luft im Ausgleichsbehälter nimmt bei Erwärmung die Volumenänderung auf. Am Kreislauf befindet sich ein Ablassventil, damit für Umbauarbeiten die Kühlflüssigkeit abgelassen werden kann. Die blauen Verbindungslinien in Abbildung 5.1 symbolisieren kalte Kühlflüssigkeit, die roten warme. Als Kühlflüssigkeit wurde Wasser verwendet.

5.2 Zwischenlösung zum Testbetrieb

Für den Testbetrieb der Maschinen wurde eine Zwischenlösung zur Ansteuerung der Kühlmittelpumpen entwickelt. Hierfür wurde die PWM-Schnittstelle der Pumpen genutzt. Zur Ansteuerung wurde ein Entwicklungskit DVK90CAN1 [3] von Atmel genutzt. Dieses

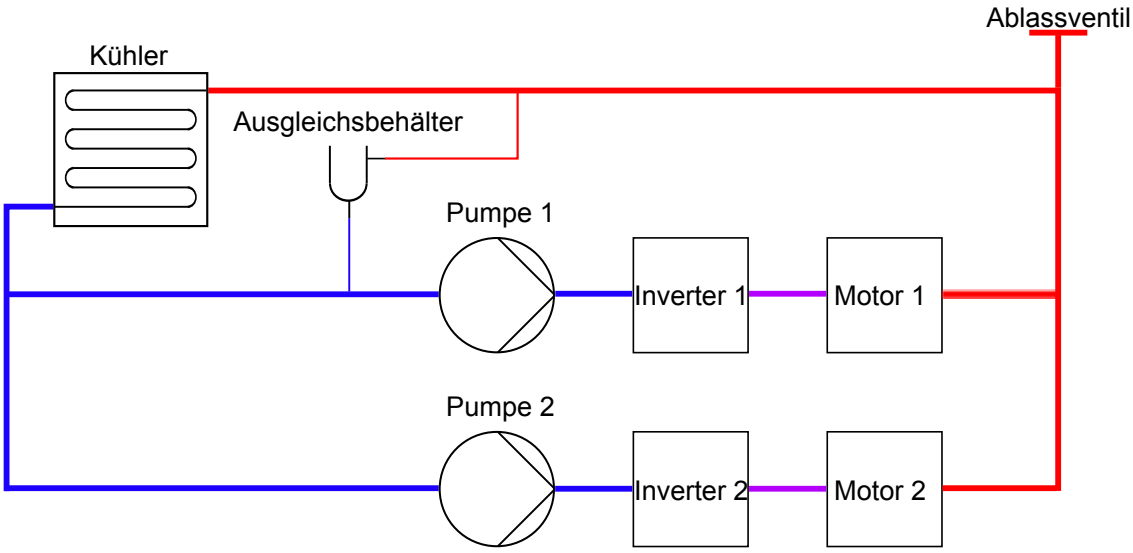


Abbildung 5.1: Kühlkreislauf

Entwicklungskit besteht aus einem Controller AT90CAN128 mit zugehöriger betriebsbereiter Peripherie. Da die Pumpe mit einer Betriebsspannung von 12 V arbeitet und der Controller mit 5 V, war es notwendig, einen Pegelwandler für die PWM-Kanäle einzusetzen. Diese Schaltung (Abbildung 5.2) arbeitet mit einem Transistor, der in Emitterschaltung betrieben wird. Liegt am Eingang ein 5 V-Pegel an, wird der Transistor durchgesteuert und der Ausgang wird auf 0 V gezogen. Liegt ein 0 V-Pegel am Eingang an, so wird der Ausgang auf 12 V gesetzt. Der Wandler arbeitet invertierend.

Zur Steuerung der Pumpen wurden die PWM-Kanäle des Controllers aktiviert. Als Frequenz wurde 100 Hz gewählt[9]. Zudem wurde die CAN-Schnittstelle des Controllers aktiviert. Diese wurde mit dem CAN Netzwerk von Inverter 1 verbunden. Empfängt der Controller nun die CAN-Nachricht INV_VCU_04, so wird aus dieser Nachricht die Temperatur des Inverters ausgelesen (Byte 3: „Inverter Temperatur“). Die Pumpendrehzahl wird nun prozentual auf den Wert eingestellt, welcher ausgelesen wurde. Wird beispielsweise eine Temperatur von 35 $^{\circ}\mathrm{C}$ gelesen, so wird die Pumpendrehzahl auf 35 % eingestellt[10]. Dadurch steigt bei steigender Temperatur die Pumpendrehzahl. Durch die Überdimensionierung der Pumpen kann so sichergestellt werden, dass die Maschinen im Testbetrieb nicht überhitzen. Durch die Momentbegrenzung am Teststand ist ohnehin nur mit einer moderaten Erwärmung zu rechnen. Die Ansteuerung mittels PWM ist allerdings nicht für die Anwendung im Fahrzeug geeignet, da die Fehlertoleranz für PWM-Signale in der Pumpe bei $\pm$ 2 % liegt ([15, Seite 3]). Dadurch ist keine präzise Ansteuerung mit PWM möglich.

[9] Die PWM-Frequenz für die Pumpen kann frei zwischen 50 Hz und 1000 Hz gewählt werden, siehe [15, Seite 3].

[10] Die Umrechnung von Tastverhaltnis zu Drehzahl befindet sich in [15, Seite 3].

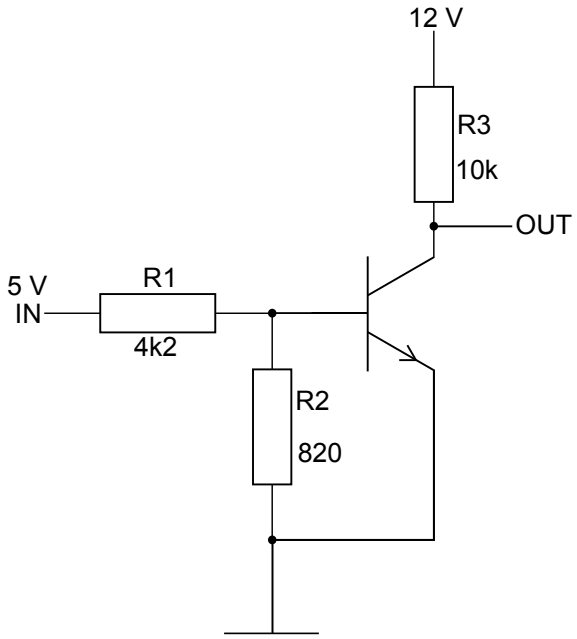


Abbildung 5.2: Pegelwandler

5.3 Ansteuerung der LIN-Schnittstelle

Die Kühlpumpen besitzen eine LIN-Schnittstelle. Zur Nutzung der LIN-Schnittstelle wurde ein LIN-Description-File (ldf-File) verfasst. Dieses ldf-File beinhaltet die Informationen über die Teilnehmerknoten, die Datenübertragungsrate sowie den Aufbau der Nachrichten. Der Aufbau der Nachrichten wurde dem Datenblatt entnommen [15, Seite 5 f.]. Die LIN-Schnittstelle der Kühlmittelpumpe kann nur mit einer Baudrate von 19200 Baud angesprochen werden [15, Seite 4]. Da das Datenblatt nur wenige Information darüber enthält, was die einzelnen Daten bedeuten, wurde dies durch Tests ermittelt. Hierfür wurde die LIN-Schnittstelle des Steuergerätes in Betrieb genommen. Als Datenbasis wurde das ldf-File genutzt. Die Drehzahl der Pumpe kann mittels dem Wert „TARVL_RPM_WAP_LIN“ vorgegeben werden. Anhand der empfangenen Daten und Messungen konnten einige Werte identifiziert werden (siehe Tabelle 5.2).

5.4 Energieeinsparung

Die Ansteuerung soll möglichst energiesparend arbeiten. Die Pumpensteuerung sollte möglichst vorausschauend arbeiten, um die Energiespeicher zu entlasten. Falls das Fahrzeug beschleunigt, wird mehr Wärme über den Kühler abgegeben, da die Luft schneller durch diesen strömt. Hierdurch wird die Kühlwassertemperatur am Ausgang des Kühlers fallen. Die Kühlpumpe könnte in solch einem Fall gedrosselt werden, um Energie einzusparen. Der Beschleunigungsvorgang kann entweder über die Geschwindigkeit oder über den aktuellen Strom erfasst werden. Zur Erfassung wurde die Ablei-

Kennzeichnung im Datenblatt	ermittelter Wert	Bemerkung
I_WAP_LIN	aktueller Strom der Pumpe	Berechnungsformel: I = I_WAP_LIN·0,22 A
TEMP_WAP_LIN	Temperatur in $^{\circ}\mathrm{F}$	Wassertemperatur oder Pumpentemperatur
U_WAP_LIN	Spannung der Pumpe	Berechnungsformel: U = U_WAP_LIN·0,1 V
AVL_VL_RPM_WAP_LIN	aktuelle Drehzahl der Pumpe	Wert entspricht Vorgabewert in TARVL_RPM_WAP_LIN
ST_TARVL_RPM_WAP_LIN		

Tabelle 5.2: LIN-Datenbedeutung

tung des Stromes genutzt. Übersteigt diese den Wert 1 $\frac{\mathrm{A}}{\mathrm{s}}$ [11], so wird die Drehzahl der Pumpe auf den Minimalwert gesetzt. Die Pumpe wird in diesem Fall nicht abgeschaltet, damit das Kühlwasser weiterhin durch den Kreislauf zirkuliert. Das Modell ist in Abbildung 5.3 zu sehen. Der damit eingesparte Strom kann mittels der Leistung abgeschätzt werden. Die maximal aufgenommene Leistung der Pumpe ergibt sich aus der Betriebsspannung und dem maximalen Strom [15, Seite 1, Current consumption in nominal duty point]:

$$P_{\max} = U \cdot I = 12\,\mathrm{V} \cdot 35\,\mathrm{A} = 420\,\mathrm{W} \tag{5.1}$$

Der DCDC-Wandler stellt die 12-Volt für das Bordnetz und die Pumpen bereit. Hierfür wandelt dieser die Spannung des Hochvoltakkumulators in 12 V um. Der DCDC-Wandler muss demnach die Leistung für die Pumpe bereitstellen. Die Maximalleistung der Pumpe entspricht demnach, bei einem angenommenen Wirkungsgrad des DCDC-Wandlers von 1 und einer konstanten Spannung von 380 V, einem Strom auf der Hochvoltseite von:

$$I = \frac{P}{U} = \frac{420\,\mathrm{W}}{380\,\mathrm{V}} = 1,1\,\mathrm{A} \tag{5.2}$$

Der maximal eingesparte Strom bei einem Beschleunigungsvorgang beträgt demnach maximal 1,1 A. Die reale Einsparung fällt jedoch geringer aus, da die Pumpe immer mit einer Mindestdrehzahl läuft. Nach Abschluss des Beschleunigungsvorgangs wird die Pumpe wieder aktiviert. Da mit einer besseren Kühlung durch den Kühler zu rechnen ist, wird die Drehzahl wahrscheinlich niedriger sein als zum Zeitpunkt der Deaktivierung.

Das Modell wurde mit einem Strom mit dreiecksförmigen Entladeverlauf (Abbildung 5.4a) und einer Pumpendrehzahl mit sinusförmigen Verlauf (Abbildung 5.4b) getestet. Der vorgegebene Strom weist hierbei zwei verschiedene Anstiege auf, um das Verhalten bei hohen und geringen Anstiegen zu testen. In Abbildung 5.4c ist nun die resultierende

[11] Dieser Wert wurde willkürlich gewählt, da keine realen Messwerte vorliegen, die eine realistische Dimensionierung ermöglicht hätten.

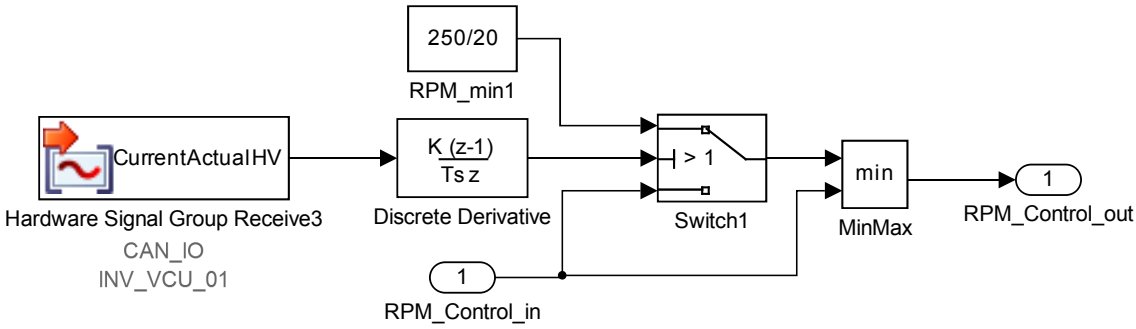


Abbildung 5.3: Stromanstiegsmodell

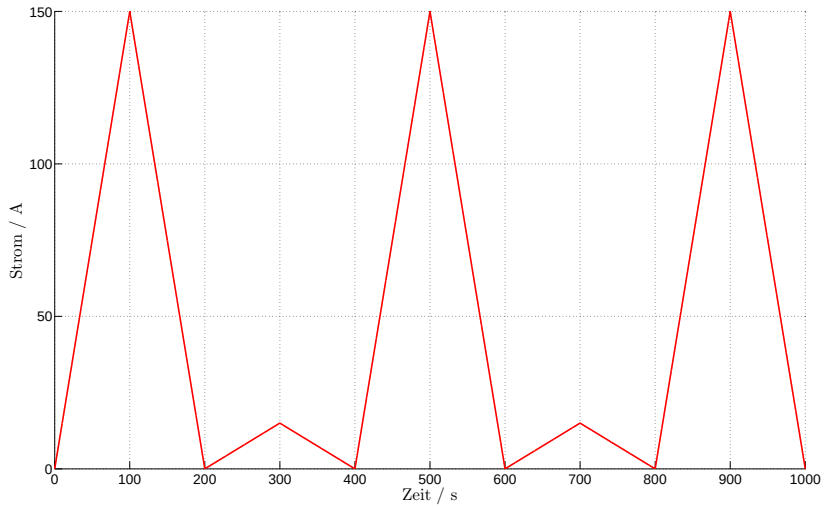
Vorgabe für die Drehzahl der Pumpe zu sehen. Es zeigt sich, dass die sinusförmige Vorgabe zu den Zeiten, in welchen der Anstieg des Stromes hoch ist, auf den Minimalwert fällt. In diesen Phasen wird Energie aus dem Akku eingespart.

5.5 Schutz vor Übertemperatur

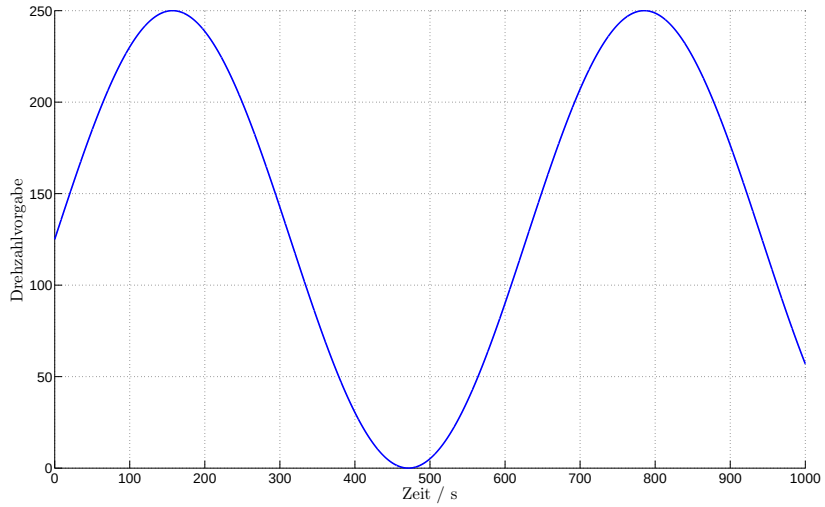
Zur Überwachung aller Grenztemperaturen (Tabelle 5.1) wurde ein Modell entworfen. Dieses Modell (Abbildung 5.5) liest die aktuellen Temperaturwerte aus CAN-Nachrichten heraus und vergleicht diese mit den zugehörigen Maximalwerten. Sollte eine Komponente eine Temperatur erreichen, welche 5 °C unter der Grenztemperatur liegt, so wird die Drehzahl der Pumpe auf 100 % gesetzt. Die 5 °C-Reserve wurde als Schutzfunktion eingebaut. Befindet sich die aktuelle Temperatur unterhalb dieser Grenze wird der eingegebene Wert (Input RPM_Control_in) als Output weitergegeben (Output Control_RPM_out). Falls eine Komponente die Grenztemperatur erreicht, so wird ein Statusbit (Abbildung 5.5, Output „State_INV“) gesetzt, welches von der Fahrzeugsteuerung ausgewertet werden soll und die Maschine abschalten soll. Zudem wird die Pumpendrehzahl auf den Maximalwert eingestellt. Dieses Modell wurde als Sub-System zwischen dem Ausgang des Reglers und der Drehzahl der Pumpe eingesetzt.

5.6 Regelung der Pumpe

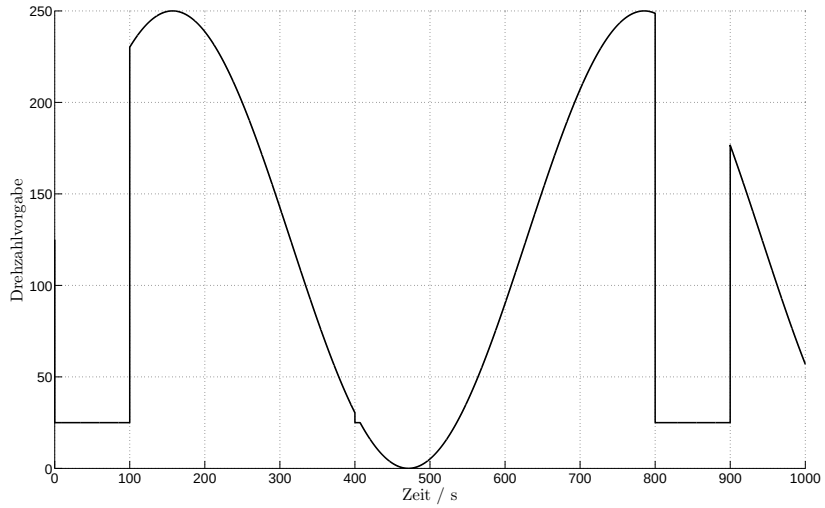
Zur Energieeinsparung soll die Drehzahl der Pumpe so eingestellt werden, dass möglichst wenig Energie aus dem Akku entnommen wird, jedoch alle Komponenten unter ihrer jeweiligen Grenztemperatur liegen. Hierfür bietet sich eine Temperaturregelung an. Der Regelkreis ist in Abbildung 5.6 dargestellt. Die Führungsgröße ist die Solltemperatur. Als Stellgröße ergibt sich die Drehzahl der Pumpe. Die Regelgröße ist die Temperatur der Komponenten. Die Rückführung ergibt sich aus der Messung der Temperatur. Die Regelstrecke ist somit das thermische Verhalten der Komponenten in Abhängigkeit der Pumpendrehzahl. Die Erwärmung der Komponenten durch variable Momentvorgaben ist hierbei als Störgröße anzusehen.



(a) Vorgabe Inverterstrom



(b) Vorgabe Pumpendrehzahl



(c) resultierende Pumpendrehzahl

Abbildung 5.4: Test Stromprobe

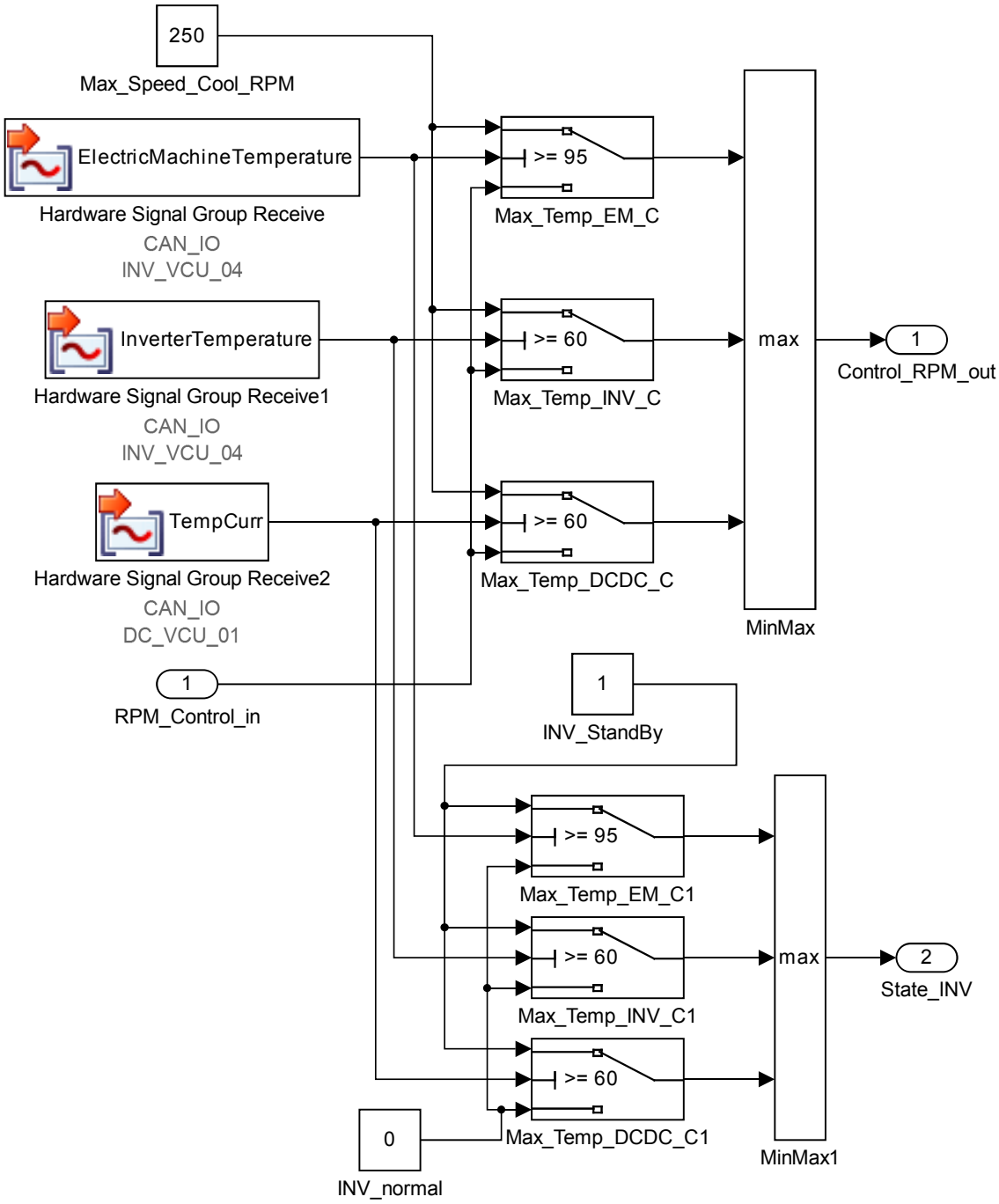


Abbildung 5.5: Maximalwertprobe

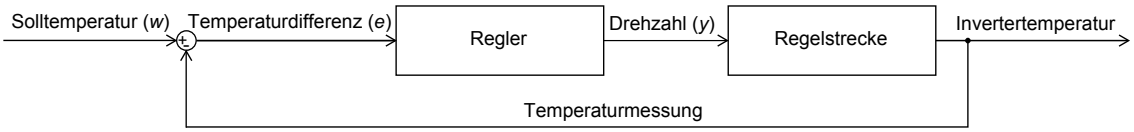


Abbildung 5.6: Regelkreis

5.6.1 Regelung mit PID-Regler

Im Testbetrieb ergab sich das der Testaufbau mit zwei Pumpen zur Reglerparametrierung ungeeignet ist, da durch die Überdimensionierung der Pumpen bei minimaler Drehzahl bereits eine ausreichende Kühlleistung einstellte. Daher wurde der Kühlkreislauf umgebaut (Abbildung 5.7). Der neue Aufbau besteht aus einer Kühlpumpe, welche die Inverter und die Motoren kühlt. Die Parallelstruktur der Inverter und Maschinen wurde beibehalten, damit beide Inverter am Einlass gekühltes Kühlwasser erhalten. Der Ausgleichsbehälter wurde in dieser Struktur in Reihe zum Kühler eingebaut, da im ersten Aufbau beobachtet werden konnte, dass das Kühlwasser zum Teil durch den Kühler und zum Teil durch den Ausgleichsbehälter floss. Dadurch war ein schlechtereres Kühlverhalten zu beobachten. Durch den Aufbau in Serie fließt nun das gesamte Kühlwasser durch den Kühler. Durch den neuen Aufbau konnte nun eine Abhängigkeit zwischen Pumpendrehzahl und Invertertemperatur ermittelt werden. Um die Drehzahl zu regeln sind nun die Streckenparameter notwendig. Um diese zu ermitteln wurde ein Sprungsignal auf die Solldrehzahl der Pumpe gegeben und der Temperaturverlauf des Inverters aufgezeichnet. Für diesen Versuch wurde eine konstante Drehzahl des Kühllüfters sowie ein konstantes Motormoment von 40 Nm gewählt. Ein höheres Moment könnte ohne eine Kühlpumpenregelung zu thermischen Beschädigungen der Komponenten führen. Vor dem Sprung wurde das System solange betrieben, bis keine Änderungen der Temperatuen mehr festzustellen waren. Als Stellgrößensprung wurde ein Sprung von 20 % auf 80 % der Drehzahl gewählt. Der Anfangswert von 20 % wurde gewählt, da bei dieser Drehzahleinstellung die minimale Durchflussrate festgestellt wurde. Bei einer kleineren Einstellung fließt nahezu kein Wasser durch das System. Abbildung 5.8 zeigt die aufgezeichnete Sprungantwort. Die blaue Kurve zeigt den Sollwert und die grüne Kurve die Temperatur des Inverters. Es ist zu erkennen, dass erst etwa 500 Sekunden nach dem Sprung der Einschwingvorgang abgeschlossen ist. Die rote Linie zeigt die eingeschwungene Temperatur. Es ist zu erkennen, dass es sich um ein System mit Ausgleich handelt, da sich die Temperatur auf einen festen Wert einschwingt. Zudem ist zu erkennen, dass eine Totzeit existiert. Diese entsteht größtenteils durch die Wassermenge im System, da das Wasser aus dem Kühler erst den Sensor im Inverter erreichen muss. In Abbildung 5.8b wurde die Ansstiegstangente (schwarz) angelegt. Die Verzugszeit (T_{u}) und die Ausgleichszeit (T_{g}) wurden für die Analyse der Regelstrecke aus der Messung entnommen. Die vollständige Messung ist in Anhang D abgebildet.

Die Streckenverstärkung K_{S}ergibt sich aus der Istwertänderung (Δx) geteilt durch die Stellgrößenänderung (Δy) (Gleichung 5.3).

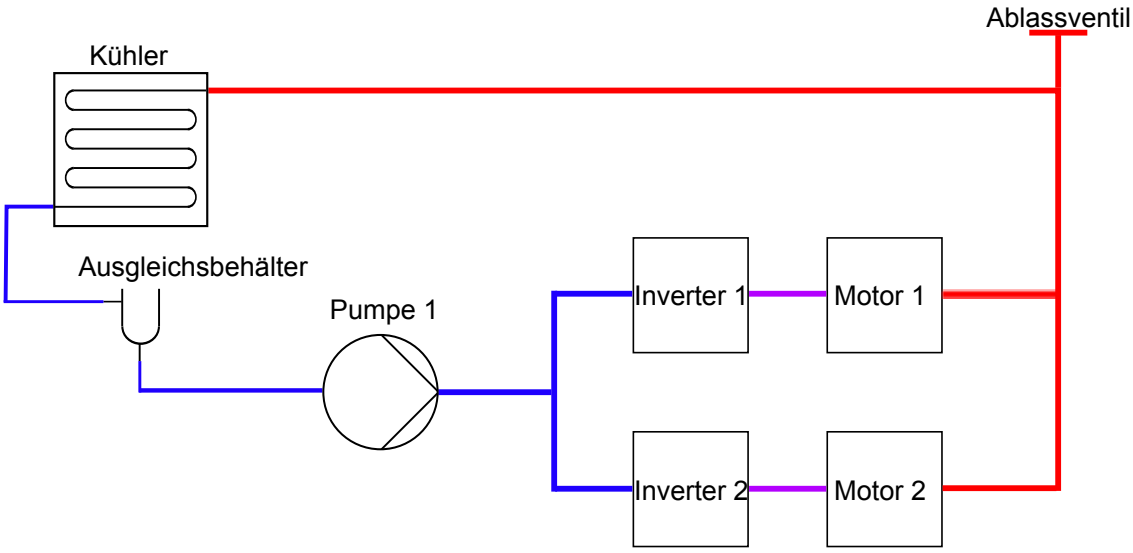


Abbildung 5.7: neuer Kühlkreislauf

$$K_S = \frac{\triangle x}{\triangle y} = \frac{(46\,°C - 59\,°C)}{(80\,\% - 20\,\%)} = \frac{-13\,°C}{60\,\%} = -0,2167 \tag{5.3}$$

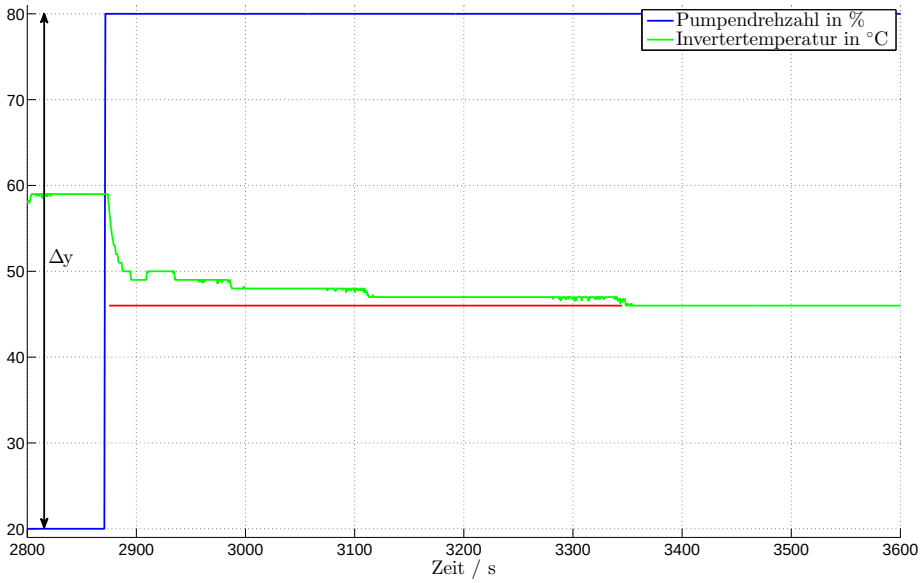
Das Streckenverhalten wurde anhand der Sprungantwort (Abbildung 5.8b) approximiert [8, Seite 206]:

$$G_S(\mathbf{s}) = \frac{K_S}{\mathbf{s} \cdot T_g + 1} \cdot e^{-\mathbf{s} \cdot T_u} = \frac{-0,2167}{\mathbf{s} \cdot 10,2\,s + 1} \cdot e^{-\mathbf{s} \cdot 2,6\,s} \tag{5.4}$$

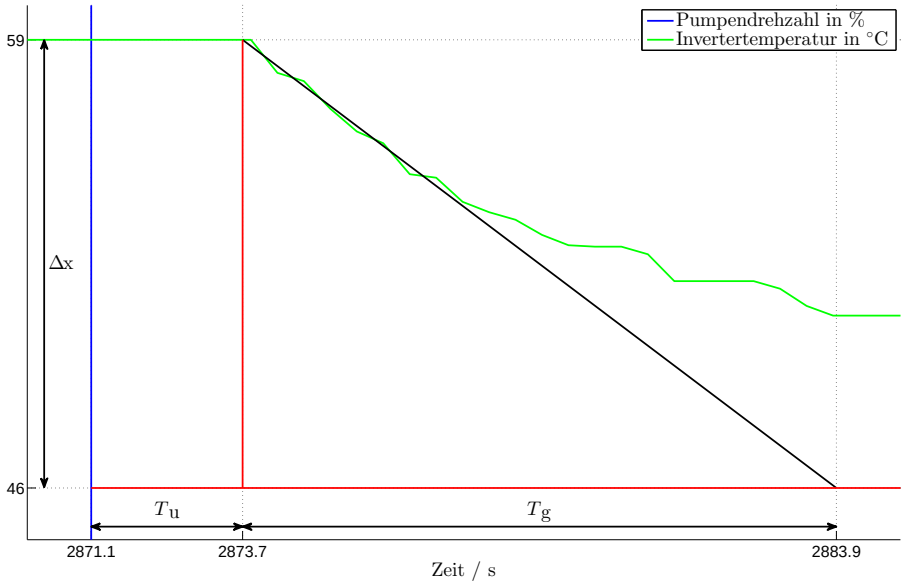
Die Totzeit (T_u) kann durch ein PTn-Glied erster Ordnung angenähert werden [9, Seite 333]. Dadurch ergibt sich folgendes Streckenverhalten:

$$G_S(\mathbf{s}) = \frac{-0,2167}{\mathbf{s} \cdot 10,2\,s + 1} \cdot \frac{1}{\mathbf{s} \cdot 2,6\,s + 1} \tag{5.5}$$

Für die Parametrierung wurden zunächst die Einstellregeln nach Chien/ Hrones/ Reswick [19, Seite 226] gewählt. Diese Parametrierung wurde gewählt, da hierbei bei geringem Aufwand mit guten Ergebnissen erwartet wurden. Die Reglerparameter berechnen sich wie folgt (Gleichungen 5.6):



(a) Vollständige Sprungantwort



(b) Ausschnitt Sprungantwort

Abbildung 5.8: Sprungantwort

$$K_\mathrm{p} = \frac{0,6}{K_\mathrm{S}} \cdot \frac{T_\mathrm{g}}{T_\mathrm{u}} = \frac{0,6}{-0,2167} \cdot \frac{10,2\,\mathrm{s}}{2,6\,\mathrm{s}} = -10,8622$$

$$T_\mathrm{n} = T_\mathrm{g} = 10,2\,\mathrm{s}$$

$$K_\mathrm{i} = \frac{K_\mathrm{p}}{T_\mathrm{n}} = \frac{-10,8622}{10,2\,\mathrm{s}} = -1,0649\,\frac{1}{\mathrm{s}} \tag{5.6}$$

$$T_\mathrm{v} = 0,5 \cdot T_\mathrm{u} = 0,5 \cdot 2,6\,\mathrm{s} = 1,3\,\mathrm{s}$$

$$K_\mathrm{d} = K_\mathrm{p} \cdot T_\mathrm{v} = -10,8622 \cdot 1,3\,\mathrm{s} = -14,1209\,\mathrm{s}$$

Zudem wurde die Parametrierung nach Ziegler/ Nichols [19, Seite 227] angewandt. Hierfür ergaben sich folgende Parameter (Gleichungen 5.7):

$$K_\mathrm{p} = \frac{1,2}{K_\mathrm{S}} \cdot \frac{T_\mathrm{g}}{T_\mathrm{u}} = \frac{1,2}{-0,2167} \cdot \frac{10,2\,\mathrm{s}}{2,6\,\mathrm{s}} = -21,7245$$

$$T_\mathrm{n} = 2 \cdot T_\mathrm{u} = 2 \cdot 2,6\,\mathrm{s} = 5,2\,\mathrm{s}$$

$$K_\mathrm{i} = \frac{K_\mathrm{p}}{T_\mathrm{n}} = \frac{-21,7245}{10,2\,\mathrm{s}} = -4,1778\,\frac{1}{\mathrm{s}} \tag{5.7}$$

$$T_\mathrm{v} = 0,5 \cdot T_\mathrm{u} = 0,5 \cdot 2,6\,\mathrm{s} = 1,3\,\mathrm{s}$$

$$K_\mathrm{d} = K_\mathrm{p} \cdot T_\mathrm{v} = -21,7245 \cdot 1,3\,\mathrm{s} = -28,2418\,\mathrm{s}$$

Um die Stabilität des Regelkreises zu bewerten wurde das Nyquist-Kriterium angewandt [9, Seite 434]. Hierfür muss zunächst die Anzahl der instabilen Pole im offenen Regelkreis ermittelt werden. Der offene Regelkreis ergibt sich aus der Multiplikation der Regelstrecke (Gleichung 5.5) mit dem gewählten Regler (Gleichung 2.11 mit den Parametern aus Gleichungen 5.6 oder 5.7). Die Übertragungsfunktion ergibt sich zu:

$$G(\mathbf{s}) = G_\mathrm{R}(\mathbf{s}) \cdot G_\mathrm{S}(\mathbf{s}) \tag{5.8}$$

Im Nenner befinden sich keine negativen Polstellen. Daher sind keine instabilen Pole vorhanden. Die Nyquistkurve darf, damit das System als stabil betrachtet werden

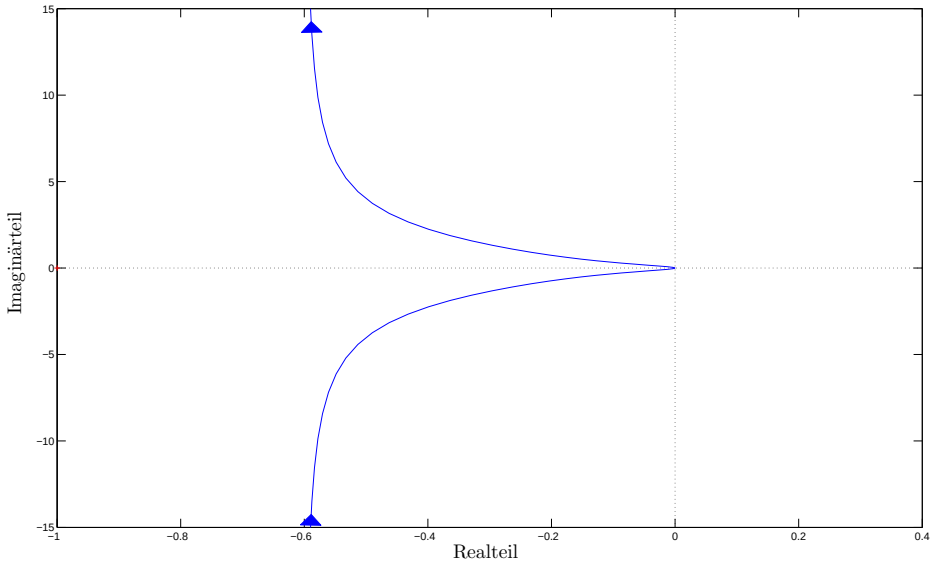


Abbildung 5.9: Nyquistdiagramm für Parametrierung nach Chien / Hrones / Reswick

kann, den Nyquistpunkt bei $-1 + j0$, nicht umschlingen. Abbildung 5.9 zeigt das Nyquistdiagramm für die Parametrierung nach Chien/ Hrones/ Reswick, Abbildung 5.10 nach Ziegler/ Nichols. In den Nyquistdiagrammen dargestellt sind die Ortskurve der Ausgangsgröße der offenen Regelkreise.

Die Nyquist-Diagramme zeigen kein Umschlingen des Punktes $-1 + j0$. Daher kann bei beiden Reglerparametrierungen von stabilen System ausgegangen werden. Die ermittelten Parameter wurden in einen PID-Regler übertragen. Als Sollwert wurden 35 $^{\circ}\mathrm{C}$ gewählt, da durch die Momentbeschränkung am Teststand die Maschinen keine hohen Temperaturen erreichen. Der Maximalwert aus den CAN-Signalen Temperatur des Inverters („InverterTemperature“ in Abbildung 5.11) sowie des DCDC-Wandlers („TempCurr“) wird bestimmt. Dieser wird als Istwert verwendet. Die Regelabweichung wird nun dem Regler zugeführt, falls diese positiv ist. Eine negative Regelabweichung würde bedeuten, dass der Istwert kleiner ist als der Sollwert. Zur Stellgröße wird der Minimalwert von 50 addiert (entspricht 20 %), um in jedem Fall eine minimale Durchflussmenge zu garantieren. Die Begrenzung („Saturation“) sorgt dafür, das die Stellgröße die Grenzwerte für die Drehzahlvorgabe einhält. Die Stellgröße wird dem LIN-Signal für die Drehzahl zugeführt. Abbildung 5.11 zeigt das Modell, welches auf das Steuergerät übertragen wurde.

Die beiden Reglerparametersätze wurden getestet. Die Lüfterdrehzahl des Kühlers war für alle Tests konstant. Es wurde eine Drehmomentabfolge für die Maschinen vorgegeben (Abbildung 5.12), welche zyklisch wiederholt wurde. Vor jedem Test war der gesamte Teststand für mehrere Stunden nicht aktiv, um gleiche Startbedingungen zu gewährleisten. Jeder Test lief solange, bis sich das System eingeschwungen hatte, mindes-

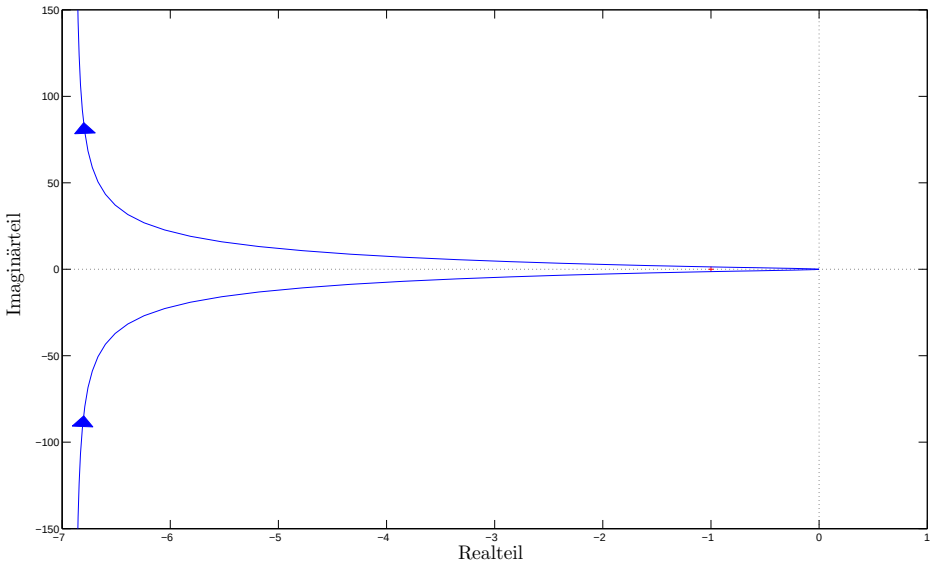


Abbildung 5.10: Nyquistdiagramm für Parametrierung nach Ziegler / Nichols

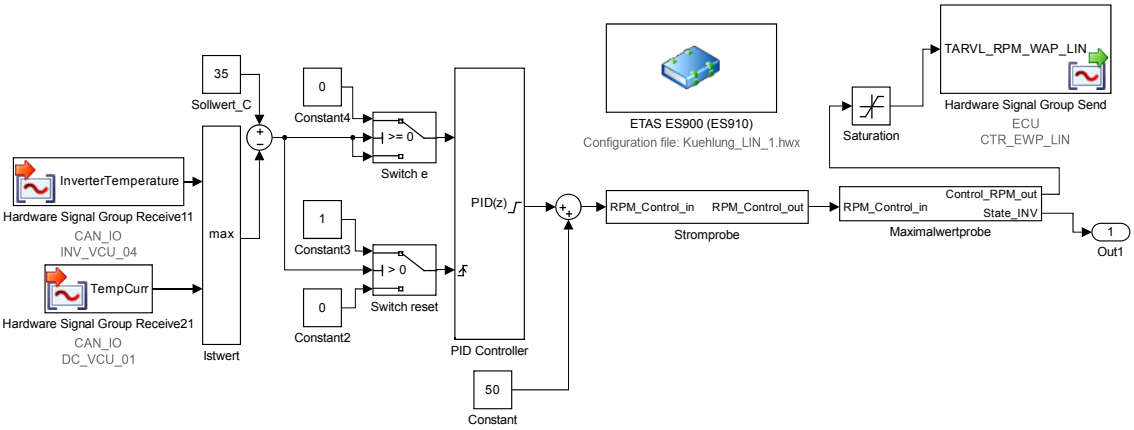


Abbildung 5.11: komplettes Reglermodell

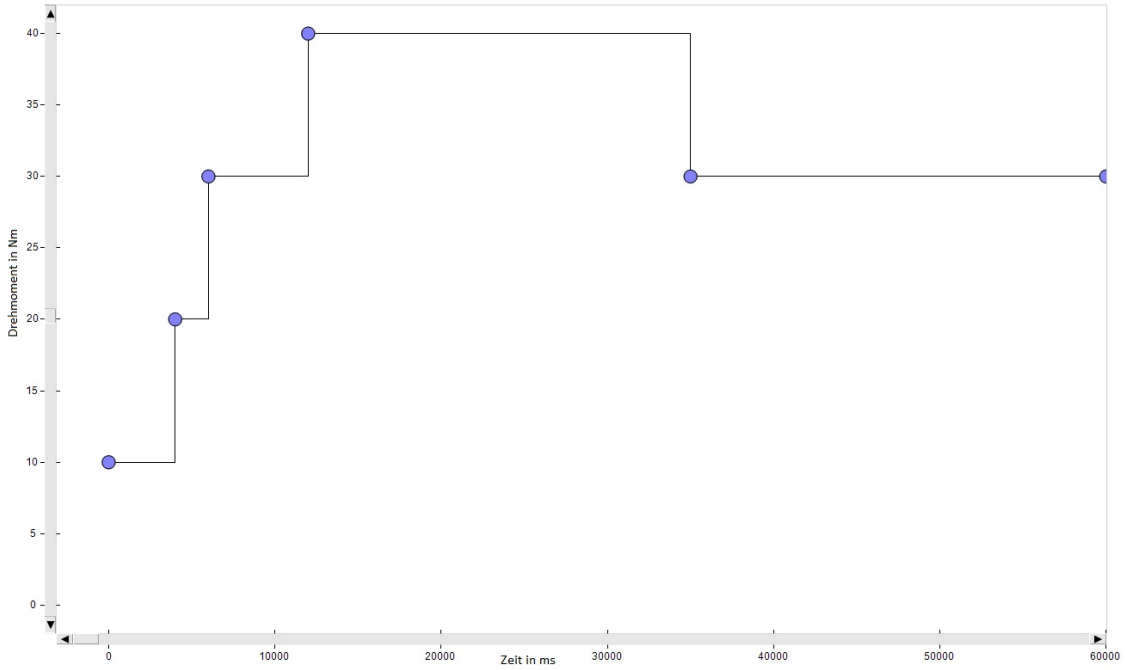
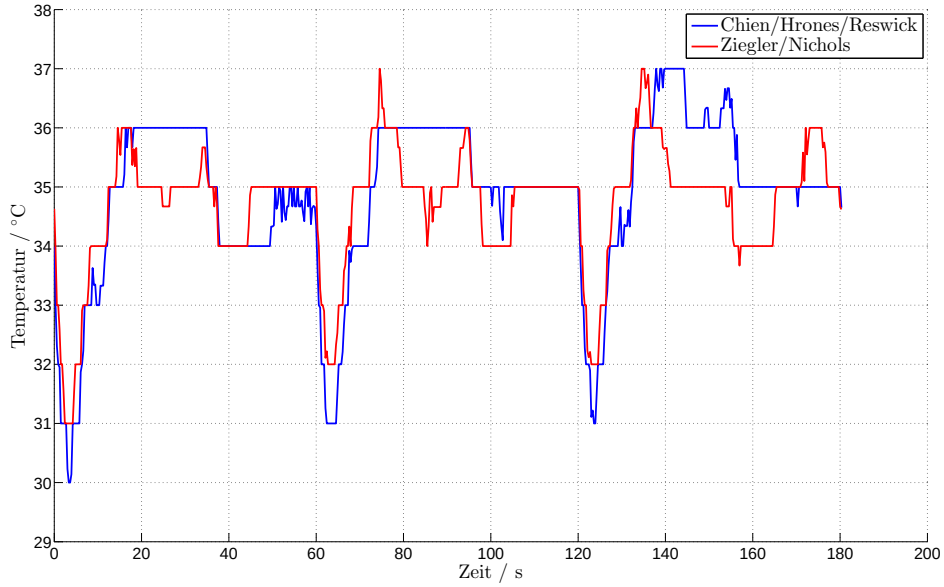


Abbildung 5.12: Momentvorgabe

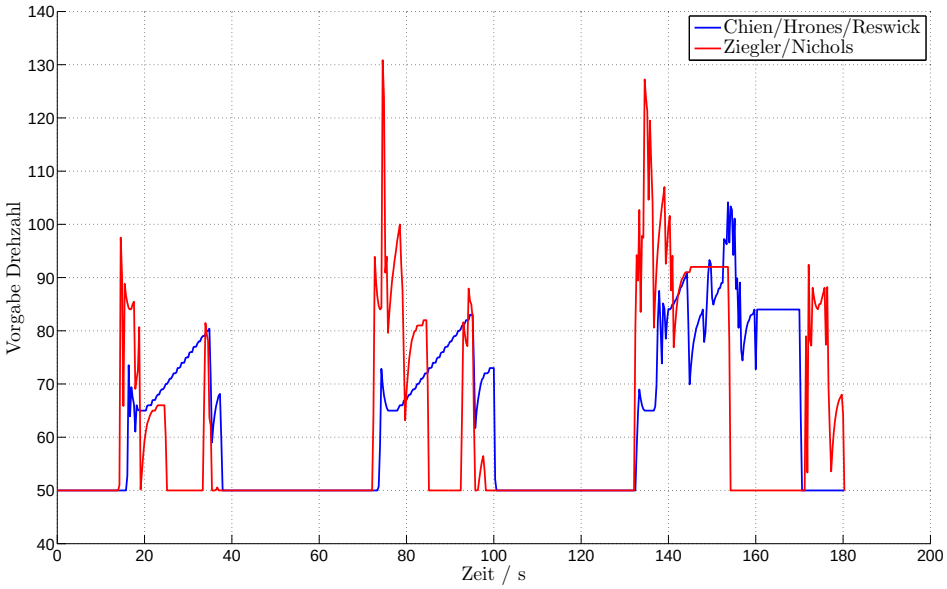
tens jedoch 10 Minuten. Ausgewertet wurden immer die letzten 3 Minuten. Abbildung 5.13a zeigt die Invertertemperaturen von Inverter 1 und Abbildung 5.13b die Drehzahl der Pumpe. Die blaue Kurve zeigt das Reglerverhalten mit den Parametern nach Chien/ Hrones/ Reswick, die rote Kurve nach Ziegler/ Nichols. Es ist zu erkennen, dass sich die Temperaturverläufe annähernd gleich verhalten. Die Solltemperatur von 35 °C wird nur selten überschritten. Es ist zudem zu erkennen, dass die Temperatur der Inverter sehr stark von dem Drehmoment abhängt. Anhand der Drehzahlen ist zu erkennen, dass bei Sollwertunterschreitung die Pumpendrehzahl auf den Minimalwert gesetzt wird. Zudem sind Sprünge in der Pumpendrehzahl zu erkennen. Diese werden durch den Stromanstiegsalgorithmus hervorgerufen (Kapitel 5.4). Für diesen Abschnitt wurde zudem die durchschnittliche Leistungsaufnahme der Pumpe anhand der Drehzahl ermittelt. Für den Parametersatz Chien/ Hrones/ Reswick ergab sich eine Drehzahl von durchschnittlich 24,99 % und für Ziegler/ Nichols 24,4 %. Da die Temperaturverläufe annähernd gleich sind und die durchschnittliche Drehzahl für Ziegler/ Nichols niedriger war, kann dieser als besserer Parametersatz angesehen werden.

5.6.2 Regelung mit F_{Rt}-Wurzelrekursion

Als Regelalgorithmus kann auch die F_{Rt}-Wurzelrekursion angewandt werden. Die Vorliegende Regelstrecke ist eine Regelstrecke mit Ausgleich, da die Regelgröße nach einem Sprung der Stellgröße sich einem festen Wert annähern wird und nicht unendlich steigt oder fällt. Gleichung 2.13 zeigt den Regelalgorithmus für diesen Fall. Für die Gewichtung wird ein Wert von 0,001 empfohlen. Die Streckenverstärkung kann auf den Wert 1 eingestellt werden, wurde hier allerdings auf den ermittelten Wert gesetzt (Gleichung



(a) Inverter-Temperatur



(b) Pumpendrehzahl

Abbildung 5.13: Auswertung PID-Regler

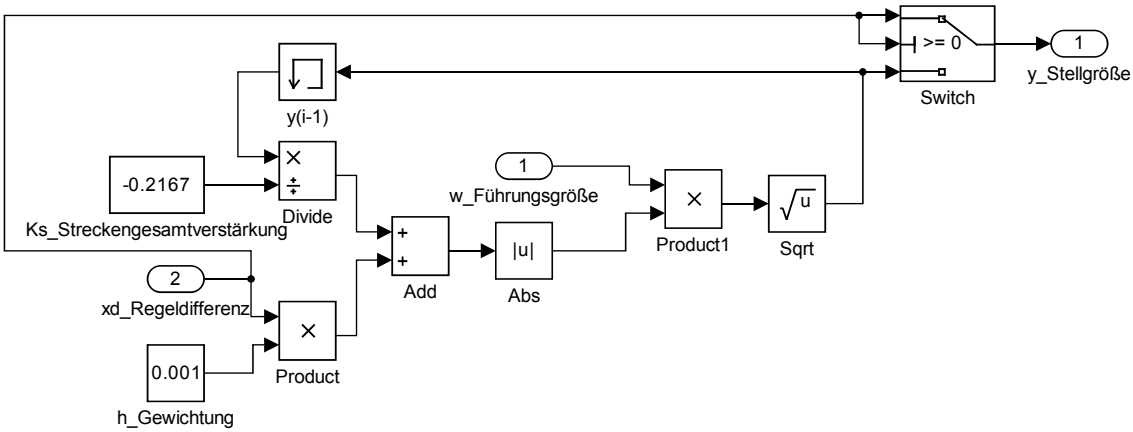


Abbildung 5.14: F_{Rt}-Wurzelrekursion Simulinkmodell

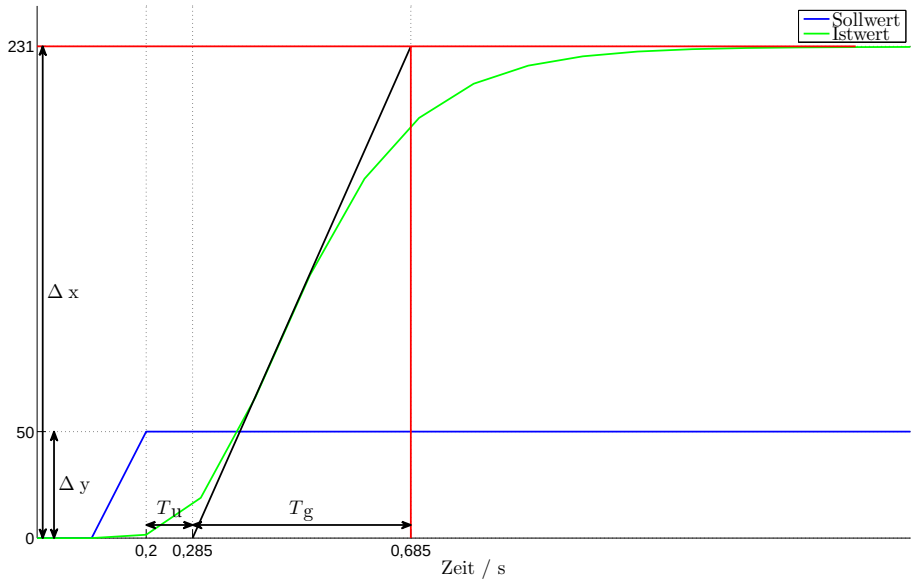


Abbildung 5.15: Sprungantwort Wurzelrekursionsregler

5.3). Der Algorithmus wurde in einem Simulinkmodell nachgebildet (Abbildung 5.14). Um die Stabilität des Regelkreises festzustellen, wurde die Sprungantwort des Reglers aufgezeichnet. Abbildung 5.15 zeigt die Sprungantwort.

Anhand der Sprungantwort wurde die Verzugszeit (T_u = 0,085 s) und die Ausgleichszeit (T_g = 0,4 s) ermittelt. Die Streckenverstärkung ergibt sich laut Gleichung 5.3 zu:

$$K_S = \frac{\triangle x}{\triangle y} = \frac{(231-0)}{(50-0)} = \frac{231}{50} = 4,62 \tag{5.9}$$

Diese Werte wurden zur Analyse des Streckenverhaltens verwendet. Es ergibt sich folgendes Übertragungsverhalten (Vergleich Gleichung 5.4):

$$G_{\mathrm{R}}(\mathbf{s}) = \frac{K_{\mathrm{S}}}{\mathbf{s} \cdot T_{\mathrm{g}} + 1} \cdot e^{-\mathbf{s} \cdot T_{\mathrm{u}}} = \frac{4,62}{\mathbf{s} \cdot 0,4\,s + 1} \cdot e^{-\mathbf{s} \cdot 0,085\,s} \tag{5.10}$$

Durch Einsetzen in Gleichung 5.8 wurde zur Bewertung der Stabilität das Nyquistdiagramm des offenen Kreises erstellt. Dies ist in Abbildung 5.16 dargestellt.

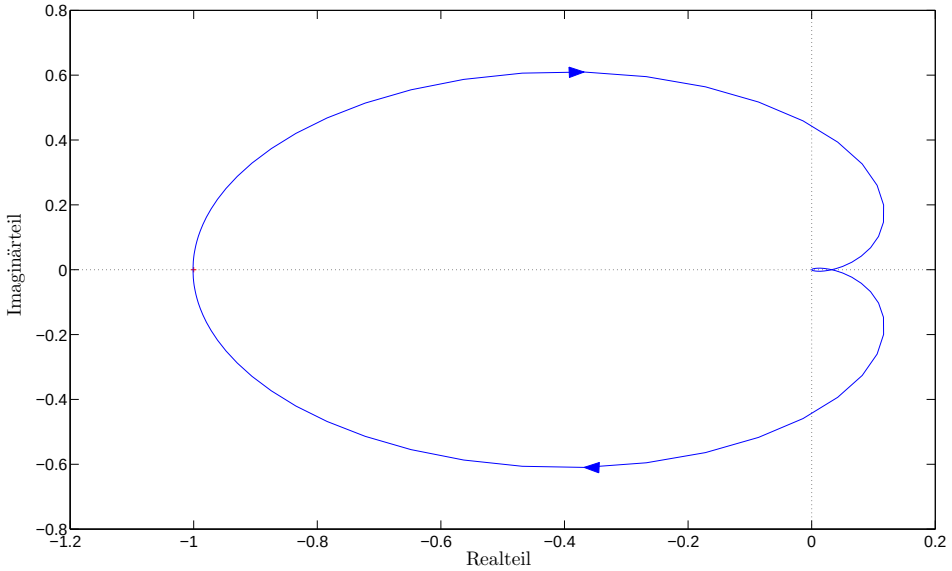


Abbildung 5.16: Nyquistdiagramm für Regelung mit Wurzelrekursion

Anhand des Nyquistdiagrammes ist nun zu erkennen, das der kritische Punkt $-1 + j0$ umschlungen wird. Da es keine instabilen Polstellen gibt, kann das System nicht als stabil angesehen werden. Der $\mathrm{F_{Rt}}$-Wurzelrekursionsalgorithmus wurde daher nicht für die Kühlung eingesetzt.

6 Visualisierung

6.1 Bedieninterface

Entgegen der ursprünglichen Planung wurde bereits ein Bedieninterface eingerichtet. Als Erweiterung zum bestehenden Interface wurde eine Visualisierung der aktuellen Zustandsgrößen eingebunden. Zudem ermöglicht diese Visualisierung die Bedienung der Akkumulatorsimulation. Die Visualisierung wurde mit CANoe angefertigt. Abbildung 6.1 zeigt diese Visualisierung. An den Rändern werden die Temperaturen der Maschinen, der Inverter, der DCDC-Wandler und des Kühlwassers für beide Maschinensätze als Zeigerinstrumente angezeigt. Befindet sich der Zeiger im roten Bereich, so liegt eine Überschreitung der Maximaltemperatur für dieses Gerät vor. In dieser Visualisierung kann der Wert für das Fahrziel eingegeben werden, welches erreicht werden soll (Feld: „Fahrmeter Sollwert“). Unterhalb dieses Feldes wird der noch verfügbare Restweg angezeigt. Zudem wird der maximale Entladestrom für das gewünschte Fahrziel angezeigt. Der dynamische Wert wird an die Inverter weitergeleitet. Der Wert „Maximalstrom“ zeigt den festen Maximalstrom für das Fahrziel an. Die aktuellen Werte für Strom, Spannung, Drehzahl und Drehmoment von Inverter 1 werden ebenso angezeigt. Der Schalter „Normalmodus / Steuermodus“ beschreibt die Variable für den SoC-Modus der Akkusimulation (siehe 4.1.4). Im Normalmodus wird der Akkumulator normal entladen. Der grüne Balken zeigt den Ladezustand an. Im Steuermodus kann durch den Regler unterhalb des Balkens ein Ladezustand vorgegeben werden. Solange der Steuermodus aktiv ist, bleibt die eingestellte Spannung stabil. Wird der Schalter zurück auf Normalmodus gestellt, so wird ab dem aktuell eingestellten Ladezustand der Akku entladen. Zudem wird die Pumpendrehzahl angezeigt.

6.2 Fahrsimulator

Zum Testen der Komponenten und zu Vorführzwecken sollte als Erweiterung zum Bedininterface ein Fahrsimulator aufgebaut werden. In diesem soll eine Geschwindigkeitskurve vorgegeben werden, welche ein Fahrer mittels der Pedalerie bei realer Motordrehung nachfahren soll. Durch die Trägheitssimulation [13] soll der Fahrer ein Feedback über das Verhalten des Antriebsstranges erhalten. Mittels eines Punktsystems soll zudem die Fahrt bewertet werden. Da die Inverter und Motoren mittels CANoe angesteuert werden, sollte der Fahrsimulator in CANoe eingebaut werden. Es ist jedoch notwendig die Soll-Geschwindigkeitskurve anzuzeigen. Dadurch kann sich der Fahrer auf Brems-, Beschleunigungs- oder Konstantgeschwindigkeitsphasen vorbereiten. CANoe kann in der Visualisierung keine statischen Kurven anzeigen. MATLAB Simulink ist in der Lage, statische Kurven anzuzeigen und dynamisch Werte hinzuzufügen. Daher wurde sich für eine parallele Simulation zwischen MATLAB und CANoe entschieden. Das Programm

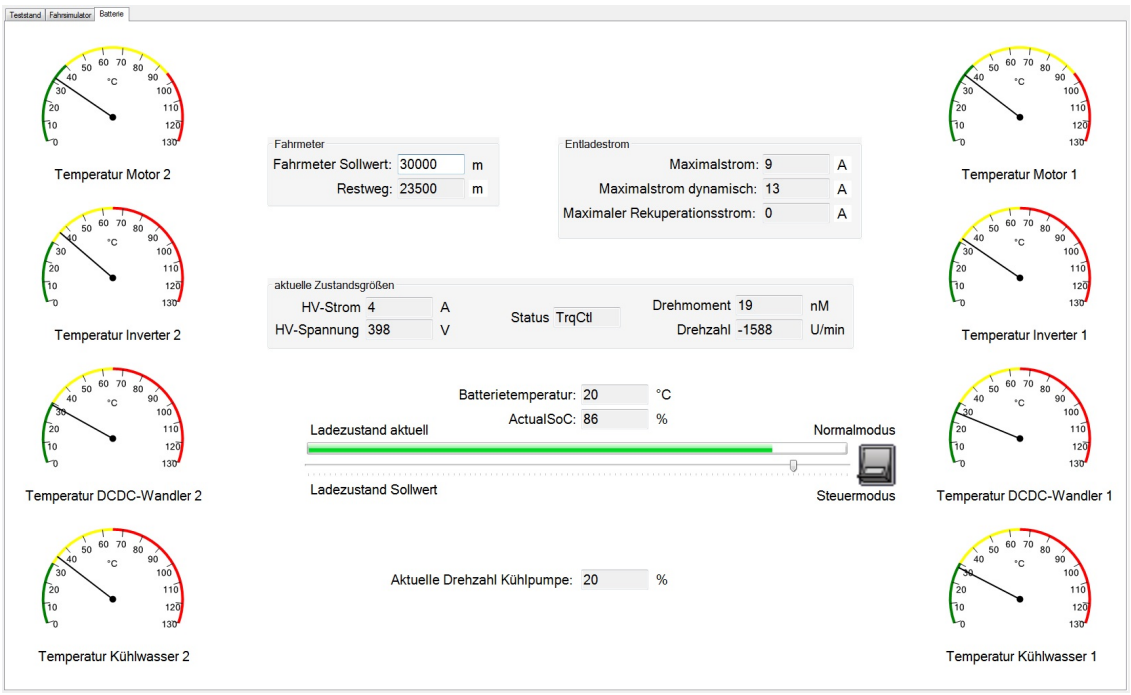


Abbildung 6.1: Visualisierung

CANoe soll den Teststand steuern und die aktuellen Zustandsgrößen anzeigen während Simulink die Berechnung der Streckenvorgabe, die Punktberechnung und die Anzeige der Geschwindigkeitskurve durchführt. Für eine Kommunikation der Programme untereinander wurden Systemvariablen in CANoe angelegt. Eine Systemvariable kann sowohl von MATLAB als auch von CANoe gelesen und beschrieben werden. Hierfür wurde ein Namensraum „Fahrsimu“ angelegt, welcher die Kommunikationsvariablen beinhaltet. Die beinhalteten Variablen mit Kurzbeschreibung befinden sich in Tabelle 6.1.

Variable	Zweck	Kommunikationsrichtung
Abweichung	beinhaltet den momentanen Mittelwert der Abweichung zwischen Soll- und Istwert in $\frac{\text{km}}{\text{h}}$	MATLAB $\rightarrow$ CANoe
Fortschritt	beinhaltet den momentanen Fortschritt der Fahrt in %	MATLAB $\rightarrow$ CANoe
Istgeschwindigkeit	beinhaltet die momentane Istgeschwindigkeit in $\frac{\text{km}}{\text{h}}$	MATLAB $\rightarrow$ CANoe[12]
Neustart	Reseten des Fahrsimulators	CANoe $\rightarrow$ MATLAB
Punkte	momentan erreichte Punktzahl	MATLAB $\rightarrow$ CANoe
Sollgeschwindigkeit	vorgegebene Sollgeschwindigkeit	MATLAB $\rightarrow$ CANoe
StartSimu	Starten des Simulators	CANoe $\rightarrow$ MATLAB
StopSimu	Fahrt abgeschlossen	MATLAB $\rightarrow$ CANoe

Tabelle 6.1: Systemvariablen im Namensraum „Fahrsimu“

[12] die aktuelle Istgeschwindigkeit des Motors wird mittels des CAN-Signals „SpeedActual“ in MATLAB in $\frac{\text{km}}{\text{h}}$ umgerechnet.

6.2.1 Der Fahrzyklus

Als Grundlage für den Fahrzyklus wurde der neue Europäische Fahrzyklus (NEFZ) [12, Seite 171 ff.] genutzt. Da dieser jedoch Phasen mit einer Geschwindigkeit von 0 $\frac{\text{km}}{\text{h}}$ beinhaltet und dies im Fahrsimulator keine Herausforderung darstellt, wurde der Zyklus abgeändert. Der abgeänderte Zyklus beinhaltet lediglich zu Beginn eine kurze Phase des Stillstandes. Im Zyklus befinden sich viele Phasen mit Geschwindigkeitsänderungen und kurze Abschnitte mit konstanten Geschwindigkeiten. Die Maximalgeschwindigkeit beträgt 50 $\frac{\text{km}}{\text{h}}$, da die maximale Drehzahl am Teststand auf 3200 $\frac{\text{U}}{\text{min}}$ begrenzt ist. Die Begrenzung resultiert aus der Momentbegrenzung. Zudem beinhaltet dieser einen Sinusteil (90 Sekunden bis 135 Sekunden) und einen starken Bremsvorgang (75 Sekunden). Die Gesamtfahrzeit beträgt 3 Minuten. Es ist möglich diesen Zyklus gegen einen beliebigen anderen auszutauschen um beispielsweise ein reales Steckenprofil nachzufahren. Der Zyklus ist in Abbildung 6.2 zu sehen.

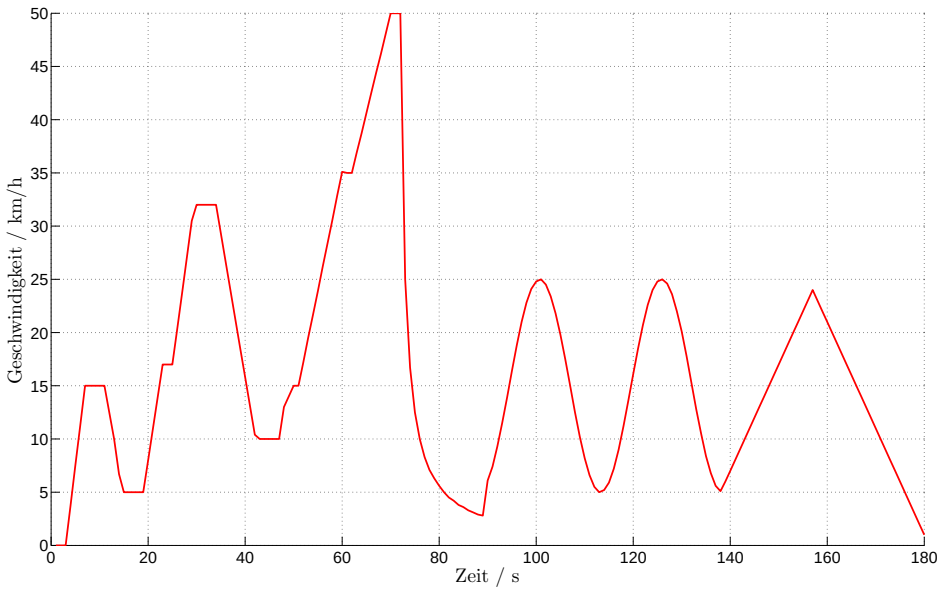


Abbildung 6.2: Fahrzyklus

6.2.2 Berechnung in MATLAB

Für die parallele Simulation ist MATLAB der „Master“, da MATLAB CANoe aufrufen kann. CANoe ist nicht in der Lage MATLAB aufzurufen. Zum Start der Simulation wird der Zyklus eingelesen und in einem Plot-Fenster angezeigt. Zudem wird der Zyklus als timeseries-Variable[13] in die Simulation weitergegeben. Die Variable „ende“ beinhaltet die Länge des Zyklus und wird ebenfalls in die Simulation weitergegeben. Zudem wird durch den Block „CANoe Simulation Step“ die Berechnung in CANoe gestartet. Die

[13] Eine timeseries-Variable ist eine Tabelle, in welcher für jeden Zeitwert in einem zeitlichen Rahmen ein Wert vorliegt. Innerhalb der Simulation wird der zur aktuellen Zeit zugehörige Wert ausgegeben.

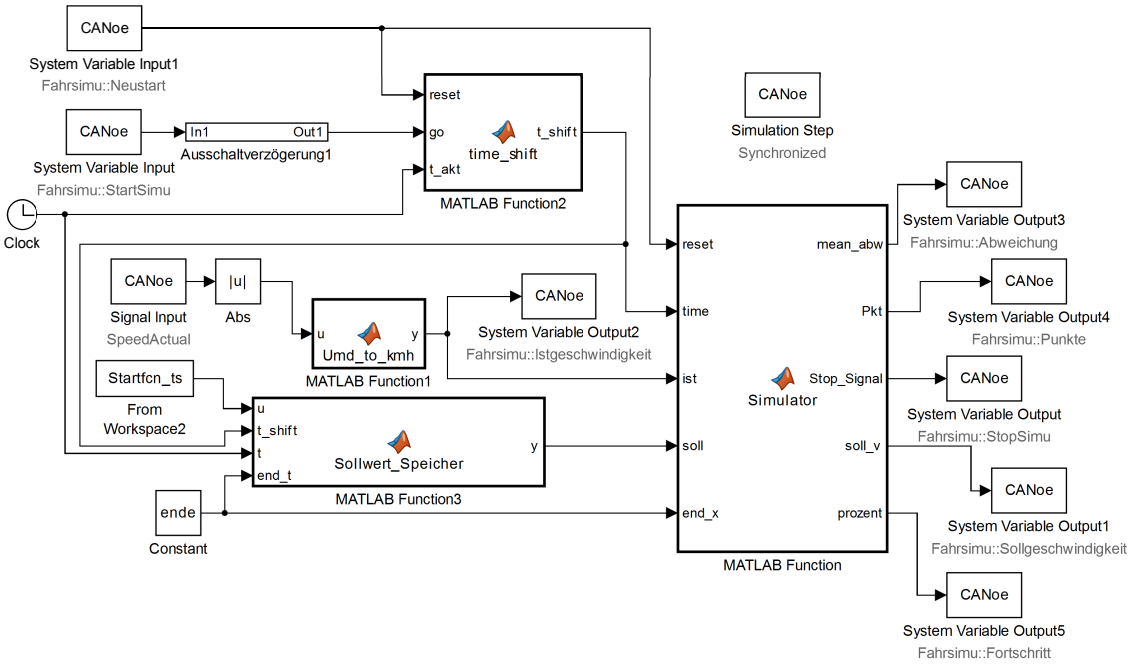


Abbildung 6.3: Simulink Modell Fahrsimulator

Einstellung „Synchronized“ bewirkt das als Zeitbasis die Simulationszeit von CANoe genutzt wird. Das Modell ist in Abbildung 6.3 abgebildet.

Die MATLAB-Funktion „Simulator“ beinhaltet die Hauptfunktion für den Fahrsimulator. Die Berechnung des Fahrsimulators wird jede Sekunde durchgeführt, da dies vollkommen ausreichend ist. Die Simulationszeit wird daher auf volle Sekunden geprüft, indem die Simulationszeit mittels Modulo auf eine Nachkommastelle gleich null geprüft wird. Falls dies zutrifft, so wird dem geöffneten Plot (welcher den Zyklus als Vorgabewert beinhaltet) ein Wertepaar bestehend aus dem aktuellen und dem letzten Geschwindigkeitswert hinzugefügt. Dadurch wird während der Fahrt zu der angezeigten Sollkurve die aktuelle Istkurve schrittweise hinzugefügt. Zudem wird der aktuelle Geschwindigkeitswert für den nächsten Durchlauf gespeichert. Die momentane Abweichung zwischen Soll- und Istwert wird errechnet und bewertet. Ist diese kleiner als 3 $\frac{\text{km}}{\text{h}}$, so werden zu der Punktzahl 2 Punkte addiert. Liegt die Abweichung unter 1 $\frac{\text{km}}{\text{h}}$, so werden 3 weitere Punkte addiert. Zudem wird anhand des aktuellen Zeitwertes der prozentuale Fortschritt errechnet. Weiterhin wird der Mittelwert der bisherigen Abweichungen von der Sollkurve ermittelt. Außerdem wird geprüft, ob die akuelle Zeit kleiner ist als die Gesamtfahrzeit. In diesem Fall wird die „Stop_Signal“-Variable gesetzt, um anzuzeigen, dass die aktuelle Fahrt beendet ist. Es wird zudem geprüft, ob die Variable „reset“ gleich 1 ist. In diesem Fall wird der Fahrsimulator für eine neue Fahrt zurückgesetzt. Der aktuelle Plot wird geschlossen und ein neuer geöffnet, in welchem die Sollkurve eingezeichnet wird. Zudem werden die Punkte und die Abweichungen genullt. Abschließend werden die Variablen für die Abweichung, die Punktzahl, das Stop-Signal, die momentane Sollgeschwindigkeit und den prozentualen Fortschritt an CANoe weitergegeben. Das komplette Script befindet sich im Anhang F.

Da die MATLAB-Funktion „Simulator“ die Simulationszeit als Bezugszeit nutzt, kann diese nur einen Durchgang berechnen. Die MATLAB-Funktion „time_shift“ wurde daher entworfen, um anstelle der Simulationszeit eine Zeitbasis für „Simulator“ zu generieren. Innerhalb dieser Funktion wird nach einem Startsignal (Systemvariable „StartSimu“) die aktuelle Simulationszeit gespeichert. Die Ausgangsvariable „time_shift“ errechnet sich aus der aktuellen Simulationszeit abzüglich der gespeicherten Zeit. Durch diese Funktion wird die Simulationszeit verschoben, so dass die Funktion „Simulator“ immer eine Zeitbasis angefangen bei null Sekunden erfährt. Durch die Eingangsvariable „reset“ (Systemvariable „Neustart“) kann der Vorgang beliebig oft wiederholt werden.

Die Funktion „Sollwert_Speicher“ zeichnet während des ersten Durchgangs die Sollwertkurve auf. Über die Eingangsvariable „time_shift“ kann die Sollwertkurve erneut aufgerufen werden. Dies ist notwendig, da die Funktion „Simulator“ zum Berechnen des Fahrsimulators zu jedem Zeitwert den entsprechenden Wert aus der timeseries-Variable benötigt, dieser aber nur zur entsprechenden Simulationszeit verfügbar ist. Gemeinsam mit der Funktion „time_shift“ wird somit die Zeitbasis und der Sollwert verzögert, so das der Simulator beliebig oft ausgeführt werden kann.

Der aktuelle Geschwindigkeitswert wird mit der MATLAB-Funktion „Umd_to_kmh“ anhand der Drehzahl errechnet. Die Geschwindigkeit ist hierbei abhängig von der Getriebeübersetzung sowie dem Radumfang. Die errechneten Kilometer pro Stunde werden an CANoe (Systemvariable „Istgeschwindigkeit“) und an den Simulator weitergegeben.

6.2.3 Anzeige in CANoe

Die Berechnung in CANoe wird durch MATLAB gestartet. Durch CANoe werden die Inverter gesteuert. In der Visualisierung (Abbildung 6.4) werden die Soll- und Istgeschwindigkeit durch analoge Instumente wiedergegeben. Zudem werden die Motortemperaturen sowie der Status der Inverter angezeigt. Die Schaltfläche „Fahrsimulator starten“ setzt die Systemvariable „StartSimu“, wodurch das MATLAB-Modell beginnt, den Simulator zu berechnen. Sobald die Systemvariable „Stop_Signal“ gesetzt wird, wird das Anzeigeelement „Fahrt abgeschlossen“ rot, um dem Fahrer mitzuteilen, dass die Fahrt beendet ist. Zudem wird durch den Ladebalken „Gesamtfortschritt“ die Systemvariable „Fortschritt“ angezeigt. Zusätzlich werden die aktuellen Punkte und die durchschnittliche Abweichung angezeigt. Die Schaltfläche „Neustart“ setzt die Systemvariable „Neustart“, wodurch die letzte Fahrt gelöscht wird. Dadurch werden die Punkte sowie die Abweichung genullt und ein neuer Plot mit der Sollkurve wird generiert. Durch „Fahrsimulator starten“ kann nun eine neue Fahrt begonnen werden.

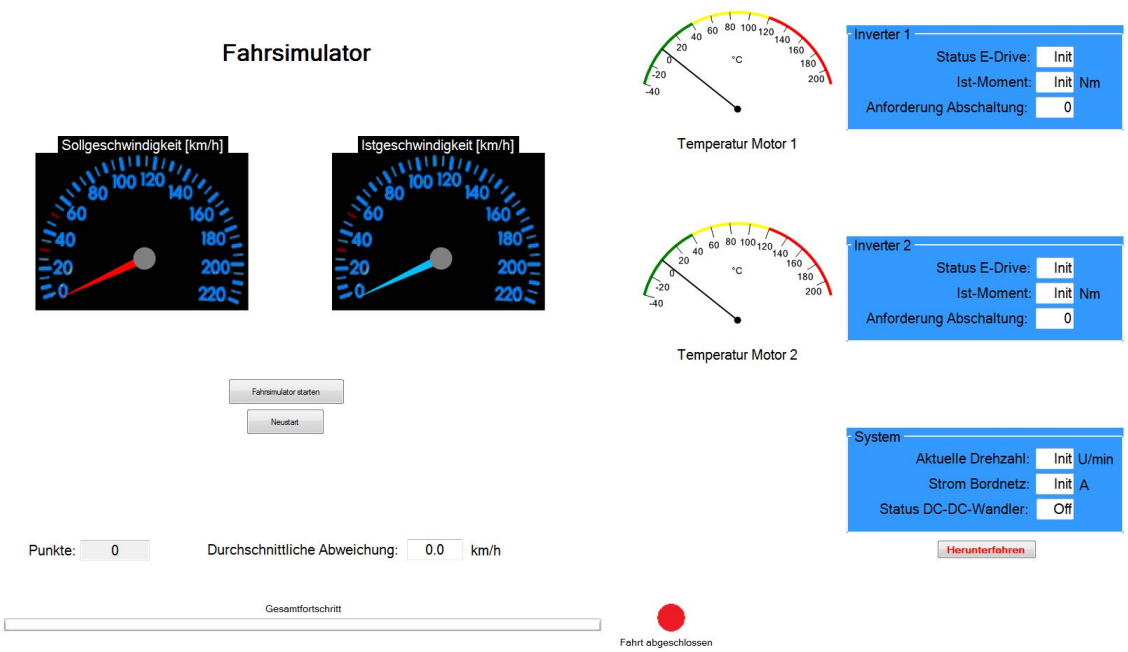


Abbildung 6.4: CANoe Visualisierung Fahrsimulator

6.2.4 Test des Fahrsimulators

Zum Testen des Simulators wurde eine Testfahrt am realen System vorgenommen. Da zu diesem Zeitpunkt die Pedalerie noch nicht zur Verfügung stand, wurde die Geschwindigkeit mittels Momentvorgabe realisiert. Das Ergebnis der Testfahrt ist in Abbildung 6.5 zu sehen. Die Simulation errechnete für diese Fahrt eine Punktzahl von 294 und eine durchschnittliche Abweichung von 4,1 $\frac{\text{km}}{\text{h}}$.

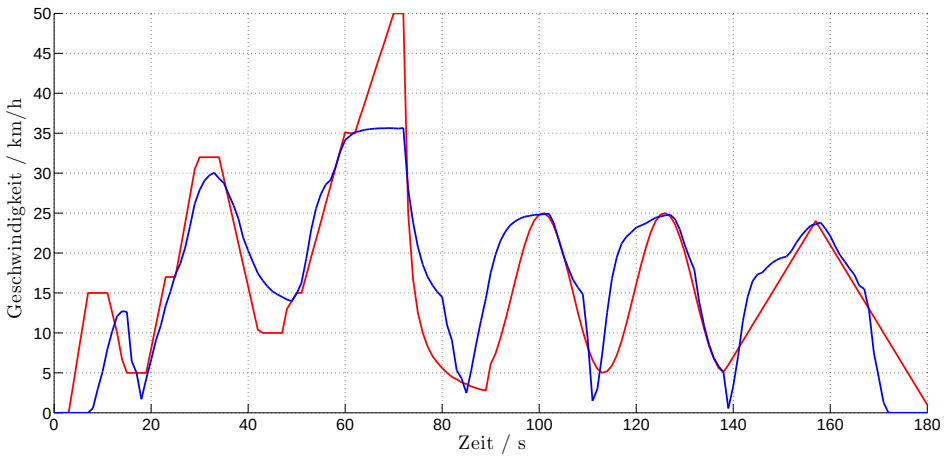


Abbildung 6.5: Testfahrt

7 Überprüfung der Algorithmen am Teststand

Die Akkumulatorsimulation sowie die Restweganzeige und die dynamische Strombegrenzung wurden am Teststand getestet. Da das Moment am Teststand begrenzt ist, wurde auch eine kleinerer Akkumulator simuliert. Der simulierte Akku besteht aus 100 Zellen in Reihe in einem Strang. Es ergibt sich eine Spannung von durchschnittlich 370 V und eine Energiemenge von 5 Ah. Die Strom-Weg-Tabelle (siehe Tabelle 4.5) wurde hierfür neu berechnet. Vorgegeben wurde ein variables Drehmoment (Abbildung 7.1) zwischen 10 und 40 Nm. Als Wegvorgabe wurden 38 km eingestellt. Hierfür ergab sich ein Maximalstrom von 4 A. Während der Tests stellte sich heraus, dass das Beschreiben der CAN-Signale „AvailableMaxDisChgCurrContinuous“, „AvailableMaxDisChgCurrLongTerm“ sowie „AvailableMaxDisChgCurrShortTerm“ keine Strombegrenzung im Inverter bewirkt. Zum damaligen Zeitpunkt konnte nicht ermittelt werden, warum der Inverter diese Vorgaben ignoriert. Für diesen Test wurde daher die Strombegrenzung manuell durchgeführt. Als Startwert für den Ladezustand wurde 90 % gewählt.

7.1 Testen der Algorithmen

Abbildung 7.2 zeigt den Inverterstrom und die Grenze durch die dynamische Strombegrenzung. Es ist zu erkennen, dass der Inverterstrom erst nach dem Fallen der Stromgrenze sinkt. Das ist auf die manuelle Bedienung zurückzuführen. Zudem ist zu erkennen, dass der berechnete Maximalstrom von 4 A überschritten werden kann, falls dieser Grenzstrom für einige Zeit unterschritten wurde. Tritt eine Überschreitung auf, so wird die Grenze wieder auf den Stromwert gesetzt, der das Erreichen des gewünschten Fahrziels sicherstellt.

Aufgezeichnet wurde ebenso die errechnete Akkuklemmspannung und die Spannung am Inverter. Abbildung 7.3 zeigt die Spannungen. Es zeigt sich, dass die Akkuspannung steigt, obwohl ein Strom aus dem Akku entnommen wird. Dies ist darauf zurückzuführen, dass der Strom sinkt und dadurch eine andere Entladekurve genutzt wird (siehe Abbildung 4.1).

Der Ladezustand wurde aufgezeichnet (Abbildung 7.4). Der Ladezustand sinkt innerhalb der Messung von 90 % auf 76 % ab. Die Messdauer betrug mehr als 13 Minuten. Der durchschnittlich entnomme Strom betrug 3,3 A. Da die Entladung bei 5 A eine Stunde gedauert hätte, ist eine Entladung von 14 % in 13 Minuten eine gute Annäherung an einen realen Akku. Zudem ist zu erkennen, dass größere Ströme den Ladezustand schneller sinken lassen als niedrige.

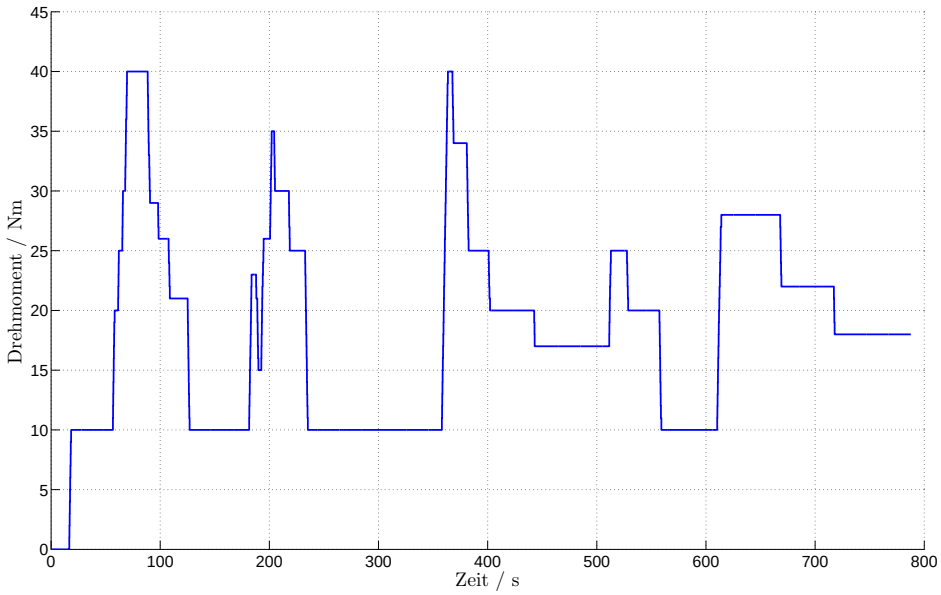


Abbildung 7.1: Momentvorgabe

Der Anzeigewert des verfügbaren Restweges wurde ebenso aufgezeichnet. Dieser ist in Abbildung 7.5 dargestellt. Es ist zu erkennen, dass die Anzeige sehr viele Sprünge zeigt. Die Anzeige im Rennfahrzeug würde demnach bei einem Bremsvorgang sprunghaft einen größeren Restweg anzeigen. Der angezeigte Wert wäre demnach nicht zu gebrauchen. Der Grund hierfür liegt wahrscheinlich in der zu häufigen Ausführung des Modells (Abbildung 4.19). Würde das Modell seltener ausgeführt werden, würde der Wert nicht so oft aktualisiert werden. Aus diesem Grund wurde das Modell im Steuergerät statt in einen 100-Millisekundentask in einen sekündlichen Task eingetragen. Dadurch wird das Modell nicht nur seltener ausgeführt, es werden auch anstelle der letzten 10 Sekunden die letzten 100 Sekunden betrachtet. Dadurch kann sich der Wert nicht so schnell ändern.

Zudem wurde auch die Drehzahl aufgezeichnet (Abbildung 7.6). Anhand der Drehzahl wurde nun die zurückgelegte Strecke errechnet. Die Durchschnittsdrehzahl betrug 1458 $\frac{\mathrm{U}}{\mathrm{min}}$. Bei einer Messdauer von über 13 Minuten wurden somit 19147 Umdrehungen (N) durchgeführt. Angenommen wurde eine Getriebeübersetzung (i) von 1:6 und einem Radradius (r) von 250 mm[14]. Anhand folgender Formel wurde die zurückgelegte Strecke errechnet:

$$s = \frac{N}{i} \cdot 2 \cdot \pi \cdot r \tag{7.1}$$

Es wurde somit eine Strecke von 5012 m zurückgelegt. Laut Restweganzeige wurde jedoch ein Weg von 6200 m zurückgelegt. Dies ergibt sich aus der Differenz zwischen Anfangs- und Endwert in Abbildung 7.5. Das Verhältnis zwischen real zurückgelegtem Weg und Anzeige ergibt sich zu 0,81. Der Unterschied zwischen den Werten resultiert

[14] Die realen Parameter lagen zu diesem Zeitpunkt nicht vor.

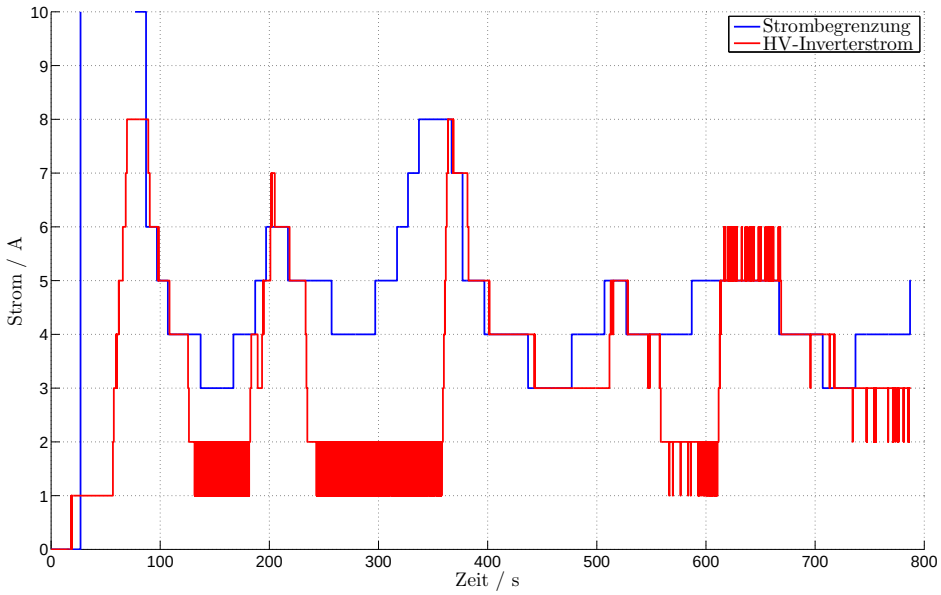


Abbildung 7.2: Inverterstrom und Strombegrenzung

daher, dass für die Berechnung des Restweges nur der statische Fall mit konstanten Strömen berücksichtigt wurde. Die Energie für das Beschleunigen des Fahrzeuges wurde nicht mit einbezogen. Um dies zu kompensieren, wurde ein Korrekturfaktor für die Restweganzeige in das Restwegmodell eingebaut. Als Faktor wurde 0,75 gewählt, da dieser Wert etwas kleiner ist als das Verhältnis zwischen realem und theoretischem Wert. Dadurch sollte die Anzeige zudem immer einen niedrigeren Wert anzeigen, als real möglich. Das neue Restwegmodell ist in Abbildung 7.7 dargestellt. Im Anhang G befindet sich die vollständige Auswertung der Wegstrecke.

Ebenso wurde die Strombegrenzung des Inverters ausgewertet. Als Vorgabe für den Gesamtfahrweg wurden 38 km eingegeben. Während des Testes wurden 14 % der Akkuladung entnommen und damit 5012 m zurückgelegt. Für 100 % Akkuladung ergibt sich daraus ein Gesamtfahrweg von 34811 m. Das Verhältniss zwischen realem und theoretischen Wert liegt bei 0,92. Um das gewünschte Fahrziel zu erreichen, muss an dieser Stelle ebenso ein Korrekturfaktor eingebunden werden. Es wurde sich für den selben Korrekturfaktor wie bei der Restweganzeige entschieden, da diese beiden Algorithmen auf der selben Grundlage beruhen. Das Modell mit Korrekturfaktor ist in Abbildung 7.8 dargestellt.

7.2 Testen der überarbeiteten Algorithmen

Die korrigierten Modelle für die Restwegberechnung und die Strombegrenzung wurden in das Steuergerät übertragen. Der Test wurde unter gleichen Startbedingungen wiederholt. Der Verlauf der Restweganzeige ist in Abbildung 7.9 dargestellt. Im Ver-

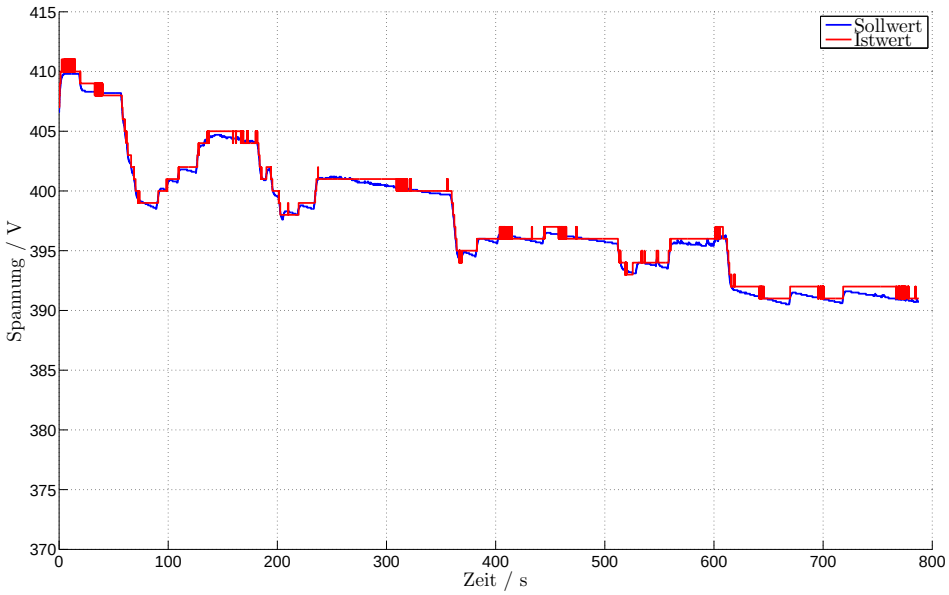


Abbildung 7.3: Akkumodellspannung und Inverterspannung

gleich mit der vorherigen Restweganzeige (Abbildung 7.5) sind nun weniger Sprünge zu erkennen. Die Anzeige ist damit stabiler. Die Differenz der angezeigten Wegstrecke betrug 5100 m. Insgesamt wurden 22281 Umdrehungen zurückgelegt, nach Gleichung 7.1 entspricht dies 5833 m. Demzufolge wurde eine größere Strecke zurückgelegt, als angezeigt wurde. Dadurch kann nun davon ausgegangen werden, dass die Anzeige weniger Stecke anzeigt, als real verfügbar ist.

Als Fahrziel wurden 38 km eingegeben. In diesem Test wurden 15,6 % der Ladung umgesetzt um 5833 m zurückzulegen. Für einen vollen Akku entspricht dies theoretisch 37392 m. Dies ist immer noch weniger als vorgegeben. Die geringe Differenz zwischen realem und theoretischem Weg kann jedoch auch auf die manuell durchgeführte Strombregrenzung zurückgeführt werden. Für weitere Tests am Teststand sind die eingestellten Korrekturfaktoren ausreichend.

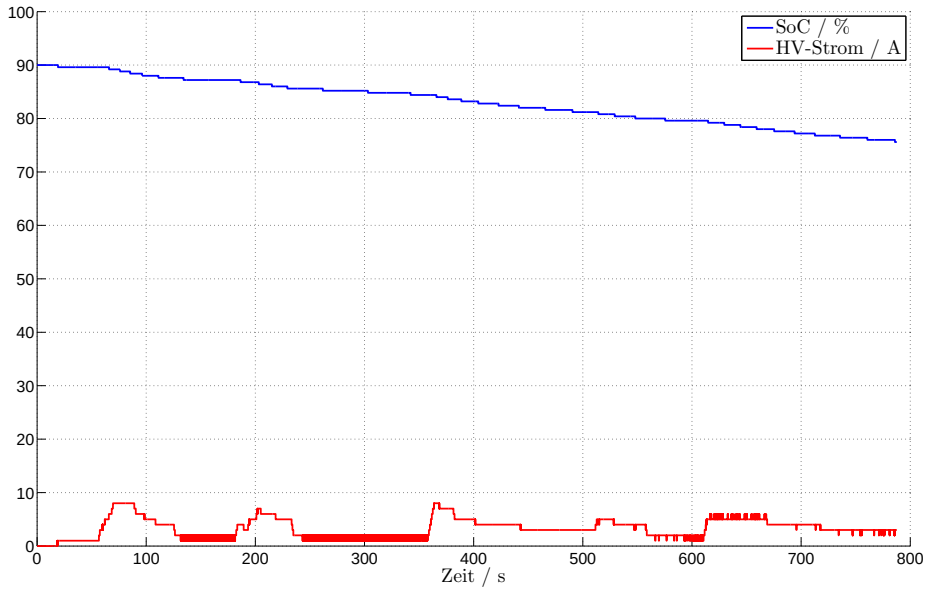


Abbildung 7.4: Ladezustand und Inverterstrom

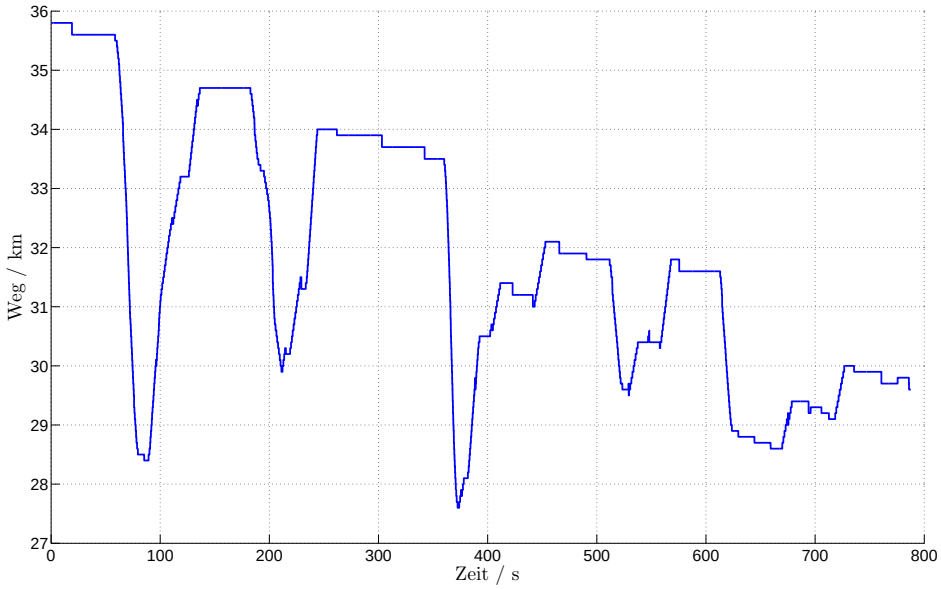


Abbildung 7.5: Restweganzeige

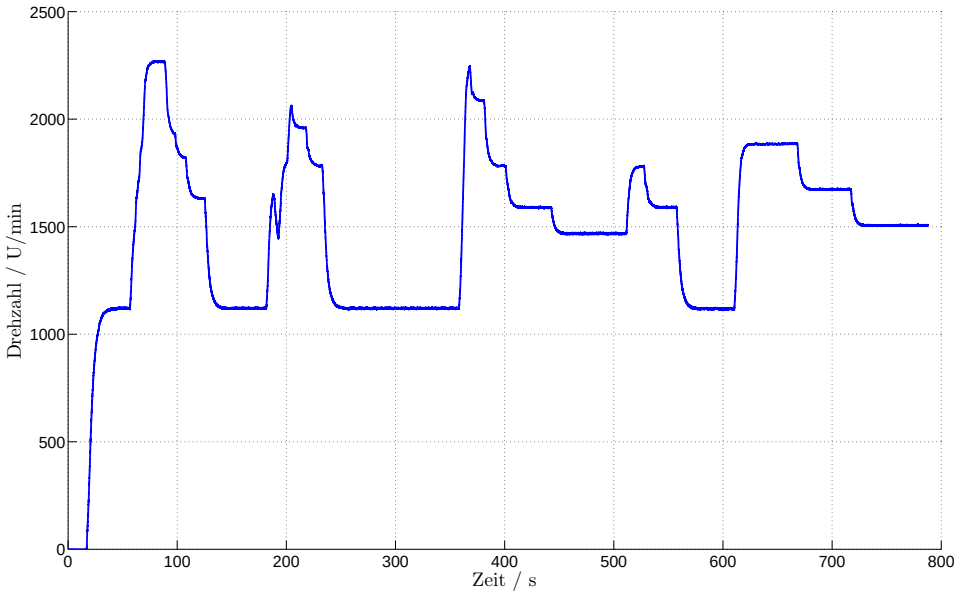


Abbildung 7.6: Drehzahl

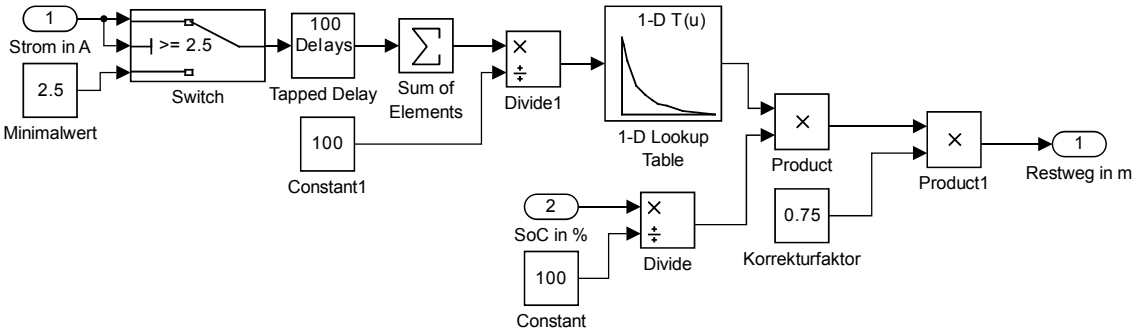


Abbildung 7.7: neues Restweg-Modell

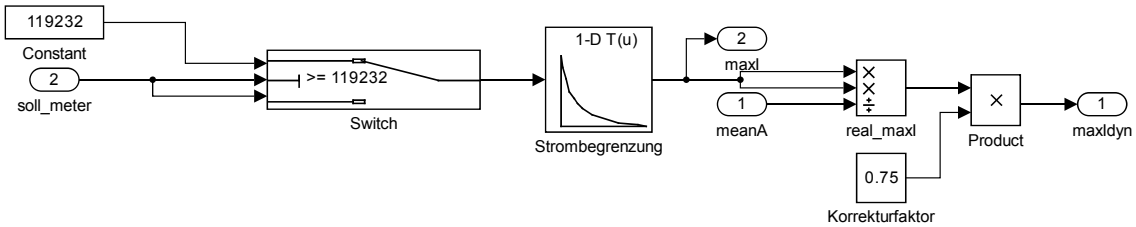


Abbildung 7.8: neues Stromgrenzenmodell

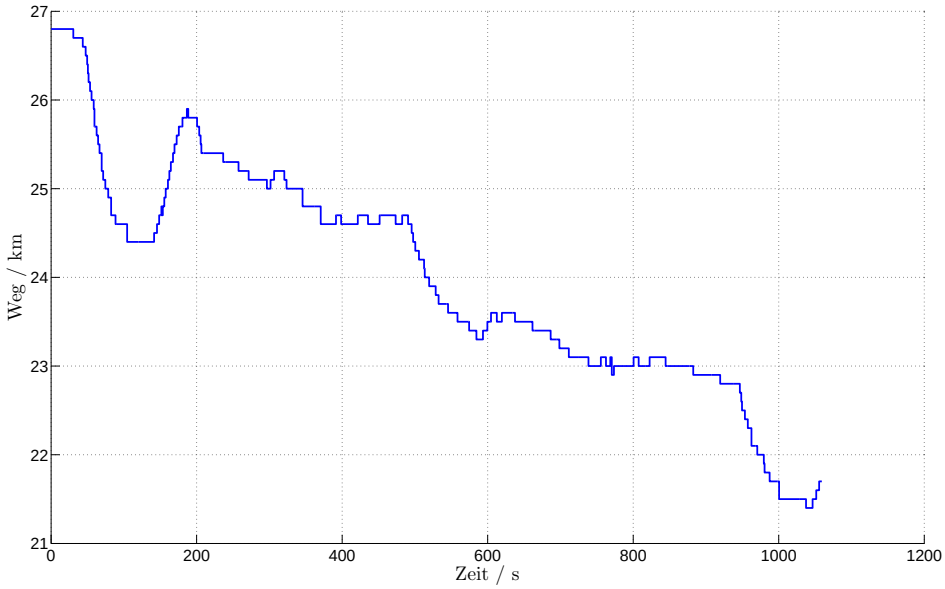


Abbildung 7.9: Restweganzeige korrigiert

8 Zusammenfassung und Ausblick

8.1 Akkumulatorsimulation

Als Grundlage für das Energiemanagment und zum Testen war es nötig einen Akkumulator zu simulieren. Die Simulation der Akkus wurde in drei Modelle aufgeteilt, um die drei relevanten Ausgangsgrößen, Klemmspannung, Temperatur und Ladezustand getrennt voneinander zu modellieren. Für das Klemmspannungsmodell wurde zunächst eine Zelle modelliert. Das Modell der Zelle verhält sich wie eine Zelle laut Datenblatt. Im Klemmspannungsmodell wurde die Zelle eingesetzt und je nach Akkugröße wurden die Zellenwerte hochskalliert. Das thermische Modell bestimmt anhand der Verlustleistung der Zelle die Erwärmung und bestimmt so die Temperatur der Zelle beziehungsweise des Akkus. Das Ladezustandsmodell summiert den entnommenen Strom auf, um somit die entnommene Ladung zu errechnen. Dadurch wird der Ladezustand bestimmt. In verschiedenen Tests zeigte sich, dass das Akkumulatormodell sich wie ein realer Akku verhält.

Das Verhalten bei kritischen Akkuzuständen wurde ebenfalls untersucht. Zur Verhinderung von Überladung wurde ein Modell entworfen, welches bei einer Akkuspannung über 410 V sowie bei einer Geschwindigkeit unter 5 $\frac{\mathrm{km}}{\mathrm{h}}$ die Rekuperation durch den Inverter verhindert. Am Teststand konnte diese Funktion bislang nicht getestet werden, da die Rekuperation noch nicht vollständig in der Fahrzeugsteuerung implementiert ist. Um eine Tiefenentladung zu verhindern, wurde ein Modell entworfen, welches bei einer Akkuspannung unter 280 V die weitere Entladung durch den Inverter deaktiviert. Diese Funktion konnte ebenso noch nicht getestet werden. Zudem wurde ein Modell entworfen, welches bei einer Akkutemperatur über 60 $^{\circ}\mathrm{C}$ die Entladung durch den Inverter deaktiviert, um die thermischer Zerstörung des Akkus zu verhindert. Diese Modelle müssen noch in die Software für die Fahrzeugsteuerung integriert werden, welche jedoch noch nicht fertiggestellt ist.

Zudem wurde ermittelt, welche verfügbare Strecke bei einem gegebenen Ladezustand theoretisch noch zurückgelegt werden kann. Angewendet wurden hierfür die Gesetze der Mechanik, um zu berechnen, wieviel Energie benötigt wird, um ein Objekt zu bewegen. Dadurch konnte für eine konstante Stromentnahme aus dem Akku der zurückgelegte Weg bestimmt werden. In einem Modell wurde dies als Tabelle hinterlegt und anhand des durchschnittlichen Stroms und des Ladezustandes konnte somit eine Abschätzung getroffen werden, wieviel Fahrweg noch verfügbar ist. Anhand der selben Tabelle wurde eine Strombegrenzung eingerichtet, mit welcher ein gewünschtes Fahrziel vorgegeben werden. Diese Strombegrenzung wertet das Verhalten des Fahrers anhand des durchschnittlich entnommenen Akkustromes aus und variiert somit den maximalen Entladestrom. Dadurch wird erreicht, dass ein sparsamer Fahrer kurzzeitig Stromspitzen

aus dem Akku entnehmen kann. Diese Algorithmen wurden getestet. Dabei stellte sich heraus, dass für beide Algorithmen ein Korrekturfaktor notwendig war. Dies ist damit zu begründen, dass die Berechnungen immer von idealen statischen Fällen ausgehen. Die Algorithmen wurden im Steuergerät implementiert.

In zukünftigen Tests muss die Akkusimulation mit realen Messergebnissen abgeglichen werden. Die Dimensionierung des Tiefpasses (Abbildung 4.4) muss vorgenommen werden. Hierfür muss der Akku (beziehungsweise eine Zelle) mit einem konstanten Entladestrom (beispielsweise 1 C) entladen werden, welcher zu einem definierten Zeitpunkt auf einen anderen Entladestrom umgeschaltet wird. Die Klemmspannung wird somit eine andere Entladekurve zeigen. Der Tiefpass soll das Verhalten im Umschaltvorgang nachbilden und könnte durch einen solchen Test dimensioniert werden. Ebenso muss das thermische Modell verifiziert werden. Hierfür müsste der Akku mit einem konstanten Strom entladen und die Erwärmung aufgezeichnet werden. Auch die Abkühlung muss real getestet werden. Die Parameter für den abgegebenen Wärmestrom und die Wärmekapazität des Akkus müssen für den realen Aufbau ermittelt werden. Auch könnte der thermische Einfluss auf die Klemmspannung eingearbeitet werden. Zudem wäre es denkbar, die Alterung der Zellen im Modell einzubeziehen. Hierfür wären weitere Modellparameter notwendig, welche das Alter und die Abnutzung beinhalten. Für die Restweganzeige und die Strombegrenzung wurden Parameter eines vorhandenen Fahrzeuges genutzt. Die realen Parameter für Fahrzeugmasse, Strömungswiderstandskoeffizient, Frontfläche sowie Antriebswirkungsgrad und Getriebeübersetzung müssen in die Modelle übernommen werden. Diese sind momentan nicht vorhanden, da das Fahrzeug noch nicht aufgebaut ist. Zudem müssen die Algorithmen für Tiefenentladeschutz und Überladungsschutz getestet werden. Diese Algorithmen müssen zudem in das Akkumulator-Managment-System eingearbeitet werden.

8.2 Kühlung

Für den Betrieb am Teststand und als Vorentwicklung für das Fahrzeug wurde ein Kühlkreislauf aufgebaut. Hierfür wurde eine Kühlpumpensteuerung entwickelt. Für den einfachen Testbetrieb wurde zunächst eine Ansteuerung mittels PWM realisiert. Diese Ansteuerung kühlt den Teststand ausreichend, so dass keine thermische Gefährdung der Komponenten auftreten kann. Dieser Aufbau ermöglichte es, erste Tests des Antriebskonzeptes vorzunehmen. Die Kühlung im Fahrzeug soll möglichst energiesparsam arbeiten. Hierfür wurde ein stetiger Regler implementiert, da dieser die Stellgröße variabel einstellen kann. Dadurch sollte erreicht werden, dass bei niedrigen Temperaturen der Komponenten die Kühlmittelpumpe eine niedrige Drehzahl einstellt. Getestet wurden ein PID-Regler mit verschiedenen Parametern. Für die Parametrierung der Regler wurde die Sprungantwort des Systems aufgezeichnet. In Tests stellte sich der PID-Regler mit Parametern nach Ziegler/ Nichols als energiesparsamste Alternative heraus. Zudem wurde ein Algorithmus eingebunden, welcher bei einem Stromanstieg die Pumpendreh-

zahl reduziert. Der Grund hierfür war, dass ein Stromanstieg eine Beschleunigung des Fahrzeug nach sich zieht. Dadurch steigt die Effekivität des Kühlers, wodurch eine niedrigerere Drehzahl der Pumpe ausreichend sein könnte. Dies konnte am Teststand jedoch nicht nachgewiesen werden, da die Drehzahl des Lüfters für den Kühler konstant ist. Außerdem wurde ein Algorithmus eingebunden, welcher Bei Erreichen der Grenztemperatur einer Komponente den Antrieb abschalten soll und die Drehzahl der Kühlpumpe auf den Maximalwert stellen soll. Dies ist zum Schutz der Komponenten gedacht. Die Tests am Teststand wurden jedoch unter gedrosselten Bedingungen durchgeführt, da die Leistung der Maschinen aus Sicherheitsgründen begrenzt ist.

Für den Teststand wäre es denkbar, den Kühlkreislauf zu überarbeiten. Die Regelung der Kühlpumpe arbeitet momentan nur mit den Werten der Antriebsmaschine, da Fahrzeugverhältnisse nachgebildet wurden. Eine seperate Teststandkühlung müsste auch die Temperaturen des zweiten Maschinensatzes berücksichtigen. Für den Teststand sind zudem keine Energiesparalgorithmen notwendig, da kein Akkumulator vorhanden ist. Auch wäre es denkbar, die Drehzahl des Kühlerlüfters zu steuern, um in Abhängigkeit der Motordrehzahl eine Geschwindigkeitssimulation einzubinden. Dadurch könnte das Kühlsystem effektiver getestet werden.

Ein weiterer möglicher Algorithmus wäre die Auswertung der Pedalerie. Dies ist jetzt noch nicht möglich, da am Teststand noch keine Pedalerie eingebunden ist. Würde beispielsweise das Gaspedal betätigt, könnte man von einer baldigen Erwärmung des Antriebsstranges ausgehen und präventiv die Drehzahl der Kühlpumpe erhöhen. Dies würde jedoch dem Stromanstiegsalgorithmus widersprechen. Es wäre denkbar an dieser Stelle eine Fallunterscheidung anhand des Systemzustandes durchzuführen. Bei einer niedrigen Systemtemperatur könnte der Stromanstiegsalgorithmus angewandt werden, bei einer hohen Temperatur der präventive Algorithmus. Auch wäre es denkbar einen der Algorithmen komplett zu entfernen. Dies müsste durch Tests festgelegt werden. Eine Betätigung des Bremspedals könnte einen Rekuperationsstrom erzeugen, der zu hoch ist, als das der Akkumulator diesen aufnehmen könnte. Diese Energie könnte man dazu nutzen, die Kühlpumpe mit voller Drehzahl laufen zu lassen um das System weiter herab zu kühlen, um damit den nächsten Beschleunigungsvorgang eine größere thermische Aufheizreserve zu bieten.

Für das Fahrzeug muss die Aufzeichnung der Sprungantwort erfolgen, da es sich um ein vollkommen anderes System als am Teststand handelt. Dementsprechend müssten die Reglerparameter neu ermittelt werden. Zudem müsste das Kühlsystem sowie die Algorithmen im Fahrzeug getestet werden.

8.3 Visualisierung

Die Visualisierung der aktuellen Zustandsgrößen ist für die Bedienung des Teststandes notwendig. Visualisiert wurden hierfür die Temperaturen der Komponenten sowie die Werte der Antriebsmaschine. Die Bedienung des Akkumulators ist ebenso eingebunden. Der Ladezustand lässt sich mittels eines Schiebereglers einstellen. Dadurch ist es nun möglich, das Verhalten des Sytems bei verschiedenen Ladezuständen zu testen. Zudem kann der Ladezustand auch auf einen festen Wert gesetzt werden, um gegebenenfalls Tests mit einer konstanten Spannung durchzuführen. Zudem wurde ein Fahrsimulator aufgebaut. Dieser besteht aus einer gemeinsamen Simulation zwischen CANoe und MATLAB. Im Fahrsimulator muss der Fahrer eine vorgegebene Geschwindigkeitskurve abfahren und erhält dafür Punkte. Das vollständige Testen des Simulators war bislang nicht möglich, da die Pedalerie noch nicht eingebunden ist. Allerdings konnten Tests mit einer Momentvorgabe durchgeführt werden. Dieser Simulator kann als Training für die Fahrer dienen. Zusätzlich kann dieser bei öffentlichen Hochschulveranstaltungen als Vorführobjekt genutzt werden.

Es wäre denkbar, in die Visualisierung weitere Zustandsgrößen einzubinden. Dies ist abhängig vom jeweiligen Versuchsaufbau. Für den Fahrsimulator könnten weitere Geschwindigkeitskurven eingebunden werden, beispielsweise Messkurven von Rennveranstaltungen.

Literaturverzeichnis

[1] *http://www.chemie.de/lexikon/Luftdichte.html*. – abgerufen am 30. 01. 2015

[2] *http://www.periodensystem.info/elemente/*. – abgerufen am 15. 01. 2015

[3] ATMEL CORPORATION: *DVK90CAN1 Hardware User Guide*, 2007

[4] BRECHMANN, DZIEIA, HÖRNEMANN, HÜBSCHER, JAGLA, KLAUE, WICKERT: *Elektrotechnik: Tabellen Energieelektronik*. westermann, 2004

[5] DIRK BERSCHIN: *Entwicklung der Spannungsregelung eines Gleichstromgenerators als Ersatz für den Lithium-Ionen Akkumulator eines Elektrorennfahrzeuges*. Hochschule Mittweida, 2014

[6] FELIX ANDRE: *Modellierung einer Li-Ionen Batterie für Hybridfahrzeug-Simulationen*. Technische Universität Berlin, 2008

[7] FORMULA STUDENT: *2015 Formula SAE Rules*. 2014

[8] HEINZ UNBEHAUEN: *Regelungstechnik 1, Klassische Verfahren zur Analyse und Synthese linearer kontinuierlicher Regelsysteme, Fuzzy-Regelsysteme*. Vieweg+Teubner Verlag, 2008

[9] JAN LUNZE: *Regelungstechnik 1: Systemtheoretische Grundlagen, Analyse und Entwurf einschleifiger Regelungen*. Springer Vieweg, 2014

[10] KOKAM CO., LTD: *Cell Specification of SLPB 11543140H5*, June 2007

[11] KONRAD ETSCHBERGER: *Controller-Area-Network Grundlagen, Protokolle, Bausteine, Anwendungen*. Carl Hanser Verlag München Wien, 2000

[12] MANFRED MITSCHKE, HENNING WALLENTOWITZ: *Dynamik der Kraftfahrzeuge*. Springer, 2014

[13] MARC NESTLER: *Realitätsnahe Ansteuerung der Antriebskomponenten eines Elektrorennfahrzeuges (FSE) auf dem Motorenprüfstand, als Vorstufe eines Fahrsimulators*. Hochschule Mittweida, 2015

[14] PETER KURZWEIL, BERNHARD FRENZEL, FLORIAN GEBHARD: *Physik Formelsammlung*. Friedr. Vieweg & Sohn Verlag, 2008

[15] PIERBURG PUMP TECHNOLOGY: *Pierburg CWA GEN III*, 2014

[16] PROF. DIPL.-ING. PETER F. ORLOWSKI: *Prozessoptimierung ohne Reglerparameter Die FRt-Wurzelrekursion*. Technische Hochschule Mittelhessen

[17] REINER KORTHAUER: *Handbuch Lithium-Ionen-Batterien*. Springer Vieweg, 2013

[18] ROBERT BOSCH GMBH: *Technische Kundenunterlage INV-CON.E-2.3 and SMG-180.1.3*, April 2014

[19] SERGE ZACHER, MANFRED REUTER: *Regelungstechnik für Ingenieure: Analyse, Simulation und Entwurf von Regelkreisen*. Springer Vieweg, 2014

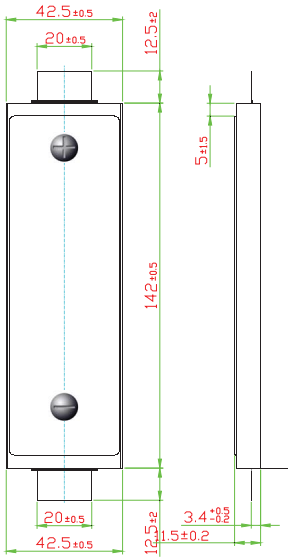
[20] TMM: *Auswertung Hector*, 2014

[21] WERNER ZIMMERMANN, RALF SCHMIDGALL: *Bussysteme in der Fahrzeugtechnik: Protokolle, Standards und Softwarearchitektur*. Springer Vieweg, 2014

Anhang A: Kokam-Zelle Datenblattauszug

Kokam™ Global Leader in Power Solution

Cell Specification of SLPB 11543140H5



Typical Capacity[1]		5.0 Ah
Nominal Voltage		3.7 V
Charge Condition	Max. Current	10.0 A
	Voltage	4.2V ± 0.03 V
Discharge Condition	Continuous Current	150.0 A
	Peak Current	250.0 A
	Cut-off Voltage	2.7 V
Cycle Life [CHA : 1.0C , DCH : 1.0C]		> 800 Cycles
Operating Temp.	Charge	0 ~ 40 ℃
	Discharge	-20 ~ 60 ℃
Dimension	Thickness (mm)	11.5 ± 0.2
	Width (mm)	42.5 ± 0.5
	Length (mm)	142.0 ± 0.5
Weight (g)		128.0 ± 4.0

1) Typical Capacity : 0.5C, 4.2 ~ 2.7V @25°C

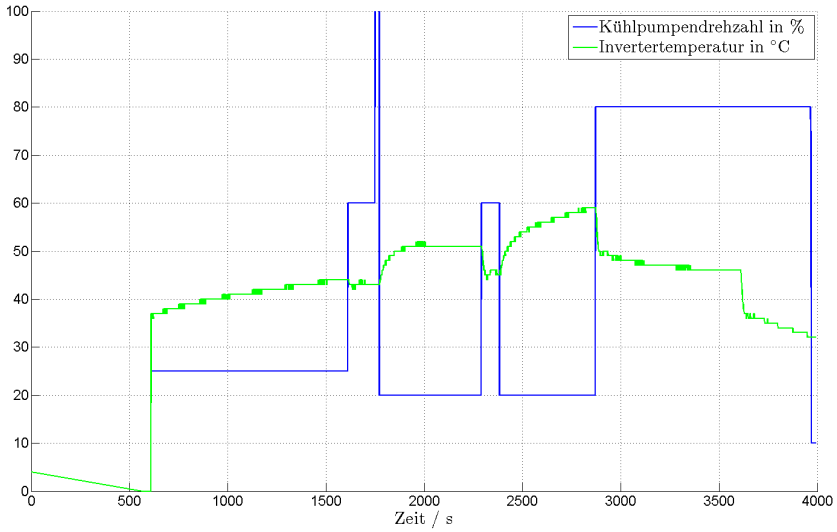
Anhang B: Script spline-Interpolation

```
%Startup-Script für Akkusimulation

%LookUp-Variable für LookUp-Tabelle vorbereiten
LookUp=[];

%Spline-Funktion aufrufen
[A]=spline_func_no_fig(C005extend);
%maximale Länge der Tabelle
maxLook=length(A);
%temp vorbereiten
temp=zeros(maxLook,1);
temp(:,1)=NaN;
%Kurve in temp schreiben
temp(1:length(A),1)=A(:,1);
%temp in LookUp schreiben
LookUp(:,1)=temp;

%Berechnung der spline-Funktion für weitere Kurven
[...]

%Übergabe an Modell-Workspace
assignin(get_param(bdroot,'modelworkspace'),'LookUp',LookUp);

%Berechnung der Filterkoeffizienten

%Transferfunktion
%   1
%  ---
%  s+1

h = tf(1,[1 1]); %Transfer-Function erstellen
hd = c2d(h, 0.001);

%%

function [N] = spline_func_no_fig(M)

%x-Werte auslesen
Cx=M(:,1);
Cx=Cx.';
%y-Werte auslesen
Cy=M(:,2);
Cy=Cy.';
%spline-Interpolation
cs=spline(Cx,Cy);
xx=0:0.01:max(Cx);
N=ppval(cs,xx);
N=N.';

end
```

Anhang C: Script Restwegberechnung

```
%Berechnung der Spannung
StartUp_Batt_U_calc
clc

%Systemvariablen
m=380;          %Masse in kg
g=9.81;         %Fallbeschleunigung in m/s²
n_r=0.03;       %Rollreibungszahl
cw=0.53;        %Strömungswiderstandskoeffizient
A=0.77;         %Fläche in m²
p=1.164;        %Dichte der Luft in kg/m³
n_Antrieb=0.85; %Antriebswirkungsgrad
Kap=15;         %Batteriekapazität in Ah
strom=[2.5 5 10 25 40 50 60 75 100 125 150];

%Berechnung der Leistungs-Geschwindigkeits-Kennlinie
v=1:0.01:70;
P_rad=v.*(m*g*n_r+cw*A*0.5*p*v.*v);

%Berechnung der Werte
delta=25;
for i=1:1:length(strom)
    value=strom(i)*U(i)*100*n_Antrieb;
    ind=find((P_rad>(value-delta)) & (P_rad<(value+delta)));
    erg((i),1)=strom(i);                    %Strom in A
    erg((i),2)=value;                       %Leistung in W
    erg((i),3)=3600/(strom(i)/Kap);         %Gesamtfahrzeit in s
    erg((i),4)=mean(ind)/100;               %v in m/s
    erg((i),5)=erg((i),4)*3.6;              %v in km/h
    erg((i),6)=erg((i),4)*erg((i),3);       %s in m
    erg((i),7)=erg((i),2)*erg((i),3)/3600;  %W in Wh
end

%csv-File anlegen
dlmwrite('I_v_Tabelle.csv',...
    ['I[A],P[W],t[s],v[m/s],v[km/h],s[m],W[Wh]'],'delimiter','');
dlmwrite('I_v_Tabelle.csv', erg,'-append','delimiter',',');

%löschen der Variablen aus dem Workspace
clear m
clear g
clear n_r
clear cw
clear A
clear p
clear n_Antrieb
clear U
clear Kap
clear strom
clear v
clear P_rad
clear delta
clear i
```

Anhang D: Messung der Sprungantwort des Kühlsystems



Anhang E: Programmierung des Steuergerätes

An dieser Stelle wird die modellbasierte Programmierung des Steuergerätes anhand eines Beispiels erläutert. Diese Anleitung soll nur als Leitfaden dienen, eine detailierte Beschreibung ist in dem Dokument *INTECRIO V4.4 GettingStarted.pdf* zu finden.

E.1 Verwendete Hardware

Für dieses Beispiel wurde folgende Hardware genutzt:

ES910: Ein Steuergerät der Firma ETAS. Dieses Steuergerät besitzt mehrere CAN- und LIN-Schnittstellen. Die Schnittstelle zum PC erfolgt über LAN. Durch das ES921-Modul wird das Steuergerät um zwei weitere CAN-Schnittstellen erweitert.

ES930: Eine Messbox der Firma ETAS. Als Erweiterung zur ES910 besitzt diese Thermoelementeingänge, digitale Eingänge, analoge Eingänge, Spannungsversorgungsausgänge, analoge Ausgänge, digitale Ausgänge sowie MOSFET-Halbbrückenschalter. Die ES930 ist für Mess- und Prüfaufgaben gedacht. Die Verbindung zur ES910 erfolgt über DaisyChain, eine ethernetbasierte Kommunikationsverbindung.

E.2 Verwendete Software

Die folgende Software ist zur Programmierung notwendig. Für jede Softwarekomponente ist eine gültige Lizenz notwendig.

HSP-UpdateTool (V 10.6.0): Ein ETAS-Tool zum Updaten und Konfigurieren der ETAS-Hardware.

INTECRIO (V 4.4.1.1028): Dieses Tool stellt aus Modulen (beispielsweise aus MATLAB-Simulationen erzeugter C-Code) ein Betriebssystem für das Steuergerät zusammen.

Daisy Chain Configurator (V 1.4.3 Release): Die Hardwarekette (DaisyChain), bestehend aus Steuergerät und Messmodulen, wird durch dieses Tool zusammengestellt und eingerichtet.

MATLAB Simulink (V R2013b (8.2.0.701) 32bit): Ein Simulationstool der Firma MathWorks. Durch den Simulink Coder kann in Verbindung mit dem erpt-grt-Target aus einem Simulationsmodell ein in Intecrio nutzbares Softwaremodul erstellt werden.

Experiment Environment (V 3.4.1.80 /1): Dieses Tool wird dazu genutzt das Steuergerät und die Programmierung zu testen.

E.3 Schritt 1: Hardware-Suche

Vor der Programmierung des Steuergerätes muss die Verbindung zur Hardware überprüft werden. Die Hardwarekomponenten müssen hierfür eingeschaltet sein und über eine Ethernet-Verbindung (100 $\frac{\text{MBits}}{\text{s}}$, fest eingestellt) mit dem PC verbunden sein. Das Programm HSP-Update-Tool muss geöffnet werden. Über *Funktionen -> Hardware suchen* wird nach verbundener Hardware gesucht (Abbildung E.1). Verbundene Hardware wird im linken Fenster angezeigt. An dieser Stelle kann nun ein Update der Gerätefirmware erfolgen. Zudem können mittels Rechtsklick auf *ES910.3* unter *System Konfiguration...* verschiedene Grundeinstellungen des Steuergerätes verändert werden.

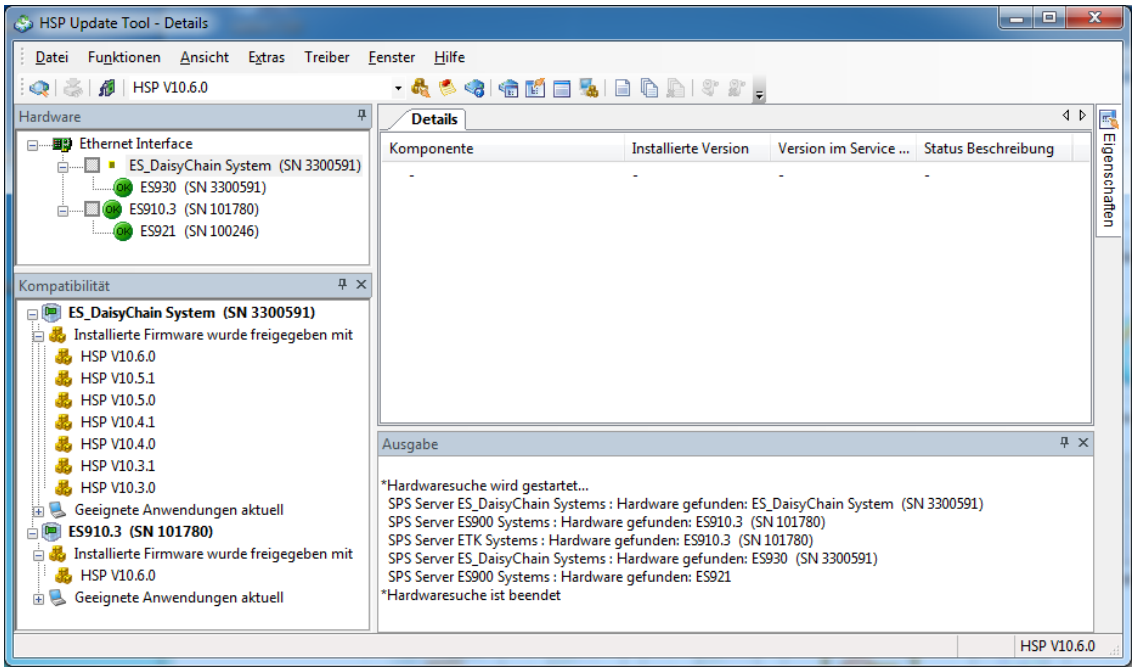


Abbildung E.1: Schritt 1: HSP Update Tool

E.4 Schritt 2: Hardware-Konfiguration

In INTECRIO muss nun ein neuer Workspace angelegt werden. Durch das Anlegen eines Workspace werden vier Ordner in INTECRIO angelegt (Abbildung E.2). Der Ordner *Hardware* beinhaltet den Unterordner *Hardware Systems*. In diesem Ordner wird die Gerätekonfiguration angelegt. Mittels *Add Hardware System...* kann nun ein *ES900 System* hinzugefügt werden. In die *ES900* muss nun mittels *Insert Target* ein *ES910 (E-Target)* eingefügt werden. Über das Kontextmenü können nun mittels *Insert...* die einzelnen Hardwarekomponenten, wie beispielsweise CAN-Controller, LIN-Controller oder DaisyChain-Module, eingefügt werden.

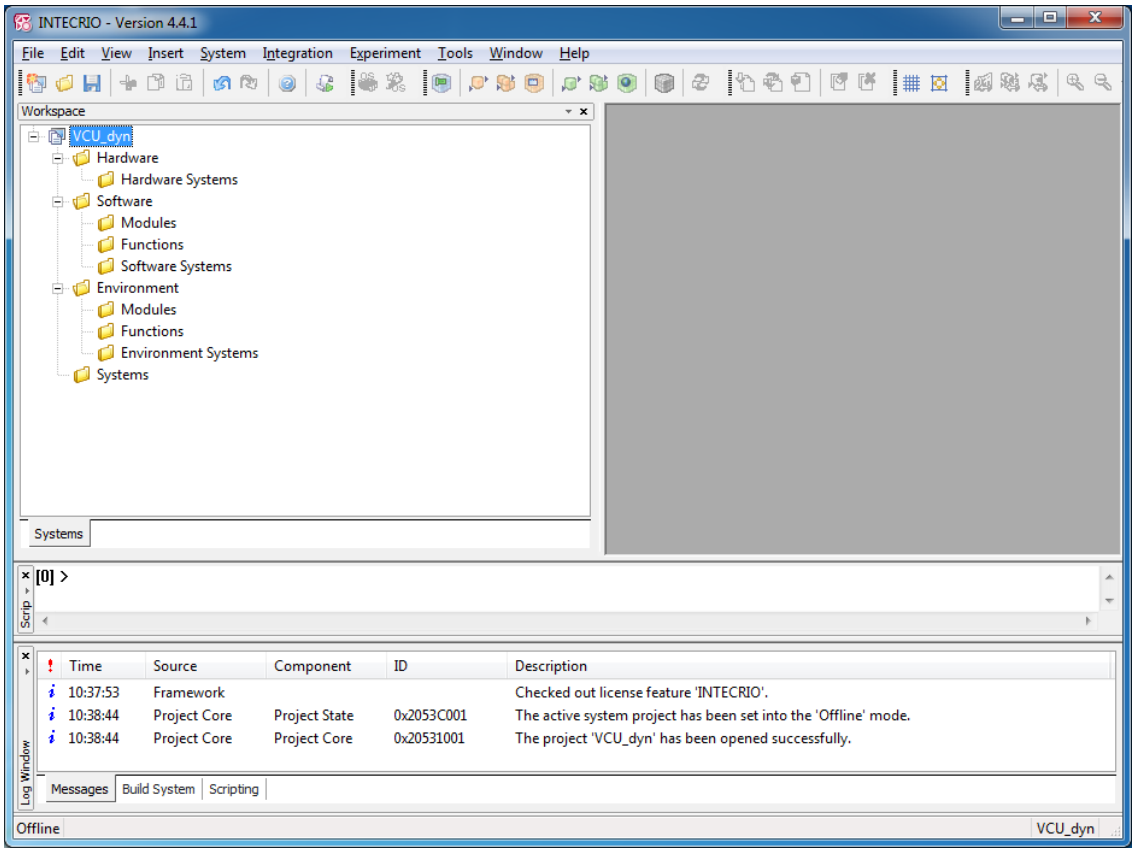


Abbildung E.2: Schritt 2: INTECRIO

Die Einbindung der ES930 erfolgt über das Programm Daisychain-Configuration. Dieses kann aus INTECRIO heraus aufgerufen werden. Dazu muss in die *ES910* ein *Daisychain*-Modul eingebunden werden. Mittels *Import Daisychain Configuration* kann eine neue Konfiguration (*Create new configuration*) erstellt werden. Die Konfiguration sollte im Projektordner abgelegt werden. Im Hardwarekonfigurationstool (Abbildung E.3) muss nun mittels *Hardware suchen* die vorhandene Hardware gesucht werden. An dieser Stelle ist es möglich, die Ein- und Ausgänge der ES930 zu konfigurieren. Mittels *Modulkombination initialisieren* muss die Konfiguration auf die ES930 geladen werden. Durch *Messung starten* kann eine Messung vorgenommen werden, in welcher die realen Eingangswerte überprüft werden können. Nach dem Initialisieren muss die Konfiguration gespeichert werden. In INTECRIO werden nun die Hardwarekanäle angezeigt (Abbildung E.4).

Ein CAN-Kanal wird eingebunden, indem in die ES930 ein CAN-Controller eingebunden wird. In diesen muss nun ein CAN I/O eingebunden werden. In dem CAN I/O kann nun mittels *Import CANdb...* eine CAN-Datenbank geladen werden[15]. Im Fenster *Node Selection* muss nun der Knoten ausgewählt werden, welcher das Steuergerät repräsentieren soll. In diesem Beispiel wurden die CAN-Datenbank aus der Teststand-Konfiguration

[15] Das Steuergerät hält sich strikt an die hinterlegten Maximal- und Minimalwerte der Signale. Liegen Initialisierungswerte außerhalb dieser Grenzen, wird der jeweilige Grenzwert und nicht der eigentliche Initialisierungswert durch das Steuergerät gesendet!

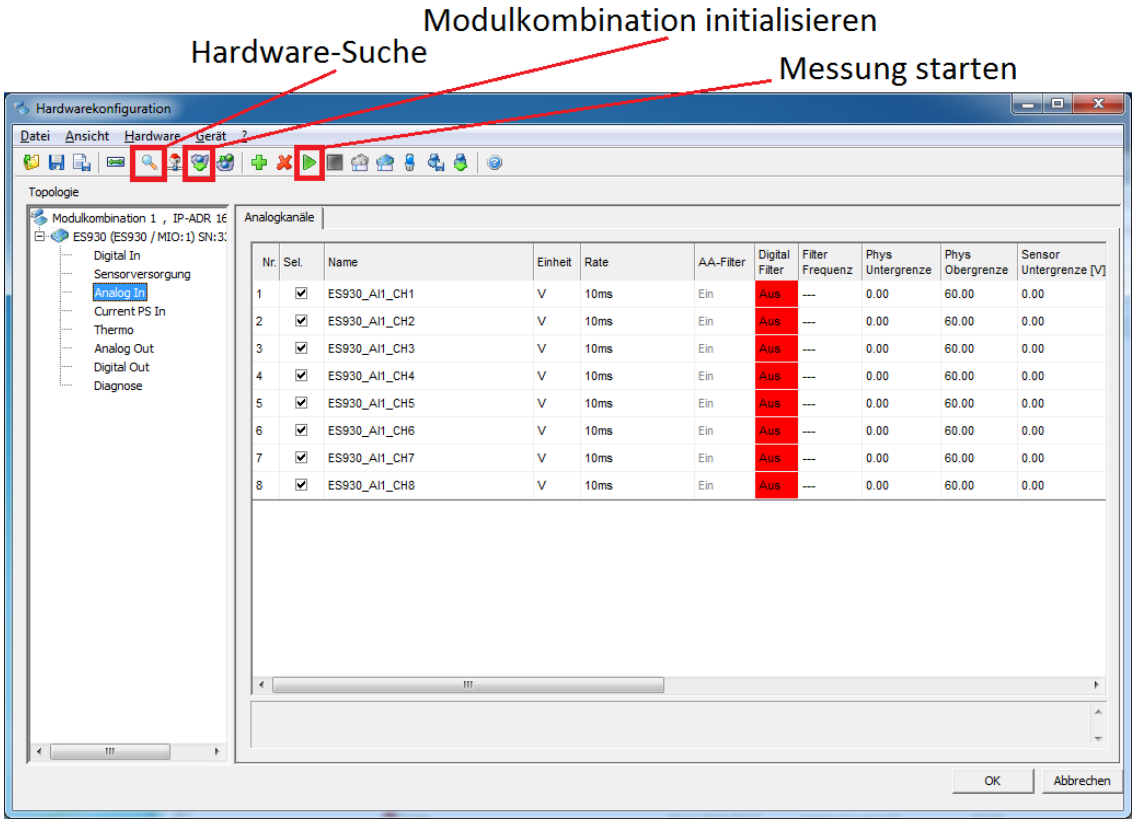


Abbildung E.3: Schritt 2: Daisychain-Konfiguration

aus CANoe geladen. Das Steuergerät repäsentiert den Knoten VCU. Dadurch werden nun alle CAN-Signale, welche VCU als Empfänger eingetragen haben als Input-Signale dargestellt. Alle Signale, welche vom Knoten VCU gesendet werden, sind nun als Outputs verfügbar. Alle Nachrichten anderer Knoten sind nicht verfügbar. Abbildung E.4 zeigt nun eine Hardwarekonfiguration mit zwei CAN-Kanälen, einer ES930 und einer LIN-Schnittstelle. Die Hardwarekonfiguration der ES900[16] muss nun exportiert werden. Das exportierte *.hwx-File wird im nächsten Schritt benötigt.

E.5 Schritt 3: Simulink-Simulation

Für die Programmierung des Steuergerätes werden MATLAB-Simulink-Modelle genutzt. Abbildung E.5 zeigt ein solches Modell. In dem Sub-System *Restwegberechnung* befindet sich eine MATLAB-Simulation (Abbildung 4.19). Für die Programmierung des Steuergerätes muss zunächst der Block *ETAS ES900 (ES910)* zur Simulation hinzugefügt werden. Dieser muss aus der Simulink-Bibliothek *ETAS Rapid Prototyping / Configuration* entnommen werden. Durch Öffnen des Blockes öffnet sich der Hardware-Konfigurator. In die ES900 muss nun die in Schritt 2 exportierte Hardwarekonfiguration (*.hwx) geladen werden[17]. Eine Kopie des Konfigurationsfiles wird automatisch im

[16] Wichtig: Es muss die Konfiguration der ES900, nicht der ES910 (E-Target) exportiert werden!
[17] Wichtig: ES900, nicht ES910(E-Target)

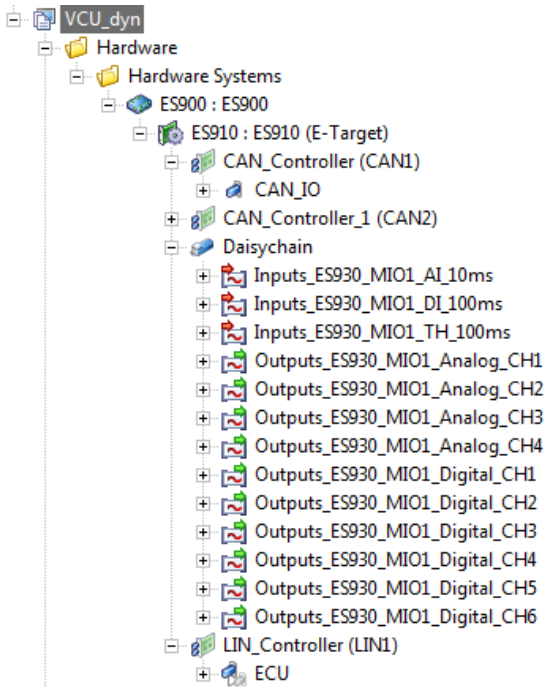


Abbildung E.4: Schritt 2: Hardwarekonfiguration

Workspace hinterlegt. Wird im Verlauf des Projektes die Hardwarekonfiguration in INTECRIO geändert, so muss die Konfiguration neu geladen werden. Zur Kommunikation mit der Hardware sind nun die Blöcke *Hardware Signal Group Receive* und *Hardware Signal Group Send* aus *ETAS Rapid Prototyping / Signal Routing* zu verwenden. Die Receive- / Send-Blöcke müssen nun den gewünschten Signalen beziehungsweise Hardwarekanälen zugeordnet werden. Die Signale sind bereits nach den Umrechnungsformeln in der CAN-Datenbank beziehungsweise der Hardwarekonfiguration skaliert.

Die fertige Simulation muss nun mit dem *erpt_grt.tlc* -Target übersetzt werden. Dieses Target erstellt automatisch die für INTECRIO nötige Scoop-IX-Schnittstellendatei. Wichtig ist hierbei die eingetragene Simulationsschrittweite. Diese wird später zur Zuordnung in den Tasks des Betriebssystems genutzt. Der Übersetzungsvorgang erstellt einen neuen Ordner im Workspace. Dieser Ordner heißt <Modellname>_erpt_grt_rtw und beinhaltet unter anderem die Scoop-IX-Schnittstellendatei <Modellname>.six, welche im nächsten Schritt benötigt wird.

E.6 Schritt 4: Softwaresystem

In INTECRIO können übersetzte Simulink-Modelle eingebunden werden. Hierfür muss in dem Workspace-Ordner *Software* unter *Modules* im Kontextmenü *Import Module...* gewählt werden. Dazu muss das in Schritt 3 erzeugte *.six-Datei ausgewählt werden. Das importierte Modul beinhaltet nun die Ein- und Ausgänge, welche im Modell (Abbildung E.5) verwendet wurden. An dieser Stelle können mehrere Modelle eingebunden

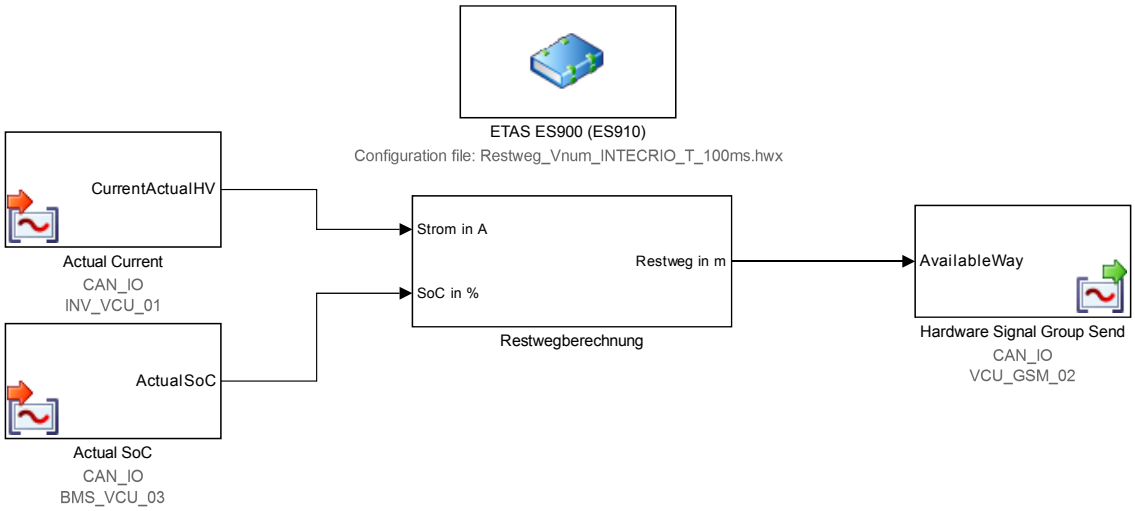


Abbildung E.5: Schritt 3: MATLAB-Simulation

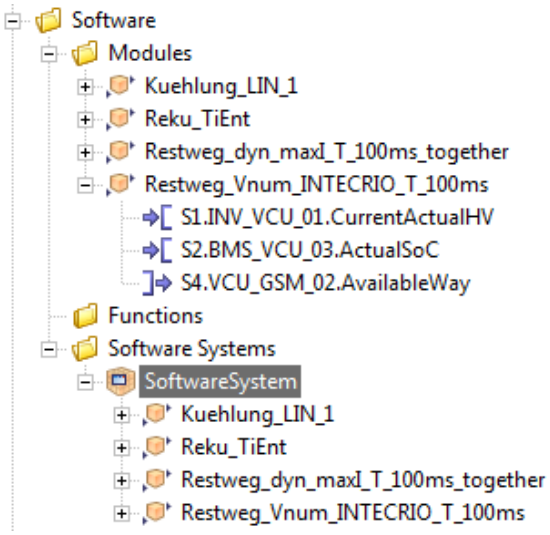


Abbildung E.6: Schritt 4: Software-Module

werden. Nun muss im Ordner *Software Systems* ein Software System mittels *Create* eingefügt werden. Im Kontextmenü unter *Modules...* können nun Module hinzugefügt werden. Abbildung E.6 zeigt nun die Module und das Softwaresystem mit eingebundenen Modulen. Durch Öffnen des Softwaresystems erscheint ein Fenster, in welchem nun die Module (Abbildung E.7, orange) mit *Scalar In Ports* und *Scalar Out Ports* (Abbildung E.7, grün) verbunden werden müssen. Besitzen die verschiedenen Module gleiche Eingänge, so können diese hier bereits mit einem einzigen Port verbunden werden.

E.7 Schritt 5: Operating System

Sobald die Hardwarekonfiguration und das Softwaresystem fertiggestellt ist, kann in INTECRIO nun ein Betriebssystem (Operating System, OS) für das Steuergerät erstellt werden. Hierfür muss im Workspaceordner *Systems* mittels *Create System...* ein Sys-

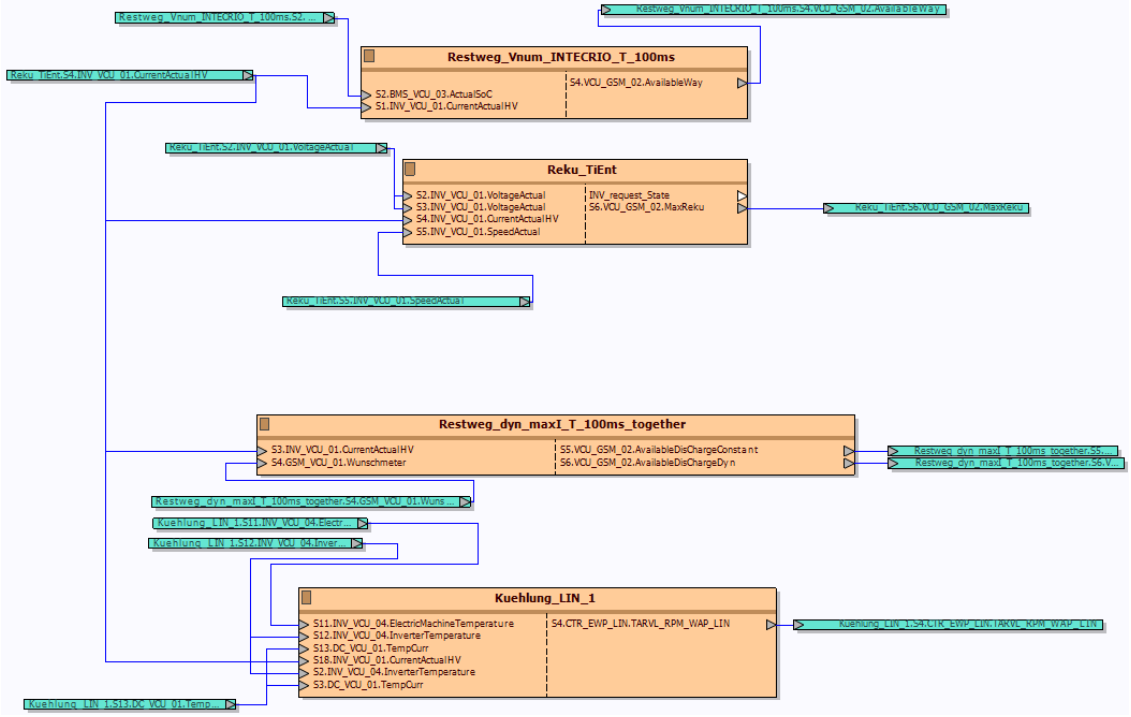


Abbildung E.7: Schritt 4: Softwaresystem

tem erstellt werden. Mittels *Add Hardware System...* aus dem Kontextmenü muss nun das erstellte Hardwaresystem (Schritt 2) eingefügt werden. Des Weiteren muss mittels *Add Software System...* das Softwaresystem hinzugefügt werden. Es kann nur ein Hardwaresystem sowie ein Softwaresystem hinzugefügt werden. Abbildung E.8 auf Seite 84 zeigt das „VCU_dyn“-System mit eingebundenem Software System und Hardwaresystem. Durch Öffnen des Systems „VCU_dyn“ wird im Editorfenster das Softwaresystem dargestellt (Abbildung E.9, gelb). Aus der Ordneransicht (Abbildung E.8) können nun die einzelnen Ein- und Ausgänge zum System hinzugefügt werden (Abbildung E.9, blau) und mit dem Softwaresystem verbunden werden. Es ist auch möglich, Hardwareeingänge direkt mit Ausgängen zu verbinden. Nachdem diese Verknüpfungen eingefügt wurden, müssen nun die Tasks angelegt werden. Dafür muss für das System die *OS-Configuration* aufgerufen werden. Über die Funktion *OS Auto mapping* könnte nun automatisch eine Konfiguration erstellt werden. Dies ist allerdings nicht zu empfehlen, da hierdurch alle in der CAN-Datenbank verzeichneten Signale eingetragen werden, auch wenn diese nicht durch das Softwaresystem beschrieben werden. Es empfiehlt sich daher, die Timer-Tasks manuell anzulegen. Ein Timer-Task (Abbildung E.10) benötigt eine Priorität und eine Periodendauer. Für die Priorität gilt: Je höher der Wert umso höher die Priorität. Die Periodendauer wird in Sekunden eingegeben. In die Timer-Tasks müssen nun die CAN-Sende- und Empfangs-Signale entsprechend Zykluszeit laut CAN-Datenbank eingetragen werden. Ein Simulationsmodul besteht aus drei Teilen, dem Startprozess (erpt_grt_Start_<Name>), dem Abschlussprozess (erpt_grt_Terminate_<Name>) und dem Schrittprozess (erpt_grt_OneStep_<Name>). Alle Start-Prozesse müssen in den Init-Task eingetragen werden. Alle Abschluss-

prozesse müssen in den Exit-Task übertragen werden. Die Schrittprozesse müssen in die entsprechenden Timer-Tasks im Unterordner *Action* eingetragen werden. Ebenso müssen die Hardware-Prozesse eingetragen werden. Das Projekt muss nun mittels *Build* übersetzt werden.

E.8 Schritt 6: Beschreiben des Steuergerätes

Nach dem erfolgreichen Übersetzen mittels *Build* kann nun aus dem Kontextmenü am System *Open Experiment...* aufgerufen werden. Das Programm Experiment Environment öffnet sich selbstständig. In diesem Programm können nun Ein- und Ausgänge sowie CAN-Signale des Steuergerätes visualisiert werden. Zudem kann das Steuergerät programmiert werden. Hierfür gibt es zwei Möglichkeiten: *Flash* und *Download*. Wird das Steuergerät durch die Funktion *Flash* programmiert, so wird das Programm dauerhaft einprogrammiert, sodass dieses auch nach einem Neustart läuft. In diesem Fall muss zur Visualisierung die *Connect*-Funktion aktiviert werden. Wird das Steuergerät jedoch mittels *Download* programmiert, so wird das Programm nur temporär ausgeführt. Nach einem Neustart ist dieses nicht mehr verfügbar. In diesem Fall ist das Steuergerät allerdings automatisch im *Connect*-Modus. Um eine Messung zu starten muss *Start Simulation* und *Start Measurement* aktiviert werden. Abbildung E.11 zeigt das Programm Experiment Environment.

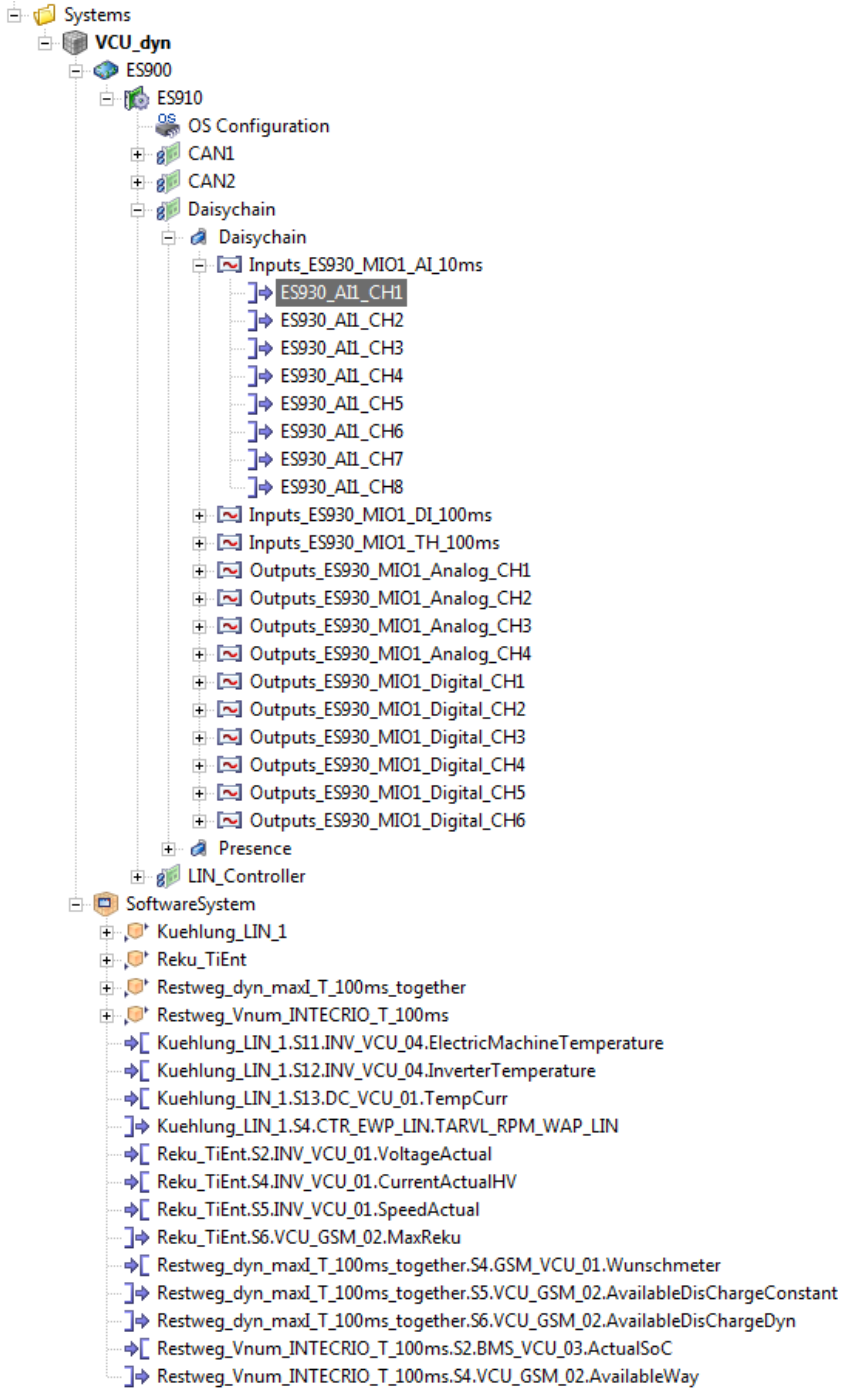


Abbildung E.8: Schritt 5: System VCU_dyn Ordner

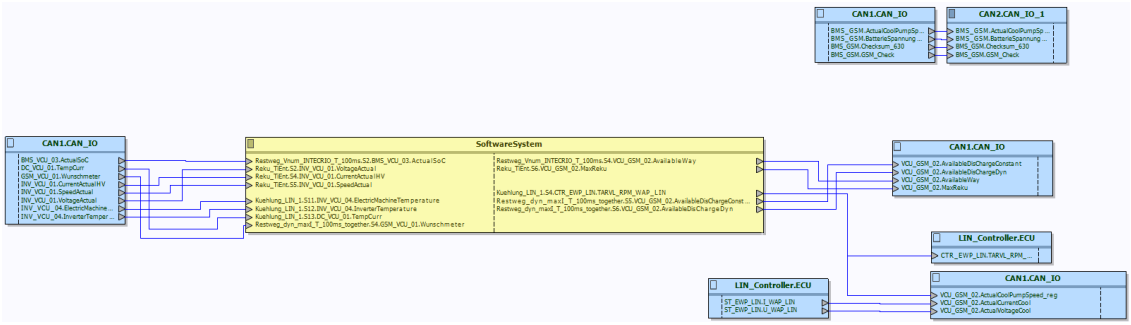


Abbildung E.9: Schritt 5: System VCU_dyn Verknüpfung

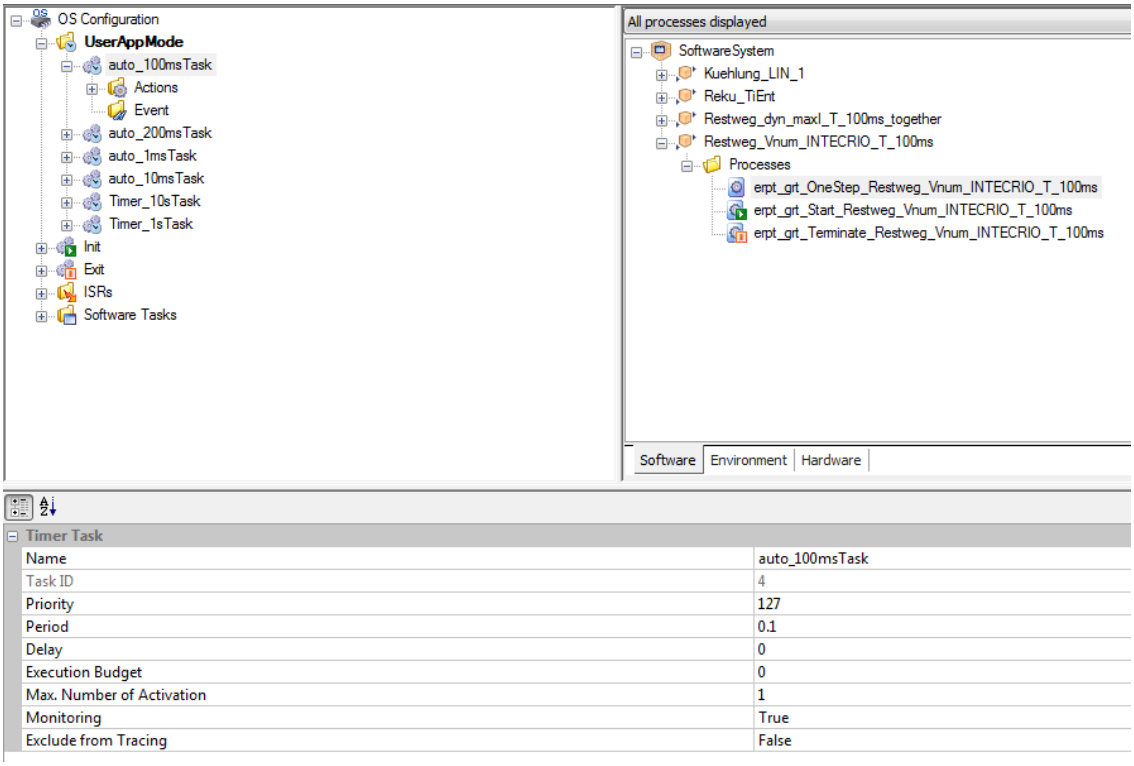


Abbildung E.10: Schritt 5: System VCU_dyn Tasks

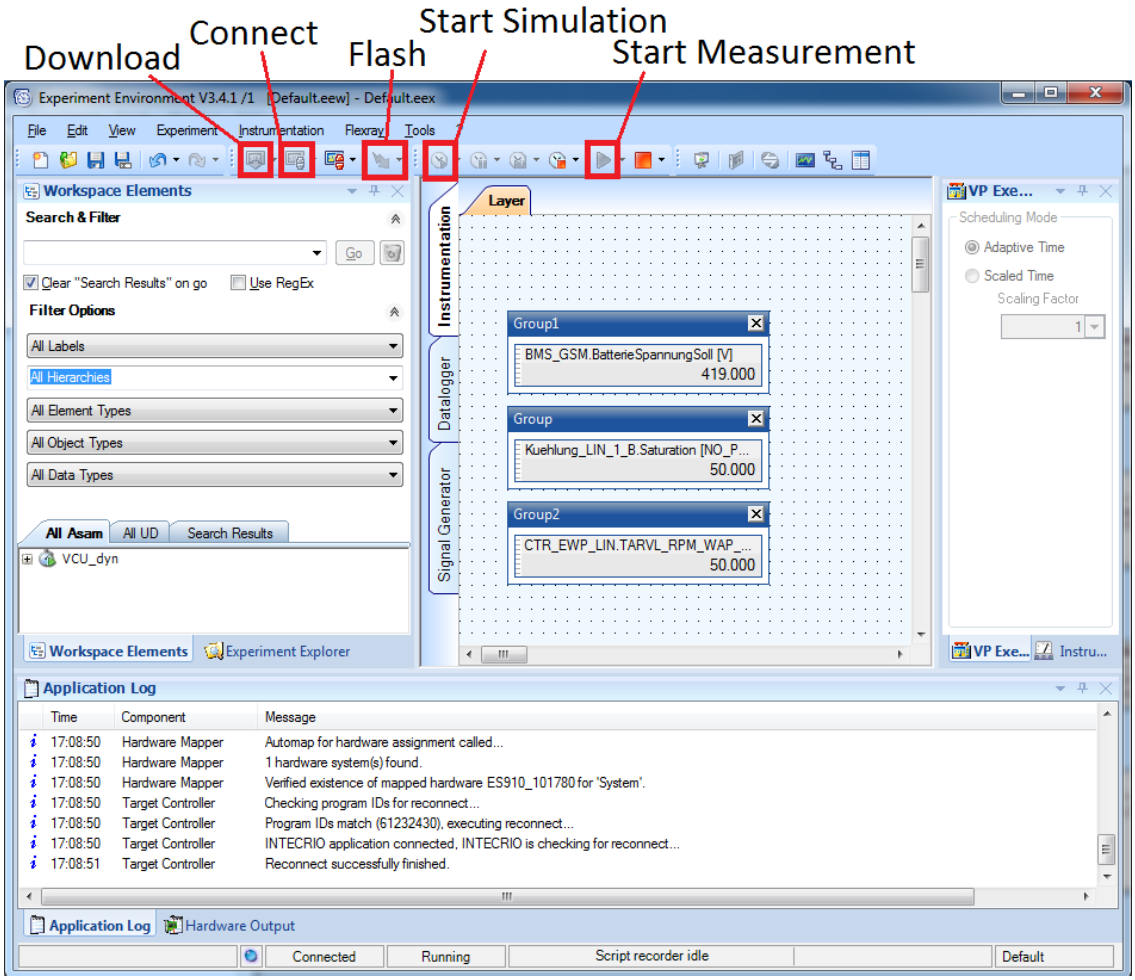


Abbildung E.11: Schritt 6: Experiment Environment

Anhang F: Script Fahrsimulator

```
function [mean_abw, Pkt, Stop_Signal, soll_v, prozent] =
                                    Simulator(reset, time, ist, soll, end_x)
%#codegen
persistent i
if isempty(i)
    i=plot(0);
end
persistent end_t
if isempty(end_t)
    end_t=180;
end
%Anlegen der statischen Variable für alten Geschwindigkeitswert
persistent ist_alt
if isempty(ist_alt)
    ist_alt=ist;
end
%Anlegen der Punkt-Variable
persistent Punkte
if isempty(Punkte)
    Punkte=0;
end
%Anlegen der Abweichungsvariable
persistent Abw
if isempty(Abw)
    Abw=zeros(end_t,1);
    %Abw=zeros(1180,1);
end
persistent soll_st
if isempty(soll_st)
    soll_st=zeros(end_t,1);
end
persistent proz
if isempty(proz)
    proz=0;
end
%Bearbeitung nur bei vollen Sekunden
time_h=time*10;
time=round(time);
time_h=round(time_h);
if(mod(time_h,10)==0)
    %Bearbeitung nur wenn t>0, da time-1 bei t=0 unmöglich
    if((time>=1)&&(time<=end_t))
        i=plot([time-1 time],[ist_alt ist]);
        set(i,'Color','blue');
        %Wertspeicherung für nächsten Durchlauf
        ist_alt=ist;
        %Abweichung bestimmen
        Abw(time,1)=abs(soll-ist);
        %falls prozentuale Abweichung unter 3 km/h
        soll_st(time)=soll;
        if (Abw(time,1)<=3)
            Punkte=Punkte+2;
            %falls prozentuale Abweichung unter 1 km/h
            if (Abw(time,1)<=1)
                Punkte=Punkte+3;
```

```
            end
        end
        proz=time/end_t*100;
    end
end

if (time>=1)
    %durchschnittliche Abweichung errechnen
    %disp(mean(Abw))
    if(time<end_t)
        mean_abw=mean(Abw(1:time));
    else
        mean_abw=mean(Abw(1:end_t));
    end
    %aktuelle Punktzahl ausgeben
    %disp(Punkte)
    Pkt=Punkte;
else
    mean_abw=0;
    Pkt=0;
end

soll_v=soll;
prozent=proz;

Stop_Signal=0;
if (time>end_t)
    Stop_Signal=1;
end

%falls Neustart
if (reset==1)
    %Variablen nullen
    Punkte=0;
    Abw=zeros(end_t,1);
    Stop_Signal=0;
    %set(i,'Visible','off');
    close
    %Sollkurve neu plotten
    figure(1);
    set(gcf,'units','normalized','position',[0,0,1,1],'MenuBar','none');
    hold on;
    grid on;
    s=plot(soll_st);
    xlabel('Zeit (Sekunden)');
    ylabel('Geschwindigkeit (km/h)');
    set(s,'Color','red');
    proz=0;
end
```

Anhang G: Berechnung der Wegstrecke

```
%% pre-plot
clc
close all

%% Daten plotten

time_value=Time10us/100000;
zeiteinheit='Zeit / s';
singleplot(1,time_value,(InverterINV_VCU_01SpeedActual)*(-1),...
    'Geschwindigkeit', zeiteinheit,'auto','Drehzahl / U/min',[0 2500]);

%% Berechnung

disp('Test 1:');
disp(' ');

disp('gefahrene Meter real');
mean_n=mean((InverterINV_VCU_01SpeedActual)*(-1));
disp( ['Mittelwert: ',num2str(mean_n),' U/min'] );
sum_n=mean_n*Time10us(end)/6000000;
                            %6000000=60*1000*100 -> min_sec*sec_ms*ms_10us
disp( ['Umdrehungen: ',num2str(sum_n),' U'] );
getriebe=6;                                                 %1:6
radius=250;                                                 %mm
weg=(sum_n/getriebe)*2*pi*radius/1000;                      %in meter
disp( ['zurückgelegte Meter: ',num2str(weg),' Meter'] );

disp(' ');
disp('gefahrene Meter laut Anzeige');
anzeige=AvailableWay(1)-AvailableWay(end);
disp( ['zurückgelegte Meter: ',num2str(anzeige),' Meter'] );

disp(' ');

disp( ['Korrekturfaktor: ',num2str(weg/anzeige),' '] );

disp(' ');

delta_SoC=BMSBMS_VCU_03ActualSoC(1)-BMSBMS_VCU_03ActualSoC(end);
disp( ['delta_Soc: ',num2str(delta_SoC),' %'] );
energie=delta_SoC*5.5/100;
disp( ['umgesetzte  Energie: ',num2str(energie),' Ah'] );
disp( ['theoretisch maxMeter: ',num2str(weg*100/delta_SoC),' Meter'] );
disp( [' gewünschte maxMeter: ',num2str(mean(Wunschmeter)),' Meter'] );
disp( ['Korrekturfaktor: ',num2str((...
    weg*100/delta_SoC)/(mean(Wunschmeter))),' '] );

%% post-plot

clear time_value zeiteinheit mean_n sum_n getriebe radius weg anzeige
clear delta_SoC energie
```

Auswertungsergebnisse:

```
Test 1:

gefahrene Meter real
Mittelwert: 1458.4628 U/min
Umdrehungen: 19147.4131 U
zurückgelegte Meter: 5012.781 Meter

gefahrene Meter laut Anzeige
zurückgelegte Meter: 6200 Meter

Korrekturfaktor: 0.80851

delta_Soc: 14.4 %
umgesetzte  Energie: 0.792 Ah
theoretisch maxMeter: 34810.9794 Meter
 gewünschte maxMeter: 38000 Meter
Korrekturfaktor: 0.91608

Test 2:

gefahrene Meter real
Mittelwert: 1262.0715 U/min
Umdrehungen: 22281.2253 U
zurückgelegte Meter: 5833.2111 Meter

gefahrene Meter laut Anzeige
zurückgelegte Meter: 5100 Meter

Korrekturfaktor: 1.1438

delta_Soc: 15.6 %
umgesetzte  Energie: 0.858 Ah
theoretisch maxMeter: 37392.3792 Meter
 gewünschte maxMeter: 38000 Meter
Korrekturfaktor: 0.98401
```

Erklärung

Hiermit erkläre ich, dass ich meine Arbeit selbstständig verfasst, keine anderen als die angegebenen Quellen und Hilfsmittel benutzt und die Arbeit noch nicht anderweitig für Prüfungszwecke vorgelegt habe.

Stellen, die wörtlich oder sinngemäß aus Quellen entnommen wurden, sind als solche kenntlich gemacht.

Mittweida, 5. März 2015