Zusammenfassung der Arbeit

Fachbereich:	Elektrotechnik und Informatik
Studiengang:	Energiesysteme und Automation
Thema:	Entwicklung einer Simulationsumgebung für den Test von Steuerungsalgorithmen für Energieflüsse in Mikronetzen.
Zusammenfassung:	Ein mögliches Konzept zur effizienten Nutzung lokal produzierter Energie (Photovoltaik) sind sub-autarke Mikronetze, die auf kleinster Netzebene lokale Systemkomponenten (einschließlich Erzeuger, Speicher und Lasten) miteinander verknüpfen. Diese Netze entnehmen oder beliefern bedarfsgerecht Energie über den angebundenen Energieverbund. Ziel ist dabei, die ausgewogene Lastverteilung und Stabilität des subautarken Systems sowie des gekoppelten Energieversorgungssystems zu erreichen. Im Rahmen des carpeDIEM Projekts wird eine Technologie entwickelt, die es ermöglicht Mikronetze hinsichtlich verschiedenen Rahmenbedingungen bestmöglich zu steuern. Die Entwicklung solcher Steuerungsalgorithmen erfordert eine geeignete Testumgebung. Dafür wurde zunächst das elektrische Energienetz der Ortschaft Dörpum in PowerFactory implementiert. Im Rahmen der Bachelorarbeit soll eine Schnittstelle zwischen PowerFactory und einem Raspberry Pi (Einplatinencomputer) entwickelt werden, damit der Steuerungsalgorithmus getestet werden kann. Eine Prüfeinrichtung dient dazu das transiente Verhalten von Photovoltaikanlagen zu modellieren. Die Messdaten dieser Prüfeinrichtung sollen ebenfalls in die neue Testumgebung integriert werden.
Verfasser:	Christian Ziegelmann und Timo Helsper
Betreuender Professor:	Prof. Dr. Carsten Lüders
WS / SS:	WS 2018/2019

Abstract of Thesis

Department:	Electrical Engineering and Computer Science
University course:	Energy Systems and Automation Engineering
Subject:	Development of a simulation environment for the test of control algorithms for energy flows in microgrids.
Abstract:	A possible concept for the efficient use of locally produced energy (photovoltaics) are sub-self-sufficient microgrids that connect local system components (including generators, storage and loads) at the smallest grid level. These networks remove or supply energy as required via the connected energy network. The aim is to achieve the balanced load distribution and stability of the sub-self-sufficient system and the coupled energy supply grid. As part of the carpeDIEM project, a technology is being developed that enables the best possible control of microgrids with regard to various framework conditions. The development of such control algorithms requires a suitable test environment. For this purpose, the electrical energy network of the village Dörpum was first implemented in PowerFactory. The bachelor thesis aims to develop an interface between PowerFactory and a Raspberry Pi (single-board computer) so that the control algorithm can be tested. A test facility serves to model the transient behavior of photovoltaic systems. The measurement data of this test facility should also be integrated into the new test environment.
Author:	Christian Ziegelmann and Timo Helsper
Attending professor:	Prof. Dr. Carsten Lüders
WS / SS:	WS 2018/2019

Inhaltsverzeichnis

1 Einleitung – Christian Ziegelmann

1.1 Ziel und Anforderung

In Deutschland hat jeder Bürger ein Grundrecht auf elektrische Energieversorgung. Das zeigt den Stellenwert den wir in unserer Gesellschaft diesem Verbrauchsgut beimessen. Um diese Versorgung gewährleisten zu können sind in Zukunft große Herausforderungen zu bewältigen. Der Wandel in der Energieproduktion erfordert eine Umstrukturierung der Netze und schnellere Kommunikation der Teilnehmer. In dieser Bachelorarbeit wird eine Simulationsumgebung für ein Stromnetz erstellt das technische Modernisierungen beinhaltet.
Dabei soll es möglich sein durch Regelung von Lasten eine Überlastung von Leitungen und anderen Systemkomponenten zu verhindern. Es wird mithilfe einer Software für Leitungsnetzsimulation eine Netzebene abgebildet. Über ein Netzwerk soll ein Algorithmus eingebunden werden dem es erlaubt ist Verbraucher zu Regeln. Diese Regelung soll zu einer optimierten Nutzung von Netzebenen führen um die Netzstabilität zu steigern. Im ersten Versuch wird das Ladeverhalten von Elektroautos mit einem Algorithmus simuliert. Dieser teilt die am Transformator zur Verfügung stehende Leistung unter den Stationen auf, die derzeit Fahrzeuge aufladen. In den Laborräumen der Technische Hochschule (TH)-Lübeck wurde der Aufbau entwickelt und Versuche ausgewertet.
Beweggrund der Forschung ist die Erweiterung von dezentralen Energie Management Systemen im Rahmen des Projektes carpeDIEM. Unser Stromnetz hat sich über die Jahrzehnte entwickelt. Durch die Energiewende und die damit verbundenen Anforderungen zur vermehrten Einspeisung erneuerbarer Energien, sind neue Techniken gefragt. Zunächst wird die Problemstellung anhand der aktuellen Netzsituation dargestellt. Im weiteren Verlauf wird eine Idee zur Besserung der aktuellen Situation vorgestellt. Der entwickelte Aufbau wird beschrieben und seine Umsetzung dokumentiert. Am Ende werden die Ergebnisse von ersten Versuchen dargestellt, und eine Einschätzung für die weitere Relevanz geben.

1.2 Entstehung von Versorgungsnetzen

Die Entdeckung des dynamoelektrischen Prinzips von Werner von Siemens und dessen Vorstellung 1867 war ein erster Schritt zur großflächigen Zurverfügungstellung elektrischer Energie für die Menschheit.
In den kommenden Jahren wurden durch die Erfindung von elektrisch betriebenen Leuchtmitteln bei den Menschen ein Bedürfnis nach jederzeit verfügbarer Beleuchten geweckt. Thomas Alva Edison startete im Jahre 1880 mit der Elektrifizierung von New York um diesem Bedürfnis nachzukommen. Parallel dazu hatte er kurz zuvor einen Kohledraht unter Vakuum in einer Glasglocke zu einer Glühlampe zusammengebaut.
Um der wachsenden Nachfrage gerecht werden zu können, mussten immer größere Strecken von dem Generator bis zum Verbraucher zurückgelegt werden. Das auf Gleichspannung basierende System von Edison kam auf Grund zu geringer Spannungen an seine Grenzen.
Nicola Tesla und George Westinghouse waren Verfechter des Wechselstroms und konnten mit Hilfe der Transformatoren ab 1885 die Wechselspannung auf ein höheres Spannungsniveau anheben. Damit ließen sich größere Distanzen der Energieübertragung bewältigen.
Michail Dolivo-Dobrowolski entwickelte Anfang der 1890er Jahre einen dreiphasigen Transformator. Damit war der bis heute gebräuchliche Drehstrom geboren.
1891 wurde erstmalig in einem dreiphasigen Leitungsnetz elektrischer Strom übertragen. Die Drehstromübertragung verlief von Lauffen nach Frankfurt über eine 176 km lange Strecke. Diese wurde mit einer Spannung von 15 kV betrieben. Ab 1900 versuchte man die Frequenz auf 50 Hz zu Normen. Heute haben über 4 Kontinente diese Frequenz angenommen.
In Europa ist ein System von 50Hz mit einer Dreiphasenspannung von 400V Effektivspannung beim Endverbraucher etabliert.

1.3 Energieversorgung

Durch die stärkere Industrialisierung und der Zunahme von Fließbandarbeit konnte der Konsum von elektrisch betriebenen Produkten stark ausgebaut werden.
Um den Bedarf an Energie decken zu können waren in Deutschland bereits 1911 mehr als 2.300 Einzelstandorte zur Produktion der Energie errichtet worden.
Zu Beginn der 70er Jahre kamen zunächst Verbraucher wie Waschmaschinen und andere Haushaltstechnik in die Privathäuser.
Mit dem gestiegenen Wohlstand und der Entwicklung des Mikroprozessors wird ein zunehmender Bedarf von Energie zur Versorgung von Rechenleistung und Kommunikationstechnologien benötigt.

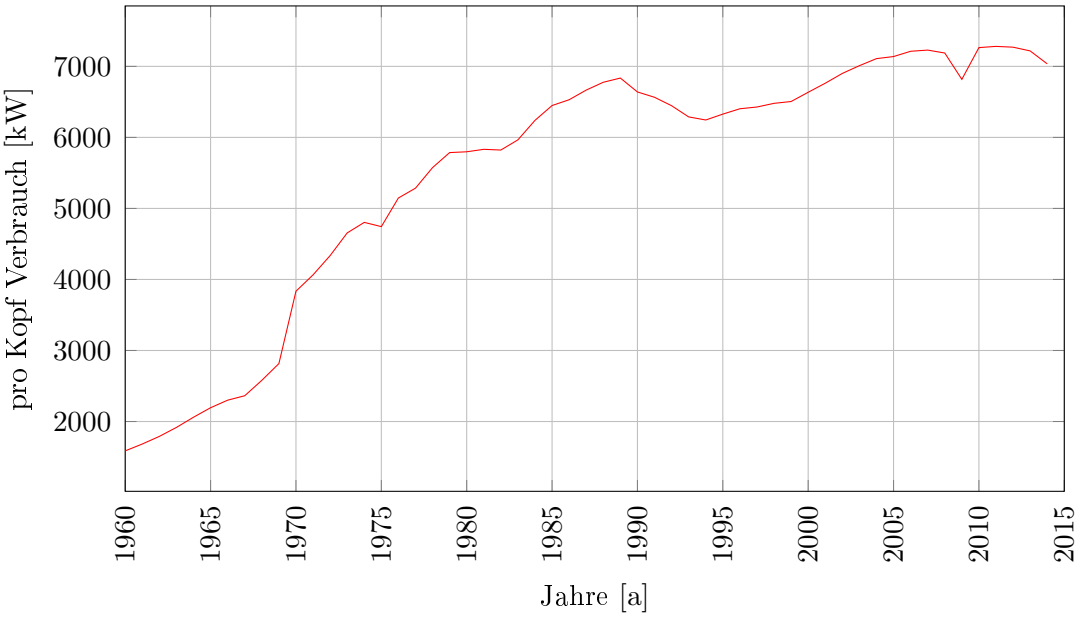


Abbildung 1.1: Elektrischer Energieverbrauch pro Kopf in Deutschland [25]

Um die Stromerzeugung immer wirtschaftlicher zu gestalten, wurden im Laufe der Elektrifizierung immer größere Kraftwerke gebaut. Dies hat sich bis zum Beginn der Energiewende nicht geändert.
Das hatte zur Folge, dass der Strom bis zu jedem Endverbraucher transportiert werden muss und die großen Trassen mit Höchstspannung den gesamten Strom mehrerer Ortschaften und Städte transportieren.
Aus diesem Grund haben Höchstspannungsleitungen eine Spannung von 380kV und werden bis in die Ortschaften runter transformiert über mehrere Ebenen bis auf 400V.

1.4 Netzebenen

Innerhalb Europas hat sich ein Netz aus vier Ebenen entwickelt.

Transportnetz

Das Transportnetz teilt sich in Deutschland in vier Zonen auf. Diese haben jeweils einen anderen Betreiber und haben sich stetig ausgeweitet. Die vier Übertragungsnetzbetreiber sind 50Hertz Transmission, Amprion, Tennet TSO und TransnetBW.
Das Transportnetz hat Höchstspannungen von 380kV und 220kV. Große Kraftwerke und sehr große Windparks speisen auf dieser Spannungsebene ein. Bis heute übertragen sie 60% der in Deutschland benötigten elektrischen Energie.

Hochspannungsnetz

Das Hochspannungsnetz hat eine Spannung von 110kV. Es bezieht seine Energie hauptsächlich aus dem Transportnetz aber auch von größeren Erzeugern. Es hat eine Versorgungsaufgabe für die unteren Spannungsebenen, sowie für große Industriebetriebe oder Einrichtungen mit enormen Energiebedarf mit eigener Trafostationen und Verteilnetzen.
Diese Netze werden von Vierteilnetzbetreibern betrieben und dienen einer groben Verteilung auf mittleren Distanzen.

Mittelspannungsnetz

Das Mittelspannungsnetz hat eine Spannung von 10kV oder 20kV. Sie sind den Hochspannungsnetzen untergeordnet und dienen als Schnittstelle zwischen den Niederspannungsnetzen mit 400V und der Hochspannung von 110kV. Sie verteilen in Ballungsgebieten die Energie auf die 10kV/400V Trafostationen und haben somit nur kurze Strecken zu überbrücken. Auf dieser Ebene sind viele Gewerbebetriebe mit höherem Energiebedarf angeschlossen. Auch Einspeiser wie Windparks, Photovoltaik (PV)-Felder oder große Biogasanlagen mit Leistung von mehreren Megawatt müssen auf dieser Ebene einspeisen.

Niederspannungsnetz

Das Niederspannungsnetz hat eine Spannung von 400V mit wenigen Ausnahmen von 600V. Angeschlossen an die Mittelspannungsebene kommt es, sowohl zu Entnahmen von Energie als auch Einspeisevorgängen, durch die im 400V Netz betriebenen PV-Anlagen. Die Trafostationen auf dieser Ebene werden bidirektional beansprucht.
Das Niederspannungsnetz verläuft bis in die Haushalte, in denen gegen Erde eine Einphasenspannung Spannung von 230V gemessen wird. Aus diesem Grunde wird der Sternpunktleiter im Gegensatz zu den höheren Spannungsebenen mitgeführt. Man bezeichnet dieses Netz deshalb auch als Vierleiternetz.

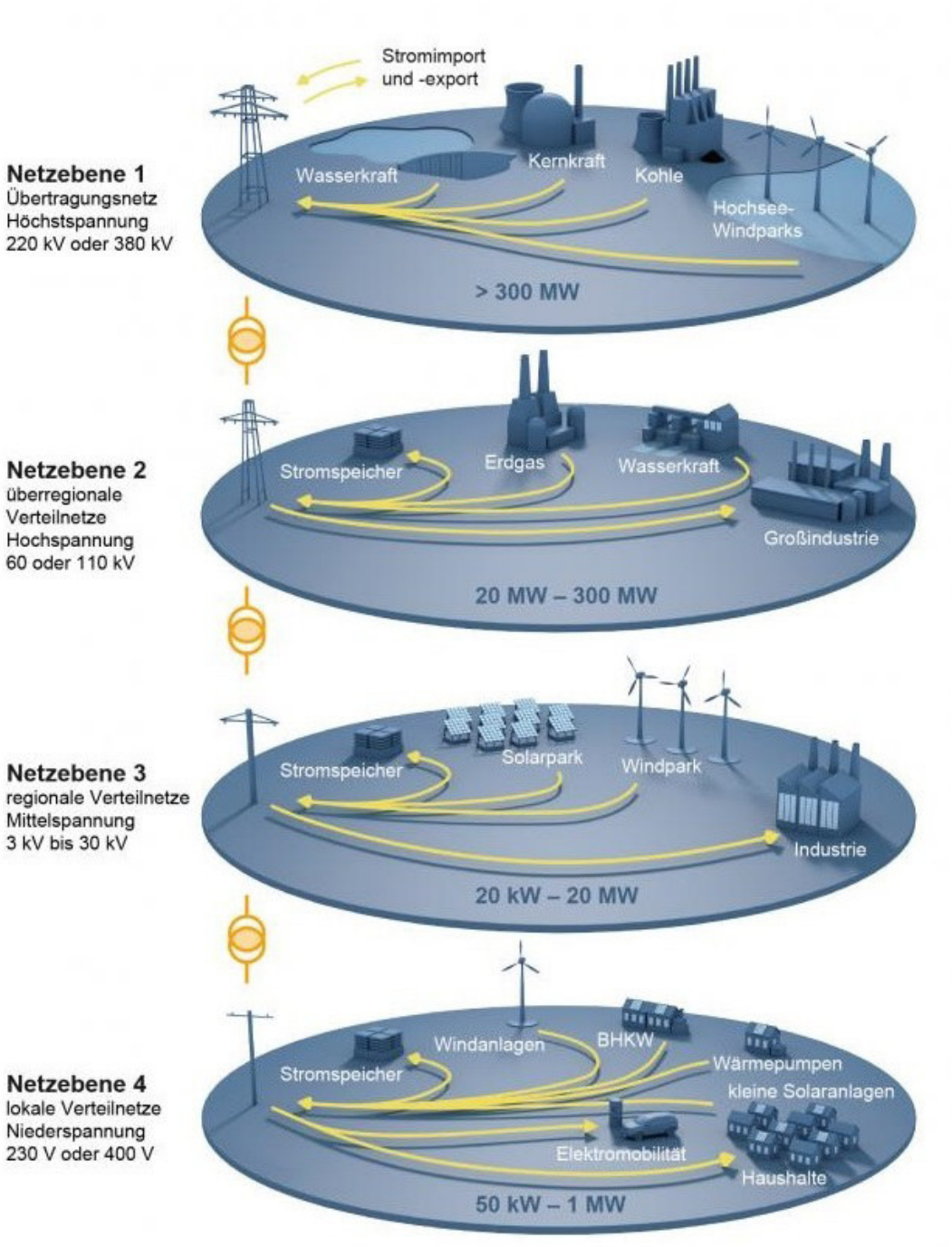


Abbildung 1.2: Die Netzebenen [8]

1.5 Engpässe bei der Energieübertragung

Für eine effiziente Verteilung und Nutzung des stetig steigenden Energiebedarfs der Verbraucher, müssen die Netze in Europa zur Bereitstellung von Energie immer weiter ausgebaut werden. Die Versorgung über große zentrale Kraftwerke zu gestalten ist nur noch eine befristete Lösung. Es müsste ein drastischer Netzausbau stattfinden.
Derzeit wird das Netz bereits massiv ausgebaut. Dies kann dem Bedarf aber nicht nachkommen. Die Ausbauarbeiten sind auch mit enormen Kosten verbunden, die vom Endkunden übernommen werden. Weiter braucht es lange Planungsphasen, um den Ausbau gestalten zu können. Die aktuelle Situation der Stromnetze zeigt weiteren Ausbaubedarf um den Anforderungen gerecht zu werden. Streckenabschnitte werden überlastet gefahren und es kommt immer wieder zu Engpässe die zu Abschaltungen und Redispatch führen. Redispatch bedeutet die Bereitstellung von Energie durch Dritte, da der von dem Verkäufer zugesagte Strom seinen Weg durch das Netz nicht findet. Enorme Mehrkosten sind die Folge, die der Endkunde tragen muss.
Auf der Internetseite `https://www.netzampel.energy` können Netzengpässe für das Bundesland Schleswig - Holstein eingesehen werden.

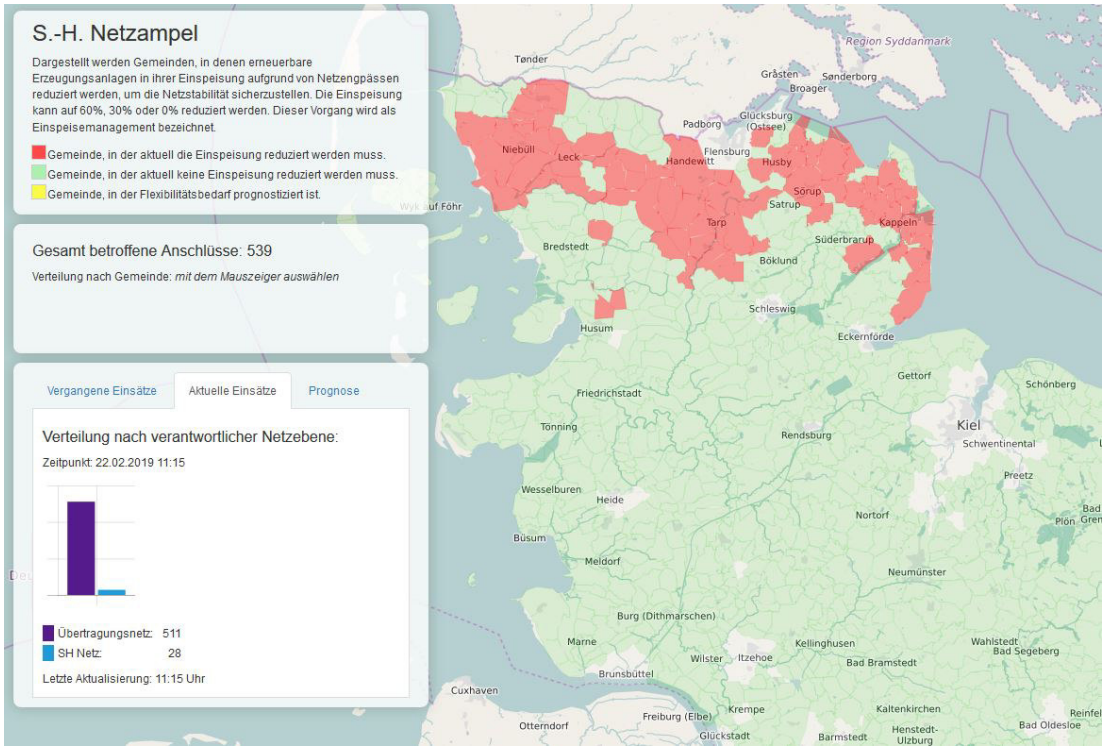


Abbildung 1.3: Netzampel Auszug von Schleswig-Holstein, vom 22.02.2019 [2]

Wie in der Grafik zu erkennen ist, hat oftmals der Norden mit Engpässen zu kämpfen. Die Angabe, dass der Engpass im Übertragungsnetz liegt, kann ein Hinweis auf Windparks sein. Wenn große Windparks auf einer höheren Netzspannung einspeisen und das Netz auslasten, kann aus niedrigeren Spannungsebenen keine Energie mehr aufgenommen werden. Die Einspeiseleistung kleiner PV-Anlagen wird ferngesteuert reduziert, wodurch mehr Verbrauch als Erzeugung in der Unterspannungsebene stattfindet.

1.6 Versorgung durch dezentrale Energieerzeugung

An dieser Stelle soll die Spannungsebene des Endverbrauchers betrachtet werden. Um die bereits installierte Infrastruktur besser nutzen zu können, benötigt es eine bessere Verteilung der Energieerzeugung. Seit Beginn der Energiewende speisen kleinere Produzenten wie z.B. PV-Anlagen aber auch Biogasanlagen, auf der Niederspannungsebene ein. Dies kann zu Entlastungen in den höher liegenden Spannungsebenen führen. Voraussetzung dafür ist die gleichzeitige Nutzung der Energie im gleichen Trafonetz.
Über 98 Prozent der PV-Anlagen in Deutschland sind an das dezentrale Niederspannungsnetz angeschlossen. [16, S.31]
Diese sind sehr wetterabhängig und belasten bei Überproduktion durch Einspeisen in das Übertragungsnetz die Leitungen darüber liegender Spannungsebenen. In den Niederspannungsnetzen kommt es oft durch die Leitungslängen zum Trafo bei Überproduktion von Solarstrom zu extremen Überspannungen. Die dezentralen Produzenten versuchen durch anheben der Spannung den Stromfluss durch den Trafo in die Verteilungsebene zu gestalten. Sind diese ohnehin schon auf einem hohen Niveau potenziert sich das Problem. Eine Lösung setzt in der Unterspannungsebene an und versucht hieraus ein Mikronetz zu gestalten bei dem es möglich sein soll durch Autarkie möglichst die übergeordneten Netze nicht zu belasten. Folgend als Mikronetze bzw. sub-autarke Mikronetze bezeichnet.

1.7 Mikronetz

Als Mikronetz wird ein aus Verbrauchern und Erzeugern bestehendes Netz bezeichnet. Es soll möglichst autark von einem Versorgungsnetz sein. Die Schnittstelle zu einem Versorgungsnetz stellt meist ein Transformators dar. Üblicherweise befindet sich ein Mikronetz auf einer Spannungsebene. Vergleichbar ist das Mikronetz mit dem Inselnetz, mit dem Unterschied, dass beim Inselnetz keine Möglichkeit besteht Unter- oder Überkapazitäten auszugleichen. Die in einem Inselnetz meist vorhandenen Speichersysteme, wie Batterien, kommen auch in diesem Mikronetz zum Einsatz und spielen eine wichtige Rolle.

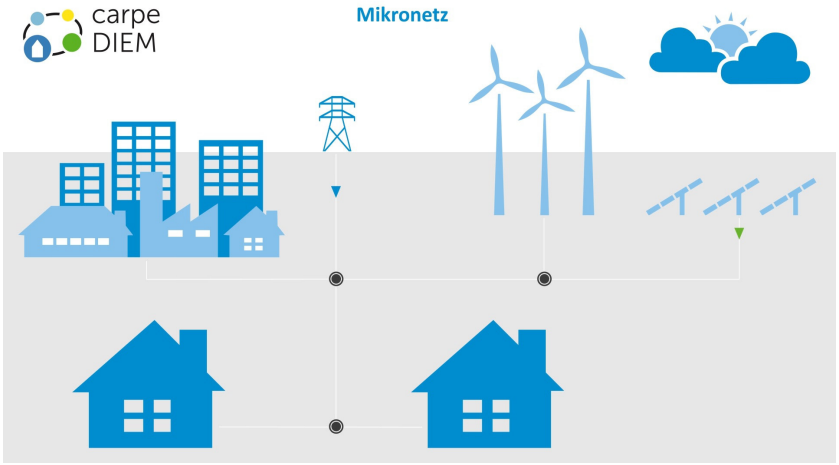


Abbildung 1.4: Mikronetz [20]

Ziel ist es über den Trafo eine möglichst kleine Leistung abzufragen oder einzuspeisen. Hauptbestandteil für solche Netze ist ein Last- und Speicher-Management, da der Fokus auf der Nutzung erneuerbarer Energien liegt. Diese machen das Mikronetz erst rentabel. Kleinere Generatoren die mit fossilen Energieträgern gespeist werden, können sich in solchen Netzen nur durchsetzten wenn sie als Blockheizkraftwerk (BHKW) eingesetzt werden und sich somit ihre Energieausbeute bzw. ihr Gesamtwirkungsgrad erhöht.
Das Management muss bei einer Überproduktion, beispielsweise aus Sonnen- oder Wind-Energie, Lasten regeln können, um produzierte Energie möglichst effizient auszunutzen. Als Lasten werden hier auch Speichermedien verstanden, die beim Laden Energie beziehen. Gegenteiliges gilt für den Fall, dass nicht genügend Energie produziert wird. Zu diesem Zeitpunkt sollten möglichst viele Verbraucher abgeschaltet werden. Bei dauerhaft wenig verfügbarer Energie, werden die Batterien wieder entladen.

1.8 carpeDIEM

Abbildung 1.5: carpeDIEM [19]

Die TH-Lübeck ist ein Projektpartner bei carpe Dezentrales Intelligentes Energiemanagement (carpeDIEM), es gehört zum Kompetenz- und Wissenschaftszentrum für intelligente Energienutzung (WiE). Das WiE beschäftigt sich mit allgemeinen Fragestellungen rund um das Thema Energiewende, Integration erneuerbarer Energien und Kommunikationstechnik.
CarpeDIEM dient dazu die durch Naturgewalten ungleichmäßige Stromproduktion besser an den Verbrauch anzupassen. Ein möglicher Ansatz, der zu diesem Zweck verfolgt wird sind die sub-autarke Mikronetze.
Im betrachteten Teilproblem aus dem Projekt carpeDIEM, wird versucht in einem exemplarischen Mikronetz das Laden von Elektrofahrzeugen so zu gestalten, dass ein Lastgang entsteht der verhindert, dass es zu kurzfristigen Überbelastungen kommt sogenanntes peak-limiting.
Es gibt in dem Projekt zwei Schwerpunkte. Zum einen soll die produzierte Energie im Mikronetz aus PV maximal ausgenutzt werden.
Zum anderen sollen die Maximale Trafoleistung, bei dem Laden von Elektrofahrzeugen nicht überschritten werden.
Seit dem das Erneuerbare-Energien-Gesetz (EEG) 2000 in Kraft getreten ist, hat sich der Ausbau der erneuerbaren Energien in Deutschland stark beschleunigt. Durch garantierte Einspeisevergütungen wurden auch viele Privatpersonen davon überzeugt hier zu

investieren. Dies führte aber auch zu einer stärkeren Produktion elektrischer Energie auf den untersten Spannungsebene. Diese war bei der Auslegung der Netze, die seit vielen Jahren bestanden, nicht mit eingeplant worden.
An diesem Punkt versucht das Projekt carpeDIEM eine Optimierung vorzunehmen. Auch unter Zugrundelegung, dass diese Entwicklung weiter anhält und sich der Anteil an elektrisch betriebenen PKWs ausdehnt, sowie eine Zunahme von Speichersystemen in Haushalten.

peak limiting vs. peak shaving

Beim peak-limiting wird das Netz beobachtet und versucht die Lasten entsprechend der aktuellen Netzauslastung anzupassen bzw. zu begrenzen. Hierbei sind alle Teile des Netzes als potenzielle Engpässe zu berücksichtigen. Wird ein Teil überlastet, findet eine Abschaltung statt.
Bei dem peak-shaving wird bereits vorher ein Fahrplan erstellt der das Netz optimal auslasten soll. Dieser basiert auf vergangenen Werten und trifft Annahmen für die Zukunft.
Der Unterschied, der zwei Verfahren besteht hauptsächlich aus der zeitlichen Betrachtung. Beim peak-shaving wird auf Grund von Erwartungen bzw. vergangenen Ereignissen eine Prognose getroffen. Beim peak-limiting werden die aktuellen Extreme in Echtzeit versucht auszugleichen.

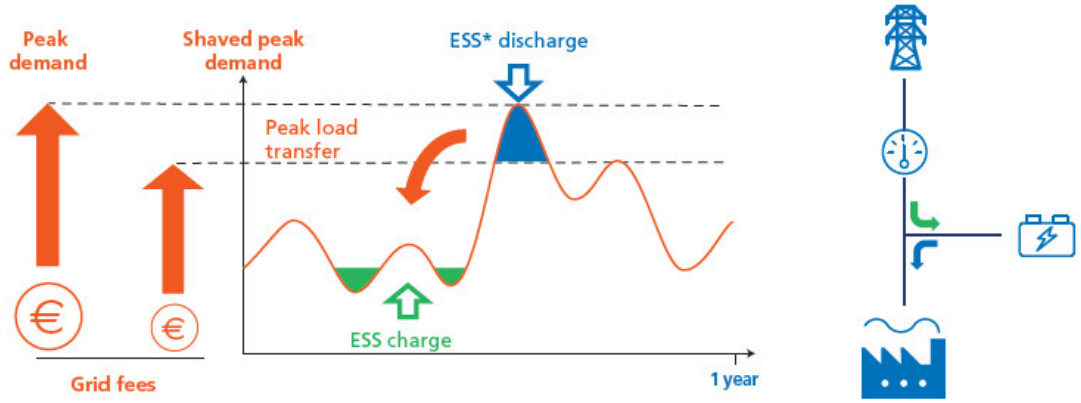


Abbildung 1.6: peak-shaving [9]

Das Bild zeigt das peak-shaving an einer exemplarischen Stelle. Es findet an dieser Stelle eine Lastverschiebung (peak-shifting) aufgrund einer prognostizierten Spitze statt. Die Verschiebung der Last wird in diesem Fall durch das Aufladen von Batterien vollzogen, die zum Zeitpunkt der Spitzenlast genutzt werden können ohne das Netz zu belasten. Über den Zeitlichen Verlauf würden weniger Netzkosten entstehen.

Batterieprojekt

In Zusammenarbeit mit den Projektpartnern des carpeDIEM wird ein Konzept zum Laden und Entladen von PKWs entwickelt. Eine Überlastung des Transformators zum übergeordneten Netz soll verhindert werden.
Beim Laden der PKW-Batterien sind große Kapazitäten in möglichst kurzer Zeit vorgesehen. Um dies zu ermöglichen wird eine sehr große Leistung in den Ladesäulen bereitgestellt. Kommt es zu einem Ladeprozess wird sofort eine hohe Leistung benötigt, die eventuell nicht mehr zur Verfügung steht, da bereits andere PKWs laden oder Haushalte diese benötigen.
Projektpartner haben hierfür einen Algorithmus entwickelt, der eine Verteilung der Leistung am Generator bzw. der im Mikronetz verfügbaren Energie regelt.

1.9 Ausbaustand der erneuerbaren Energien

Erneuerbare-Energien-Gesetz

Das Erneuerbare-Energien-Gesetz ,kurz EEG, soll den Ausbau erneuerbarer Energien regeln und fördern. Eingeführt wurde das Gesetz im Jahre 2000 und ist seither einigen Änderungen untergangen.
Sie betrafen meist die stärkere Förderung einzelner Energieträger und deren Vergütungen. Auch Eigenverbrauchsanlagen oder die Erneuerung von Bestandsanlagen, sogenanntes Repowering, sind hier geregelt.
Die letzte Neufassung des Gesetzes kam Juli 2014 und die letzte Änderung Dezember 2018.
Dies zeigt wie viel politisches Engagement und Aktualität hinter dem Projekt steht. Neue Technologien sollen schnell in unser Netz integriert werden, um eine schnelle Besserung der Netzstabilität und Belastbarkeit zu erzielen.
Der starke Ausbau der erneuerbaren Energien hatte durch die Förderung einen zu Beginn massiven Anstieg der Umlage nach sich gezogen. Dies hat sich in den letzten Jahren etwas beruhigt. Durch sinkende Einspeisevergütungen wurde eine bremsende Wirkung erzielt, was zuletzt zu einem bleibenden Niveau führte. Um diese Kosten, die als Umlage jeder Energieverbraucher mittragen sollte, nicht in unbegrenzte Ausmaße steigen zu lassen, wird diese kontinuierlich gesenkt.
Derzeit muss zum Bau größerer Anlagenparks an Ausschreibungen teilgenommen werden, bei denen die geringsten Forderungen an Zusatzvergütung den Zuschlag erhalten. Kontrolliert und dirigiert wird dies durch die Bundesnetzagentur.
Bereits zu Beginn der Einführung dieser Umlage wurde gewarnt, dass die Wirtschaft zu einem großen Teil diese Mehrbelastung nicht tragen kann. Unter der Annahme, dass viele Arbeitsplätze dieser Umlage zum Opfer fallen würden, wurden die meisten Industrien und große Gewerbebetriebe von der Umlage befreit. Dies hatte zur Folge, dass die Privathaushalte umso stärker belastet werden.
Im Jahr 2018 betrug die Umlage für die erneuerbaren Energien, kurz EEG-Umlage, bei 6,79 Cent. Das entspricht 23% vom durchschnittlichen Gesamtstrompreis von 29,42 Cent in Deutschland.

Ausbau von Photovoltaikanlagen

Durch die Verabschiedung des EEG, welches die bevorzugte Einspeisung und garantierte Preise regelt, wurde der Ausbau von PV-Anlagen massiv vorangetrieben. Dies gipfelte in den Jahren 2010 - 2012. Im Jahre 2012 kam es zu einem Rekordausbau mit ca. 7,6 GWp. In diesen Jahren wurden durch den Preisverfall der Module am Markt eine hohe Rendite ermöglicht.
Die Bundesregierung befürchtete eine Explosion der EEG Umlage und somit ein mögliche Ablehnung der Bevölkerung gegen die Energiewende und den Ausbau der erneuerbaren Energien.
Philipp Rösler von der FDP, in der Rolle des Bundesminister für Wirtschaft und Technologie, warb für eine Novellierung des EEG mit stark reduzierten Vergütungssätzen. Folglich wurde der Ausbau der solaren Strahlungsenergie stark dezimiert. Dies hatte zur Folge, dass die Industrie, die sich mit den Preisen auf die Abnahme großer Mengen eingestellt hatte, stark belastet wurde und es zu Konsolidierung kam.

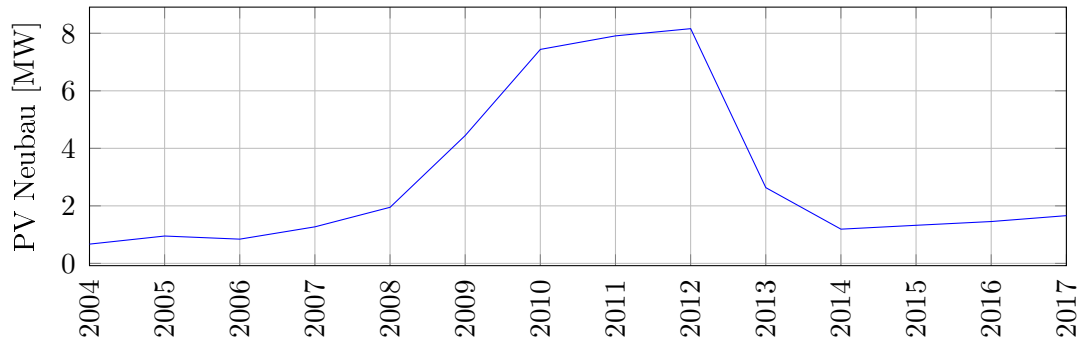


Abbildung 1.7: Jährlicher Zubau von PV-Leistung in Deutschland [3]

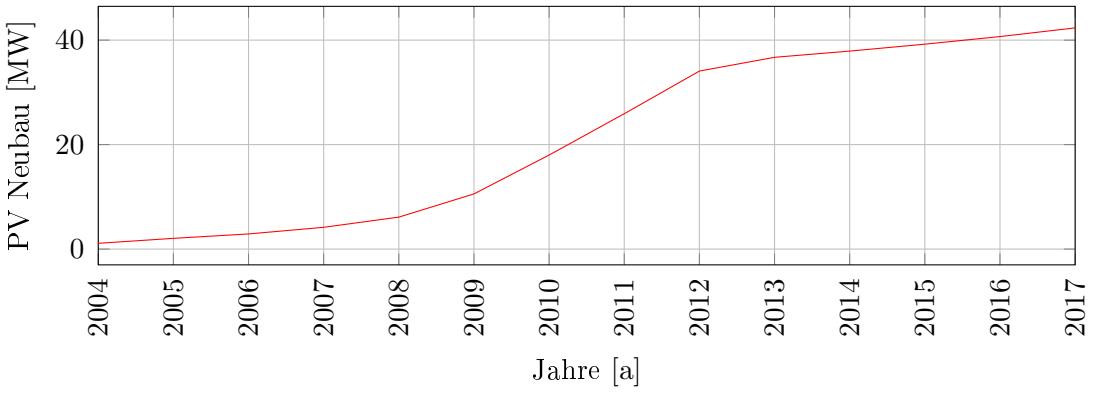


Abbildung 1.8: Gesamt installierte PV-Leistung in Deutschland [3]

Ausbau von Batteriesystemen

Ein nicht neues aber immer attraktiveres System sind Speichermedien. Um eine Verringerung des Leistungsbedarfs aus dem Netz zu gestalten oder eigenproduzierte Sonnenenergie zu speichern.
Es sind in diesem Bereich, getrieben von der aktuellen Problematik, viele verschiedene Ansätze verfolgt worden. Dies hat zu sehr viel effizienteren und auch langlebigeren Speichermedien geführt. Diese werden vermehrt von Privatpersonen im kleinen dezentralen Bereich eingesetzt.

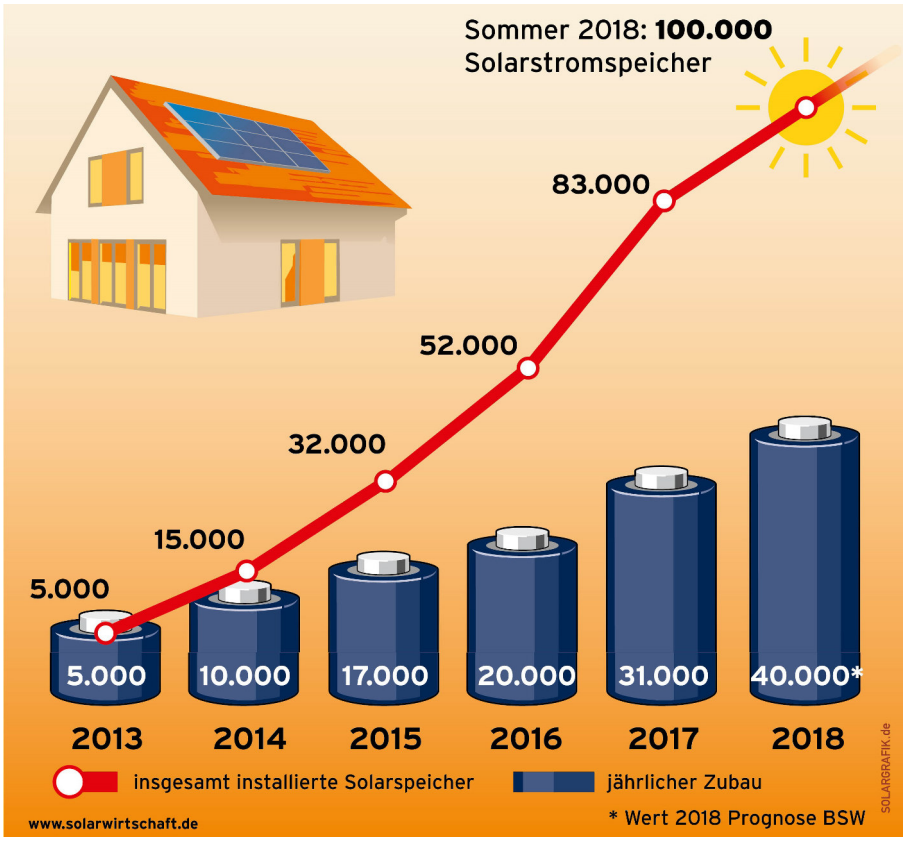


Abbildung 1.9: PV-Anlagen mit Speicher in Deutschland [6]

In den neueren Novellierung des EEG werden Speichersysteme durch erhöhte Förderungen besser gestellt als zuvor. Die Politik zeigt hier Einsehen, dass es große Potentiale für die Netzstabilität durch die Verbreitung solcher Technologien gibt.
Die Daten, der für die Überwachung verantwortlichen Bundesnetzagentur, zeigen einen kontinuierlichen Ausbau der PV-Anlagen in Deutschland, wobei auch ein erheblicher Teil in Kombination mit einem Speichermedium installiert wird.
In dem Projekt von carpeDIEM werden diese Speichermedien als zentrale Lösung für eine Reduzierung der steigenden Gesamtnetzkapazität angesehen. Die Entwicklung von intelligenten Algorithmen zum Steuern und Vernetzen dieser Speicher wird fokussiert. Im Folgenden wurde eine Simulationsumgebung entwickelt, um solche Algorithmen zu testen und ihr Einwirken auf das Netz abschätzen zu können.

2 Konzeptentwicklung – Timo Helsper

Das Konzept beinhaltet das Erstellen einer Simulationsumgebung. Diese Simulationsumgebung enthält ein Mikronetz, welches für den Steueralgorithmus ausgelegt ist. Mit Hilfe des Sonnensimulators kann ein Sonnenverlauf simuliert und aufgenommen werden. Diese Messwerte müssen für die Simulationsumgebung aufgearbeitet und in die Simulationsumgebung integriert werden. Zusätzlich soll es ein Haushaltslastprofil geben, welches das Leben in einem Haushalt widerspiegelt. Zum Testen des Steueralgorithmus werden zwei Ladestationen für Elektromobilität in die Simulationsumgebung mit integriert.

2.1 Anforderungen

Damit der Steuerungsalgorithmus richtig funktioniert, ist es erforderlich, dass einige Grundelemente vorhanden sind. Zusätzlich sollen die ausgegebenen Werte realitätsnah sein. Folgende Punkte sind für die Simulationsumgebung notwendig:

- Simulationsumgebung in PowerFactory
  - Transformator
  - Zwei Haushalte
  - Zwei Photovoltaik-Anlagen
  - Zwei Ladestationen für Elektromobilität
- Datenübertragung über das vorhandene Netz der Technischen Hochschule
- Java Server
  - Implementierung des Steueralgorithmus
  - Bereitstellung der Werte für den Steueralgorithmus
  - Pro Haushalt ein Steueralgorithmus
  - Pro Haushalt einen eigenen Raspberry Pi
- Sonnensimulator
  - Bereitstellung der Messwerte
  - Bereitstellung des Services für die Datenübertragung

2.2 Lösung Simulationsumgebung

Um der Realität nahe zu kommen, werden sinnhaft zwei Häuser erstellt. Ein Haus besitzt jeweils ein Lastprofil von einem Haushalt, ein sogenanntes Haushaltslastprofil (HLP). Auf dem Dach befindet sich eine PV-Anlage. Die Werte für die PV-Anlage werden von dem Sonnensimulator bereitgestellt und hier als Erzeugerlastprofil (ELP) benannt.
Zusätzlich hat das Haus noch eine Ladestation für Elektroautos. Die Ladezeiten für diese Ladestation werden durch das Verhalten der Anwohner angenommen und sind in dem Ladezeitenprofil (LZP) Prozentual angegeben. Die Werte von diesen drei Bestandteilen sind einzeln schreib- und lesefähig. Diesen Aufbau gibt es zweimal, um zwei Häuser zu simulieren.
Die beiden Häuser sind über ein Kabel mit einer Sammelschiene verbunden. An dieser Sammelschiene ist ein Transformator (Trafo) angeschlossen, welcher die Verbindung zu dem externen Netz darstellt. Die Auslastung des Trafos ist bereitgestellt, da diese von dem Steueralgorithmus benötigt wird.

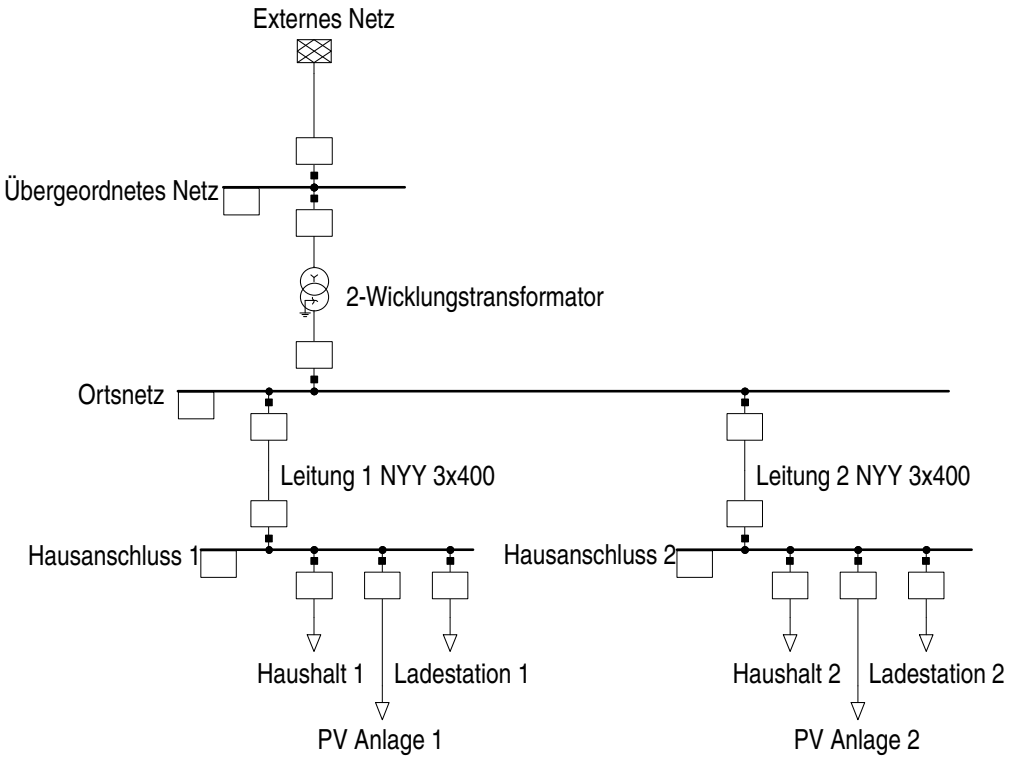


Abbildung 2.1: Aufbau der Simulationsumgebung

Durch die Anforderungen an die Simulationsumgebung ist der Aufbau aus Abbildung 2.1 entstanden.
Anschließend werden die Variablen festgelegt, welche überprüft werden sollen.

- Leistung
  - Transformatorwirkleistung auf der Oberspannungsseite
  - Pro Haushalt, die Wirkleistung
  - Pro Photovoltaik, die Wirkleistung
  - Pro Ladestation, die Wirkleistung
- Auslastung der Kabel in Prozent
- Die Frequenzänderung in Hertz
- Spannungsänderung zwischen Leiter und Erde in Volt

2.3 Lösung Datenübertragung

Für die Datenübertragung der Werte von dem Sonnensimulator wird anfangs überlegt, Open Platform Communications Unified Architecture (OPC UA) zu verwenden. Es findet in der Industrie immer wieder Anwendung.
Die speicherprogrammierbare Steuerung (SPS) sowie PowerFactory unterstützt den Standard OPC UA. So werden erste Versuche unternommen, Daten über ein OPC UA Server auszutauschen. Im Laufe dieser Versuche stellt sich heraus, dass der Standard OPC UA von PowerFactory nur gegen Aufpreis zur Verfügung steht.
Eine Alternative ist nun, ein Raspberry Pi (RPI) als ein Haushalt zu sehen. Somit werden alle Daten von dem RPI verarbeitet und für PowerFactory bereitgestellt. Der RPI stellt somit die Schnittstelle zwischen externen Werten und PowerFacotry.
Der Sonnensimulator stellt seine Werte per File Transfer Protocol (FTP)-Server bereit. Das HLP sowie das LZP des Autos liegen als Text-Datei auf dem RPI. Einzig die Berechnung des Algorithmus findet auf einem externen Server statt.

3 Versuchsstand – Christian Ziegelmann

In unserem Teilprojekt des carpeDIEM möchten wir die Simulation eines Mikronetzes konzipieren. Es soll möglich sein auf der 400V-Spannungsebene Simulationen durchzuführen und einen Einblick auf die Auswirkungen auf die überliegenden Spannungsebenen zu bekommen.
Hierzu wurde am Standort Lübeck eine PV-Wettersimulation gebaut, die hardwaremäßig Sonneneinstrahlung und Wolkenzug simulieren kann. Diese Werte werden übernommen und bilden die Grundlage der PV-Energie, die im Mikronetz zur Verfügung steht. Ziel soll es sein, Verbraucher und andere Lasten so zu steuern und regeln, dass ein möglichst homogener Lastfluss im Mikronetz entsteht. Zu diesem Zweck werden Verbraucher mit unterschiedlichen Lastprofilen eingebunden, um deren Auswirkung zu erfassen.
Weitergehend wollen wir mit unseren Projektpartnern ein System abbilden können, dass vom Wetter und dem eingebunden Algorithmen beeinflusst werden kann. Diese Algorithmen versuchen durch das aktive Eingreifen in Verbrauch und Speicherung eine optimale Nutzung des Netzes zu erzielen.
Zusätzlich ist es ein Anliegen jederzeit individuelle Vorgaben des Endverbrauchers einzulesen und diese sofort mit zu integrieren. Hierbei handelt es sich um vom Endverbraucher angeforderte Leistungen. Ein Beispiel ist die Programmierung von Ladezuständen der PKWs zu definierten Zeiten.
Auch das transiente Verhalten von PV-Anlangen soll betrachtet werden. Dabei wird untersucht, wie sich der Wolkenfluss, aus der Wettersimulation, auf den Energiefluss und besonders auf Strom, Spannung und Frequenz auswirkt.

3.1 Schematischer Aufbau

Im Labor der TH-Lübeck wird die Simulation in mehrere Stationen durchgeführt.

Hardware:

- Speicherprogrammierbare Steuerung (SPS) verbunden mit der Sonnensimulation
- Raspberry PI Computer
- Personal Computer (PC) mit der Software PowerFactory

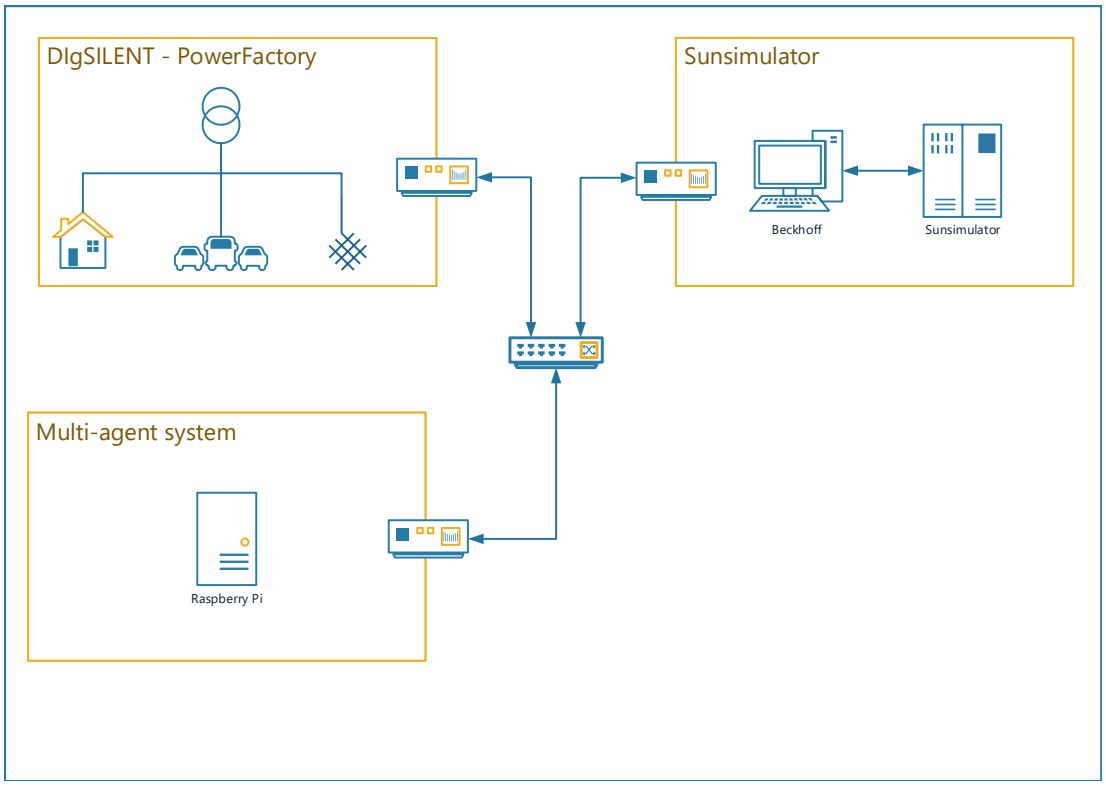


Abbildung 3.1: Schematische Übersicht des Versuchsaufbaus

Die Kommunikation der Teilnehmer untereinander läuft über das Intranet. Alle Teilnehmer haben eine Netzwerkkarte mit Ethernet-Port und RJ45 Steckverbindung. Die Protokolle, Transmission Control Protocol (TCP)/Internet Protocol (IP) werden über das Ethernet übertragen. Dafür werden die an der TH-Lübeck installierten Router verwendet. Diese sind über Dosen in den Steckerleisten bereits vorbedrahtet.
Um eine dezentrale Umgebung zu simulieren werden Raspberry PI Computer in dem Laborraum verteilt platziert. Lastprofile für Haushalte und Ladezyklen von elektrisch betriebenen Personenkraftwagen (Pkw), werden auf diesen hinterlegt.
Das Profil für die PV-Anlage auf den privaten Haushalten wird in der Sonnenstandsimulation erzeugt. Über Faktoren, die die installierte Leistung angibt, werden die Daten für diese Produzenten errechnet.
Während der Simulation werden zyklisch Berechnungen der Lastflüsse durchgeführt.

Bei jedem Zyklus werden alle aktuellen Profilwerte abgefragt. Alle Werte werden in der Simulationsumgebung zusammengefügt. Im Ergebnis können alle Lastflüsse im Netz betrachtet werden.
Eine in diesem System eingebaute Steuerung kann einzelne Lasten, abweichend von den Profilwerten, zuweisen. Es kann ein intelligentes Netz mit optimierter Nutzung lokal produzierter Energie getestet werden.

3.1.1 Sonnensimulation

Angeschlossen sind:

- PV-Modul
- Wechselrichter
- Batterie
- 12V Lampen
- Nebelmaschine

Die Sonnensimulation besteht im wesentlichen aus einem PV-Modul, Lampen, und einer Nebelmaschine.
Die Lampen produzieren die Einstrahlungsleistung, die sich im PV-Modul zu elektrischer Energie umwandelt. Hierbei wird versucht dem Spektrum der Sonne möglichst nahe zu kommen. Um einen Tagesverlauf richtig widerzuspiegeln, werden mittels einer Nebelmaschine Wolken erzeugt, die die Einstrahlung auf das PV-Modul reduzieren. Durch Wolkenbildung und Absaugen der Wolken lassen sich verschiedene Wetterszenarien simulieren.
Es lässt sich darstellen, wie abrupt die Energieschwankungen im Netz sind. Der Verlauf der Sonne wird durch Dimmen der Lampen realisiert. Ein Programm auf der SPS speichert den Simulationszyklus. Diese Daten werden anschließend als Referenz in der Simulation von PowerFactory genutzt.

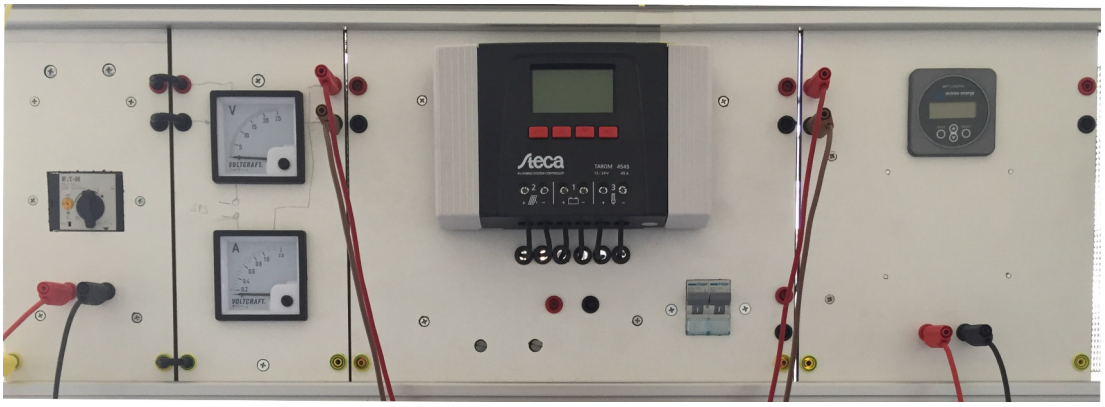


Abbildung 3.2: Wechselrichter Anschluss an dem Versuchsstand

Abbildung 3.3: Lampen und PV-Modul der Sonnensimulation

Speicherprogrammierbare Steuerung

Die Speicherprogrammierbare Steuerung, kurz SPS, die zum Betrieb der Wettersimulation zum Einsatz kommt, ist eine Beckhoff CX9010, wie sie im Abbildung 3.4 zu sehen ist. Diese hat als Betriebssystem eine Windows Variante.

Gesteuert werden von der SPS:

- Lampen:
  - Hallogenstrahler
  - Light Emitting Plasma (LEP) Lampe
- Lüfter zur Kühlung
- Nebelmaschine
- Radiallüfter zum Absaugen des Nebels

Ausgelesen werden von der SPS:

- Strom und Spannung des PV-Moduls
- Temperatur des PV-Moduls
- Strom und Spannung von den Verbrauchern an der Batterie
- Umgebungstemperatur

Aufbauend auf dem Steuerungsprogramm des Herrn Biwersi, wurde die Software um die Speicherung der Ausgabewerte in Zeitdiskreten Schritten erweitert.

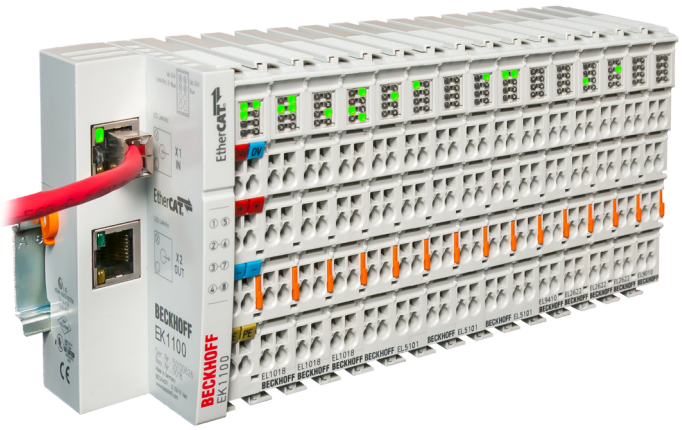


Abbildung 3.4: Beckhoff SPS mit I/O Karten [17]

3.1.2 Raspberry Pi

Die Raspberry Pi aus Abbildung 3.5 sind kleine Einplatinencomputer. Hier wurde die zweite und dritte Generation der Platine eingesetzt. In diesem Versuchsaufbau dienen sie zur Simulation der Verbraucher. Auf Ihnen werden Lastprofile gespeichert und der Verbrauch kann jederzeit, beispielsweise durch Algorithmen, beeinflusst werden. Angeschlossen sind sie an das Netzwerk sowie mit einer $5V$ Gleichspannungsquelle mit $2A$ verbunden.

Abbildung 3.5: Raspberry PI [10]

3.1.3 PowerFactory

Ein PC mit Windows 10 hat die Version PowerFactory 15 installiert. Es handelt sich um die Education-Version, die für einmalig 150€ von Forschungseinrichtungen und Hochschulen zur nicht kommerzielle Nutzung erworben werden kann. Eine genauere Beschreibung ist im Kapitel 6.1.

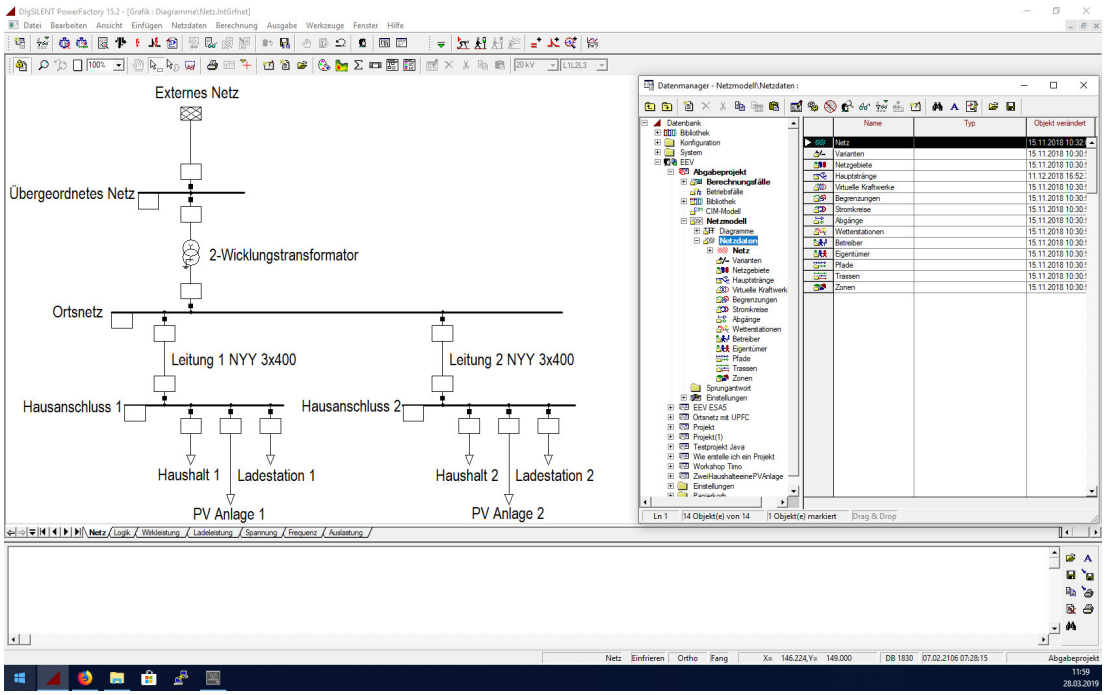


Abbildung 3.6: Simulationsumgebung auf einem Windows 10 PC [15]

4 Datenübertragung – Christian Ziegelmann

4.1 Übersicht

Es sollen alle Daten im Intranet, sowie im Internet über TCP/IP übertragen werden. Im folgenden Kapitel werden diese Protokolle aufgezeigt und ihre Anwendung für die Übertragung deutlich gemacht.
Dazu werden Aufbau des IP sowie TCP dargestellt und die für das Projekt wichtigen Teile aufgezeigt.

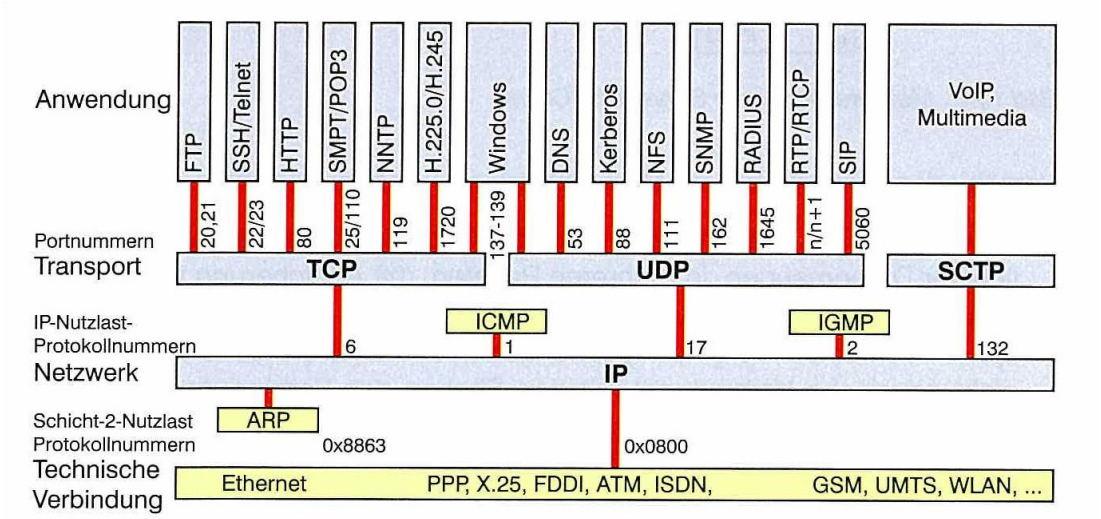


Abbildung 4.1: Architektur Aufbau der Transportschichten [11, S.25]

In der Abbildung 4.1 wird ein Überblick über gängige Protokolle, die zur Datenübertragung dienen, gegeben. Die im Versuchsaufbau gesendeten Daten werden im Ethernet mit den Protokollen IP, TCP und FTP übertragen.
Das Ethernet ist die technische Umsetzung der Anbindung einer Netzwerkschnittstelle.

4.2 Internet Protocol

Das Internet Protocol (IP), ist für den Transport der Datenpakete vom Sender zum Empfänger verantwortlich. Es wird in der Request for Comments (RFC) 791 definiert. In dieser ist der Protokollkopf und die Reihenfolge der Bits definiert. Die RFC wird von der Internet Society (ISOC) herausgegeben. Die ISOC ist eine Nichtregierungsorganisation für die Pflege und Weiterentwicklung der Internetinfrastruktur.
Das Internet basiert auf der Idee der IP-Adressen. Es ermöglicht jeden Teilnehmer zu identifizieren, um ihm Nachrichten zustellen zu können. Derzeit wird hauptsächlich mit der IPv4 (Version 4) gearbeitet. Der mögliche Adressbereich der Version 4 ist bereits ausgeschöpft. Eine neue Version 6 wurde zur Erweiterung des Adressraumes entworfen. Die Umstellung von Version 4 auf 6 ist noch nicht vollzogen. Deshalb gibt es interne und externe Adressräume. Das bedeutet, dass die im Heimnetzwerk zugewiesene Adresse nicht ins Internet übermittelt wird. Der Router bekommt eine eigene Adresse zugewiesen mit der dieser im Internet sichtbar ist. Um eine Anfrage im Internet durchzustellen wird ein Port am Router geöffnet, welcher zusammen mit der Router-Adresse übertragen wird. Nachdem der Router die Antwort erhält wird diese im Heimnetzwerk zugestellt. Da wir mit IPv4 arbeiten, wird auch nur diese dargestellt.

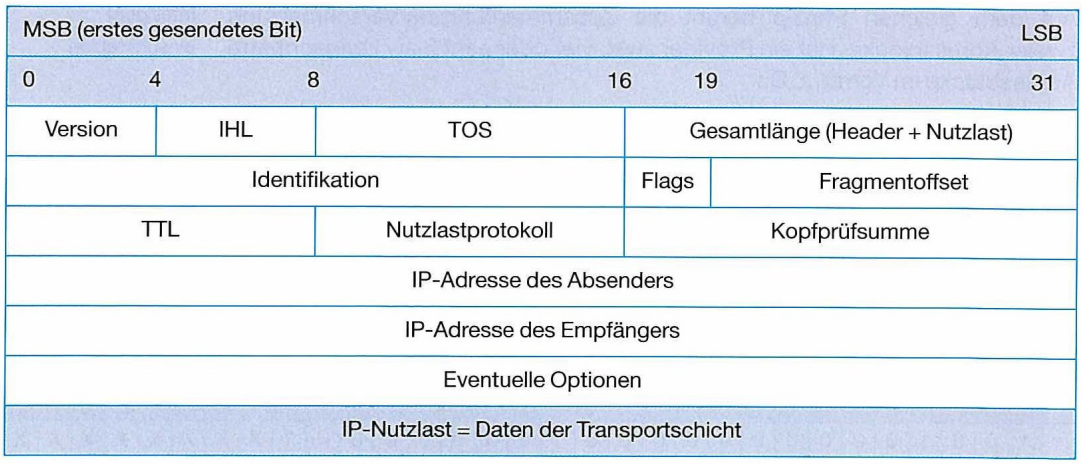


Abbildung 4.2: IP-Protokollkopf [11, S.34]

Feld		Bedeutung/Anmerkung
Version	(4 Bits)	Die Versionsnummer; aktuell ist Version 4; die Vorgänger werden nicht mehr verwendet. Version 5 wird übersprungen und Version 6 steht vor ihrer Einführung.
IHL	(4 Bits)	(**I**P-**H**eader **L**ength) Länge des Headers, gemessen in 32-Bit-Gruppen: 5 <= IHL <= 15
TOS	(8 Bits)	(**T**ype **O**f **S**ervice) Anforderung höherer Protokollschichten an eine bestimmte Dienstgüte, z. B. minimale Verzögerung/minimale Kosten/maximale Zuverlässigkeit/maximaler Durchsatz
Gesamtlänge	(16 Bits)	Maximale Länge eines IP-Datagramms ist 65 535 Oktette; jedes System sollte 576 Oktetts große IP-Pakete verarbeiten können.
Identifikation	(16 Bits)	siehe nachfolgenden Text
Flags	(3 Bits)	
Fragmentoffset	(13 Bits)	
TTL	(8 Bits)	(**T**ime **T**o **L**ive) Begrenzt die Anzahl der Weiterleitungen, um Endlosschleifen fehlgeleiteter Pakete zu vermeiden.
Nutzlastprotokoll	(8 Bits)	1=ICMP, 2=IGMP, 6=TCP, 17=UDP nach RFC 1700 (als wichtigste); siehe auch Bild 1.22.
Kopfprüfsumme	(16 Bits)	Ein Prüfwert, den die sendende Instanz aus allen Oktetten des Headers errechnet und einträgt. Die empfangende Instanz rechnet auch, vergleicht mit dem gesendeten Prüfwert und verwirft das Paket bei fehlender Übereinstimmung.

Abbildung 4.3: IP-Protokollkopf und Beschreibung [11, S.35]

4.3 Port

Das englische Wort Port, übersetzt Hafen, lässt sich außerdem mit dem Wort Tor oder Durchlass definieren.
Ein Port wird im Netzwerkprotokoll angegeben, um die Zuordnung der Informationen zu einer bestimmten Anwendung zu gewährleisten. Es existiert eine Liste der Ports. Diese Port-Nummern werden von der Internet Assigned Numbers Authority (IANA) vergeben. Die Gesamtmenge der Ports beläuft sich auf 65.536. Diese teilen sich auf in drei Bereiche:

- **System Ports**
  Die ersten 0-1023 Ports wurden von der IANA für bestimmte Dienste und Zwecke festgelegt. Ein bekanntes Beispiel ist der Port 80 für eine Hypertext Transfer Protocol (HTTP) Übertragung.

- **Registered Ports**
  Die Ports zwischen 1024 und 49151 können von Programmen und Anwendungen auf Anfrage an die IANA reserviert werden. Sie sollen als vergebene Ports für definierte Programme die Kommunikation dieser sichern stellen.

- **Dynamic Ports**
  Die restlichen Ports zwischen 49152 und 65535 sind dynamisch und können von Programmen frei genutzt werden.

Zum Aufbau einer TCP Verbindung, werden dem Protokollkopf zwei Ports mitgegeben, die des Absenders und die des Empfängers.
Ports können auch Netzwerkprotokolle und entsprechende Netzwerkdienste identifizieren. Für eine Verbindung mit FTP sind Port 20 und 21 als reservierte Systemports vorgesehen.

4.4 Transmission Control Protocol

Tranmission Control Protkol (TCP), dient zur verlustfreien verbindungsorientierten Datenübertragung zwischen zwei Stationen. Daten jeglicher Art werden in ein TCP-Pakete gewandelt, um sie übertragen zu können.

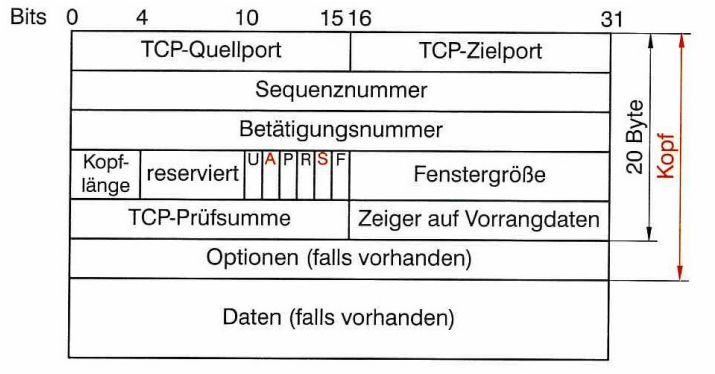


Abbildung 4.4: TCP Protokollkopf [11, S.60]

Zum Verbindungsaufbau kommen Flags zum Einsatz, diese sind in TCP-Protokollkopf zu finden (siehe Abbildung 4.4). Sind diese ausgetauscht worden, kommt es zu einer Verbindung. Bei der Datenübertragung kommt es pro Paket zu einer Bestätigung des Empfängers. Jedes Paket bekommt eine Sequenznummer. Bei Paketverlust wird anhand der Sequenznummer die ausstehende Bestätigung erkannt und entsprechendes Paket erneut gesendet.
Der Verbindungsabbau kann von beiden Seiten ausgelöst werden dies geschieht mit Hilfe von Finish (FIN)- und Acknowledgement (ACK)-Flags. Hier wird das Beenden der Verbindung mit FIN angezeigt und mit ACK-Flag betätigt. Das TCP ist komplex und wurde entwickelt um lange andauernde Verbindungen zu ermöglichen. Wie das Aufbauen einer Internet Verbindung (http). Diese Verbindungen haben eine lange Bestandszeit.

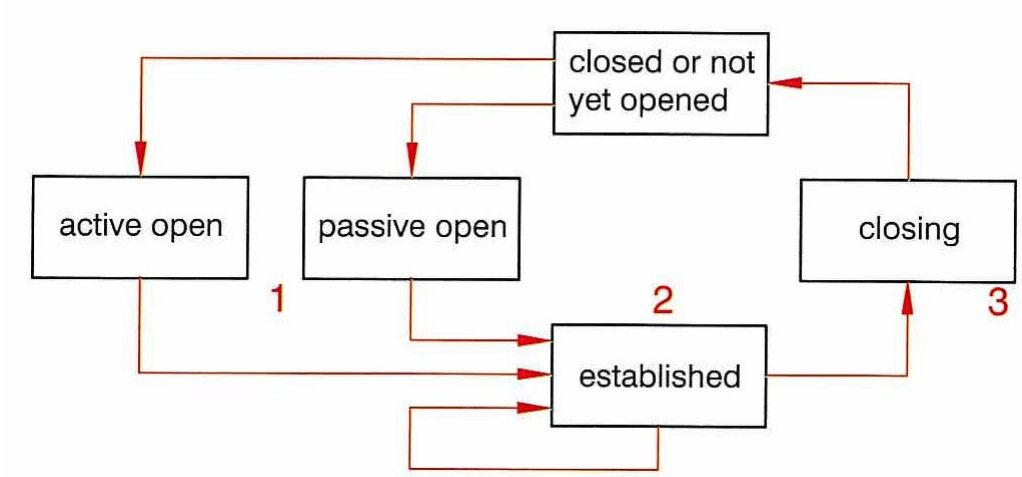


Abbildung 4.5: TPC Verbindung [11, S.60]

In der Abbildung 4.5 sind drei Schritte aufgezeigt. Im ersten wird die Verbindung etabliert. Es gibt zwei verschieden Verbindungstypen. Das "passive open" wird von dem Server angefordert. Eine gestartete Applikation sendet eine TCP-Sequenz ohne Zielport nur der Quellport wird eingetragen. Vom Client wird ein "active open" angefordert dieser enthält sowohl Quell- als auch Zielport. Kommt es zu einer Übereinstimmung von "active open" Zielport und "passive open" Quellport wird eine Verbindung aufgebaut. In Schritt zwei bleibt die Verbindung so lange etabliert bis sie in Schritt drei von einem Teilnehmer geschlossen wird.
Bei nicht so aufwändigen Verbindungen kommt das User Datagram Protocol (UDP) zum Einsatz. Es wird häufig für einfache Anfragen benutzt. UDP ist ein verbindungsloses Protokoll bei dem Daten verloren gehen können. Fehlerkorrektur und Verbindungsabbau entfällt.

4.5 File Transfer Protocol

FTP, ist eins der ältesten und stabilsten Protokolle des Internets. Es wurde 1971 entwickelt und ab 1985 in der RFC 959 definiert. Hier sind Statuscodes und FTP-Kommandos deklariert.
Das FTP hat im Internet eine große Bedeutung bekommen, um Dateien in einen Web-

space hoch oder runter zu laden. Auch in anderen Netzwerken hat sich dieses Protokoll etabliert. Das FTP hat sich aufgrund seiner Stabilität und Vorteile bis in die heutige Zeit halten können.

- Es werden zwischen Client und Server TCP-Verbindungen aufgebaut.
- Die Port Auswahl kann eindeutig definiert werden. Im Gegensatz stehen moderne Cloud-Lösungen, bei denen es oft zu einem Wechsel kommt.
- Über FTP ist auch bei größeren Dateien eine sichere Übertragung gewährleistet.
- Der Zugriff auf einen FTP-Server kann sowohl für jeden zugänglich, aber auch über die Abfrage von Benutzernamen und Passwort gestaltet werden.
- Verschlüsselung der Übertragung ist mit Secure Sockets Layer (SSL) und Secure Shell (SSH) Verfahren möglich. Die Bezeichnung ändert sich hier in FTP über SSL (FTPS) und SSH File Transfer Protocol (SFTP)
- FTP ist von Betriebssystemen unabhängig

Das FTP hat zwei Kommunikationskanäle. Der Erste wird genutzt um Befehle und Statusmeldungen zu übersenden. Auf jedes Kommando antwortet der FTP-Server mit zugehöriger Statusmeldung.
Der Zweite parallel genutzte Kanal wird zur Datenübertragung eingesetzt. Dies hat den großen Vorteil, dass während einer Übertragung weiterhin uneingeschränkt kommuniziert werden kann. Es müssen auch in der Datenbandbreite der Daten keine Platzhalter für Befehlssätze reserviert werden.
FTP arbeitet nach dem Client-Server-Prinzip. Das bedeutet eine Seite ist der FTP Server (Bereitsteller) an den der Client (Bezieher) eine Anfrage stellt.

4.6 Übertragung

Bei einem Verbindungsaufbau wird von dem Client ein IP-Protokollkopf erstellt. Dieser beinhaltet die Absender- und Ziel-Adresse der Kommunikationspartner. In dem Protokollkopf wird unter Nutzlastprotokoll der Binärcode für die Dezimalzahl sechs eingetragen und damit TCP ausgewählt. Die Nutzdaten des IP-Protokollkopfs beginnen mit dem TCP-Protokollkopf. Hier werden die Quell- und Zielports hinterlegt. Der Zielport der am Server angesprochen wird ist grundsätzlich 21. Der Quellport wird zufällig eingestellt aus einem begrenzten Portbereich.
Das Bild 4.6 zeigt wie zunächst die Anfrage als Terminal over Network (Telnet)-Protokoll gesendet wird. Hierbei handelt es sich lediglich um ein textbasiertes Kommando, dass aus der FTP Kommandoliste ausgewählt wird. Während der Anfrage Kommunikation wird die Berechtigung des Zugriffs (Benutzername und Passwort) und die Art der Übertragung festgelegt. Unterschieden wird zwischen aktivem und passiven FTP.

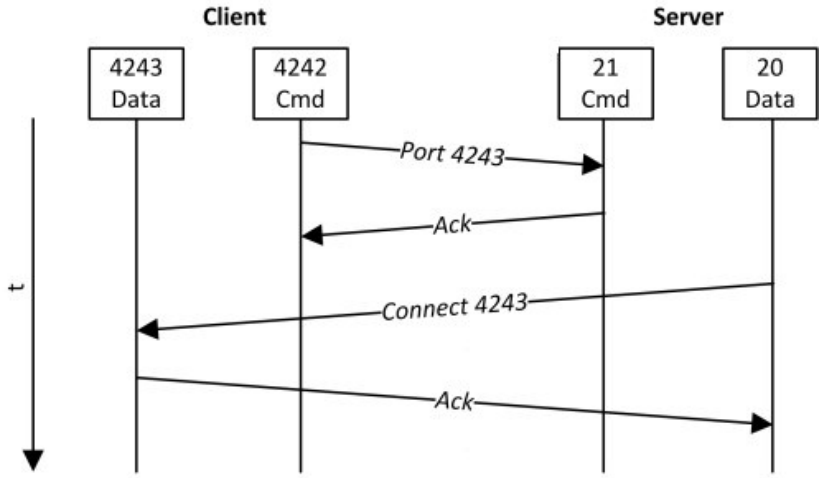


Abbildung 4.6: aktives FTP [18]

Bei dem aktiven FTP wird nach der Anfrage einer Datei über die Kommandoverbindung eine weitere Verbindung hergestellt. Diese Verbindung wird vom Server erstellt und enthält den Quellport 20. Der Ziel Port ist N+1, wobei N für den Kommandoport des Clients steht.

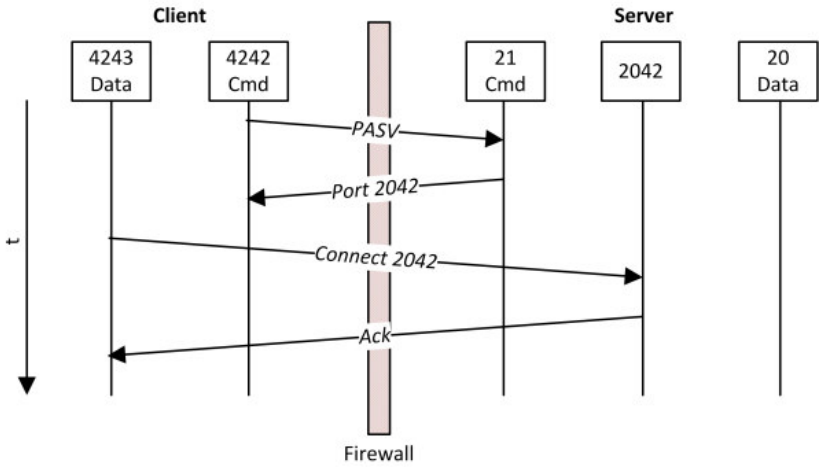


Abbildung 4.7: passives FTP [18]

Sollte eine Firewall oder ein Router das Durchstellen der Ports verhindern, kann die Methode des passiven FTP genutzt werden. In diesem Fall sendet der Client bei seiner Anfrage das Kommando PASV mit.
Der Server sendet eine Portnummer über den Kommunikationskanal dem er die Daten schicken möchte. Anschließend wird die Datenverbindung vom Client etabliert. Siehe 4.7. Da das TCP eine Vollduplex-Kommunikation aufbaut, können beide Partner gleichermaßen senden und empfangen.

5 Java Server – Timo Helsper

Zukünftige Ladestationen haben einen Anschluss an das Internet, um z.B. die Kostenabrechnung der Kunden, Zustand der Ladesäule oder andere Informationen an den Betreiber zu übermitteln. Um die Daten zu verwalten und zu transportieren, werden häufig Server verwendet. Da ein Java Server leicht aufzubauen ist, werden für diese Abschlussarbeit mehrere Java Server erstellt. In den folgenden Abschnitten wird über Java und den Java Server berichtet.

5.1 Java Server

Java bietet durch seine universelle Programmiersprache die Möglichkeit, nahezu plattformunabhängig zu agieren. Durch diese Eigenschaft ist Java im täglichen Leben und somit auch in der Industrie Java stark vertreten. Die Plattformunabhängigkeit ist deswegen wichtig, da Server für spätere Einsätze auf verschiedene Betriebssysteme laufen können. Ein weiteres Merkmal von Java ist, dass es sich hierbei um eine objektorientierte Sprache handelt.
In dem Programmablauf von Java gibt es die Möglichkeit einen Server zu öffnen. Dieses eröffnet die Option, mit externen Programmen zu kommunizieren. Ein einfaches Beispiel für das Nutzen eines Servers ist der Austausch von Nachrichten. Diese Nachrichten können simpel sein (z.B.:"Hallo Welt!") oder mehrere Informationen beinhalten (z.B.:"Hallo Lübeck. Die Temperatur beträgt 23 Grad Celsius"). Wie auch in der Industrie wird der Java Server genutzt, um mehrere Informationen mit PowerFactory auszutauschen. Während der Arbeit mit Java wurden hauptsächlich zwei Quellen zu Rate gezogen. ([1] - Grundkurs JAVA sowie [28] - Java ist auch eine Insel). Als Programmierumgebung wurde Eclipse verwendet [7].

5.2 Ablauf des Servers

Um den Prozess vom Server besser zu verstehen, wird nun in der folgende Auflistung der Ablauf schemenhaft dargestellt. Hierbei ist zu beachten, dass die Punkte 6 bis 8 solange durchgeführt werden, bis die Simulation beendet ist. Anschließend wird mit Punkt 9 fortgefahren.

1. Einlesen der Setup Textdatei
2. Einlesen der lokalen Lastprofile
3. Einlesen der externen Lastprofile
4. Algorithmus initialisieren
5. Erstellen eines Servers
6. Auf Datenverbindung warten
7. Daten verarbeiten
8. Daten zurückgeben
9. Ggf. Server schließen
10. Programm Ende

Einen genauen Programmablauf des Servers in Zusammenarbeit mit der Simulationsumgebung ist unter Kapitel 10.4 zu finden.

5.3 Automatisches Setup des Server

Im Laufe der Entwicklung des Servers ist aufgefallen, dass wiederkehrende Parameter, leicht modifiziert, immer wieder Verwendung finden. Um zu vermeiden, dass sich der Endanwender mit dem Quellcode auseinander setzen muss, wird eine Setup Textdatei eingerichtet. Der Aufbau der Textdatei ist einfach gehalten, sodass der Endanwender und der Java Server den Inhalt gut verstehen kann. In jeder Zeile steht ein variabler Name und ein variabler Wert. Der Name und der Wert sind mit einem Semikolon getrennt. Kommas dürfen, genau so wie Umlaute und Sonderzeichen, nicht eingegeben werden.

Die Setup Textdatei befindet sich leicht zugänglich auf dem RPI und beinhaltet folgende Parameter:

- Der Port auf dem der Server laufen soll.
- Der Name der Textdatei mit dem Lastprofil vom Haushalt.
- Die maximale Simulationszeit von PowerFactory.
- Die IP-Adresse vom FTP Server.
- Der Port vom FTP Server.
- Der Login Name vom FTP Server.
- Das Passwort vom FTP Server.
- Den Namen der Textdatei mit dem PV-Erzeugerprofil.
- Faktor für das PV-Erzeugerprofil.
- Faktor für das Lastprofil vom Haushalt.
- Den Namen des Rapberry Pi (muss einzigartig sein)
- Die maximale Ladeleistung der Ladestation
- Eine eindeutige Identifikation für die Datenübertragung

Beim Starten des Servers werden nun die Parameter der Textdatei an den Server übergeben und in die einzelnen Variablen geschrieben.

5.4 Lastprofil des Haushaltes

In einer Textdatei ist das Lastprofil des Haushaltes abgebildet. Hierbei wird der Tagesablauf in Minutenschritten wiedergegeben. Der erste Wert ist der aktuelle Zeitstempel in Minuten, der zweite Wert ist die aktuelle Leistung in kW. Diese zwei Werte sind mit einem Semikolon getrennt. Das Dezimaltrennzeichen ist ein Dezimalpunkt. Nach den beiden Werte folgt ein Zeilenumbruch. Mit Hilfe der Tabelle 5.1 wird in einem kleinen Ausschnitt ein exemplarisches Lastprofil eines Haushaltes dargestellt.

Zeitstempel;Wert
[...]
682;0.667
683;0.647
684;0.626
[...]

Tabelle 5.1: Ausschnitt aus dem Lastprofil des Haushaltes

5.5 Erzeugerprofil der Photovoltaikanlage

Das Erzeugerprofil der PV-Anlage liegt auf einer SPS. Hierbei stellt die SPS einen FTP-Server zur Verfügung, über den es möglich ist, diese Werte abzufragen. Das Format ist das Gleiche wie beim Lastprofil des Haushaltes (Tabelle: 5.1). Hierbei wurde aber im Laufe der Entwicklung noch die Spannung und der Strom in korrekter Formatierung an die jeweilige Zeile angehängt. Diese Werte können für zukünftige Simulationen genutzt werden und sind derzeit ohne Belangen. Der Java Server baut eine Verbindung mit dem FTP-Server auf und lädt sich dann die Daten auf den Java Server. Die Daten werden für die anschließende Simulation ausgewertet und für eine Abfrage bereitgestellt.

5.6 Von der Datei zum Array

Die drei Dateien (Setup Datei, Lastprofil des Hauses, Erzeugerprofil der PV-Anlage) liegen dem Server am Anfang nur im Rohformat vor. Rohformat bedeutet, der komplette Inhalt einer Text Datei liegt in einer einzigen Zeile vor. Damit die Werte an PowerFactory übergeben werden können, muss die Zeile getrennt werden und die einzelnen Werte in ein Array gespeichert werden. Dieses geschieht bei allen drei Varianten ähnlich.

```
String fileContent = null;
String zeile[] = null;
String ausgabe[][] = null;
int limit = 0;
[...]
zeile = fileContent.split("\n");
limit = zeile.length;
ausgabe = new String[limit][2]
for(int i = 0; i != limit ;i++)
        {
        ausgabe[i] = zeile[i].split(";",2);
        }
```

Listing 5.1: String in ein zweidimensionales Array aufteilen

Dieser Code und alle weiteren enthalten für eine bessere Übersicht keine Kommentare, sondern werden im Text genauer erklärt.
Zu sehen ist im Codeausschnitt, wie der eingelesene String (fileContent) zuerst in Zeilen aufgeteilt wird. Dazu wird im String nach Zeilenumbrüchen geschaut (\n). Auffällig ist hier, dass die speicherprogrammierbare Steuerung im Gegensatz zum Texteditor ein anderen Zeilenumbruch macht. Ein \n setzt den Cursor nur auf die erste Position der Zeile. Ein \r führt ein Return aus. Das heißt der Cursor wandert eine Zeile weiter runter. Während der Editor von Windows den Cursor erst auf den Anfang gesetzt hat und dann eine Zeile runtergegangen ist, hat die SPS noch einen Schritt mehr gemacht. Der Cursor wird auf die erste Stelle gesetzt, dann wird der Cursor eine Stelle nach unten gestellt, und zum Schluss wird der Cursor nochmals auf die erste Stelle gesetzt. Dieses ergibt, je nach Ausleseart, einen Fehler. Im Laufe der Entwicklung werden diese Fehler auf beiden Seiten behoben.

Der Java Server wird so weiter entwickelt, dass beim Einlesen nur ein Zurücksetzen auf die erste Stelle berücksichtigt wird. Parallel dazu wird auf der SPS dieses erneute Zurücksetzen abgeschaltet.
Java unterscheidet beim Einlesen einer Datei zwischen Byte und String. Das Lastprofil des Haushaltes wird stringweise eingelesen. Hierbei achtet Java selber auf \n und \r. Das Einlesen des FTP Servers wird byteweise gemacht. Somit erkennt Java \r oder \n nicht als Sonderzeichen. Als Lösung daraus entsteht, \r aus dem langen String zu entfernen. So können die Bytes problemlos in ein String formiert werden. Würde dieses nicht gemacht werden, wird ein \r oder \n als nicht Sondersymbol abgespeichert. Nachfolgende Programme würden nun fälschlicherweise einen Zeilenumbruch erzeugen.
Wichtig ist außerdem: Die Symbole \r oder \n dürfen nicht in einer Kombination im Variablennamen auftauchen, da es sonst zu Fehler kommen wird.

5.7 Einbindung des Steueralgorithmus

Ziel dieser Simulationsumgebung ist es, für einen Steueralgorithmus die perfekte Umgebung zu schaffen, damit dieser getestet werden kann. Der Steueralgorithmus wird als Java Datei zur Verfügung gestellt, welche in die Server Datei integriert wird. Es stehen damit ausgewählte Funktionen zur Verfügung.
Damit der Server mit dem Algorithmus richtig arbeiten kann, muss ein Ladeagent angelegt werden. Dieser Ladeagent vermittelt über einen externen Server die einzelnen Auslastungen der Ladestationen. Gleichzeitig wird mit dem Ladeagenten festgelegt, wie viel Leistung der Ladestation zur Verfügung gestellt wird.
Für die komplette Erstellung eines Ladeagenten braucht es mehrere Parameter. Auf jedem RPI befindet sich ein Ladeagent, der einen im System einzigartigen Namen hat. Der nächste Parameter legt die maximale Ladeleistung der Ladesäule fest. Dieses ist vergleichbar mit der Leistung einer Säule, z.B. ob es sich um eine 11kW oder 22kW Ladesäule handelt.
Neben der maximalen Ladeleistung wird auch die maximale Transformatorleistung mit übergeben. Zum Schluss benötigt der Ladeagent noch eine Internet Adresse, worüber er mit anderen Ladestationen arbeiten kann. Sind diese Dinge gestellt, kann der Steueralgorithmus in dieser Form starten. (Hierbei handelt es sich um eine Entwicklungsversion, die sich noch am Anfang befindet.)
Während der Simulation wird dem Agenten als erstes die Reserveleistung des Transformators übermittelt. Im nächsten Schritt wird dem Agenten mitgeteilt, zu wie viel Prozent die Ladestation jetzt ausgelastet sein darf.
Der Agent gibt dem Server als Rückgabewert einen Leistungswert, mit dem die Ladestation nun laden soll. Wenn zu viel Leistung aus dem Netz entnommen wird, werden einzelne Ladestationen in der Leistung angepasst.
Ziel des Steueralgorithmus ist es, das Netz nicht zu überlasten.

5.8 Erstellen eines Servers

Unter Java kann man in wenigen Schritten einen Server erstellen. Dafür ist der sogenannte Server Socket zuständig. Er stellt den Port bereit. PowerFactory kann über diesen Port auf den Server lesen und schreiben.
Sobald der Server den Port geöffnet hat, wartet Java auf eine eingehende Verbindung. Dabei ist zu beachten, nicht mehrere Server über den gleichen Port laufen zu lassen, da es hier zu Problemen kommen kann. Diese Probleme können verhindert werden, wenn die Verbindung am Ende beendet und der Port wieder geschlossen wird.

```
ServerSocket serverSocket = new ServerSocket(port);
Socket socket = serverSocket.accept();
[...]
socket.close();
serverSocket.close();
```

Listing 5.2: Server starten und beenden

In diesem aufgeführten Beispiel wird ein neuer Server Socket erstellt. In den Klammern wird der Port festgelegt. Anschließend wird auf eine eingehende Verbindung gewartet. Danach folgt ein Programmablauf, der an späterer Stelle genauer erwähnt wird. (Kapitel: 10.4)
Hat das Programm alle Befehle abgearbeitet, wird die Verbindung geschlossen und im Anschluss kann, je nach Bedarf, der Server auf eine neue Verbindung warten oder den Server schließen.

5.9 Datenaustausch mit PowerFactory

PowerFactory schickt an den Server einen String, der alle Werte für diesen Simulationsschritt enthält. Alle Werte sind mit einem Semikolon voneinander getrennt. Der eingehende Datenstrom wird aufgeteilt und vom String in Double konvertiert. (Kapitel: 5.6)
Nach der Datenverarbeitung werden die einzelnen Doublewerte wieder in ein String konvertiert. Die Strings werden jetzt zusammen gesetzt. Damit PowerFactory die Werte wieder richtig zuordnen kann, werden die einzelnen Werte mit einem Semikolon getrennt.

5.10 Datenverarbeitung

Die erhaltenen Daten werden nun entsprechend der Vorgaben des Endbenutzers mit einem Faktor versehen.
Danach wird vom Trafo ermittelt, wie viel Kapazität dieser noch hat. Diese Kapazität wird an dem Steueralgorithmus übergeben. Sind die Werte vollständig, kommuniziert der Steueralgorithmus mit einem Server in Hamburg. Dort findet die eigentliche Berechnung statt. Hier in Hamburg sammelt der Server alle Werte der Trafoauslastung zusammen und berechnet daraus für die jeweilige Ladestation die Ladeleistung. Die jeweilige Ladeleistung wird an den Java Server zurückgegeben, dieser bereitet die Werte nun durch einen Faktor so auf, dass die Werte an PowerFactory übergeben werden können.

```
String name = "RaspberryPi1"
String server = "Server_Hamburg"
Float pmaxTotal = 35;
Float pmaxLocal = 22;
chargeAgent =
        new ChargeAgent(name, pmaxTotal, pmaxLocal, server);
[...]
chargeAgent.setPmaxTotal(Trafoueber)
actualCharging = chargeAgent.setChargePowerPercent(75);
```

Listing 5.3: Verbindung mit dem Server auf dem sich der Algorithmus befindet

In diesem Codeausschnitt ist exemplarisch aufgeführt, wie der Steueralgorithmus initialisiert und aufgerufen wird. Als Erstes werden zwei Variablen angelegt. Der Name muss einzigartig sein und spiegelt die Ladestation wider. Hinter der Variable ”server”, verbirgt sich die IP-Adresse, unter der der Server für die Berechnung des Steueralgorithmus zu finden ist. Unter ”pmaxTotal” wird die maximale Trafoleistung in kVA angegeben. ”pmaxLocal” spiegelt die maximale Ladestation Leistung in kW wieder.
Sind diese Werte angegeben, so kann ein neuer Ladeagent ”chargeAgent” mit diesen Werten erstellt werden. Dieses muss vor dem Starten des Java Servers geschehen. Als nächstes wird dem Ladeagent die zur Verfügung stehende Restleistung ”Trafoueber” übermittelt. Zum Schluss wird noch angegeben, wie viel Prozent Leistung abgerufen werden soll (Hier 75%). Nach der Berechnung wird der aktuelle Ladewert ”actualCharging” in kW ausgegeben.

6 Simulationsumgebung – Timo Helsper

Um neue Entwicklung zu testen, ist häufig ein realitätsnaher Versuchsaufbau nötig. So können Schwächen in der Entwicklung erkannt und behoben werden. Da diese Aufbauten schnell teuer werden und auch auf Veränderungen unflexibel reagieren können, wird oft vorab am Computer simuliert.
Für die Simulation von Energienetzen wird das Programm PowerFactory eingesetzt. Es bietet die Möglichkeit ein Versuchsnetz nach unseren Bedürfnissen zu erstellen und jeder Zeit anzupassen.

6.1 PowerFactory

PowerFactory ist von der Firma DIgSILENT [4] entwickelt und bietet die Möglichkeit ein Stromnetz zu simulieren. So kann ein kleines Ortsnetz mit einer externen Einspeisung aufgebaut werden oder ein komplettes Stadtnetz mit mehreren Solarkraftwerken. Auch sind eigene Szenarien möglich.
Mit einem bestehenden Stromnetz stehen mehrere Simulationsfunktionen zur Verfügung. Im folgendem Text wird die Lastflussberechnung und anschließend die eingesetzte RMS-Simulation (Zeitbereichssimulation) vorgestellt.

6.2 Aufbauen einer Simulationsumgebung

Bevor eine Simulation durchgeführt werden kann, muss als erstes eine Simulationsumgebung geschaffen werden. Dazu wird ein neues Projekt angelegt. Nach der Vergabe eines Projektnamens wird ein Name für das Netz angegeben und die Nennfrequenz festgelegt. Anschließend öffnet sich die Hauptansicht. Auf der rechten Seite befindet sich eine Symbolleiste. Aus dieser Symbolleiste kann das gewünschte Netz zusammen gebaut werden. Als Beispiel folgt nun ein einfacher Netzaufbau.
Im ersten Schritt wird die Sammelschiene (Einfachsammelschienensystem) ausgewählt. Durch einen Linksklick erfolgt die Platzierung der Sammelschiene. Es werden zwei Sammelschienen untereinander benötigt. Diese beiden Sammelschienen werden über einen 2-Wicklungstransformator verbunden. Beim Versuch, die Leitung mit der Sammelschiene zu verbinden, kommt die Frage vom System, welche Anschlussklemme hier verwendet werden soll. Exemplarisch wird der erste Anschluss verwendet. Anschließend müssen die Spannungsebenen vergeben werden.

Ein Doppelklick auf die Sammelschiene öffnet das Eigenschaftsfenster. Die Leiter-Leiter Spannung muss im kV Bereich eingestellt werden. Das wird an der zweiten Sammelschiene wiederholt, so dass folgende Einstellungen vorliegen:

- Oberspannungsseite: 10kV
- Unterspannungsseite: 0,4kV

Mit einen Doppelklick auf dem Trafo öffnet sich das Eigenschaftsfenster vom Trafo. Durch einen Klick auf den Typ erscheint ein Dropdown Menü. Bei der Auswahl "Globalen Typ auswählen" öffnet sich ein neues Fenster. Folgende Einstellung ist vorzunehmen:

50Hz -> Verteilung -> 10kV -> 0,01 MVA 10/0,4kV

Das Fenster kann wieder mit "Ok" geschlossen werden.
Auf der Oberspannungsseite wird über die rechte Symbolleiste ein externes Netz hinzugefügt. Dieses externe Netz spiegelt das Übertragungsnetz wider und stellt alle benötigten Leistungen bereit. Auf der Unterspannungsseite ist eine "Allgemeine Last" hinzuzufügen.
Durch einen Doppelklick auf die "Allgemeine Last" öffnet sich dessen Einstellung. Unter dem Reiter "Lastfluss" wird eine Wirkleistung von 0,01MW eingestellt. Das Fenster wird über den Button "Ok" geschlossen.
Über den Button "Lastfluss berechnen", welcher sich in der oberen Symbolleiste befindet, kann der Lastfluss berechnet werden. Das Fenster kann mit dem Button "Ausführen" geschlossen werden.
Der Trafo wird rot hervorgehoben, da dieser zu über 100% ausgelastet wird.
Hierbei handelt es sich um ein exemplarischen Aufbau. Weitere Informationen können in der Dokumentation zu PowerFactory gefunden werden [5]. Der verwendete Simulationsaufbau ist größer und wird mit einer anderen Simulationsart berechnet.

6.3 Simulationsarten

6.3.1 Lastflussberechnung

Für die Lastflussberechnung werden die Lastflüsse zu einem festgelegten Zeitpunkt analysiert. Idealerweise werden kritische Zeitpunkte ausgewählt. Mit der Lastflussberechnung wird die Auslastung an der Sammelschiene oder am Trafo angezeigt. Auch kann man Leitungen zu- oder wegschalten, um einen Ausfall zu simulieren.
Mit Hilfe dieser Lastflussberechnung können Schwachstellen in einem System aufgezeigt werden. Es ist auch erkennbar, wie warm oder wie stark eine Leitung ausgelastet ist. Auch ist deutlich, wie hoch die Wirk- und Blindleistungsflüsse sind und in welcher Phase diese liegen.

6.3.2 RMS-Simulation

Für die Root Mean Square (RMS)-Simulation muss ein Zeitbereich angegeben werden. Dieses hat den Vorteil, dass man nun Laständerungen mit berücksichtigen kann. Der Name RMS leitet sich aus der Abkürzung Root Mean Square ab. Dabei handelt es sich um eine Simulation mit Effektivwerten. Bis hierhin unterscheidet sich diese Simulation nicht von der Lastflussberechnung.
Der Vorteil an der RMS-Simulation ist, dass nicht nur ein Zeitpunkt betrachtet wird, sondern ein Zeitraum. Es handelt sich um eine Verkettung von Lastflussberechnung über einen Zeitraum. Jeder Zeitpunkt wird einzeln betrachtet.
Um die Simulation zu starten, muss ein Startzeitpunkt, die Schrittweite und ein Stoppzeitpunkt definiert werden. Wichtig ist hier, dass die zu untersuchenden Variablen vor Simulationsstart ausgewählt werden. Nach der anschließenden Simulation kann mit Hilfe des Diagramms eine Auswertung dargestellt werden.
Aus der Abbildung 6.1 ist das Ergebnis der Messung sichtbar. Erkennbar ist in rot ein Haushalt in Kombination mit einer PV-Anlage. Tagsüber speist der Haushalt in das Netz ein und abends bezieht der Haushalt wieder Leistung aus dem Netz. Die andere Kurve spiegelt den Leistungsverlauf des Trafos über den Tag wider: Die Leistungsentnahme (negativ) und die Einspeisung (positiv) in das übergeordnete Netz.
Insgesamt ist hier ein Tagesablauf simuliert.
Für den späteren Versuch wurden folgende Punkte gewählt:

- Startzeitpunkt 0 für 0:00 Uhr
- Schrittweite 1 für 1 Minute
- Stoppzeitpunkt 1439 für 23:59 Uhr

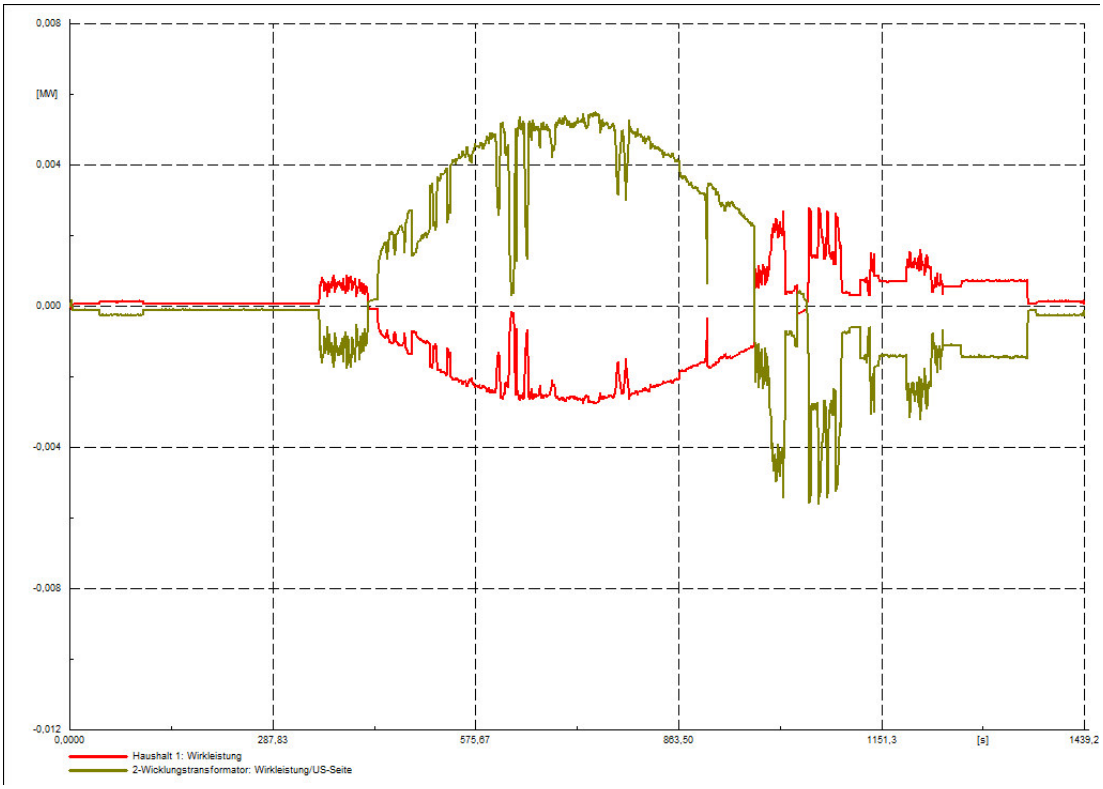


Abbildung 6.1: Diagramm einer RMS-Simulation

6.4 Betrachtung der Lastabweichung

Während mehreren Messungen ist aufgefallen, dass die ausgehende Leistung nicht der eingestellten Leistung entspricht. Um diese Abweichung besser zu erklären, werden drei Messpunkte der Ladestation 2 genauer untersucht. Ladestation 2 ist vom Modell eine allgemeine Last. PowerFactory stellt für jedes Modell ein Technisches Datenblatt zur Verfügung. In diesem Datenblatt [24] wird unter anderem beschrieben, wie die Leistung berechnet wird.
Ausschlaggebend für eine Leistungsabweichung sind Frequenz- und Spannungsabweichung an der simulierten Last. Frequenzabweichungen sind in dem vorhandenen Netz nur minimal vorhanden und sind bei den Stichproben bei 50Hz. Somit kann der Fokus auf die Spannungsänderung gelegt werden.
Spannungsänderungen treten auf, wenn Last zu- oder weggeschaltet wird. Dieses ist später in einer Auswertung sichtbar (z.B. Abbildung: 10.4). Nicht zu vermeiden sind Leitungslängen von 5km, die auch für einen geringen Spannungsabfall sorgen.
In dem Technischen Datenblatt finden sich viele Formeln für die Berechnung dieser Abweichung. Leider werden nicht alle Variablen erklärt, was es rechnerisch nicht nachvollziehbar macht.

6.5 Netzaufbau

Für eine erfolgreiche Simulation muss das Netz mit mehreren Anlagenteilen aufgebaut werden. Folgenden Anlagenteile werden verwendet:

- Externes 20kV Netz als Einspeisung
- Mittelspannungschiene für den Anschluss des Ortstrafos
- Ortstrafo (20/0,4kV 0,05 MVA)
- Niederspannungsschiene als Ortsnetz
- Zwei Abgänge an eine weitere Niederspannungsschiene als Hausanschlusskasten
- Pro Abgang einen Anschluss für den Haushalt, die PV-Anlage und die Ladestation

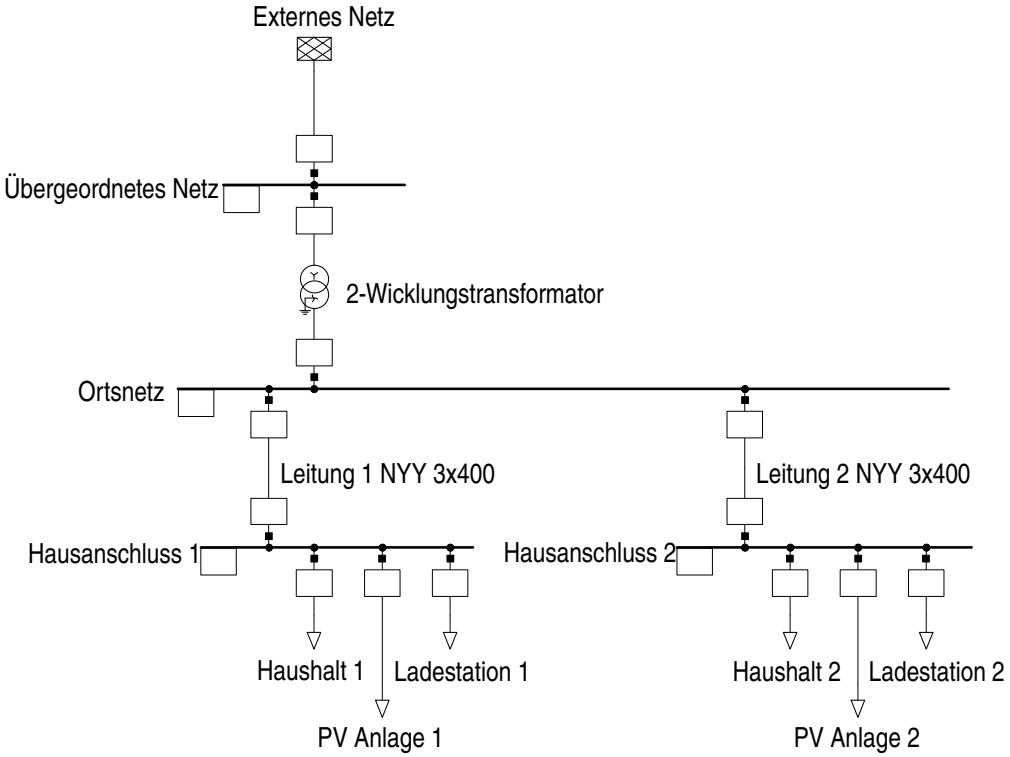


Abbildung 6.2: Aufbau des Netzes

Aus der Abbildung 6.2 können die aufgezählten Anlagenteile wieder gefunden werden. Sie stellen ein kleines Netz dar, mit zwei Haushalten und je einer PV-Anlage sowie einer Ladestation. Das externe Netz dient dazu, fehlenden bzw. überschüssigen Strom zu importieren bzw. exportieren. Die Mittelspannungsschiene ist das Verbindungsstück zwischen Ortstrafo und dem externen Netz. Der Ortstrafo kann von der Leistung variabel angepasst werden. Er stellt die Schnittstelle von der Mittelspannung zur Niederspannung da.

Bei der späteren grafischen Auswertung ist folgendes zu beachten: Der Graph kann invertiert sein, je nachdem, ob man den Messpunkt im Hochspannungsbereich oder im Niederspannungsbereich gelegt hat. Der Lastfluss muss dementsprechend immer geprüft werden. Weiteres im Kapitel 6.8.
Der Ortstrafo speist in das Niederspannungsnetz ein, in der sich die Niederspannungsschiene befindet. Diese simuliert das bestehende Ortsnetz und ist als Ideal zu betrachten. Verluste werden an den Abgängen berechnet. Diese Niederspannungsschiene dient dazu, mehrere Haushalte zu sammeln und mit dem Trafo zu verbinden.
Die Verbindungsleitung zwischen der Niederspannungsschiene und der Stromschiene für den Haushalt wird so konfiguriert, dass das Material und die Länge mit angegeben werden, um Verluste simulieren zu können.
An den Stromschienen für die einzelnen Haushalte liegen drei allgemeine Lastabgänge dran. Die Lastabgänge sind anpassbar. Hierbei kann es sich um ein Haushaltslastprofil, Erzeugerlastprofil oder um die Ladestation handeln. Auf der späteren Grafik (Kapitel: 6.8) sind Lastflüsse, die aus der Stromschiene in den Lastabgang fließen, positiv. Lastflüsse aus dem Lastabgang in Richtung Stromschiene sind negativ.

6.6 Verdrahtungsplan

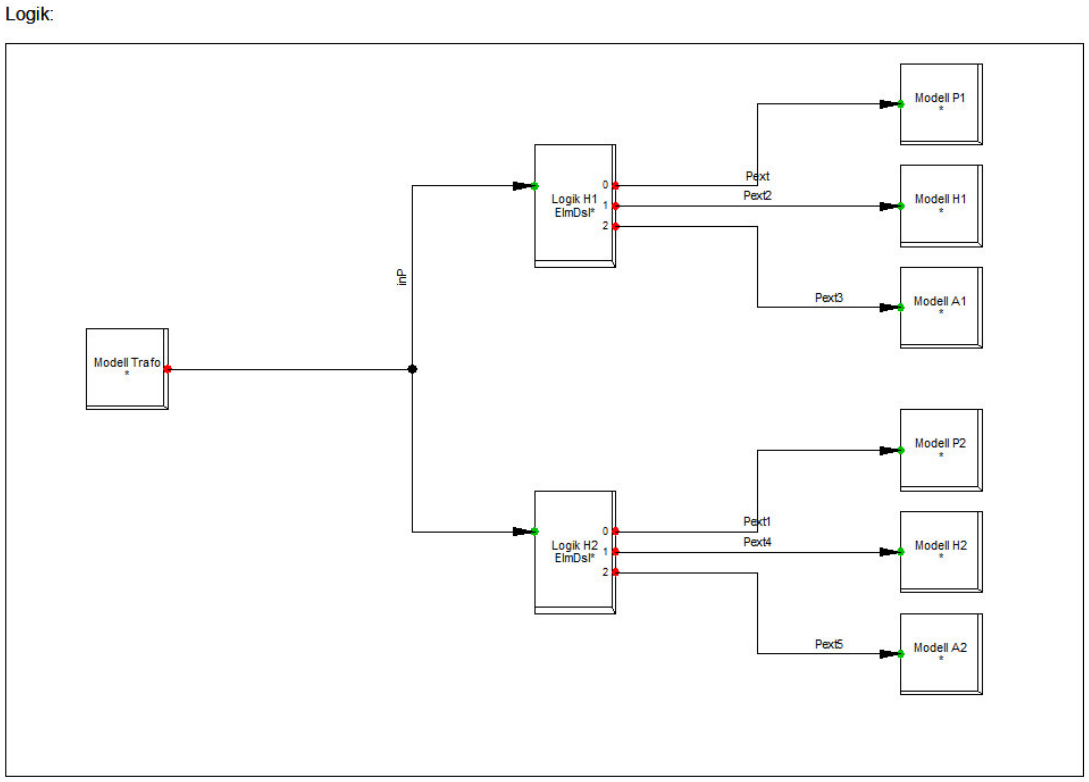


Abbildung 6.3: Übersicht Verdrahtungsplan

Im Verdrahtungsplan bekommen die Lastabgänge ihre Werte vom Controller für den aktuellen Simulationsschritt zugewiesen. Der Controller ist eine Blockdefinition mit hinterlegter Funktion. Aus der Abbildung 6.3 ist zu erkennen, dass vom Trafo (Links) Werte zu dem Controller gehen (Mitte). Von dort aus gehen drei Abgänge zu jeweils einem weiteren Block (Rechts). Die drei Blöcke spiegeln einen Haushalt wider (PV-Anlage, Haushaltslast, Ladestation). Der Datenfluss geht hierbei von links nach rechts. In den nächsten Unterkapiteln werden die einzelnen Blöcke genauer erklärt.

6.6.1 Einschub für Verbraucher und Erzeuger

In der Abbildung 6.3 sind am linken und rechen Rand Blöcke. Diese Blöcke sind sogenannte Einschübe, welche zum Beispiel Anlagenteile, wie einen Trafo, darstellen. Hier wird festgelegt, welche Werte ausgegeben werden sollen. Die Zuordnung, dass es sich hierbei um ein Trafo handelt, findet an anderer Stelle statt.

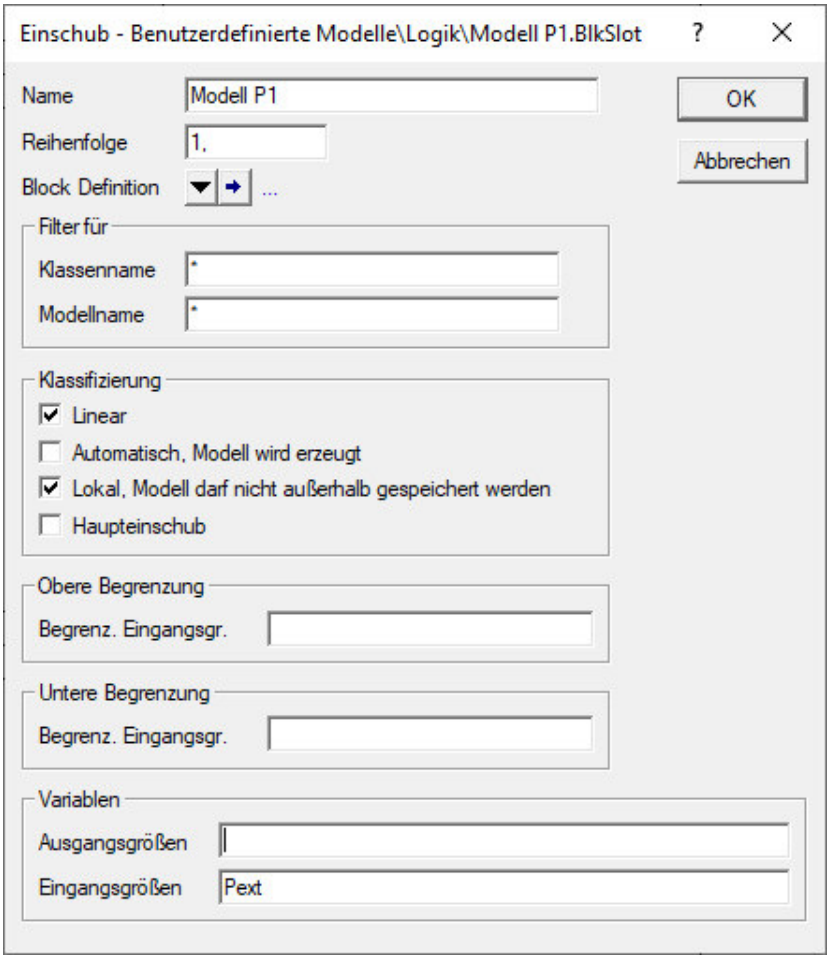


Abbildung 6.4: Modell eines Einschubes

Abbildung 6.4 zeigt einen Einschub mit geöffnetem Fenster. Exemplarisch wird das Modell der PV-Anlage im Haushalt 1 gewählt. Beachtung sollten die Variablen erhalten, welche sich am Ende des Fensters befinden. In die Zeilen für Ausgangsgrößen bzw. Eingangsgrößen können bestimmte Variablennamen eingetragen werden. Der Variablennamen legt fest, welcher Wert ausgelesen oder eingelesen werden soll. Da dieser Einschub die erste PV-Anlage widerspiegelt, bekommt die Anlage eine Eingangsgröße. Der Name "Pext" ist nicht zufällig gewählt. Jedes Anlage hat unterschiedliche Variablen. Jede Variable hat eine bestimmte Eigenschaft und muss zu der Anlage passen. Zum Beispiel kann der PV-Anlage keine Drehzahl zugeordnet werden.

6.6.2 Blockdefinition

Die Blockdefinition bietet die Möglichkeit, eigene Funktionen in das Programm zu integrieren. Es werden die Werte an eine Funktion in einer Dynamic Link Library (DLL)-Datei übermittelt und die DLL-Datei kann Werte zurückgeben. Im Beispiel Listing:6.1 ist eine solche Zuordnung aufgeführt.

```
1 inc(outH)=0
2 inc(outP)=0
3 inA=time()
4 inB=inP
5 outAH=outH
```

Listing 6.1: Codezeile in der Blockdefinition

Zeile 1 und Zeile 2 spiegeln die Anfangsbedingung wider. Hier werden alle Werte auf 0 gesetzt. In der Zeile 3 wird die Variable inA auf aktuelle Simulationszeit gesetzt. Zeile 4 bekommt über die Variable inP die aktuelle Auslastung vom Trafo. Dieser Wert wird dann in die Variable inB geschrieben Die Zeile 5 schreibt den Ausgang für die Haushaltslast. inA, inB und outH werden in der DLL-Datei weiter verarbeitet.
Die Funktion ruft mit jeden Simulationsschritt die DLL-Datei auf. Hierbei wird die aktuelle Simulationszeit und die Trafobelastung übergeben. Als Rückgabewert wird die aktuelle Last für den jeweiligen Simulationsschritt zurückgegeben. Bei diesem Code handelt es sich um eine Erweiterung von der Arbeit von Herrn Buchmann.

6.7 DLL Datei

In der DLL-Datei befinden sich die Funktionen, die von der Blockdefinition aufgerufen werden. Einmal aufgerufen, wird eine Verbindung zu einem Java Server hergestellt. Steht diese Verbindung, werden die Werte (aktuelle Simulationszeit, aktuelle Trafolast) an den Java Server übergeben. Der Server gibt die aktuellen Leistungswerte für das Haus, die PV-Anlage und die Ladestation wieder zurück. Diese aktuellen Lastwerte werden an die Blockdefinition geschickt und die Verbindung mit dem Java Server beendet.
Für jeden Haushalt gibt es eine eigene Funktion in der DLL Datei, da diese auf einen anderen Server zugreift.

6.8 Erstellen einer grafische Auswertung

Nach dem Ende der Simulation können in der grafischen Auswertung die Lastflüsse betrachtet werden. Für eine Untersuchung müssen hierbei die Variablen vor Simulationsbeginn ausgewählt werden. Anschließend wird eine Grafik erstellt und das Diagramm mit einem Namen versehen. Die Skalierung wird später durchgeführt. In die untere Tabelle werden die Kurven eingetragen, die genauer angeschaut werden sollen. Danach wird das Fenster geschlossen.
Nach der erfolgreichen Simulation gibt es in der oberen Symbolleiste im Grafikbereich zwei Skalierungsbuttons, die eine automatische Skalierung ermöglichen und somit die Grafik übersichtlich gestaltet.

7 Automatisierung – Christian Ziegelmann

Das elektrotechnische Wörterbuch vom, Deutsches Institut für Normung (DIN) und International Electrotechnical Commission (IEC) 60050-351 [1] definiert den Begriff „selbsttätig/automatisch“: „Einen Prozess oder eine Einrichtung, der oder die unter festgelegten Bedingungen ohne menschliches Eingreifen abläuft oder arbeitet.“

7.1 Automatisierungspyramide

In der Literatur wird die Automation meist über die Analogie als Pyramide dargestellt. In dieser sollen zwei Aspekte herausgestellt werden.
Zum einen die Datenmenge, zum anderen der hierarchische Aufbau. Die Pyramide hat an ihrer Spitze die Unternehmensführung und fußt auf der Produktion. Hierbei stellt das breitere Fundament das größte Datenaufkommen mit Anforderungen an Datenübertragung und Geschwindigkeiten bis zur Echtzeit. Die nach oben aufsteigenden Ebenen, sind jeweils auch hierarchisch übergeordnet. Sie weisen Prozesse an und kontrollieren diese. Die Auswertung der Daten geht über die technische Fertigung und Prozesskontrolle hinaus, bis in die Unternehmensleitung und Produktplatzierung hinein. So wird versucht die gesamte Firmenkonstruktion einzubeziehen.
Dabei können Produktanpassungen oder Produktionsmengen dem Markt automatisiert angepasst werden und auch automatisch erkannt werden.
Die Vorstellung einer komplett autarken Produktion liegt hier nicht fern und findet auch statt. Ein einfaches Beispiel wären überwachte Lagerbestände, die die Produktion drosseln oder beschleunigen ohne Mitwirken von Menschen. Der Mensch spielt in dieser Darstellung der Automatisierung eine einmalige Rolle, in der er den Markt analysiert und Produkte entwickelt, die Absatz finden. Anschließend laufen die Produktionsprozesse automatisiert ab.

Abbildung 7.1: Automatisierungspyramide [23]

7.2 Industrie 4.0

Mit dem Begriff der Industrie 4.0 soll eine erneute Industrielle Revolution ausgerufen werden. Wie es einst die Dampfmaschine als treibender Faktor war.

- **Industrie 1.0:** Anfang des 19. Jahrhunderts führten Maschinen angetrieben durch Dampfmaschinen einzelne Bewegungen und Arbeiten aus, wie Webstühle.
- **Industrie 2.0:** Ende des 19. Jahrhunderts wurden Fließbandarbeit eingeführt. Durch die massenhafte Fertigung auf solchen Fließbändern, kommt es zu beschleunigten Fertigungen und erheblich geringeren Produktionskosten pro Stück.
- **Industrie 3.0:** Ab den 70er Jahren wurden Industrie-PCs und Mikroprozessoren in die Fertigungsprozesse eingebunden. Eine weitere Verkürzung der Prozessabläufe war die Folge. Hinzukamen genauere Fertigung und das Auswerten von Prozessen durch Datenerfassung.

Der Begriff der Industrie 4.0 soll eine Vernetzung der Maschinen und Geräte darstellen, die der Industrie 3.0 vergleichbar ist. Die Erweiterung findet auf einer höheren Ebene statt. Dies macht die Differenzierung dieser Schlagworte etwas kompliziert.[23]
Basierend auf der Idee des kompletten Datenaustausches sollen Geräte eigenständiger und autonom arbeiten können. Dazu werden den Maschinen und Geräten mehr Rechte und Möglichkeiten der Kommunikation eingeräumt. Die benötigten Informationen sollen eigenständig aus dem Netzwerk entnommen werden. Selbstdiagnosen sind möglich und bei Bedarf können Bauteile eigenständig geordert werden. Die Daten des Teilnehmers werden dem Netzwerk für andere zur Verfügung gestellt. Dies soll die autarke Fertigung ein Stück näher bringen.

7.3 Internet of Things

Im privaten Sektor ist der Begriff des Internet of Things (IoT) eng mit der Idee Industrie 4.0 verwoben. Hierbei handelt es sich meist um Geräte aus dem Konsumerbreich, die ihre Dienstleistung ohne Eingreifen von Menschen leisten sollen. Ein gutes Beispiel ist der Kühlschrank, der die Milch online bestellt, sobald diese leer wird.

7.4 Automatisierung einer Sonnensimulation

Für die Entstehung der Wettersimulation in den Laborräumen der TH-Lübeck wurde darauf geachtet eine Steuerung zu wählen die allen Ansprüchen der Industrie 3.0 genügt.

- **Industrie-PC (IPC):** Es soll möglichst lange Laufzeiten gewährleistet werden. Ausfälle durch Programmfehler werden durch Laufzeitkontrollen überwacht.
- **Netzwerkfähigkeit:** Ermöglicht die Integration der Steuerung in einem Netzwerk, um Daten extern auszulesen und Eingaben zu übergeben.
- **Erweiterbarkeit:** Die Summe von Ein- und Ausgängen kann den Bedürfnissen des Prozesses angepasst werden.
- **Human Machine Interface (HMI):** Eine visuelle Darstellung der aktuellen Prozesse, sowie das Beeinflussen des Ablaufes.
- **Autarker Betrieb:** Dem System ist es möglich seinen Ablauf, ohne das Einwirken eines Menschen, dauerhaft auszuführen.

7.5 Die Beckhoff CX9010

Zur Programmierung der Beckhoff CX9010, wird The Windows Control and Automation Technology (TwinCAT) genutzt. Diese Programmiersoftware ist eine Visualstudio-Applikation.
Die SPS kann in den üblichen Programmiersprachen wie Funktionsplan (FUP), Anweisungsliste (AWL), Kontaktplan (KOP) und Strukturierter Text (ST) programmiert werden. Weiterhin kann eine visuelle Programmieroberfläche angelegt werden. Mit dieser lässt sich die Steuerung über ein externes Display oder direkt aus der Programmierumgebung bedienen.
In unserem Fall werden Programmierung und Bedienung aus der Software TwinCAT heraus ausgeführt, welche auf einem Windows 7 Desktop PC installiert wurde.
Die Kommunikation zwischen der SPS und dem Desktop PC findet über TCP/IP statt. Die SPS ist standardmäßig mit einer Netzwerkkarte versehen.
Außerdem wurde die SPS mit Einschüben, für digitale Ein- und Ausgängen, sowie speziellen Multimeterkarten erweitert. Diese Karten haben Eingänge über die Strom und Spannung direkt gemessen werden können. In der Programmierung ist keine explizite Konvertierung von Werten nötig, diese werden korrekt skaliert übergeben.

8 Datenaustausch – Timo Helsper

Um einen Datenaustausch zwischen mehreren Computern zu realisieren, stehen verschiedene Datenübertragungsarten zur Verfügung. Die Funktionweisen der Übertragungsarten werden im Kapitel 4 näher betrachtet.
In diesem Kapitel wird der Datenaustausch sowie der gewählte Weg der Daten erklärt.

8.1 Vorgaben

Zur Verfügung steht ein Ethernet Netzwerk. Alle Teilnehmer bekommen eine eindeutige IP und sind über ein Switch miteinander verbunden. Der meiste Datenverkehr findet im Netz der TH statt und benötigt somit keine Anbindung an das World Wide Web (WWW). Nur der für uns zur Verfügung gestellte Steueralgorithmus benötigt eine Verbindung zu einem Server in Hamburg.

8.2 FTP

Der Beckhoff Computer bietet uns die Möglichkeit, Daten auf einem FTP Server bereitzustellen. Der Java Server kann sich auf den FTP Server mit Anmeldename und Anmeldepasswort anmelden und die benötigten Daten herunterladen.

8.3 DLL-Datei

Windows stellt verschiedene DLL Dateien zur Verfügung. In einer DLL Datei können verschiedene Funktionen hinterlegt sein. Da meist mehrere Funktionen hinterlegt sind, spricht man von einer Bibliothek. Programme können nun eine Verbindung zu dieser DLL Datei aufbauen und deren Funktionen benutzen. Dadurch ergibt sich der Vorteil, dass einzelne Funktionen nicht mehr selber im Hauptprogramm hinterlegt sein müssen und somit Speicherplatz sparen. Der Nachteil dabei ist, dass ohne die DLL Datei das Hauptprogramm nicht mehr funktioniert.
Die benötigte DLL Datei ist nur für Funktionen in PowerFactory vorgesehen. Hierbei können verschiedene Projekte auf die DLL Datei zugreifen. PowerFactory bietet somit die Möglichkeit, auch projektübergreifend dieselbe Funktion abzufragen.

8.4 Datenfluss

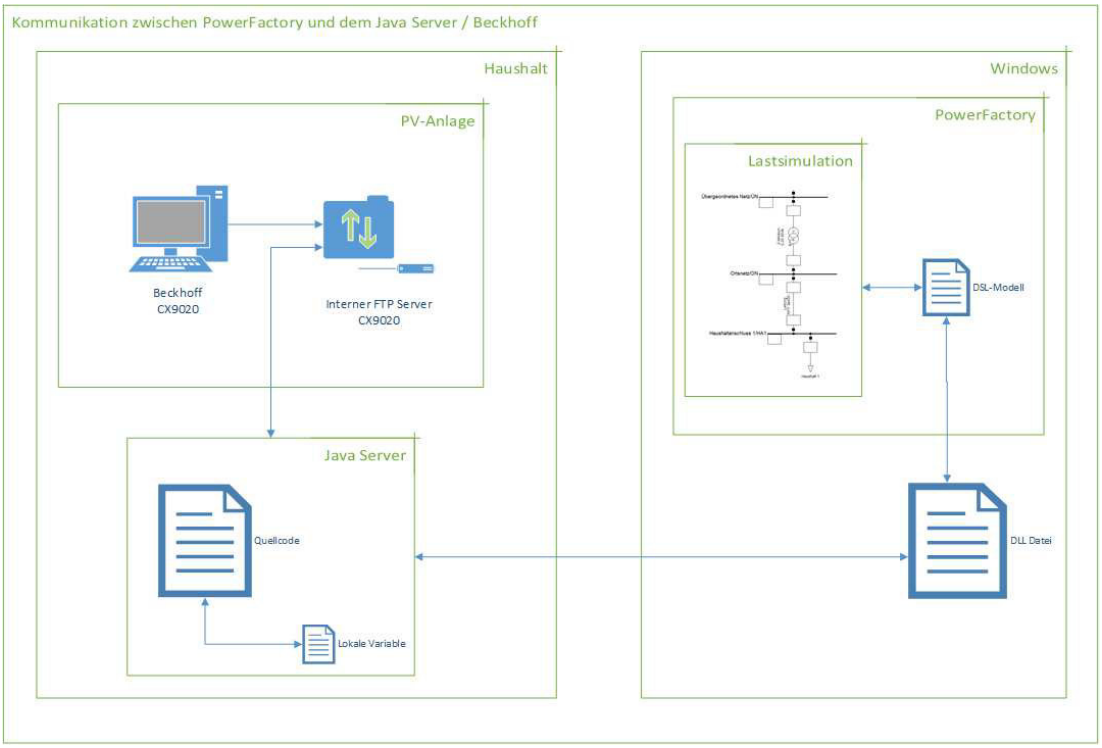


Abbildung 8.1: Übersicht Datenfluss

Aus der Abbildung 8.1 ist zu erkennen, wie die Kommunikation zwischen PowerFactory und dem Java Server abläuft. Bevor die Simulation in PowerFactory ausgeführt werden kann, müssen die Java Server gestartet werden.
Bei der Initialisierung der Server werden die Lastprofile, die als Textdatei hinterlegt sind, eingelesen. Zusätzlich wird eine Verbindung zu dem FTP Server des Sonnensimulators aufgebaut, um die Leistungswerte der Photovoltaik abzufragen.
Sind die Werte alle eingelesen, wartet der Server auf die erste eingehende Verbindung von PowerFactory. PowerFactory verarbeitet seine Simulation in mehreren Schritten. In jedem einzelnem Schritt wird in PowerFactory eine Blockdefinition aufgerufen (Kap.: 6.6.2). Diese Blockdefinition ist z.B. mit dem Lastabgang eines Haus verbunden.
Die Blockdefinition enthält eine Funktion, welche eine DLL Datei aufruft. Die DLL Datei (Kap.: 6.7) hat mehrere Funktionen. Je nach Funktionsaufruf wird zu einem bestimmten Java Server eine Verbindung aufgebaut.
Der Java Server bekommt nun mehrere Parameter. Diese Parameter verarbeitet der Java Server und das Resultat wird an die DLL Datei zurückgeschickt. Anschließend wird das Ergebnis aus der DLL Datei an die Blockdefinition übergeben. Zum Schluss schickt die Blockdefinition das Ergebnis an den jeweiligen Lastabgang. Damit kann jetzt der restliche Lastfluss in PowerFactory simuliert werden.

8.5 Open Platform Communications Unified Architecture

Die Geräte werden mit dem Einzug von Industrie 4.0 immer weiter vernetzt. Das hat die Folge, dass ein Gerät an mehrere, andere Geräte sendet oder mehrere Geräte an ein Gerät senden. Hierbei fallen nicht nur viele Daten an, auch muss das Datenübertragungsprotokoll bekannt sein.
Da setzt Open Platform Communications Unified Architecture (OPC UA) an. Plattformunabhängig tritt OPC UA als Schnittstelle zwischen den Geräten auf.
OPC UA tauscht, soweit eingestellt, automatisch zwischen mehreren Geräten Informationen aus. Man spricht auch von einer Machine-to-Machine (M2M) Kommunikation. Das Protokoll für die Kommunikation ist TCP/IP.
OPC UA bietet die Möglichkeit, verschiedene Protokolle zu verstehen. Dabei handelt es sich um eine Middelware, die sich in der Industrie bewährt hat. Dieses zeigt sich auch in dem Punkt, dass Hersteller ihre Protokolle für OPC UA bereitstellen, so dass eine Anwendung mit mehrere Systemen möglich ist. Somit bietet der OPC UA-Server eine Möglichkeit, Daten zu sammeln und für andere Teilnehmer zur Verfügung zu stellen. Diese und weitere Informationen ist über die Firma [12] - ascolab GmbH oder im Buch [21] - OPC Unified Architecture zu finden.
Versuchsweise wird eine Datenverbindung zwischen einem Windows Computer und einem Beckhoff Computer hergestellt. Der Windows Computer ist der Zentrale OPC UA-Server, die Beckhoff der OPC UA-Client. Dieses gestaltet sich anfangs schwierig. Beim Aufsetzen des OPC UA-Servers müssen zusätzliche Treiber installiert werden, damit das Protokoll von der Beckhoff verstanden wird.
Es stellt sich heraus, dass PowerFactory nur gegen Aufpreis OPC UA unterstützt. Deswegen wird das Programm nicht weiter verfolgt.
Im Laufe des Projektes hat cbb software GmbH [13], für die Kommunikation des Algorithmus, Open Process Communication Simplified Architecture (OPC SA) [14] entwickelt. Auf Grund des weiten Fortschritts der Abschlussarbeit und dessen zeitlichen Limits wird entschlossen, OPC SA nicht zu implementieren. OPC SA kann direkt auf dem Raspberry Pi (RPI) installiert werden und benötigt, anders als die jetzige Lösung, keinen externen Server.

9 Erweiterung der Sonnensimulation – Christian Ziegelmann

9.1 Programmiersprachen

Es gibt vier, hauptsächlich in der SPS Programmierung vorkommende Programmiersprachen. Jeder Funktionsblock kann dabei nur in einer Sprache geschrieben werden. Aufgerufen werden die Funktionsbausteine unabhängig ihrer Programmiersprache hintereinander. Auch können Funktionsbausteine andere Funktionsbausteine in ihrem Ablauf integrieren, unabhängig ihrer Sprache.

FUP

Funktionsplan oder Funktionsbausteinsprache (FBS) ist eine grafische Programmiersprache. Es werden die einzelnen Funktionen mit einender durch Linien verknüpft. Die Funktionen, die als Blöcke dargestellt werden, haben für ihre Ein- und Ausgangsvariablen jeweils einen Anschluss. An diese Anschlüsse können alle Datentypen angeschlossen werden, wenn sie, der des Bausteins entsprechen.
Diese Programmiersprache bietet Vorteile bei logischen Verknüpfungen und Nachvollziehbarkeit der Abläufe.

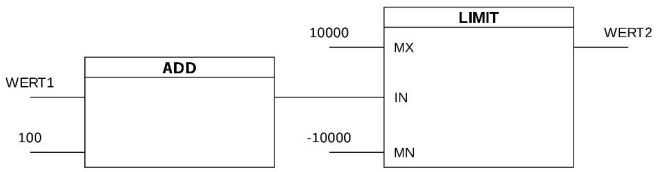


Abbildung 9.1: FUP Beispiel [26]

KOP

Kontaktplan oder Ladder Diagram (LD) ist eine grafische Programmiersprache, die sich an dem Schaltplan orientiert. So kann hier in den meisten Fällen ein Schaltplan direkt abgebildet werden, um seine Funktion in die SPS zu implementieren.

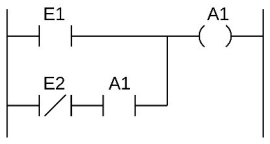


Abbildung 9.2: KOP Beispiel [22]

AWL

Anweisungsliste oder Instruction List (IL) ist eine textbasierte Programmiersprache. Sie besteht aus einer Befehlsabkürzung und einem Wert in jeder Zeile. Sie wird hauptsächlich eingesetzt, um Eingabewerte oder gespeicherte Werte miteinander zu verknüpfen. Größere Anweisungen sind unübersichtlich und schwer nachzuvollziehen. Oft gleichen sich die Befehlskürzel der verschiedenen Hersteller nicht.

```
LD      INPUT 1
AND     INPUT 2
ST      OUTPUT
```

Listing 9.1: AWL Beispiel

ST

Strukturierter Text oder Structured Control Language (SCL) ist eine textbasierte Programmiersprache ähnlich einer Hochsprache. Diese Programmiersprache ist umfangreicher als die AWL. In ihr lassen sich rekursive Programmabläufe implementieren und mathematische Funktionen einfacher abbilden.

```
IF      Bedingung1 THEN
        Anweisung1;
ELSIF   Bedingung2 THEN
        Anweisung2;
ELSE
        Anweisung3;
END_IF;
```

Listing 9.2: ST Beispiel

9.2 Funktion der Steuerung

Die Sonnensimulation kann über ein Baustein der Beckhoff-Datenbank direkt die Sonnenbahn errechnen.
Hierfür notwendig ist die Eingabe von Datum und Uhrzeit, sowie Längen und Breitengrad des Standortes auf der Erde. Es wird der jeweilige Sonnenstand ausgegeben, der mittels einer Formel in die Strahlungsstärke der Lampen umgewandelt wird. Hierbei entspricht der höchste Sonnenstand der maximalen Lampeneinstellung.
Um ein genaueres Ergebnis zu erzielen wird eine Plasma Lampe erst ab einem bestimmten Sonnenstand hinzu geschaltet. Diese Werte wurden durch Versuche heraus erarbeitet.
Die nun im Modul erzeugte Energie, wird von einem Wechselrichter mit Maximum Power Point (MPP)-Tracker zu einer 12V Batterie zugeführt. Eine Lampe, zum entladen der Batterie, kann angeschaltet werden.
In der Benutzeroberfläche dem Man Machine Interface (MMI) werden sowohl die oben Angegebenen Werte der Zeit, sowie der zu simulierende Ort eingestellt. Während die Anlage im Betrieb ist lassen sich in Echtzeit die Werte der bereitgestellten Leistung, sowie Strom und Spannung ablesen.
Aus der Simulation heraus, lässt sich eine Nebelmaschine individuell einschalten. Der produzierte Dampf bewirkt eine Wolkenbildung zwischen Glasscheibe und PV-Modul. Die Wolken reduzierten den Lichteinfall auf das Modul, woraufhin die Leistung einbricht.
Die Steuerung für die Wettersimulation wurde hauptsächlich in der Programmiersprache FUP durchgeführt. Einzelne Funktionsbausteine für Sonnenstand, Zeitverlauf und eine Visualisierung waren programmiert.

9.3 Erweiterung der Steuerung

Um die Daten aus der Sonnensimaltion nutzen zu können, müssen diese der Simulationsumgebung zugänglich gemacht werden. Werte wie Zeit und weitere von der SPS erfassten Parameter sollen von der Simulationsumgebung ausgelesen werden.
Die Beckhoff-SPS bietet diverse Möglichkeiten der Netzwerkkommunikation. Für die Übertragung von Informationen muss zunächst eine Netzwerkverbindung zwischen der SPS und der Simulationsumgebung eingereicht werden.

Anforderungen an die Erweiterung sind:

1. Auswahl einer Speichermethode
2. Erstellung einer Datenerfassungsfunktion
3. Erweiterung der Bedienungsoberfläche
4. Bereitstellen der Daten via Netzwerkzugriff

9.3.1 Auswahl einer Speichermethode

Für die Aufnahme der Werte wird eine Datei angelegt die alle Werte aufnimmt. Dies hat mehrere Vorteile. Die Simulation kann beliebig oft mit den gleichen Werten wiederholt werden. Zusätzlich lassen sich die Werte extern mit anderen Programmen verarbeiten, um auch andere Auswertungen der Sonnensimulation zu ermöglichen. Es gab mehrere Kriterien, die hauptsächlich zur Entscheidung herangezogen wurden.

- **Wenig Speicherbedarf**
  Als Hauptkriterium zur Auswahl des richtigen Formates stand zu erst ein geringer Speicherplatzbedarf. Ziel ist es, dass die SPS ohne erweiterten Speicher und ohne, dass die Werte auf einem Server abgelegt werden, arbeiten kann. Die SPS bleibt somit weiterhin in allen Funktionen autark und man benötigt erst dann Netzwerkzugang, wenn hier von Außen darauf zugegriffen werden soll. Siehe Kapitel 5.3 Automatisches Setup des Server.

- **Plattformübergreifende Kommunikation**
  Zweites Kriterium war die plattformübergreifende Lesbarkeit und Kompatibilität des Formates. Diese Daten sollen in diesem Projekt auf einem Java-Server verarbeitet werden. Aber auch eine Darstellung in Microsoft Exel oder anderen Kalkulationsprogrammen soll möglich sein.

- **Freiheit von variablen Längen und Typ**
  Da die Reihenfolge und die Datentypen nicht genau bekannt sind wird eine hohe Flexibilität benötigt.

Alle Kriterien werden von der Comma Separated Values (CSV)-Datei erfüllt.

Dateiformat

CSV ist ein Datentyp ohne allgemeinen Standard für die Formatierung, jedoch wird es im RFC 4180 grundlegend beschrieben. Die Beschreibung ist aus dem Jahre 2005. Die lange Existenz ist auch ein Grund für seine starke Verbreitung.
In einer CSV-Datei werden eine Reihe Schriftzeichen hintereinander erstellt wie in einer Text-Datei. Ihre Kodierung ist nicht genormt, meist kommt 7-Bit American Standard Code for Information Interchange (ASCII) zum Einsatz. Ein wiederkehrendes Trennzeichen gibt die Möglichkeit, Werte zu separieren. Länge oder Datentyp von einzelnen Datensätzen sind hierbei nicht festgelegt. Es gibt weiter keine Information im Datensatz, der hierüber Auskunft gibt, dass macht dieses Format sehr schlank.
Da die Daten, wie in einer Text-Datei üblich, zeilenweise angeordnet sind, können diese Dateien auch von einem gewöhnlichen txt-reader geöffnet werden. Erkennt der PC die Datei automatisch und beschriftet diese als *.txt*, werden beim öffnen die Zeilenumbrüche nicht angezeigt. Nach einem Zeilenumbruch handelt es sich um den beginn eines neuen Datensatzes.
Dieser Zeilenumbruch ist in der CSV-Datei mit $CRLF$ definiert und kommt im ASCII unter Windows vor. In Java würde dieser mit $/r/n$ ausgelesen. Es verbirgt sich jedoch immer der gleich Hexadezimalsystem (Hex)-Wert dahinter.
Durch eine vorangegangene Einigung der zu übergebenden Werte und ihrer Reihenfolge, können diese in der Datei mit z.B. Semikolon getrennt werden. Und sind so einzeln auslesbar und Zeilenwiese zusammengehörig.

Konzipierung der Wertetabelle

In der Wertetabelle sollen alle Werte gespeichert werden können, die die Sonnenstandsimulation erfasst. Für die Berechnungen in der Simulation sind derzeit nur Zeitpunkt und Leistung des PV-Modules wichtig. Für die Übersichtlichkeit und um Fehler zu vermeiden, wurden nur die Werte aufgenommen, die momentan benötigt werden.
Die genutzte Version von PowerFactory simuliert in Minutenintervallen. Zu diesem Zweck wird ein Tag in Minuten aufgeteilt. Während des Verlaufes der Simulation werden diese von 1 bis 1440 hoch gezählt. Um die Zuordnung der Werte richtig zu gestalten, haben wir diese Art der Zeiterfassung übernommen. Sie wird pro Datensatz als erstes in die CSV-Datei geschrieben. Hierbei könnte man auch von einer fortlaufenden Nummerierung sprechen.
Der zweite von uns benötigte Wert wird durch die Anfügung eines Semikolon getrennt. Um auch hier auf minimalen Speicherplatz zu achten, werden die Werte auf zwei Nachkommastellen beschränkt. Die Länge und Größe hat keinen Einfluss auf die Weiterverarbeitung.
Anschließend wird ein Zeilenumbruch angefügt. Damit ist ein Datensatz geschrieben. Dieser Datensatz spiegelt eine Momentaufnahme in der Minute wider und wird zu Beginn der Minute ausgelöst.

Zeitstempel	Leistung des Modules	Wert X
in Minuten des Tages	in Watt	Einheit Wert X

Tabelle 9.1: Aufbau einer Zeile der csv Datei

[...]
801;2.49$CRLF$
802;2.43$CRLF$
804;2.38$CRLF$
[...]

Tabelle 9.2: Ausschnitt aus dem Aufzeichnung der Sonnensimulation

Der Ausschnitt zeigt die Werte eines Tages im November in der Zeit ab ca. 13 Uhr.
Während des Projektes wurde die CSV-Datei um mehrere Werte erweitert. Da die in der Simulation erstellte Datei, auch in anderen Bachelorarbeiten mit eingebunden wird. Durch die beschriebene Struktur entstehen daraus keine Nachteile für die, in dieser Arbeit, beschriebene Simulation in PowerFactory.

9.3.2 Erstellung einer Datenerfassungsfunktion

Im ersten Schritt wurden die Funktionen zur Erfassung der Werte im Betrieb erstellt. Es können die bereits in der gegebenen Programmierung existierenden Variablen verwendet werden. Die bereits existierenden Betriebsmodi werden eingebunden.
Programmiert wird eine Funktion zum Erstellen und Beschreiben einer CSV-Datei. Das Schreiben der Werte in die Datei ist an die interne Zeitangabe gekoppelt. Andere Variablen, wie Anzahl aufzunehmender Werte, können in der Benutzeroberfläche geändert werden. Sowie das Starten und Beenden der Aufzeichnung wird im MMI gesteuert.

Programmierbausteine definieren

Aus der Beckhoff Infosys, dem Online Handbuch, lassen sich die hierzu benötigten Bausteine entnehmen. Die unter `https://infosys.beckhoff.com/` erreichbare Webseite beinhaltet alle Informationen zu den TwinCAT Versionen 2 und 3. Sie beinhaltet alle Funktionsbausteine, die sich in der Bibliothek befinden. Hier werden auf Deutsch und Englisch alle eingehenden und ausgehenden Variablen erklärt. Es handelt sich hier oft um Abkürzungen, die zu verwenden sind. An dieser Stelle wird die Abhängigkeit von dieser Plattform offensichtlich. Bei umso komplexeren Bausteinen wird die Programmiersprache FUP immer anschaulicher. Zum einen werden alle Variablen im Baustein angezeigt. Zum anderen können während der Laufzeit hier die aktuellen Werte einfach abgelesen werden. Dies erleichtert die Fehlersuche und Inbetriebnahme erheblich.

Implementierung

In der Übersicht ist unter der SPS bereits das SPS-Projekt, *SPS Sonnensimulator DC*, angelegt. In diesem befindet sich das gesamte Projekt, dass während der Laufzeit ausgeführt wird. Alle Funktionen und Variablen sind hier zu finden.

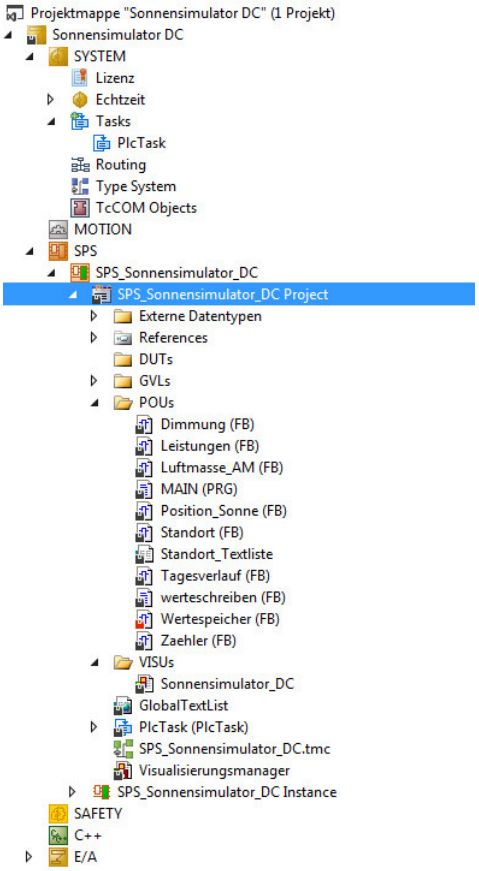


Abbildung 9.3: Sonnensimulator Projektstruktur

Es wurden unter bestimmten Reitern Änderungen Vorgenommen.

- **Globale Variablenliste (GVL)**
  Die definierten Variablen können aus allen Bausteinen, sowie MMI und von außerhalb bearbeitet und gelesen werden.

- **Programming Organization Units (POUs)**
  Diese enthalten alle Funktionen, Funktionsbausteine und Programme

- **Visualisierung (VISU)**
  Enthält alle grafischen Programmierungen für das, MMI.

Unter dem POUs befindet sich der Baustein MAIN, dieser ist fest vorgegeben und dient zum Aufruf aller Funktionen, die durchlaufen werden sollen.
Um eine neue Funktion anzulegen, wird unter 'Hinzufügen' POU ausgewählt. Dieser

wird als Funktionsbaustein angelegt und der Name vergeben, sowie die Implementierungssprache ausgewählt. Die neu erzeugte POU besteht aus einzelnen Netzwerken, die der Übersicht dienen. Es empfiehlt sich die einzelnen Netzwerke möglichst schlank zu halten. Es ist notwendig im Ablauf des Bausteins darauf zu achten, dass Aktionen die für ein späteres Netzwerk benötigte Werte ausgeben in einem Netzwerk zu platzieren, dass eine kleinere Nummerierung trägt und somit im Ablauf oberhalb steht.

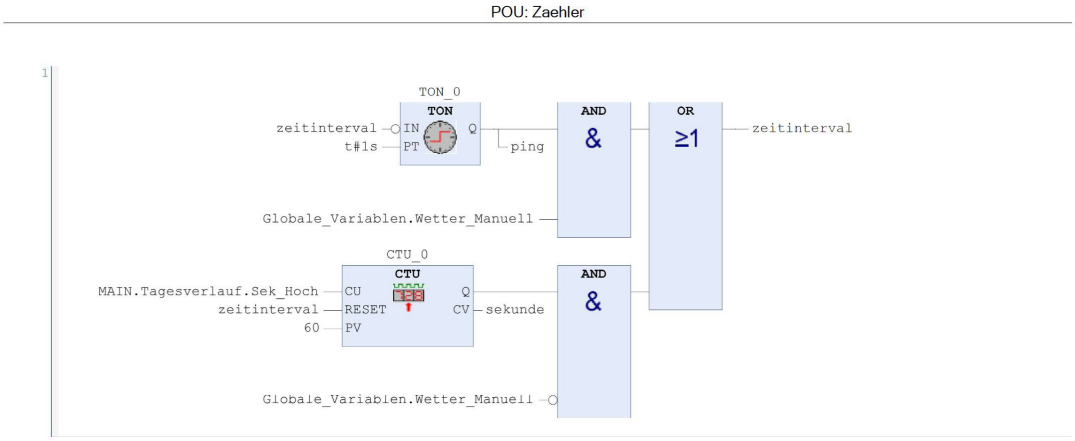
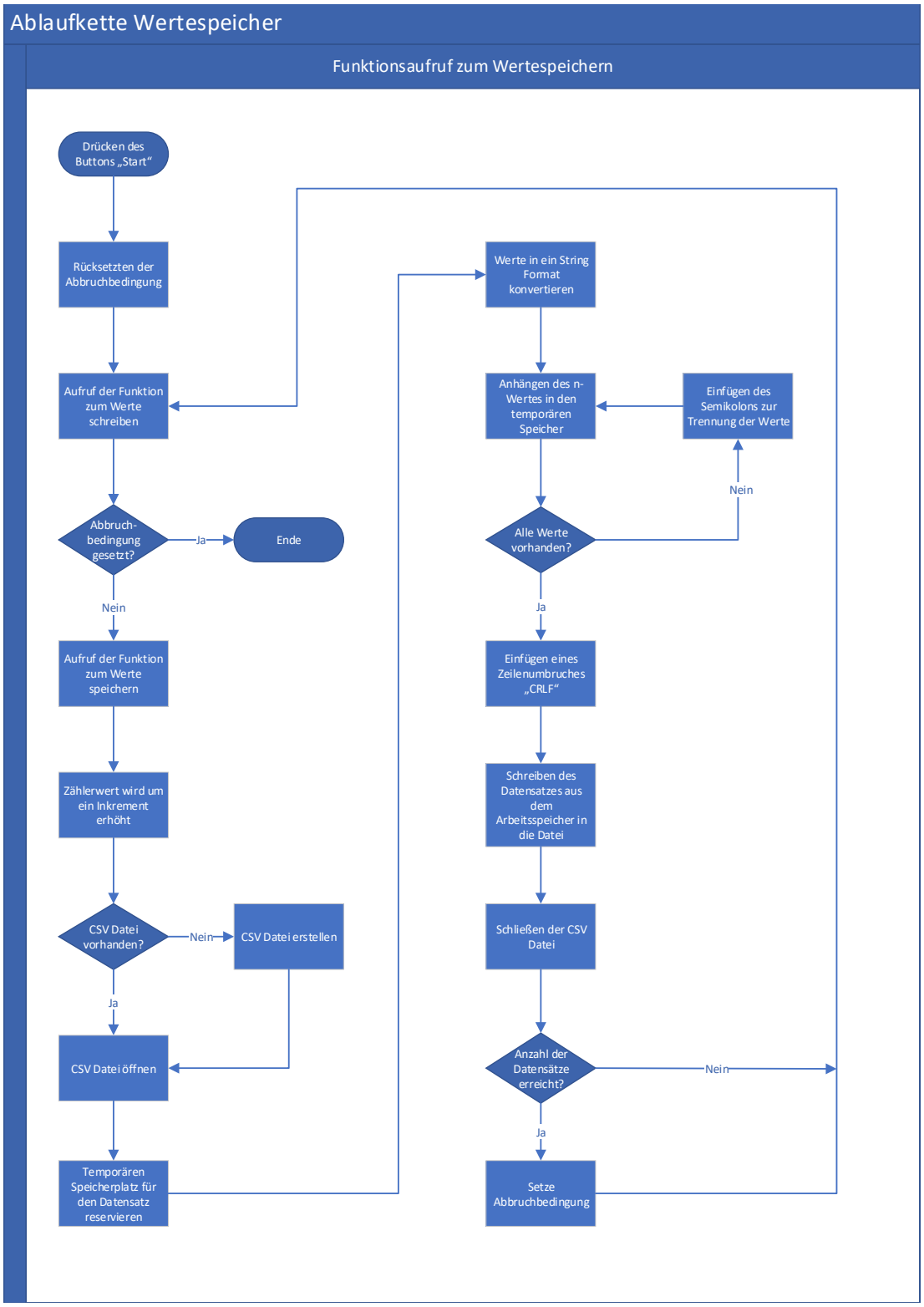


Abbildung 9.4: Exemplarischer Funktionsbaustein in einem Netzwerk

Funktion Wertespeicher

Die Funktion Wertespeicher besteht aus 11 Netzwerken, um die Daten während des Programmablaufs in die CSV-Datei zu schreiben. Die Aufstellung sieht vor, zwei Spalten zu generieren, die von links nach rechts laufenden Nummern und aktuelle Leistung eingetragen werden. Für jede Aufnahme gibt es eine weitere Zeile.
Diese Herangehensweise ist darauf ausgelegt leicht erweiterbar und modifizierbar zu sein. So könnte in der ersten Zeile die Uhrzeit stehen und weitere Werte wie Spannung und Strom separat eingefügt werden.
Kommt es zu einer Speicherung, wird dieser Prozess zeitgesteuert in regelmäßigen Intervallen aufgerufen. Es kommt zu folgendem Ablauf.


Ablaufkette Wertespeicher
Funktionsaufruf zum Wertespeichern
Drücken des Buttons „Start“
Rücksetzten der Abbruchbedingung
Aufruf der Funktion zum Werte schreiben
Abbruch-bedingung gesetzt?
Ja
Ende
Nein
Aufruf der Funktion zum Werte speichern
Zählerwert wird um ein Inkrement erhöht
CSV Datei vorhanden?
Nein
CSV Datei erstellen
Ja
CSV Datei öffnen
Temporären Speicherplatz für den Datensatz reservieren
Werte in ein String Format konvertieren
Anhängen des n-Wertes in den temporären Speicher
Einfügen des Semikolons zur Trennung der Werte
Nein
Alle Werte vorhanden?
Ja
Einfügen eines Zeilenumbruches „CRLF“
Schreiben des Datensatzes aus dem Arbeitsspeicher in die Datei
Schließen der CSV Datei
Anzahl der Datensätze erreicht?
Nein
Ja
Setze Abbruchbedingung

Ist die eingestellte Anzahl an Werten aufgenommen, wird die Initialisierungs-Variable ENDEspeicherung auf TRUE gesetzt. Der Ablauf des Programmes findet weiterhin ohne Aufnahme von Werten statt.
Ist der gesamte Simulationsprozess abgelaufen sind die Werte in der eingestellten Datei zu finden. Dieser befindet sich fest implementiert im Ordner /Hard Disk/FTP/... .

Aufruf des Funktionsbausteins Wertespeicher

Der Funktionsbaustein Wertespeicher findet sich in der MAIN Funktion nicht. Vor dem Aufruf des Bausteins findet eine Abfrage der Startbedingung statt.
Soll die Speicherung ausgeführt werden, wird im MMI der Start Button betätigt, der die globale Variable ENDEspeicherung auf FALSE setzt.
Um diese Bedingungen abzufragen wird in der MAIN Funktion der Funktionsbaustein Werteschreiben implementiert. Dieser hat als Variable nur den Funktionsbaustein Wertespeicher.

```
FUNCTION_BLOCK werteschreiben

VAR
        Wertepeichern : Wertespeicher;
END_VAR

IF MAIN.werteschreiben.Wertespeichern.Startbutton = TRUE
        THEN Globale_Variablen.ENDEspeicherung := FALSE;
END_IF;

IF Globale_Variablen.ENDEspeicherung = FALSE
        THEN Wertespeichern();
END_IF;
```

Listing 9.3: Aufrufbedingung der Werteaufnahme

9.3.3 Erweiterung der Bedienungsoberfläche

Die bereits vorhanden Visualisierung wurde übernommen und erweitert. Dem Design wurde hierzu gefolgt und nur eine neue Kaskade mit Bedienelementen hinzugefügt. Hier können folgende Einstellungen vorgenommen und abgelesen werden.

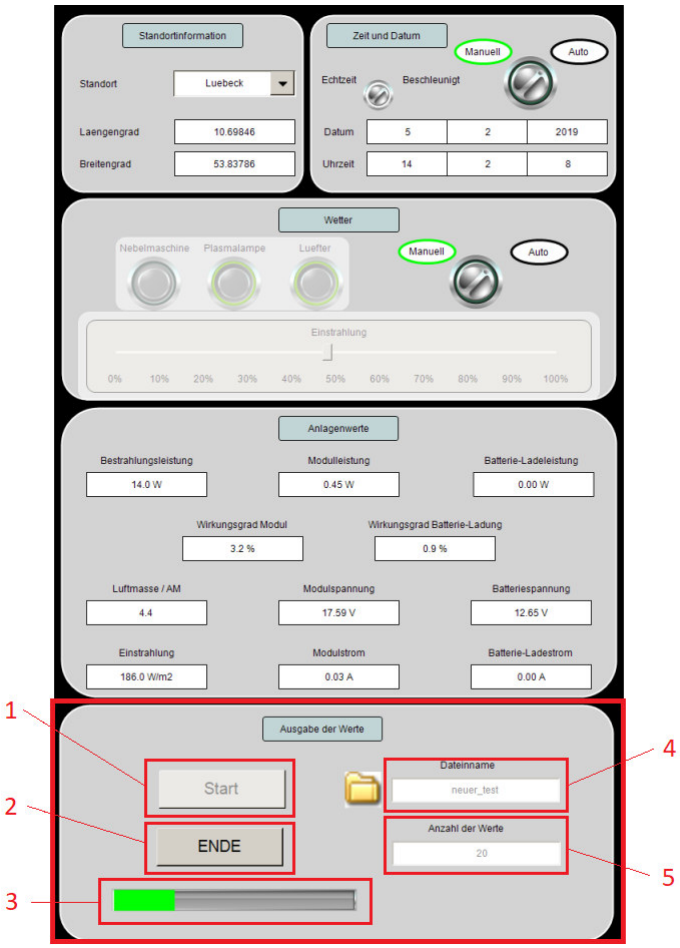


Abbildung 9.5: Visualisierung des MMI

1. Ein Bedienelement, zum Starten der Aufzeichnung
2. Ein Bedienelement, zum Abbruch der Aufzeichnung
3. Während der Aufzeichnung wird der Fortschritt visuell dargestellt
4. Name der CSV-Datei
5. Anzahl der Werte, die aufgenommen werden sollen

Die Bedienung der Elemente 1, 4 und 5 ist während der Speicherung nicht möglich. Um dies deutlich zu machen werden die Elemente ausgegraut.

Hand und Automatik-Betrieb

Bereits zwei Betriebsmodi waren in der Programmierung vorhanden.
Zum einen der Manuellmodus. Bei diesem ist der Simulationsstand unabhängig von dem zuvor erwähnten Baustein, der zuständig für den Sonnenstand ist. Über Schieberegler in der Simulation lassen sich die Halogenstrahler analog auf jeden Wert einstellen, sowie die Plasmalampe und die Nebelmaschine über Buttons an- und ausschalten.
Im Automatikmodus kann man, bis auf die Nebelmaschine diese Dinge nicht beeinflussen. Die Lampen werden dem Tagesverlauf entsprechend hoch und herunter gefahren. Eine Besonderheit gibt es im Automatikmodus. Er kann eine beschleunigte Simulation durchführen.
Hierzu wird der Prozessablauf nur von der Zeit beeinflusst, die die CPU benötigt, um das Programm abzuarbeiten. Somit kann ein 24 Stunden Tag in ca. 35 Minuten durchlaufen werden.
In der Programmierung wurden alle drei Fälle berücksichtigt. Hierzu findet eine Auswahl statt, zwischen Hand und Automatik-Betrieb.
Im Handbetrieb wird jede Sekunde ein Wert aufgenommen.
Im Automatikbetrieb ist die Aufnahme von der simulierten Zeit abhängig und findet jede Simulierte Minute einmal statt.

9.3.4 Bereitstellen der Daten via Netzwerkzugriff

FTP-Zugriff konfigurieren
Die Beckhoff SPS CX9020 arbeitet mit einem Betriebssystem von Microsoft. Windows Embedded Compact 7 ist eine kompakte Variante des Desktop-Betriebssystems. Dies macht die Bedienung einfach und die Oberfläche vertraut. Die in jeder Windows Version vorhandene Möglichkeit des Zugriffs per FTP ist auch hier implementiert.
Aktiviert und Konfiguriert wird der FTP-Server in der Systemeinstellung. Die ist über die grafische Bedienoberfläche aufzurufen. Unter dem Reiter FTP können folgende Parameter eingestellt werden.

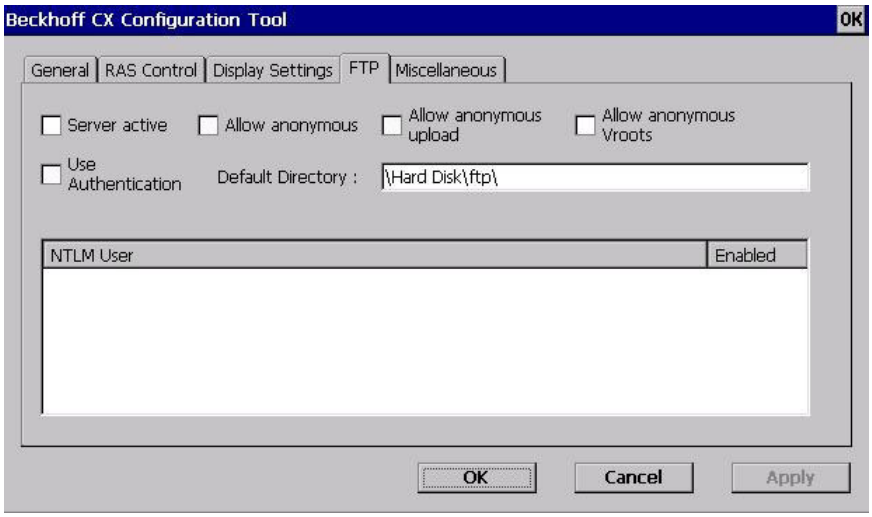


Abbildung 9.6: Beckhoff FTP-Konfiguration

- **Server active**
  Dies aktiviert allgemein die FTP Verbindung.
- **Allow anonymous**
  Wenn aktiviert kann jeder unter der IP-Adresse auf die Ordner und Dateien zugreifen.
- **Allow anonymous upload**
  Eröffnet einer unbekannten Quelle die Möglichkeit Daten auf den Server hoch zu laden.
- **Allow anonymous Vroots**
  Die Ordnerstruktur bekommt einen Verweis auf eine andere Maschine.
- **Use Authentification**
  Nutzer müssen sich über Namen und Passwort auf der Zielmaschine einloggen.
- **Default Directory**
  Ist das Verzeichnis, dass bei Zugriff geöffnet wird. Hierbei handelt es sich, um den am höchsten zu erreichenden Ordner innerhalb der Ordnerstruktur auf dem Server.

Die vorgenommene Einstellung beschränkt sich auf die Aktivierung des Servers und den Passwort geschützten Zugriff auf diesen. Ein Benutzer mit dem Namen "eev" wurde erstellt. Dieser hat das gleichnamige Passwort "eev". Jetzt kann mit Benutzernamen und Passwort von Außen auf den freigegebenen Ordner und dessen Inhalte via FTP zugegriffen werden. Dazu wird von einem Windows-PC in der Ordnerstruktur FTP:\\und die IP-Adresse eingegeben. Sollte die Eingabeaufforderung für Benutzer und Passwort nicht automatisch starten, kann diese mit der rechten-Maustaste unter „Anmelden als...“ aufgerufen werden.

10 Versuchsdurchführung – Timo Helsper

In diesem Kapitel werden zuerst die letzten Vorbereitungen getroffen um den Steueralgorithmus zu testen. Sind diese Abgeschlossen kann der Steueralgorithmus nun genauer betrachtet werden.

10.1 Profile

In den nächsten Unterkapiteln wird der Aufbau der Simulationsumgebung gezeigt. Um einen ersten Einblick zu erlangen, ist es wichtig zu wissen, wie ein Haushalt aufgebaut ist.
Es gibt in dieser Simulationsumgebung insgesamt zwei Haushalte die voneinander unabhängig Arbeiten aber an dem gleichen Trafo angeschlossen sind. Beide Zuleitung zu den Häusern sind 5km lang. Jeder Haushalt hat auf dem Dach eine PV-Anlage. Neben der PV-Anlage hat jeder Haushalt eine Ladestation für ein oder mehrere Elektrofahrzeuge. Die Haushalte selber haben noch ein eigenes Haushaltslastprofil.

10.1.1 Erzeugerlastprofil

Als Grundlage für alle weitere Simulationen muss eine Zeit intensivste Messung durchgeführt werden. Hierbei benötigt der Sonnensimulator für eine vollständige Tagessimulation ca. 35-40 min. Zusätzlich müssen die Randbedingungen festgelegt werden.
Die Einstellungen im Sonnensimulator wurden so gewählt, dass einmal der längste Tag, der 21.06.2019, und einmal der kürzeste Tag, der 21.12.2019, simuliert werden. Eine Verschattung durch Nebel findet nicht statt. Die Simulation startet um 00:00 des Tages und nimmt die Werte im Minuten Takt auf. Am Ende des Tages sind dann 1439 Werte vorhanden.
Die Werte für die jeweilige Minute steht in einer Zeile. Mehrere Werte werden mit einem Semikolon getrennt und Dezimalstellen haben ein Punkt statt Komma. Die Eingabe sieht wie folgt aus:

1;2.00;3.00;4.00

Die erste Stelle (1) ist eine Fortlaufende Nummer.
Die zweite Stelle (2.00) repräsentiert die Leistung des Photovoltaik Moduls.
Die dritte Stelle (3.00) repräsentiert die Spannung des Photovoltaik Moduls.
Die vierte Stelle (4.00) repräsentiert den Strom des Photovoltaik Moduls.

Die Werte werden gesammelt in einer Textdatei. Diese Textdatei liegt auf einem FTP-Server, welcher von Raspberry Pi (RPI) erreichbar ist. Dieses Profil wird auch Erzeugerlastprofil (ELP) genannt.
Der erste Haushalt hat eine PV Anlage von 6kWp auf dem Dach und der zweite Haushalt eine 7kWp Anlage auf dem Dach. Somit kann Tabelle 10.1 aufgestellt werden.

Haushaltsname	PV-Leistung	Ladestationleistung	Bewohner
Haushalt 1	6kWp	- - -	- - -
Haushalt 2	7kWp	- - -	- - -

Tabelle 10.1: Übersicht der Haushalt 1

10.1.2 Ladezeitenprofil

Über die Häuser und das Verhalten der Bewohner kann nun ein Ladezeitenprofil (LZP) erstellt werden. Hier bei wurde der Tagesablauf der Bewohner genauer angeschaut. Wichtig hierbei war, dass der Bewohner immer ein volles Auto vor der Tür stehen hat. Deswegen wurden die Autos immer geladen wenn der Bewohner aufsteht, von außerhalb wieder nach Hause kommt oder Abends vor dem schlafen gehen. Hierbei wurde weiterhin angenommen, dass die Familien ein Elektroauto haben, welches in der Familie geteilt wird. Eine Rücksichtnahme auf den Ladestand der im Auto befindlichen Batterie wurde nicht getroffen. Auch aus diesem Grund, wurde veränderliche Ladeleistungen nicht mit in das Ladeprofil mit eingebunden. Die Ladestation lädt somit immer mit der vollem, ihm Verfügung stehenden Leistung. Somit kann folgende Tabelle 10.2 aufgestellt werden.

Haushaltsname	PV-Leistung	Ladestationleistung	Bewohner
Haushalt 1	6kWp	11kW	- - -
Haushalt 2	7kWp	22kW	- - -

Tabelle 10.2: Übersicht der Haushalt 2

10.1.3 Haushaltslastprofil

Basierend auf dem Messungen aus dem Kapitel 10.1.1, werden nun die passende Haushaltslastprofil (HLP) erstellt. Hier bei ist das Datum wichtig. Mit dem vorhandenem Datum kann in einem Programm ein HLP erstellt werden. Dieses Programm heißt Load Profile Generator. [27] Das fertige HLP spiegelt ein Tag in diesem Haushalt wichtig.
Von jedem Haushalt gibt es zwei HLP. Ein HLP wird für den 21.06.2019 berechnet und das andere HLP wird für den 21.12.2019 berechnet. Somit steht ein Langer Sommertag zur Verfügung und ein kurzer Wintertag. Der Sommertag ist ein Freitag und der Wintertag ist ein Samstag. Die Wochentage sind in sofern wichtig. Das am Wochenende andere Tätigkeiten, wie zum Beispiel: Schwimmen oder Einkaufen unternommen werden, während unter der Woche viel gearbeitet wird.
Im ersten Haushalt wohnt ein Pärchen. Beide Arbeiten unter der Woche und sind nicht anwesend.
Im zweiten Haushalt wohnt ein Pärchen mit zwei Kindern und zwei Senioren. Das Pärchen arbeitet unter der Woche und die Senioren passen auf die Kinder auf.

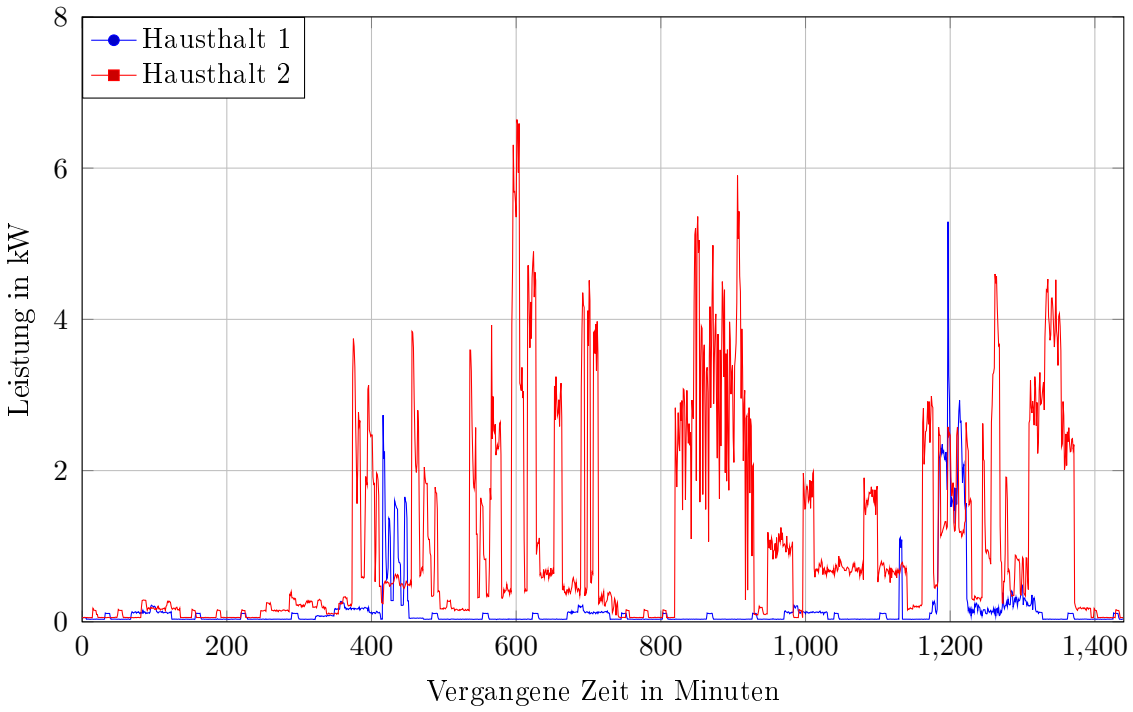


Abbildung 10.1: Lastprofil zweier Haushalte

Aus der Abbildung 10.1 ist zu erkennen, wie unterschiedlich beide Haushalte sind. Beide Lastverläufe sind für den selben Tag simuliert. Hier wurde der 21.06.2019 gewählt. Dieser Tag ist ein Freitag.
Während Haushalt 1 nur morgens und abends hohe Leistungsentnahmen hat, hat Haushalt 2 über den Tag verteilt immer wieder hohe Leistungsentnahmen. Dieses ist dadurch durch zu begründen, dass hier zwei Grundverschiedene Haushalte aufeinander treffen. Trotz der unterschiedlichen Familien ist dieses Szenario insofern Realistisch, das besonderes auf dem Ländlichen Gebiet immer wieder so oder solch ähnliche Konstellationen anzutreffen sind.
Zum Abschluss wird die Tabelle 10.2 um die Anzahl der Bewohner ergänzt, sodass die Tabelle 10.3 zustande kommt.

Haushaltsname	PV-Leistung	Ladestationleistung	Bewohner
Haushalt 1	6kWp	11kW	2
Haushalt 2	7kWp	22kW	6

Tabelle 10.3: Übersicht der Haushalt 3

10.2 Steuerung der Ladeleistung

In Zusammenarbeit mit dem carpeDIEM Projekt, wurde ein Steueralgorithmus für diesen Abschluss zur Verfügung gestellt. Dieser Steueralgorithmus findet in den folgenden Versuchen Anwendung. Der volle Umfang dieses Algorithmus stand während des Versuches nicht zur Verfügung und war daher nur eingeschränkt nutzbar. Auch befand sich der Algorithmus noch in der Anfangsentwicklung. Draus folgt, dass die nachfolgende Versuche nicht immer das gewünschte Ergebnis liefern. Während dieser Abschlussarbeit wurde immer wieder Feedback zu den Algorithmus gegeben und die Ergebnisse wurden geteilt um eine bessere Entwicklung zu ermöglichen.

10.3 Vorbereitung der Versuche

Es wurden zwei verschiedene Versuche durchgeführt, um den Algorithmus zu testen. Hierzu wurden zwei unterschiedliche Tage ausgewählt. Mit diesen zwei Tagen und den dazugehörigen Profilen, wurde die Setup Textdatei konfiguriert. Um nun die Randbedienung nicht zu verändert, wird die Setup Textdatei nicht mehr verändert. In der Java Serverdatei wurde nur der Algorithmus abgeschaltet bzw. dazugeschaltet. So kann der gleiche Tag exakt zwei mal unter fast den gleichen Bedingungen simuliert werden. Einmal mit Algorithmus und einmal ohne Algorithmus.
Bevor die Simulation starten kann, wird die Setup Textdatei auf den jeweiligen Java Server kopiert. Anschließend wird der Java Server gestartet. In der Konsole des Java Servers ist erkennbar, ob der Server läuft oder ob es ein Problem gab. Ein typischer Fehler, kann ein falscher Namen einer Profildatei sein. Hier muss auf die korrekte Schreibweise geachtet werden.
Wenn der Java Server läuft, kann auf die Oberfläche von PowerFactory gewechselt werden. Hier ist die Einstellung RMS Simulation zu wählen. Als nächstes wird der Startzeitpunkt festgelegt und die Schrittweite. Wichtig hierbei ist zu wissen, dass PowerFactory in Sekunden rechnet. Eine simulierte Sekunde entspricht eine reale Minute.

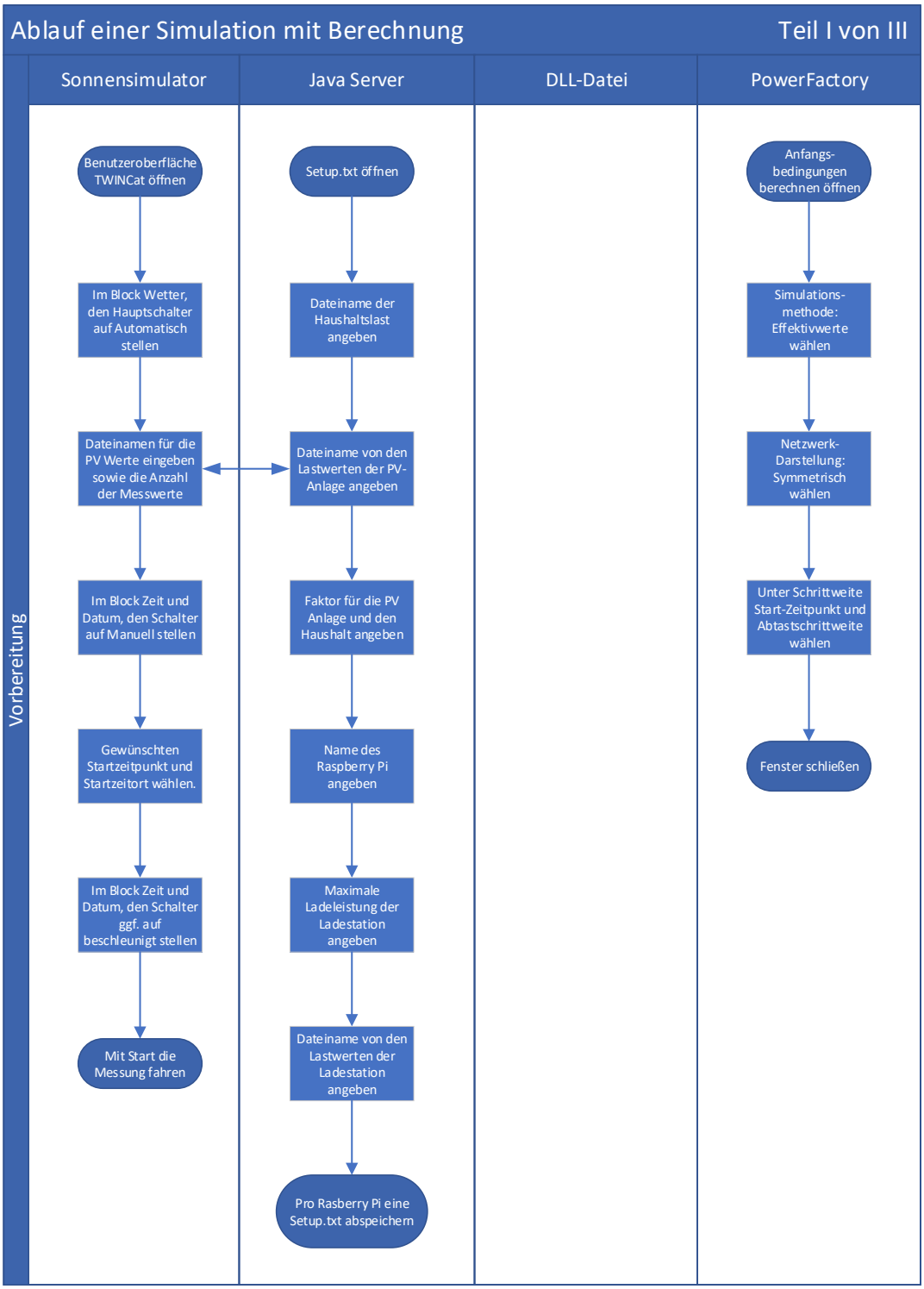
10.4 Ablauf des Versuches

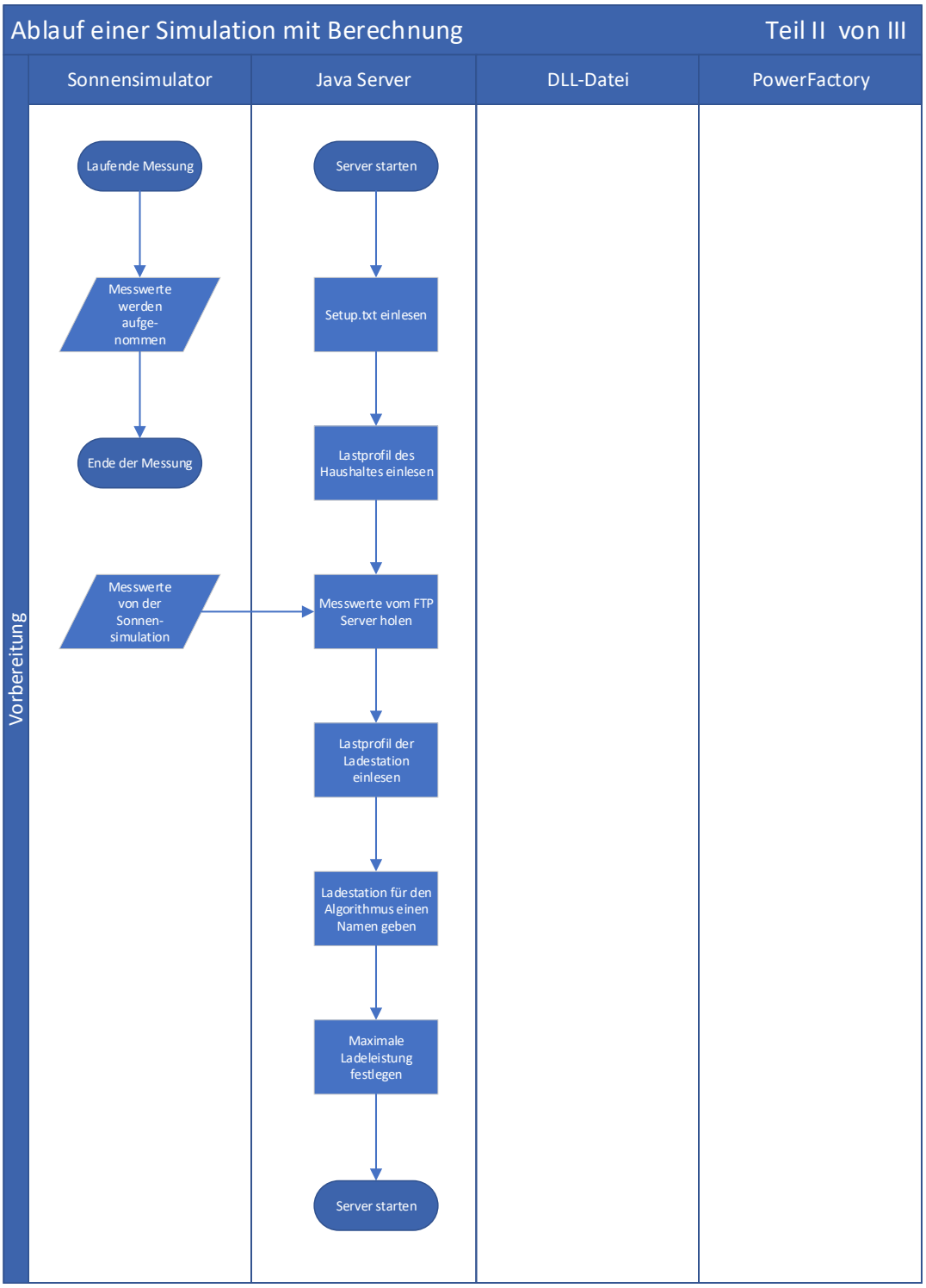
Als erstes müssen zwei Zeitpunkte fest gelegt werden, wann die Simulation starten soll. Anschließend wird die Benutzerobefläche TwinCat gestartet um eine Sonnensimulation durchzuführen. Das Wetter wird auf Automatisch gestellt, und anschließend wird ein Dateiname eingegeben. Der Dateiname ist wichtig, da dieser später wieder benötigt wird. Danach wurde angegeben wie viele Messwerte aufgenommen werden sollen. Hier wurde die Einstellung 1440 getroffen. Im Anschluss wurde das Startdatum gewählt. An diesem Zusammenhang kam im ersten Durchlauf der 21.06.2019 dran, danach der 21.12.2019. Beide Startpunkte waren um 00:00 Uhr. Ausserdem musste noch ein gewünschter Ort fest gelegt werden. In diesem Fall wurde Lübeck gewählt. Anschließend wurde die Simulation noch auf beschleunigt gestellt und die Messung startete. Dieses dauert im Schnitt 35 bis 40 Minuten und muss einmal für den Sommer Tag und einmal für den Wintertag durchgeführt werden.
Parallel kann das HLP erstellt werden. Dieses muss zweimal gemacht werden, für jeden Haushalt einmal. Unter dem Abschnitt 10.1.3 ist der Vorgang genauer zu finden. Nach dem erstellen des HLP, wurde ein passendes Ladeverhalten für die Elektrofahrzeuge erstellt (Abschnitt 10.1.2). Weiterhin muss die Setup Datei angepasst werden (Abschnitt 5.3). Viele Punkte sind vordefiniert und müssen nicht geändert werden. Folgende Anpassungen müssen getroffen werden:

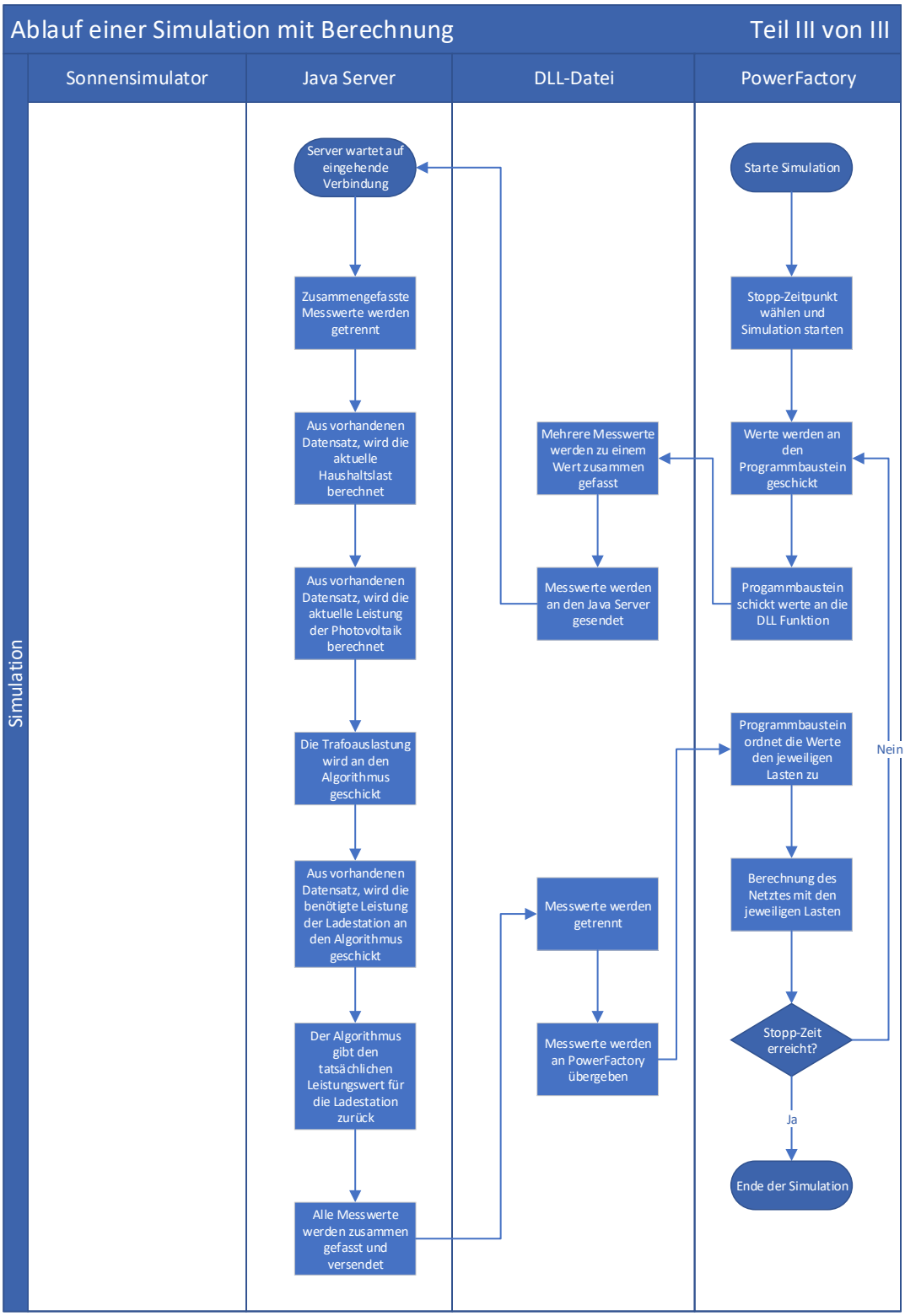
- Der Name der Textdatei, in der das Haushaltslastprofil hinterlegt ist, muss in die Setup Datei eingetragen werden.
- Der Name der Textdatei, in der das Erzeugerlastprofil der PV-Anlage hinterlegt ist, muss in die Setup Datei eingetragen werden.
- Da es sich bei den Sonnensimulator um ein Modell handelt, muss das Erzeugerlastprofil mit einem Faktor versehen werden. Somit kann die kleine 10Wp Anlage zu einer großen 6kWp Anlage skaliert werden.
- Der Raspberry Pi muss eine dem Haushalt zugeordneten Adresse bekommen.
- Zum Schluss muss festgelegt werden, welche Maximale Ladeleistung die Ladestation abgegeben darf

Ist die Erstellung der Setup Datei fertig, so muss dieses für den anderen Haushalt angepasst wiederholt werden. Anschließend wird die fertigen Setup Datei zusammen mit dem HLP auf den passenden RPI kopiert.
In PowerFactory können nun schon die erste Vorbereitung beginnen, um die Simulation zu starten. Nach dem das Projekt geöffnet wurde, werden die Anfangsbedienung festgelegt. Dabei ist die Simulationsmethode Effektivwerte zu wählen. Anschließend muss unter dem Punkt Netzwerkdarstellung Symmetrisch ausgewählt werden. Um die Vorbereitung abzuschließend muss unter dem Punkt Schrittweite der Startzeitpunkt '0' und die Abtastschrittweite von '1' gewählt werden. Danach können die Eingaben bestätigt werden und das Fenster kann geschlossen werden.

Ist der Sonnensimulator fertig mit den Messungen, werden die Werte automatisch auf dem FTP-Server bereitgestellt und es bedarf keiner weiteren Bearbeitung durch den Anwender. Die Simulation ist nun bereit gestartet zu werden.
Als erstes werden die beiden Java Server gestartet. Diese lesen als die Setup Datei ein. Danach wird das HLP auf den Server geladen. Nach diesem Vorgang, wird eine Verbindung zum FTP-Server aufgebaut und die Messwerte werden vom FTP-Server auf den Java Server kopiert. Anschließend wird das Ladeprofil der Ladestation eingelesen und der Algorithmus bekommt die Namen der Haushalte zugeschrieben und die Maximale Ladeleistung. Der Server ist nun betriebsbereit. Alle Werte sind nochmal in der Konsole dargestellt und können Kontrolliert werden.
Die Simulation kann nun in PowerFactory gestartet werden. Dazu aus der Symbolleiste die Auswahl "Simulation starten..." auszuwählen. Im Anschluss öffnet sich ein Fenster und fragt nach dem Stoppzeitpunkt. Hier ist die Zeit '1439' einzutragen. Anschließend werden die aktuellen zu simulierten Werte an einen Programmbaustein geschickt. Die Anfangsbedienung der Werte beträgt 0. Der Programmbaustein speichert sich die Werte intern ab und gibt diese Werte weiter an eine Externe DLL Datei. In dieser DLL Datei werden die Werte zusammen zu einem String gebaut und es findet Verbindungsaufbau zum Java Server statt. Steht diese Verbindung, erhält der Java Server den String. Dieser String wird auf dem Java Server wieder getrennt und die einzelnen Werte werden den jeweiligen Variablen zugeordnet. Als erstes liest der Java Server mit der aktuellen Zeit den Datensatz mit den Haushaltswerten aus. Der passende Wert wird bereitgestellt. Im Anschluss wird auch aus einem anderen Datensatz mit der gleichen Zeit die aktuelle Leistung der PV-Anlage ausgelesen. Auch dieser Wert wird bereitgestellt. Sollte nun der Algorithmus aktiv sein, wird nun an den Algorithmus die Verfügbare Restkapazität des Trafos übermittelt. Zusätzlich wird an Algorithmus die gewünschte Ladeleistung übermittelt. Hat der Algorithmus die Werte von allen Stationen erhalten, so kann die Berechnet Ladeleistung Bereitgestellt werden. Sollte der Algorithmus nicht vorhanden sein, wird nur geprüft ob geladen werden soll oder nicht. Alle bereitgestellten Werte, werden nun zusammen gefasst und an die DLL Datei zurück übergeben, welche diese weiter an die jeweiligen Programmbausteine schickt. Die Programmbausteine weißen nun die Werte den jeweiligen Lasten zu wodurch der Lastfluss zu einen bestimmten Zeitpunkt berechnet werden kann. Dieses wiederholt sich solange bis der Stoppzeipunkt erreicht ist.
Anschließend kann unter den Grafischen Reitern die zuvor eingestellten Variablen genauer betrachtet werden. Dieses ist in weiteren Kapitel nochmals genauer erwähnt.


Ablauf einer Simulation mit Berechnung
Teil I von III
Sonnensimulator
Java Server
DLL-Datei
PowerFactory
Vorbereitung
Benutzeroberfläche TWINCat öffnen
Im Block Wetter, den Hauptschalter auf Automatisch stellen
Dateinamen für die PV Werte eingeben sowie die Anzahl der Messwerte
Im Block Zeit und Datum, den Schalter auf Manuell stellen
Gewünschten Startzeitpunkt und Startzeitort wählen.
Im Block Zeit und Datum, den Schalter ggf. auf beschleunigt stellen
Mit Start die Messung fahren
Setup.txt öffnen
Dateiname der Haushaltslast angeben
Dateiname von den Lastwerten der PV-Anlage angeben
Faktor für die PV Anlage und den Haushalt angeben
Name des Raspberry Pi angeben
Maximale Ladeleistung der Ladestation angeben
Dateiname von den Lastwerten der Ladestation angeben
Pro Rasberry Pi eine Setup.txt abspeichern
Anfangs-bedingungen berechnen öffnen
Simulations-methode: Effektivwerte wählen
Netzwerk-Darstellung: Symmetrisch wählen
Unter Schrittweite Start-Zeitpunkt und Abtastschrittweite wählen
Fenster schließen


Ablauf einer Simulation mit Berechnung
Teil II von III
Sonnensimulator
Java Server
DLL-Datei
PowerFactory
Vorbereitung
Laufende Messung
Messwerte werden aufge-nommen
Ende der Messung
Messwerte von der Sonnen-simulation
Server starten
Setup.txt einlesen
Lastprofil des Haushaltes einlesen
Messwerte vom FTP Server holen
Lastprofil der Ladestation einlesen
Ladestation für den Algorithmus einen Namen geben
Maximale Ladeleistung festlegen
Server starten


Ablauf einer Simulation mit Berechnung
Teil III von III
Sonnensimulator
Java Server
DLL-Datei
PowerFactory
Simulation
Server wartet auf eingehende Verbindung
Zusammengefasste Messwerte werden getrennt
Aus vorhandenen Datensatz, wird die aktuelle Haushaltslast berechnet
Aus vorhandenen Datensatz, wird die aktuelle Leistung der Photovoltaik berechnet
Die Trafoauslastung wird an den Algorithmus geschickt
Aus vorhandenen Datensatz, wird die benötigte Leistung der Ladestation an den Algorithmus geschickt
Der Algorithmus gibt den tatsächlichen Leistungswert für die Ladestation zurück
Alle Messwerte werden zusammen gefasst und versendet
Mehrere Messwerte werden zu einem Wert zusammen gefasst
Messwerte werden an den Java Server gesendet
Messwerte werden getrennt
Messwerte werden an PowerFactory übergeben
Starte Simulation
Stopp-Zeitpunkt wählen und Simulation starten
Werte werden an den Programmbaustein geschickt
Progammbaustein schickt werte an die DLL Funktion
Programmbaustein ordnet die Werte den jeweiligen Lasten zu
Berechnung des Netztes mit den jeweiligen Lasten
Stopp-Zeit erreicht?
Nein
Ja
Ende der Simulation

10.5 Versuch 1

Im ersten Versuch wird ein Langer Sommertag betrachtet. Der Tag startet bei 0:00 Uhr und endet um 23:59 Uhr. Es handelt sich hierbei um den 21.06.2019 welches ein Freitag ist. Im ersten Durchlauf ist der Steueralgorithmus ausgeschaltet im zweiten ist dieser eingeschaltet.

10.5.1 Versuch 1.1

Versuch ohne Steueralgorithmus

Übersicht

Um die folgenden Auswertung besser zur Verstehen, zeigt Abbildung 10.2 eine Übersicht des Versuchsergebnis.

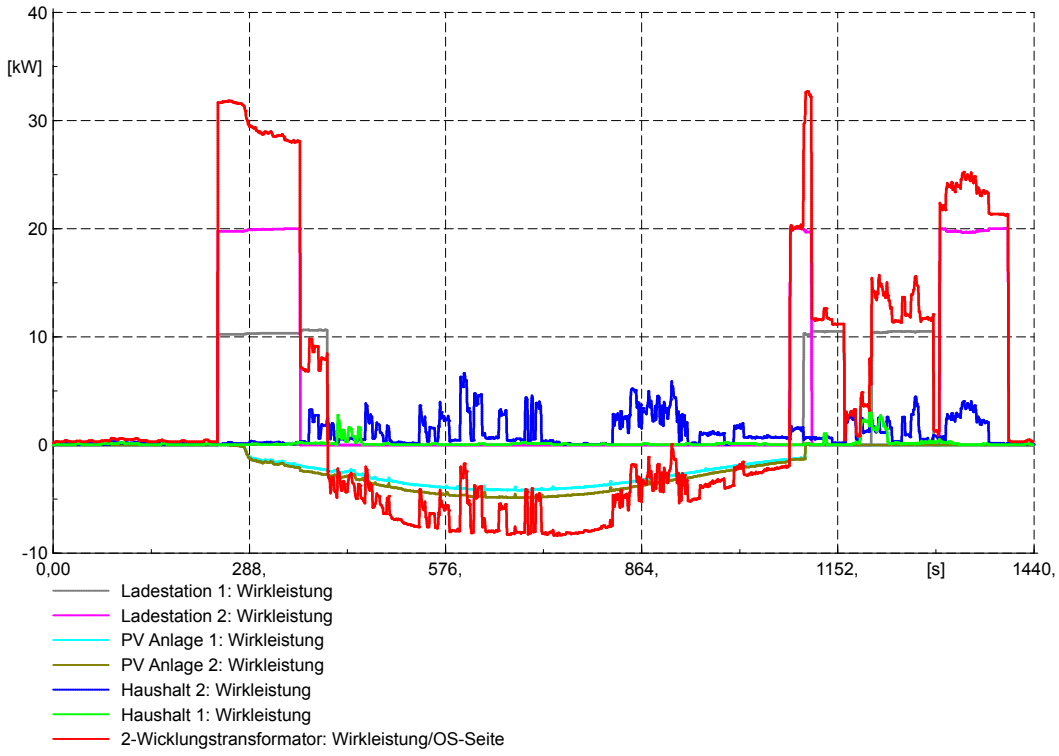


Abbildung 10.2: Versuch 1.1 Langer Tag ohne Algorithmus

An diesem Tag werden beide Autos morgens geladen. Dieses geschieht bei der Simulationszeit um die 288. Danach steht Haushalt zwei als erstes auf, gefolgt von Haushalt eins. Bei jedem Haushalt gehen nun zwei Personen arbeiten. Besonderes bei Haushalt eins ist dieses erkennbar, da kaum weitere Leistungsentnahmen erkennbar ist.
Haushalt zwei hat über den Tag hinweg immer wieder Leistungsentnahmen. Dieses ist damit zu begründen, das sich Zuhause noch zwei Senioren und zwei Kinder befinden. Abends gegen Simulationszeit 1152 kommen die Haushalte wieder zusammen und es

wird kurz geladen. Nach einer kurzen Ladepause, werden die Elektrofahrzeuge das letzte mal vor Mitternacht geladen. Während der ganzen Simulation, ist der Algorithmus ausgeschaltet.
Bei der näheren Betrachtung des Trafos ist zu erkennen, das morgens und abends Lastspitzen entstehen. Diese liegen teilweise über 30kW. Zwischen den Morgen und Abendstunden speist die PV-Anlage über den Transformator mehr ins Netz ein, als der eigentliche Verbrauch ist. Morgens und Abends wird dafür zum Laden viel Leistung aus dem Netz entnommen.

Leitungsauslastung

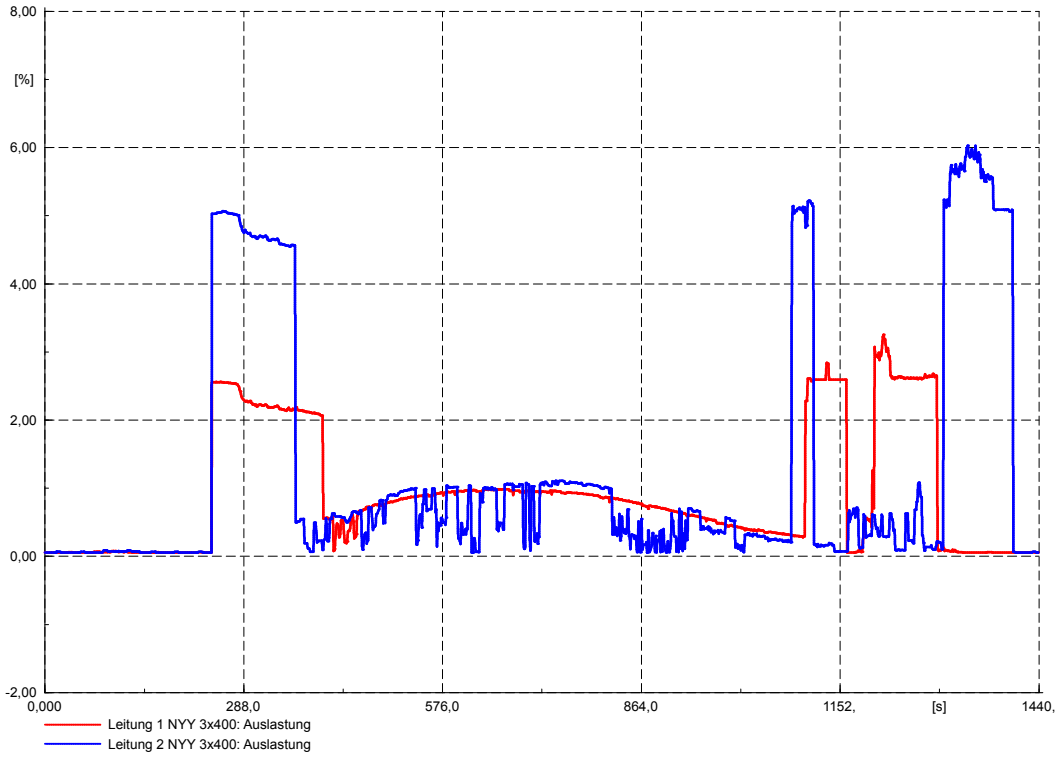


Abbildung 10.3: Versuch 1.1 Übersicht Leitungsauslastung

Betrachtet man nun die Auslastung der Zuleitung (Abbildung: 10.3), so ist zu erkennen, dass besonderes zu den Ladezeiten die Auslastung der Leitung am höchsten ist. Hier erreicht die Auslastung der Leitung zu spitzen Zeiten 6%. Die Leitung ist somit 5 bis 6 mal stärker Ausgelastet als zur Mittagszeit, während die PV-Anlage einspeist. Besonderes ältere Haushalte, deren Zuleitung noch nicht an größere Leistung angepasst sind, könnten an die Grenzen kommen. Hierbei muss auch angemerkt werden, dass höhere Entnahme auch bei neuen Häusern zu Problemen führen kann. Die Leitungen sind für eine gewissen spitzen Last ausgelegt. Haushalt 1 (rot), welches nur eine 11kW Ladestation ist von dem Problem nicht so stark betroffen wie Haushalt 2 (blau). Haushalt 2 (blau) hat eine 22kW Ladestation und auf Grund der Familiären Situation (Mehrgenerationenhaus) ist es nicht unwahrscheinlich, wenn die Familie sich ein weitere Fahrzeug anschafft.

Während die Leitung im Haushalt 1 (rot) keiner starken Schwankung unterliegen, ist bei Haushalt 2 (blau) immer wieder Einbrücke der Leitung zu betrachten. Dieses ist immer dann der Fall, wenn besonderes zu Mittagszeit die Erzeugte Leistung und die entnommene Leistung sich ausgleichen.

Spannungsbetrachtung

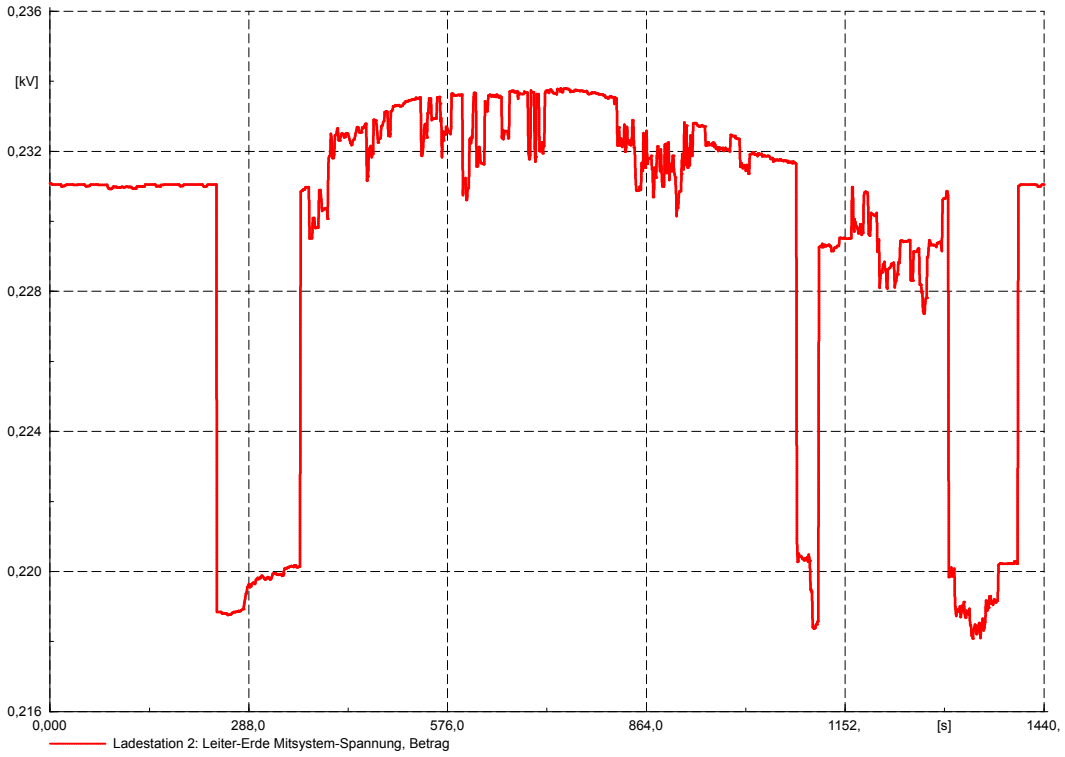


Abbildung 10.4: Versuch 1.1 Übersicht Spannungseinbrüche

Im gleichen Zusammenhang der Ladeleistung, kann eine Aussage zu den Spannungseinbrüchen genommen werden.
Aus der Abbildung 10.4 ist zu erkennen, dass die Spannung zu den Ladezeiten auf ca. 218V fällt, während die Spannung beim einspeisen der PV-Anlage auf knapp 234V ansteigt. Dieses ist eine Differenz von 16V über den Tag betrachtet.
Wichtig hierbei auch zu wissen ist, das der Transformator früher nur immer eingespeist hat, um das Spannungsband zu halten. Mit entfernteren Häusern nahm die Spannung berechenbar ab. Nun kommen neu auftreten Leistungsspitzen, die dafür sorgen, dass die mehr Spannung abfällt. Zusätzlich treten in Zeiten, wo wenig Leistungsentnahmen in Wohngegenden statt findet, Einspeisungen durch Erneuerbare Energie statt. Hier muss der Trafo zukünftig flexibel reagieren können.

Frequenzbetrachtung

Im nächsten Schritt kann nun die Frequenzabweichung betrachtet werden (Abbildung: 10.5. Dabei ist zu erkennen, dass beim Einschalten bzw. Ausschalten der Ladestationen die Frequenz um die 5 bis 6mHz abweicht. Dieses ist dadurch zu begründen, da plötzlich viel Leistung auf einmal gefordert wird bzw. nicht mehr gefordert wird.
Eine Schwankung um 6 mHz ist noch in einem Bereich, in dem keine Regelung statt findet. Diese Regelung findet erst ab einer Frequenzabweichung von 10mHz statt. Diese Frequenzabweichung ist in diesem Simulationsaufbau zu vernachlässigen, muss aber bei größeren Netzen Beachtung geschenkt werden.

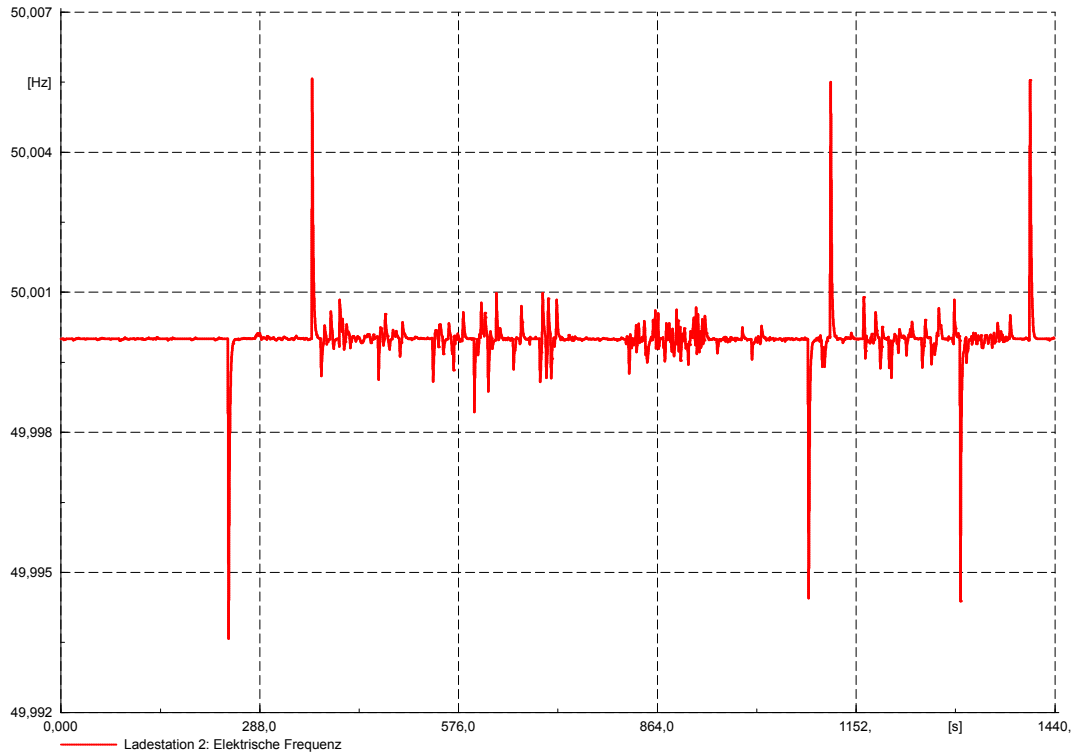


Abbildung 10.5: Versuch 1.1 Übersicht Frequnzabweichung

10.5.2 Versuch 1.2

Versuch ohne Steueralgorithmus

Übersicht

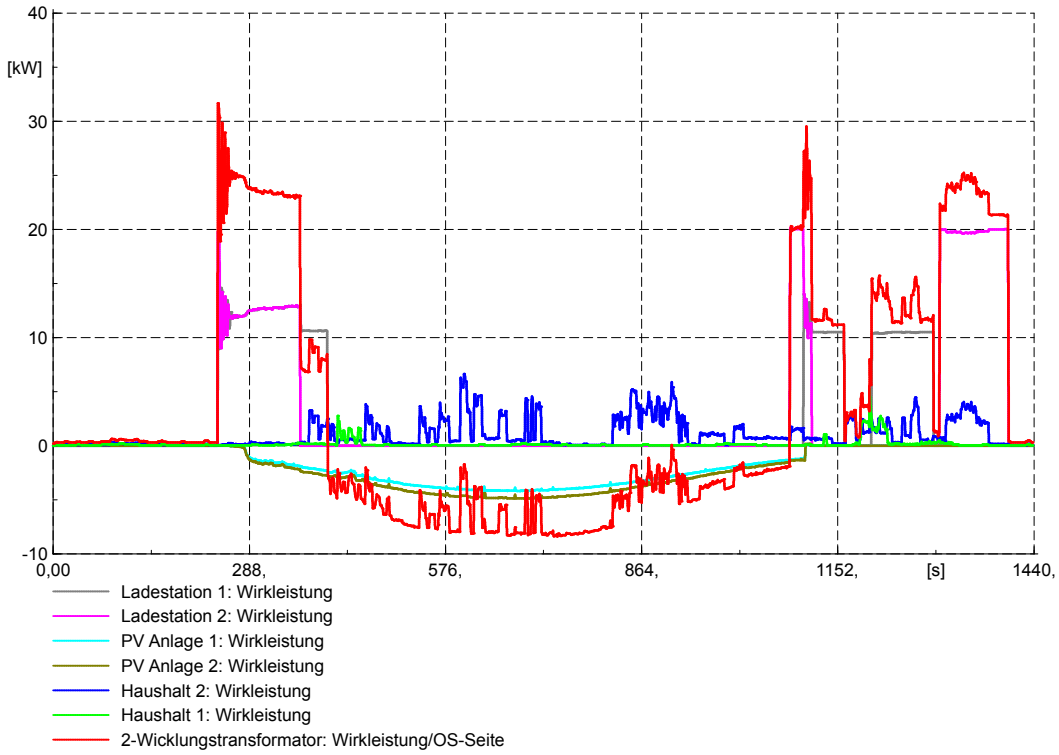


Abbildung 10.6: Versuch 1.2 Langer Tag mit Algorithmus

Im zweiten Teil dieses Versuches werden alle Bedingung gleich gehalten. Nun wird aber zusätzlich noch der Ladealgorithmus dazugeschaltet. In der Abbildung 10.6 ist die Übersicht der Messung noch einmal dargestellt. Auch bei dieser Simulation gibt es immer noch Lastspitzen, diese fallen aber nur morgens kürzer aus, da hier beide Ladestationen gleichzeitig über einen längeren Zeitraum laden. Der Restliche Verlauf des Tages ist mit dem vorherigen Versuch identisch und bedarf keiner weiteren Betrachtung. Es wird nun viel mehr Wert auf die Ladestationen gelegt werden und deren Veränderung.

Gegenüberstellung des Ladeverhalten

Im direkten Vergleich steht nun die abgegebene Ladeleistung einmal ohne Algorithmus (Abbildung 10.7) und mit Algorithmus (Abbildung 10.8) gegenüber. Der blaue Verlauf spiegelt das Ladeverhalten an der Ladestation 1 (11kW) wieder und der magenta Verlauf spiegelt das Ladeverhalten an der Ladestation 2 (22kW) wieder.

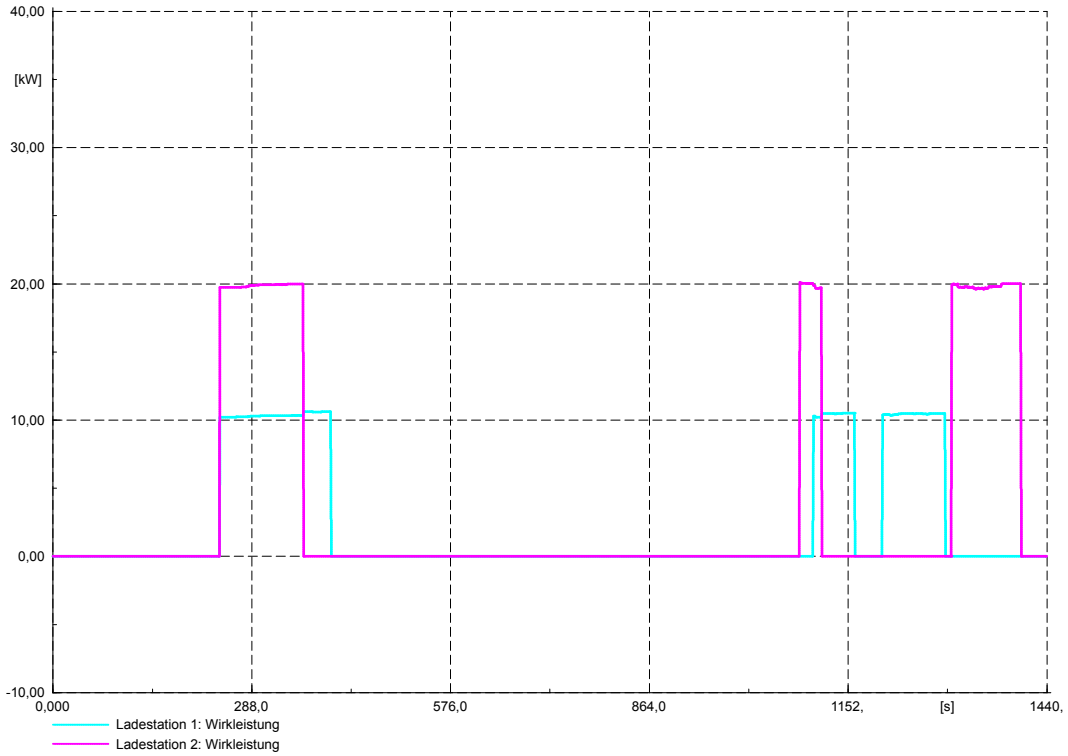


Abbildung 10.7: Versuch 1.2 Ladeverhalten ohne Algorithmus

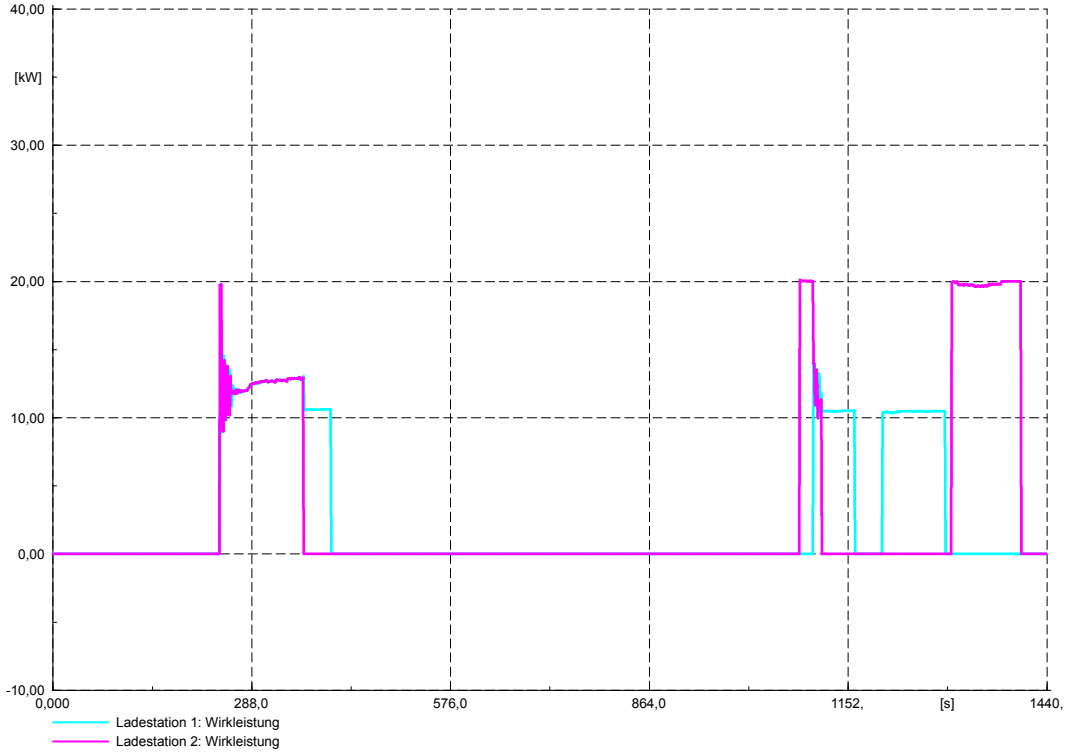


Abbildung 10.8: Versuch 1.2 Ladeverhalten mit Algorithmus

Der erste Ladezeitpunkt überschneidet sich hier und würde eine Gesamtleistung von 33kW benötigten. Im ersten Versuch, wird die Ladeleistung nicht limitiert. Auch findet in den Abendlichen Stunden keine Limitierung statt, sodass über den Abend hinweg immer hohe Ladeleistungen verlangt werden.
Im zweiten Versuch ist das Eingreifen des Algorithmus beim ersten Ladezeitpunkt deutlich zu erkennen. Einmal die Spitzenleistung erreicht, gleichen sich beide Ladeleistung an und bleiben solange gleich, bis eine Ladestation aufhört zu laden. Der Grund für die Limitierung könnte sein, dass die Angeforderte Leistung, die halbe Trafoleistung überschreitet. Des weiteren fällt auf, dass die Ladeleistung beim eingreifen des Algorithmus immer wieder schwankt. Da es sich hierbei um eine neue Simulationsumgebung handelt, kann dieses Problem auch von der Seite PowerFactory stammen. Um dieses genauer zu untersuchen, wurde versucht den Fehler unter anderen Parameter zu reproduzieren.
In der Abbildung 10.9 sind die Ergebnisse der ersten Untersuchung. Hierbei wurden zwei Ladestationen (türkis) im Netz Simuliert die je eine Ladeleistung von 40kW haben. Anschließend wurde die Leistung in 25% Schritten erhöht. Zwischen jeder Erhöhung wurde der Ladeleistung wieder auf 0% gesetzt. Der Trafo (rot) hat eine maximale Leistung von 50kVA.
Im ersten Versuch wurden die beiden Ladestationen mit 25% ihrer Leistung gefahren. Dieses ist in Summe 20kW. Hierbei treten keine Auffälligkeiten auf.
Im zweiten Versuch wurde die Ladestation auf 50% ihrer Leistung gefahren. Dieses ist in Summe 40kW. Auffällig hierbei ist, dass die Ladeleistung schwinkt. Dieser Versuch wurde nochmal mit 75% (60kW) und 100% (80kW) auch hier schwinkt die Ladeleistung. Um die Schwingungen genauer zu Untersuchen, wurde nun die Ladeleistung mit einer Rampe angefahren um den gleichen Zielwert wieder zu erreichen. (Vergleiche Abbildung 10.9).
Vergleicht man nun beide nun beide Abbildungen fällt auf, dass sich an der Ladestation mit einer Gesamtentnahme von 20kW nichts ändert. Bei den anderen drei Stufen fällt auf das die Schwingungen deutlich weniger geworden sind. Die Leistung hierbei überschreitet nicht 25kW. Dieses einspricht der halben Trafoleistung. Somit begrenzt der Algorithmus die maximale Ladeleistung auf die halbe maximale Trafoleistung. Des weiteren kommt der Algorithmus mit einer Laständerung besser zurecht, wenn diese langsam hochfährt. Sprunghafte Änderungen führen zu einem Schwingverhalten, welches den Trafo an sein Limit bringen kann.

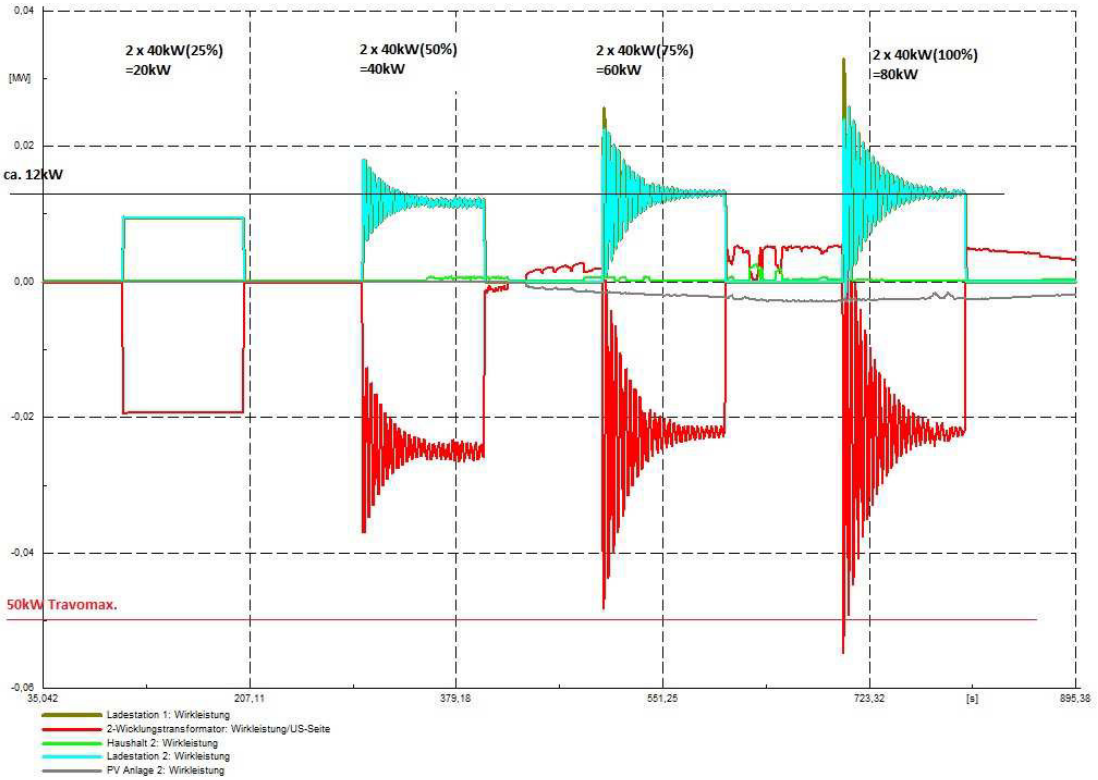


Abbildung 10.9: Versuch 1.2 Ladeverhalten ohne Algorithmus

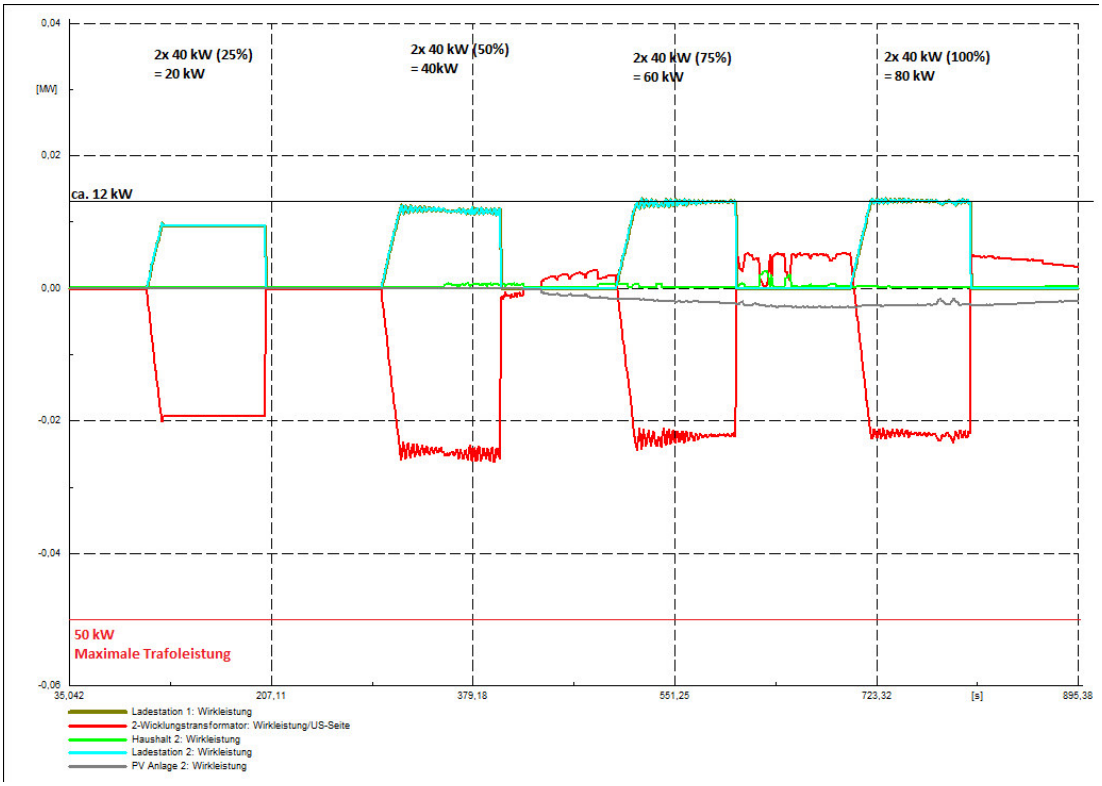


Abbildung 10.10: Versuch 1.2 Ladeverhalten mit Algorithmus

Um nochmal zurückzukommen auf Abbildung 10.8 wurde das Schwingverhalten und die Absenkung der Ladeleistung von Ladestation 2 genauer untersucht. Ein Absenken der Ladeleistung ist dadurch zu verhindern, in dem der Trafo ausgetauscht wird. Der neue Trafo muss mindestens die doppelte Leistung haben, als die Maximale Ladeleistung im Netz. Dieses hat aber die Folge, dass der Trafo beim nicht benutzen der Ladestationen im unteren Leistungsbereich fährt. Hierzu wurde auch Simulationen einmal mit angepassten Trafo durchgeführt. Als Ergebnis fiel auf, dass die Leerlaufverluste überhand nahmen. Somit muss der Trafo an die Ladestation angepasst werden, sodass die Leerlaufverluste möglichst gering sind und die beim Einsatz der Maximalen Ladeleistung der Trafo nicht überlastet wird. Weiterhin auffällig ist, dass die Ladestation 1 einen anstieg der maximalen Ladeleistung zeigt. Die Besonderheit hierbei ist, dass es sich um eine 11 kW Ladestation handelt, der Algorithmus die Ladeleistung auf ca. 14kW anhebt. Dieses gab zu Anlass, nochmals die Implementierung des Algorithmus zu prüfen. Untersuchungen ergaben, dass in der Programmierung die Grenzen eindeutig festgelegt wurden und auch in späteren Simulationsverlauf die Grenzen immer eingehalten werden.
Da es sich bei dem hier eingesetzten Algorithmus um ein Modell handelt, welches noch nicht vollständig entwickelt wurde, haben wir den Entwickler unsere Ergebnisse mitgeteilt.

Gegenüberstellung der Spannungsänderung und Frequenzänderung

Vergleicht man die Spannungsänderung mit (Abbildung: 10.11) und ohne (Abbildung: 10.12) Algorithmus fällt auf, das die Spannungseinbrüche geringer ausfallen. Auch fallen hier die Leistungsschwankungen auf, die sich in der Spannungsübersicht widerspiegeln. Diese Spannungsschwankungen liegen bei einer Differenz von ungefähr 6V innerhalb kurzer Zeit.
Ist das Schwingverhalten vom Algorithmus beendet und hat einen stabilen Wert angenommen, kann der Wert mit der vorherigen Messung verglichen werden. Bezugnehmend auf die letzten Werte, so fällt auf, das die Spannung um 4V weniger einbricht. Somit ist das Netz stabiler bzw. hat geringere Spannungsänderungen.
Den Einfluss von Leistungsänderungen hat nicht nur folgen auf die Spannungsänderung sondern auch auf die Frequenzänderung (Abbildung: 10.13 und 10.14). Hier spiegelt sich die Leistungsschwankungen wieder. Die Frequenz hat mal eine positive, mal eine negative Abweichung von der 50Hz Marke. Betrachtet man aber den Frequenzgang über den Tag hinweg, sind kleine Abweichung immer wieder präsent und verkraftbar. Einzig allein störend ist, dass die Abweichung größer ausfällt, wenn die Ladestation in Benutzung ist. Auch wenn hier die Änderungen zu keinem Einsatz von Regelleistung führen würden, so ist im großen Maßstab zu denken. Wie verhält sich der Algorithmus wenn 10 Ladestationen gleichzeitig laden? Fraglich ist auch ob die Trägheit des Netzes die Folgen ausgleichen kann.

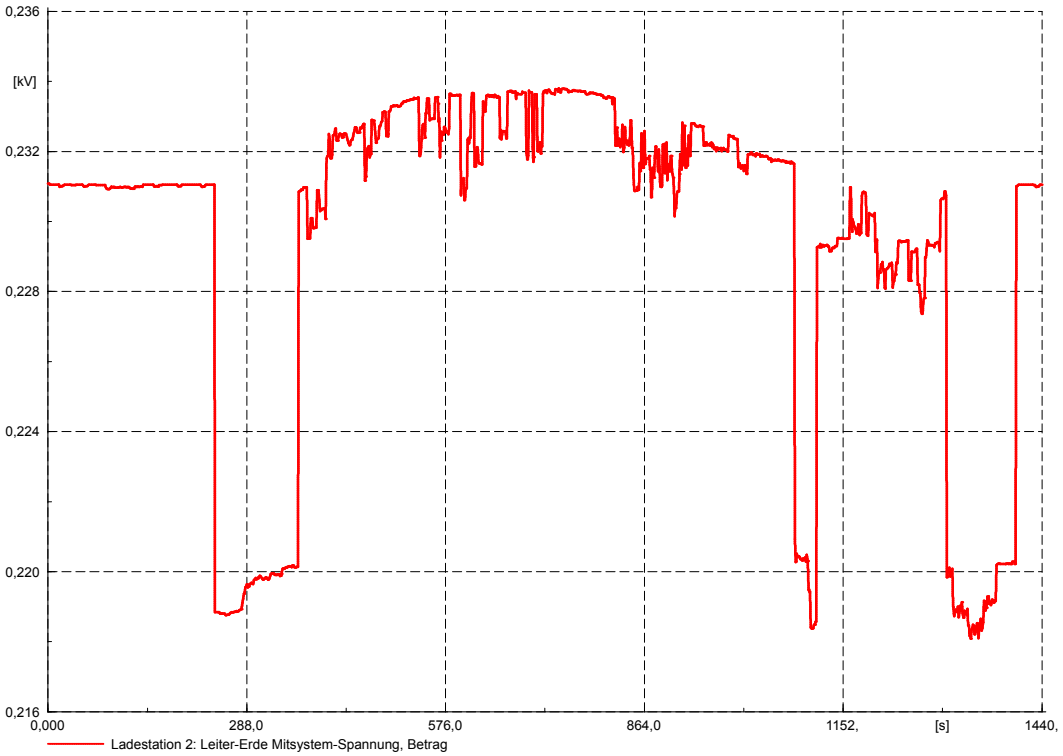


Abbildung 10.11: Versuch 1.2 Spannungsänderung ohne Algorithmus

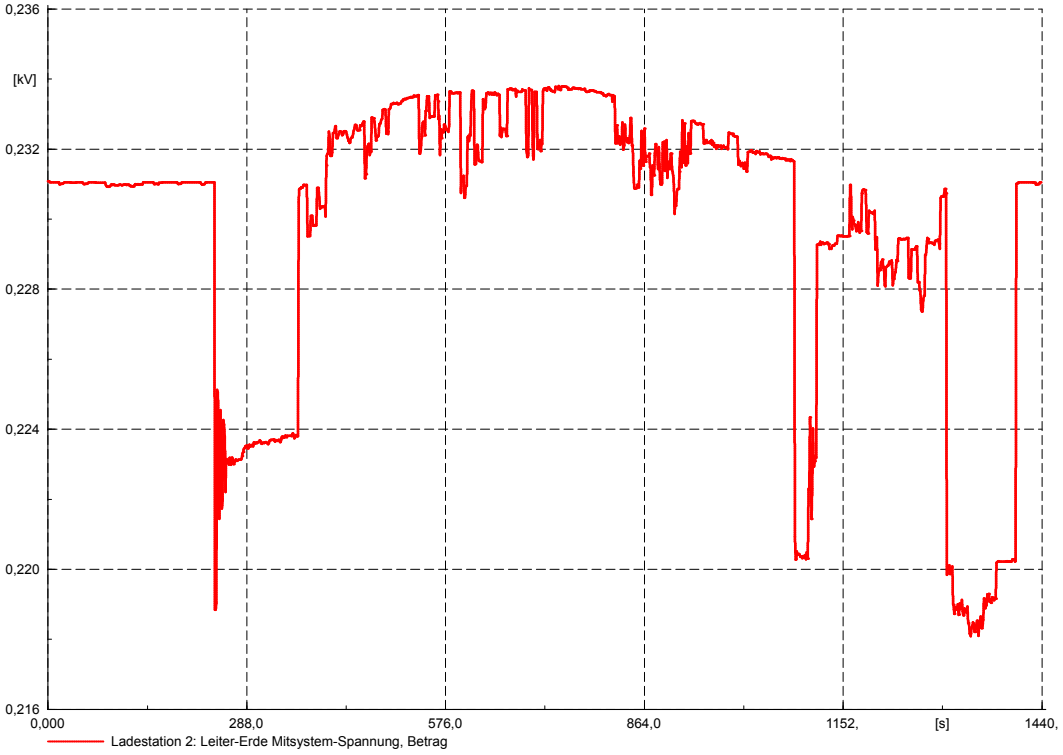


Abbildung 10.12: Versuch 1.2 Spannungsänderung mit Algorithmus

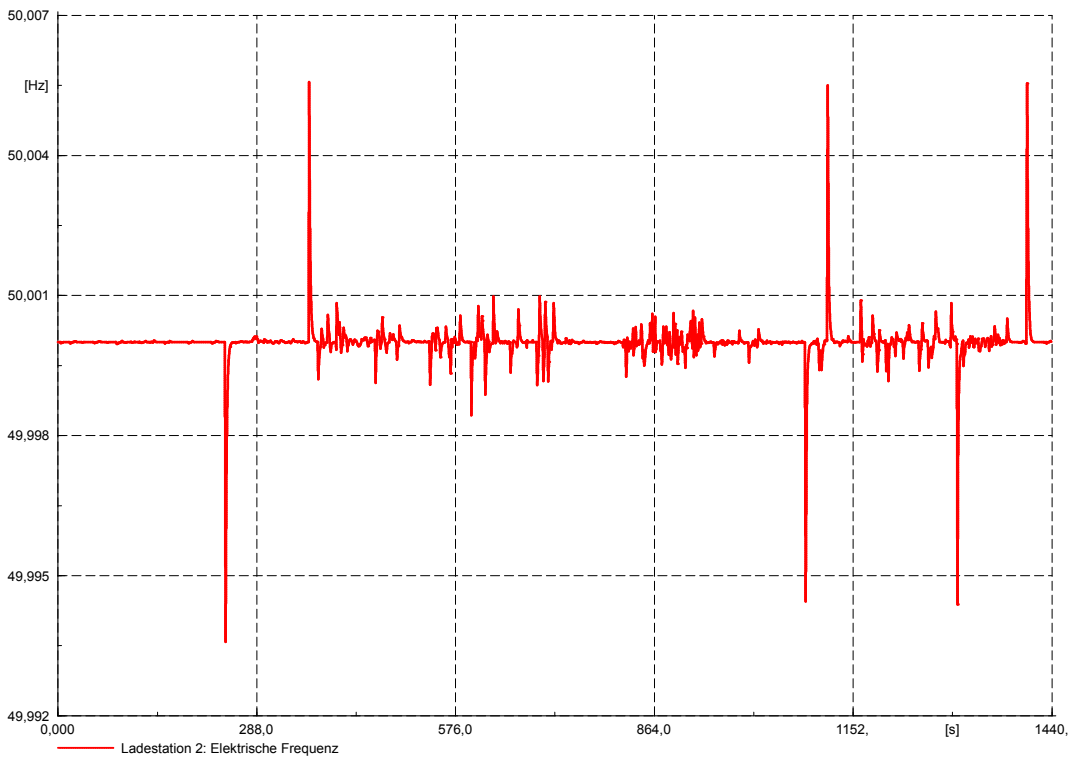


Abbildung 10.13: Versuch 1.2 Frequenzänderung ohne Algorithmus

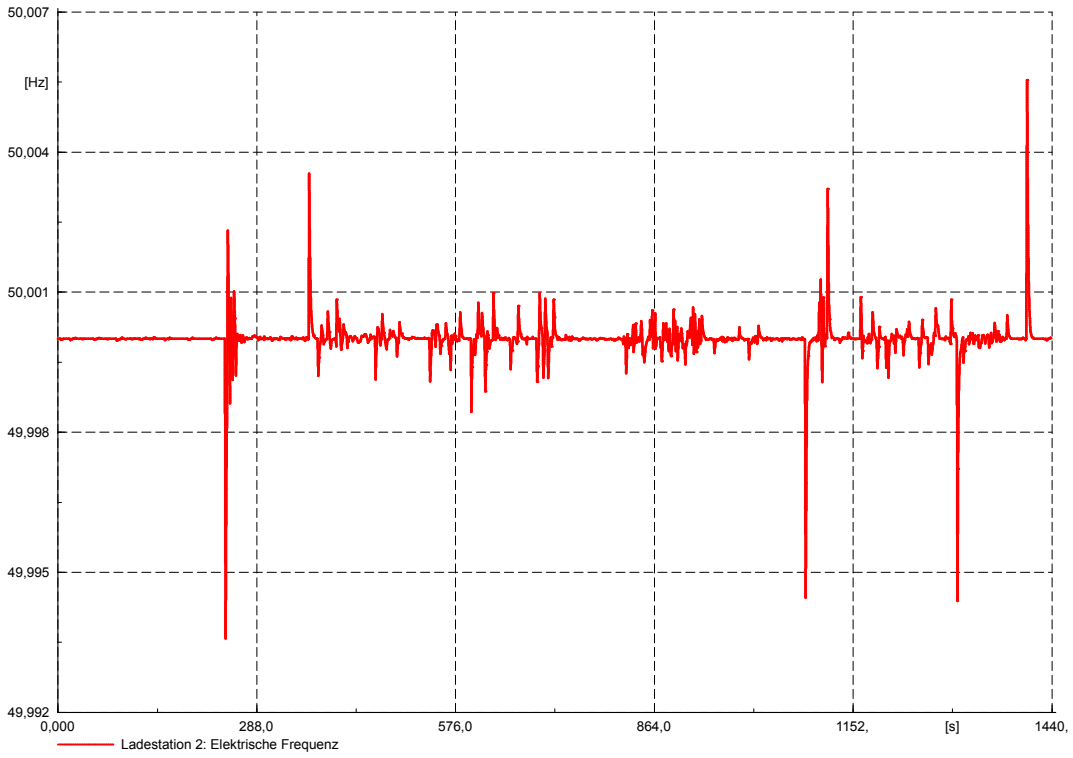


Abbildung 10.14: Versuch 1.2 Frequenzänderung mit Algorithmus

10.6 Versuch 2

Im zweiten Versuch wird ein kurzer Wintertag (21.12.2019) betrachtet. Der Tag startet bei 00:00 Uhr und endet um 23:59 Uhr. Anderes als im Versuch eins, ist dieser Tag ein Samstag, deswegen werden sich die Haushaltslastprofile vom ersten Versuch unterscheiden.
Der Übersichtshalber werden beide Versuche direkt mit einander verglichen.
Aus den Abbildung 10.15 und 10.16 sind am Tagesanfang die gleichen Ladeprobleme zu erkennen wie im ersten Versuch. Entweder wird eine hohe Leistung aus dem Übergeordneten Netz genommen oder Algorithmus sorgt für ein Schwingverhalten beim Laden und hebt die Ladestation 1 über ihre Ladelimit an. Anderes als im Versuch 1 wird nun mitten am Tag noch einmal geladen, dieses mal jede Station komplett einzeln. Gleichzeitig fällt die PV-Anlagenleistung an diesem Tag gering aus, da die Sonne erst spät aufgeht und früh wieder untergeht. Auch sind die Haushaltslasten über den Tag größer, da die Personen am Wochenende mehr zu Hause sind und in der Winterzeit mehr Licht eingeschaltet ist.
Zieht man nun Grafiken aus dem ersten Versuch zu (Abbildung 10.2 und 10.6), stellt man fest, dass diese starke Ähnlichkeit haben. Aus diesem Grund wird die Leitungsauslastung direkt gegenüber gestellt.
Vergleicht man nun die Abbildung 10.17 und 10.18, so ist zu erkennen das die Leitung nur beim Eingreifen des Algorithmus entlastet wird. Sollte im Netz keine weitere Ladestationen in Betrieb sein, wird die Leitung mit der vollen Leistung belastet. Der Algorithmus nimmt auf die Entlastung der Leitung somit keine Rücksicht. Dieses Erkenntnis ist insofern nicht neu, da dem Algorithmus zu keinem Zeitpunkt die aktuelle Belastung zur Verfügung gestellt wurde.

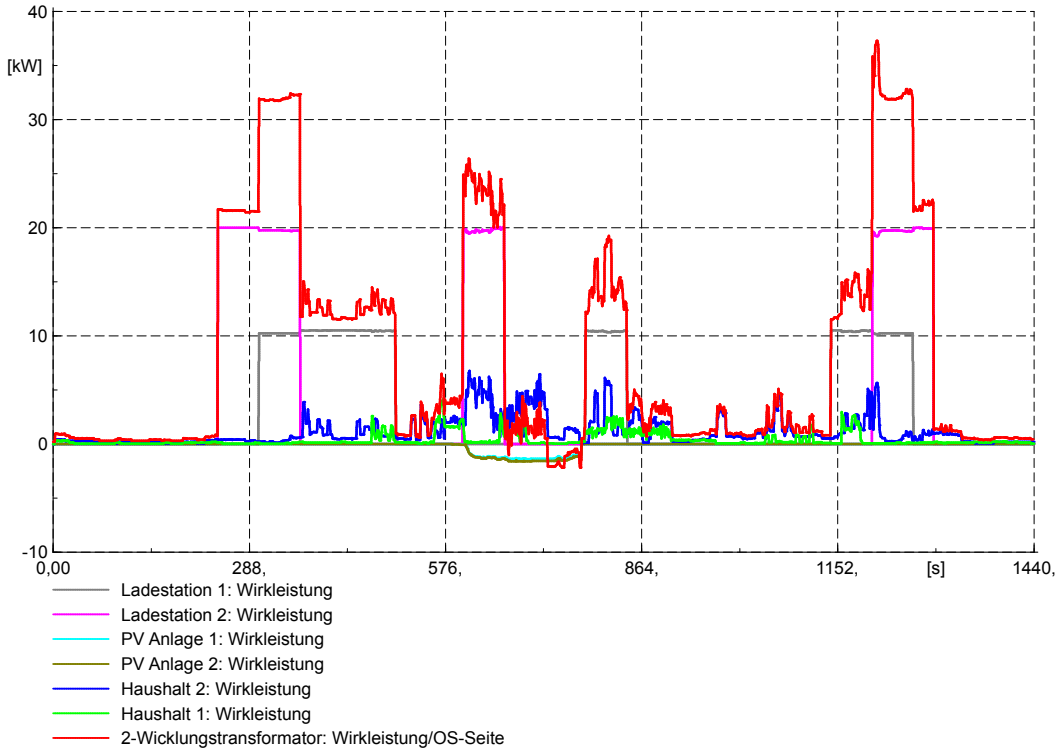


Abbildung 10.15: Versuch 2 Übersicht ohne Algorithmus

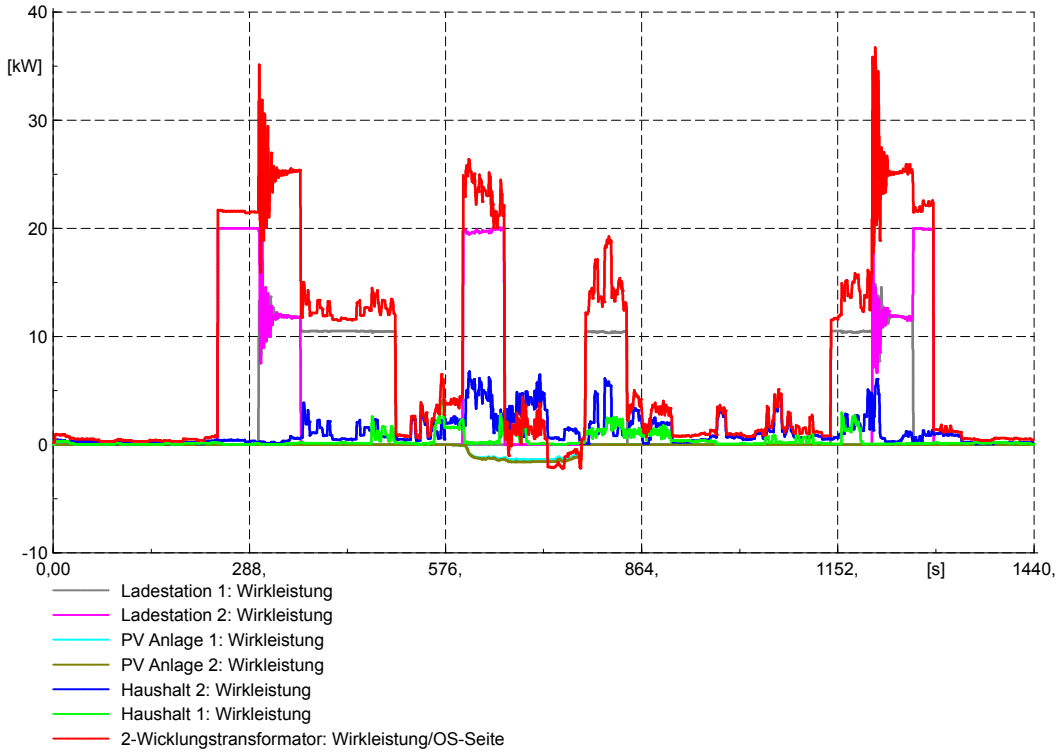


Abbildung 10.16: Versuch 2 Übersicht mit Algorithmus

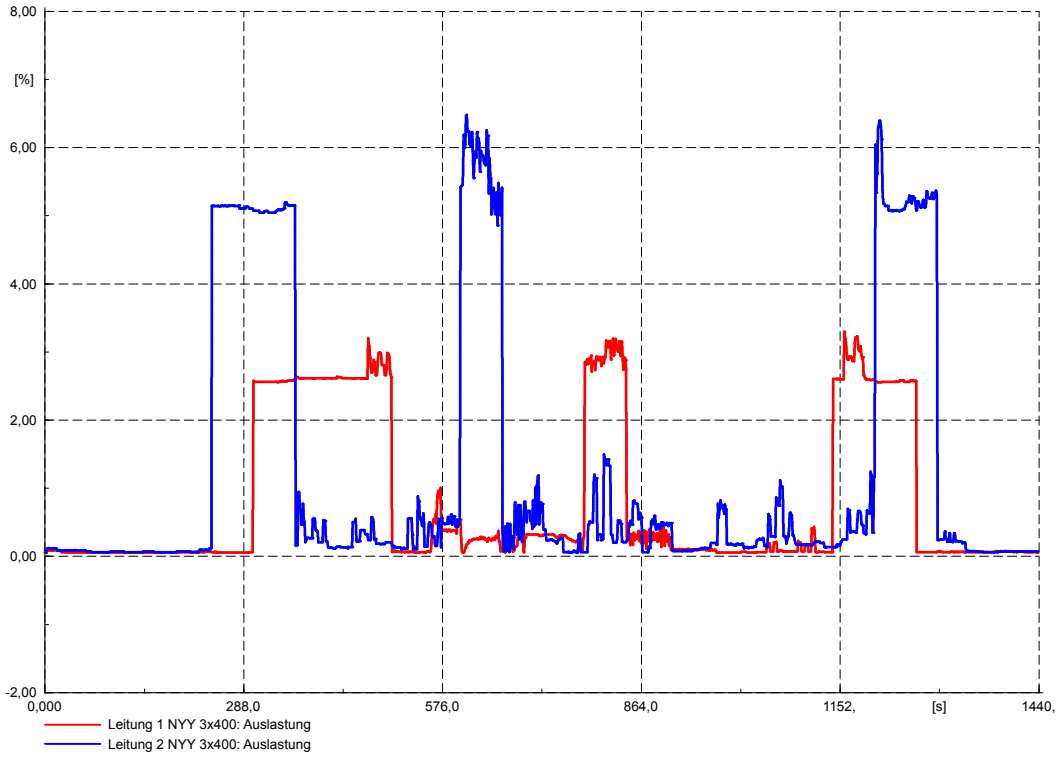


Abbildung 10.17: Versuch 2 Leitungsauslastung ohne Algorithmus

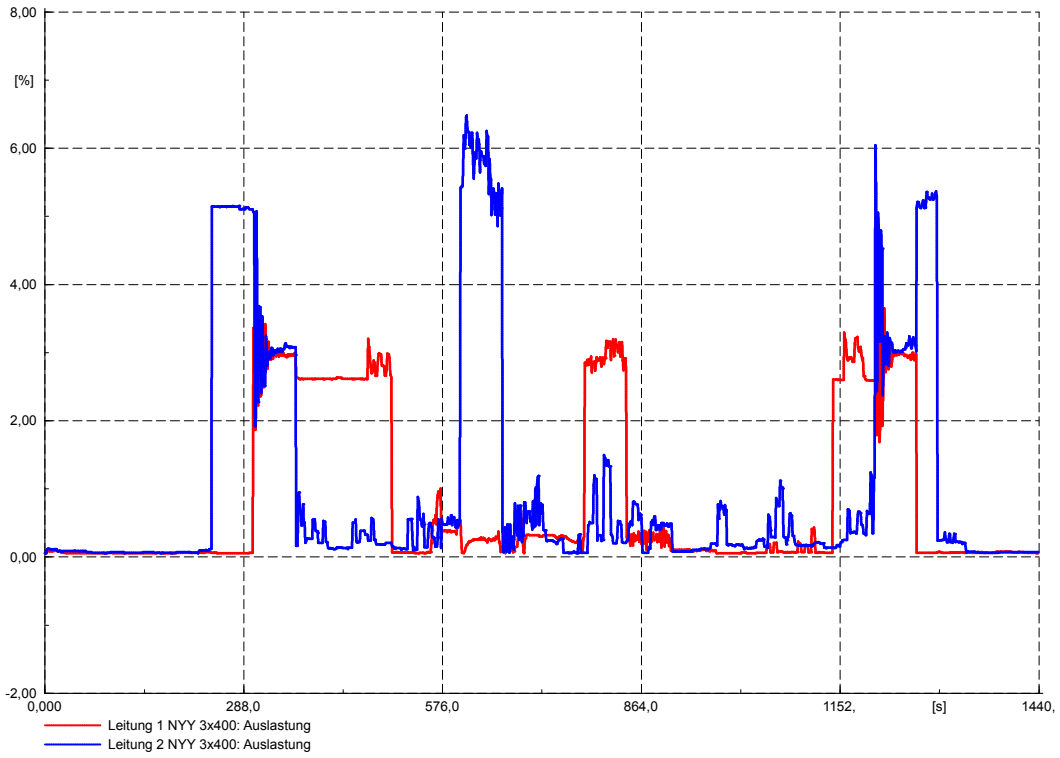


Abbildung 10.18: Versuch 2 Leitungsauslastung mit Algorithmus

10.7 Resümee

Zusammenfassend kann gesagt werden, dass es sich hierbei um eine nutzbringende Simulationsumgebung handelt. Die einzelnen Lastflüsse können bei Bedarf Detailreich untersucht werden. Auch das Zusammenspiel mit dem Java und FTP Server läuft Problemfrei. Die einzelnen Szenarien lassen sich leicht anpassen. Diese Konstellation bietet dem Algorithmus eine Ideale Simulationsumgebung.
Bezugnehmend auf die durchgeführten Versuche ist zu erkennen, dass der Algorithmus dazugeschaltet wurde. Bei Vernachlässigung des Schwingverhalten oder des Anheben der Ladeleistung, begrenzt der Algorithmus die Ladeleistung konsequent. Dabei wird stets vermieden, die Auslastung des Trafos über 50% zu bringen.
Durch die Limitierung der Maximalen Leistung durch den Algorithmus, besteht die Möglichkeit, dass Netz zu Spitzenzeiten vor Spannungseinbrüchen zu bewahren. Auch würde ein Anfahren der Leistung das Netz zusätzlich Entlasten. Die plötzlichen Lastwechsel fielen schon während der Simulation im Frequenzdiagramm negativ auf.

11 Zusammenfassung und Ausblick – Christian Ziegelmann

Auswertung

Um eine fehlerhafte Programmierung in der Simulation zu erkennen wurden zu Beginn verschiedene Test Szenarien probiert. Ohne Werte von Außen einzulesen haben Berechnungen stattgefunden. Die Testszenarien waren einfach aufgebaut und konnten sich gut überprüfen lassen. Die anschließend eingebundenen Werte aus dem Netzwerk haben zu den gleichen Ergebnissen geführt. Daraus konnte gefolgert werden, dass die Werte richtig übertragen und eingelesen wurden. Auch der Datentypen und die Umrechnung der Werte in die richtige Einheit wurden überprüft.
Zur Optimierung der Werte wurde der Server mit dem Algorithmus in die Berechnungen einbezogen. Dieser hat noch nicht seine endgültige Form erreicht und die Tests haben einige Probleme aufgezeigt. Dies hat die Relevanz für eine Testumgebung weiter hervorgehoben. In der Realität hätten die übertragenen Werte schnell zu einer Zerstörung von elektrischen Bauteilen führen können. In der weiteren Entwicklung des Projektes wird sich zeigen, wie gut ein solcher Algorithmus optimiert werden kann.
Übertragungen und Zugriff über FTP auf den Beckhoff-Server waren immer möglich und es kam nicht zu Ausfällen oder Störungen. Das Protokoll läuft stabil und sicher in unserm Laboraufbau und ist für weitere Entwicklungen zu empfehlen.
Die Ergebnisse der Simulation geben erste Denkanstöße, welche weiteren Ansätze einer Optimierung verfolgt werden können. In den nach der Simulation erzeugten Diagrammen ist zu erkennen, dass während der Zeiten lokaler Energieproduktion kaum Lasten betrieben werden. An dieser Stelle wären Zwischenspeicher ein denkbarer Ansatz für ein Vorhalten der Energie im Mikronetz. Beim Aufladen der Fahrzeuge über Nacht sollte eine Informationsaustausch mit den höheren Netzebenen gestaltet werden. Zum Zeitpunkt der Überproduktionen von Windenergie würde das Laden von Speichern die Netze entlasten. Die Einschränkung für diese Art der Speicherung ist, dass sie nur bei nahe gelegenen Mikronetzen in Frage kommt. Weiter ist es denkbar, dass nicht benutzte Fahrzeuge, die über den Tag durch PV-Energie geladen wurden, ihre gespeicherte Energie zu Spitzenzeiten an andere Fahrzeuge abgeben. In dieser sehr komplexen Lösung müssen weit mehr Parameter in eine Simulation fließen, als es in dieser Simulation der Fall war. Die Besitzer müssten ihre PKW für einen solchen Prozess zur Verfügung stellen. Erst damit würde das speichern der Energie möglich. Diese Informationen müssen im Algorithmus bedacht und vorausschauend verarbeitet werden.

Bewertung

Die Generierung eines komplett autarken Mikronetzes scheint bei der Vielzahl an möglichen kurzfristigen Laständerungen unmöglich. Eine Optimierung der Netzstruktur nach dem Bottom-Up-Ansatz, der das Problem von einer niedrigen Hierarchieebe angeht, ist dennoch ein sehr wichtiger Schritt für die Netzsicherheit. Bei einem wachsenden Konsum kann die Lösung nicht alleine im Netzausbau liegen. Auch der schon jetzt große Anteil von Netzkosten am Strompreis zeigt, dass die hier betriebene Forschung ein wichtiger Schritt zur Erarbeitung des Stromnetzes der Zukunft ist.
Der in unserem Versuch betriebene Algorithmus und die Simulationen haben gezeigt, dass es sich um ein sehr komplexes Thema handelt, welches nur Branchenübergreifend zu lösen ist. Die mathematische Herausforderung ist ebenso groß wie die Einführung von Standards beim Informationsaustausch.
In diesem Versuch ist deutlich geworden, dass ein enormer Informationsfluss zur Optimierung und Steuerung eines Netzes erforderlich ist.
Die Optimierung muss bei großen Gebäudekomplexen beginnen wie z.B. der TH-Lübeck. Eine optimierte Nutzung lokal produzierter Energie mit Speichern ist ein erster Schritt. Die Steuerung der Verbraucher muss zukünftig stärker mit der Produktion gekoppelt sein.
Vorschriften zum Bau neuer Gebäude versuchen eine Reduzierung der Energieverbräuche auf unterer Netzebene zu gestalten. Es werden Niedrigenergiehäuser gebaut. Diese verfügen oft über Ladestationen für Elektroautos. Die Vernetzung der Häuser steht dennoch aus und es sind wenige bis keine Schnittstellen für den Informationsfluss gegeben. Die allgemeine Internetanbindung dieser Häuser muss für die Kommunikation von Energiewerten erweitert werden. Die Nutzer müssen sich mit der Nutzung ihrer Daten einverstanden erklären. Auch der größere Einsatz von intelligenten Zählern, die den aktuellen Leistungsbezug übermitteln können, wäre eine Option für Datenaustausch. In diesem Fall sind auch anonymisierte Werte eine Option.
Eine funktionierende Infrastruktur wird in unserem Projekt als gegeben vorausgesetzt. Die in unserem Versuchsaufbau genutzte Kommunikationstechnik auf dem aktuellem Stand der Technik und findet auf allen Automatisierungsebenen bereits Anwendung. Das angedachte System kann auch große Ballungsgebiete optimieren. Umso wichtiger ist es den ungewollten Zugriff zu vermeiden. Jede Schnittstelle stellt eine potenzielle Schwachstelle zur Beeinflussung des Netzes dar. Über die verwendeten Protokolle sind die aktuellen Sicherheitsstandards einzuhalten. Eine Verschlüsselung der übertragenen Daten sollte zu jeder Zeit gewährleistet sein. Die Kommunikation in unserem Labor wurde unzureichend verschlüsselt und müsste einer Optimierung unterzogen werden. Ein Beispiel ist die Übertragung von FTP-Anmeldedaten, hier wird sowohl der Benutzername als auch das Passwort in Klartext übertragen und kann leicht ausgelesen werden.
Mit einer besseren Kommunikation des Netzes ist auch ein Vorhalten von dynamisch einschaltbaren Lasten denkbar, die genau dann betrieben werden, wenn es zu Überkapazitäten im Netz kommt. Eine Möglichkeit ist die Weiterentwickelung der Netzampel, die ihre Information auf einer Plattform bereitstellt. Verschiedene Broker können diese Informationen nutzen um nahegelegene Lasten ein- oder auszuschalten.

Projektentwicklung

Im Laufe des Projektes hat der Projektpartner cbb software GmbH eine Software entwickelt, die auf dem Prinzip der OPC SA-Architektur basiert. Diese Software soll die Kommunikation der Ladesäulen untereinander ermöglichen. Durch die Änderungen kann eine dezentrale Struktur erzeugt werden bei der es nicht nötig ist einen Server bereitzustellen, der als Broker fungiert. Der Algorithmus muss dafür so modifiziert werden, dass eine lokale Berechnung zur Anpassung der Leistung stattfindet. Die Anpassung, sowie die Variablen und Parameterliste der auszutauschenden Daten wird derzeit erstellt und anschließend in die Kommunikationssoftware eingebunden. Auf unseren Laborversuch hatten diese Änderungen keinen Einfluss. Ausgewertet wurde nur das System, welches sich auf einem zentralen Server befindet.
Für die kommende Entwicklung einer Simulationsumgebung für sub-autarke-Mikronetze kann die hier entworfene Simulationsumgebung weiter betrieben werden. Ein dezentraler Algorithmus ist in der Laborumgebung jederzeit implementierbar. Eigene Projekte an der TH-Lübeck könnten in Zukunft unabhängig entwickelt und getestet werden. Es sind verschiedenste Herangehensweisen zur Optimierung vorstellbar. Der Versuchsstand ist dafür nicht von Lizenzen oder Partnerfirmen abhängig und somit für fast alle Ideen modellierbar.
Eine Idee zur weiteren Entwicklung des Laborversuches ist der Aufbau von eigenen dezentralen Strukturen, die auf den Raspberry PI Computern implementiert werden und ein publish–subscribe Prinzip haben. Eine Opensource Software aus diesem Bereich ist z.B. akka. Diese auf Java basierende Kommunikationssoftware wird derzeit in dem Projekt NEW4.0 angewendet. Das NEW4.0 ist ein Forschungsprojekt, dass sich ebenfalls im WiE befindet. Die weitere Forschung in dem Bereich der intelligenten Energienetze ist gegenwärtig erforderlich.
Dieses Projekt hat durch die Nachbildungen von Leistungsflüssen verschiedener Szenarien das Verständnis für den Einfluss von Mikronetzen auf die Systemstabilität gesteigert. Die Relevanz des Projektes zur Entwicklung einer solchen Simulationsumgebung hat den Fokus auf die Komplexität der Energieversorgung geschärft. Eine Projektarbeit mit mehreren Partner hat neue Betrachtungsweisen auf die Auswirkungen der Energiewende gefördert. All das hat die Bedeutung hervorgehoben, Lösungen zu finden die, zur Stabilisierung der Versorgung mit elektronischer Energie führen.

Abkürzungsverzeichnis

ACK	Acknowledgement
ASCII	American Standard Code for Information Interchange
AWL	Anweisungsliste
BHKW	Blockheizkraftwerk
carpeDIEM	Dezentrales Intelligentes Energiemanagement
CSV	Comma Separated Values
DIN	Deutsches Institut für Normung
DLL	Dynamic Link Library
EEG	Erneuerbare-Energien-Gesetz
ELP	Erzeugerlastprofil
FBS	Funktionsbausteinsprache
FIN	Finish
FTP	File Transfer Protocol
FTPS	FTP über SSL
FUP	Funktionsplan
GVL	Globale Variablenliste
Hex	Hexadezimalsystem
HLP	Haushaltslastprofil
HMI	Human Machine Interface
HTTP	Hypertext Transfer Protocol
IANA	Internet Assigned Numbers Authority
IEC	International Electrotechnical Commission
IL	Instruction List
IoT	Internet of Things
IP	Internet Protocol

IPC	Industrie-PC
ISOC	Internet Society
KOP	Kontaktplan
LD	Ladder Diagram
LEP	Light Emitting Plasma
LZP	Ladezeitenprofil
M2M	Machine-to-Machine
MMI	Man Machine Interface
MPP	Maximum Power Point
OPC SA	Open Process Communication Simplified Architecture
OPC UA	Open Platform Communications Unified Architecture
PC	Personal Computer
Pkw	Personenkraftwagen
POU	Programming Organization Unit
PV	Photovoltaik
RFC	Request for Comments
RMS	Root Mean Square
RPI	Raspberry Pi
SCL	Structured Control Language
SFTP	SSH File Transfer Protocol
SPS	Speicherprogrammierbare Steuerung
SSH	Secure Shell
SSL	Secure Sockets Layer
ST	Strukturierter Text
TCP	Transmission Control Protocol
Telnet	Terminal over Network
TH	Technische Hochschule
Trafo	Transformator
TwinCAT	The Windows Control and Automation Technology

UDP	User Datagram Protocol
VISU	Visualisierung
WiE	Wissenschaftszentrum für intelligente Energienutzung
WWW	World Wide Web

Abbildungsverzeichnis

Tabellenverzeichnis

Literatur

[1] Dietmar Abts. *Grundkurs JAVA*. Springer Fachmedien Wiesbaden, 2018. DOI: 10.1007/978-3-658-21907-9.

[2] Schleswig-Holstein Netz AG. *Netzampel*. Hrsg. von Ove Struck. 25. März 2019. URL: https://www.netzampel.energy/.

[3] Bundesnetzagentur. *EEG-Registerdaten*. Hrsg. von Fiete Wulff. 28. Feb. 2019. URL: https://www.bundesnetzagentur.de/DE/Sachgebiete/ElektrizitaetundGas/Unternehmen_Institutionen/ErneuerbareEnergien/ZahlenDatenInformationen/EEG_Registerdaten/EEG_RegDaten_FoerdSaetze.html.

[4] DIgSILENT. *DIgSILENT*. 2. Apr. 2019. URL: https://www.digsilent.de/de/home.html.

[5] DIgSILENT. „PowerFactory 15“. In: *PowerFactory15 Benutzerhandbuch*.

[6] Bundesverband Solarwirtschaft e.V. *Solarstromspeicher*. Hrsg. von Cybay New Media GmbH. 25. März 2019. URL: https://www.solarwirtschaft.de/specials/100000-solarstromspeicher.html.

[7] Eclipse. *Eclipse*. 2. Apr. 2019. URL: https://www.eclipse.org/.

[8] EnergieAgentur.NRW. *Spannungsebenen*. Hrsg. von Herr Dipl.-Ing. Frank Schäfer. 25. März 2019. URL: https://www.energieagentur.nrw/netze/netzinfrastruktur.

[9] EDF Renewable Energy. *peak shaving*. Hrsg. von Rebekka Schuster. 25. März 2019. URL: https://www.edf-re.de/en/peak-shaving-service/what-is-peak-shaving/.

[10] Raspberry Pi Foundation. *raspberry-pi-3*. 25. März 2019. URL: https://www.raspberrypi.org/products/raspberry-pi-3-model-b/.

[11] Werner Frisch, Klaus Hegemann und Udo Schaefer. *Basiswissen IT-Berufe. Vernetzte IT-Systeme. Schülerband*. Bildungsverlag Eins GmbH, 1. Sep. 2018. 577 S. ISBN: 3427111601. URL: https://www.ebook.de/de/product/28192365/werner_frisch_klaus_hegemann_udo_schaefer_basiswissen_it_berufe_vernetzte_it_systeme_schuelerband.html.

[12] ascolab GmbH. *OPC UA Konzepte*. 9. Jan. 2019. URL: http://www.ascolab.com.

[13] cbb software GmbH. *cbb Automoitie / Prüftechnik / Automatisierung / Energie*. 2. Apr. 2019. URL: https://www.cbb.de/home/.

[14] cbb software GmbH. *Open Process Communication Simplified Architecture - opcsa*. 2. Apr. 2019. URL: https://opcsa.de/.

[15] DIgSILENT GmbH. *PowerFactory Netzmodel*. 25. März 2019. URL: https://www.digsilent.de/de/network-model-manager.html.

[16] Dr. Harry Wirth Fraunhofer ISE. „Aktuelle Fakten zur Photovoltaik in Deutschland“. In: 18. März 2019. URL: www.pv-fakten.de.

[17] Beckhoff Automation GmbH & Co. KG. *BeckhoffSPS*. 25. März 2019. URL: https://www.kisspng.com/png-beckhoff-automation-gmbh-co-kg-ethercat-input-outp-1402074/preview.html.

[18] Beckhoff Automation GmbH & Co. KG. *Grundlagen zum File Transfer Protocol*. 25. März 2019. URL: https://infosys.beckhoff.de/index.php?content=../content/1031/tf6300_tc3_ftp/82912779.html&id=.

[19] Technische Hochschule Lübeck. *CarpediemLogo*. Hrsg. von Carpediem. 25. Feb. 2019. URL: http://www.project-carpediem.eu/.

[20] Technische Hochschule Lübeck. *Mikronetz*. Hrsg. von Carpediem. 25. Feb. 2019. URL: http://www.project-carpediem.eu/.

[21] Wolfgang Mahnke, Stefan-Helmut Leitner und Matthias Damm. *OPC Unified Architecture*. Springer Berlin Heidelberg, 2009. DOI: 10.1007/978-3-540-68899-0.

[22] MatthiasDD. *einfacher Kontaktplan als SPS-Programm*. 25. März 2019. URL: https://de.wikipedia.org/wiki/Kontaktplan#/media/File:Kontaktplan_Software.svg.

[23] IMK - Ingenieurbüro für Mechatronik und Kybernetik. *Forschung & Entwicklung Industrie 4.0*. 25. März 2019. URL: https://imk.engineering/industrie-4.0.html.

[24] DIgSILENT PowerFactory. *General Load*. Techn. Ber.

[25] IEA Statistics. *Electric power consumption*. Hrsg. von worldbank. 25. März 2019. URL: https://data.worldbank.org/indicator/EG.USE.ELEC.KH.PC?locations=DE.

[26] Stkl. *Einfaches Beispiel Funktionsplan*. 25. März 2019. URL: https://de.wikipedia.org/wiki/Funktionsbausteinsprache.

[27] Professur Technische Thermodynamik Developed by Noah Pflugradt TU Chemnitz. *Load Profile Generator*. Hrsg. von Noah Pflugradt. 2010-2017.

[28] Chrisitan Ullenboom. *Java ist auch eine Insel*. 2017. ISBN: 978-3-8362-5869-2. URL: https://www.amazon.com/Java-ist-auch-eine-Insel/dp/3836258692?SubscriptionId=AKIAIOBINVZYXZQZ2U3A&tag=chimbori05-20&linkCode=xm2&camp=2025&creative=165953&creativeASIN=3836258692.