Rene Griesser, BSc

Automatisierung eines Kompaktleistungsschalter-Labordemonstrators mit Python Kommunikation

MASTERARBEIT

zur Erlangung des akademischen Grades

Diplom-Ingenieur

Masterstudium

Elektrotechnik-Wirtschaft

eingereicht an der

Technischen Universität Graz

Betreuer

Univ.-Prof. DDipl.-Ing. Dr.techn. Robert Schürhuber

Institut für Elektrische Anlagen und Netze

Dipl.-Ing. Daniel Herbst, BSc

Graz, September 2021

Danksagung

In erster Linie möchte ich mich bei Herrn Prof. Robert Schürhuber, den Betreuer meiner Arbeit, bedanken der diese erst ermöglicht hat. Zudem möchte ich mich an dieser Stelle für die Organisation sowie Bereitstellung der verwendeten Siemens Software recht herzlich bedanken.

Weiters möchte ich mich bei meinem Co-Betreuer, Herrn Dipl.-Ing. Daniel Herbst, für die Betreuung und Unterstützung in allen Belangen bedanken.

Zu guter Letzt möchte ich mich an dieser Stelle auch bei meiner Familie und meinen Freunden bedanken, die mich während meines Studiums in allen Bereichen unterstützt haben.

EIDESSTATTLICHE ERKLÄRUNG

Ich erkläre an Eides statt, dass ich die vorliegende Arbeit selbstständig verfasst, andere als die angegebenen Quellen/Hilfsmittel nicht benutzt, und die den benutzten Quellen wörtlich und inhaltlich entnommenen Stellen als solche kenntlich gemacht habe. Das in TUGRAZonline hochgeladene Textdokument ist mit der vorliegenden Masterarbeit identisch.

Graz, am 20.09.2021

Rene Griesser, BSc

Kurzfassung

Bedingt durch die rasant steigende Anzahl an Elektrofahrzeugen, Photovoltaik Anlagen und anderen erneuerbaren Energiequellen- sowie senken steht das bestehende Energieversorgungsnetz vor großen Herausforderungen. Der Netzausbau gilt aufgrund der in der Bevölkerung nicht immer vorherrschenden Akzeptanz als eine der letzten Maßnahmen. Dies ruht unter anderem auf der enormen Kapitalintensität, aber auch auf der teilweise jahrelangen Erschließung neuer Übertragungswege. Demgemäß wird in den meisten Fällen zuerst eine bestmögliche Optimierung des bestehenden Netzes – ohne zwingenden Ausbau – versucht. Mitunter durch die stetige Zunahme an Elektrofahrzeugen, welche sich bis in das Jahr 2030 enorm intensivieren wird, unterliegt hier vor allem das Niederspannungsnetz einem großen Stresstest. Dahingehend untersucht der Kern dieser Arbeit, inwieweit eine temporäre Rekonfiguration des Niederspannungsnetzes zielführend ist und damit netzentlastend wirken kann. Dafür ist ein Steuerungsalgorithmus in der Programmiersprache Python entwickelt worden, mit welchem verschiedenste Netztopologien computergestützt und in Echtzeit umgeschaltet werden können. Die dafür notwendigen Schalthandlungen werden hierbei an einem Laboraufbau, welcher einen in Ring verschalteten Niederspannungsnetzausläufer repräsentiert, getestet.

Damit wird eine Möglichkeit geschaffen, Unterschiede zwischen einzelnen Netztopologien aufzuzeigen, um in weiterer Folge Schlüsse zu ziehen, ob eine temporäre Rekonfiguration netzentlastend respektive bauteilentlastend wirken kann. Weiters wird im Zuge dieser Arbeit ein Leitungsmodell projektiert, das in den bestehenden Kompaktleistungsschalter-Laboraufbau zur Simulation verschiedener Leitungslängen zwischen einzelnen Leistungsschaltern integriert werden kann. Diese Arbeit soll mit der Bereitstellung des Ansteuerungs- sowie Auswertealgorithmus eine zusätzliche Hilfestellung bieten, um in weiterer Folge Themen im Bereich der temporären Vermaschung am Institut für Elektrische Anlagen und Netze an der Technischen Universität Graz analysieren und bewerten zu können.

Schlüsselwörter: temporäre Vermaschung, Netz-Rekonfiguration, PoSyCo, Netztopologie, Niederspannungsnetz, Elektromobilität, Netzoptimierung, Ansteuerungsalgorithmus, Python, Siemens-Sentron, Kompaktleistungsschalter, radiales Netz, Ringnetz, Vermaschung

Abstract

Due to the rapidly increasing number of electric vehicles, photovoltaic systems and other renewable energy sources as well as sinks, the existing energy supply grid is facing great challenges. Grid expansion is considered one of the last measures to be taken on basis of the lack of acceptance among the population. This is, among other things, because of the enormous capital intensity, but also to the fact that in some cases it takes years to develop new transmission routes. Accordingly, in most cases, the first step is to optimise the existing grid as best as possible without mandatory expansion. Partly due to the steady increase in electric vehicles, which will intensify enormously by the year 2030, the low-voltage grid in particular is subject to a major stress test. With respect to this, the core of this thesis examines the extent to which temporary reconfiguration of the low-voltage grid can be effective and thus relieve the load on the grid. For this purpose, a control algorithm has been developed in the programming language Python with which various grid topologies can be switched over computer-aided and in real time. The necessary switching operations are tested on a laboratory setup (which represents two low-voltage grid feeders connected into a ring).
This provides an opportunity to show differences between individual grid topologies in order to subsequently draw conclusions as to whether a temporary reconfiguration can have a grid-relieving or a component-relieving effect. Furthermore, in the course of this work a line model will be designed, which can be integrated into the existing compact circuit breaker laboratory setup for the simulation of different line lengths between individual circuit breakers. By developing the control and evaluation algorithm, this thesis is intended to provide additional assistance to the Institute for Electrical Systems and Networks at the Graz University of Technology in order to be able to analyse and evaluate topics in the field of temporary meshing in the future.

Keywords: temporary meshing, grid reconfiguration, PoSyCo, grid topology, low-voltage grid, electromobility, grid optimisation, control algorithm, Python, Siemens-Sentron, compact circuit breaker, radial grid, ring-structured grid, meshing

Inhaltsverzeichnis

Abkürzungsverzeichnis

A	Ampere, SI-Basiseinheit der elektrischen Stromstärke
AC	en. alternating current, de. Wechselstrom
APG	Austrian Power Grid; österreichscher Übertragungsnetzbetreiber
csv	en. comma-separated values, de. Komma getrennte Werte
DLL	en. Dynamic Link Library; de. dynamische Programmbibliothek
en.	Englisch, englische Sprache
EV	en. Electric Vehicle, de. Elektrofahrzeug
EVCS	en. Electric Vehicle Charging Station, de. Ladestation für Elektrofahrzeuge
GSDML	Generic Station Description Markup Language
Hz	Hertz, SI-Einheit der Frequenz, Anzahl von Schwingungen je Sekunde
IT-System	Netzsystem für die elektrische Energieversorgung, isoliert betriebener Generatorsternpunkt (oder sehr hochohmig) verbunden mit Betriebserder, Betriebsmittel stehen mit Anlagenerder in Verbindung
KPI	en. Key Performance Indicators, de. wichtige Leistungsindikatoren
kW	Kilowatt
kWp	Kilowatt peak
LIB	Library Datei; für statische und/oder dynamische Bibliotheken
m	Meter, geometrische Längeneinheit
N	Neutralleiter, N-Leiter
nF	Nanofarad, dezimaler Bruchteil von der abgeleiteten SI-Einheit für die elektrische Kapazität Farad
OCPD	en. overcurrent protection device, de. Überstromschutzeinrichtung
p.a.	per annum bzw. pro anno; pro Jahr
PE	en. Protective Earth, de. Schutzerdung; Schutzerdungsleiter
PEN	en. Protective Earth Neutral, de. kombinierter Neutral- und Schutzerdungsleiter
PoSyCo	Power System Cognification, Forschungsprojekt
PV	Photovoltaik
p.u.	per unit
SEO	en. Stored Energy Operator, de. Operator für gespeicherte Energie
SPS	Speicherprogrammierbare Steuerung

TN-System	Netzsystem der elektrischen Energieversorgung, Generatorsternpunkt mit Betriebserder verbunden, Betriebsmittel sind über PE-Leiter oder PEN-Leiter mit dem Betriebserder verbunden
TN-C-System	Netzsystem der elektrischen Energieversorgung, Generatorsternpunkt mit Betriebserder verbunden, Betriebsmittel sind über PEN-Leiter mit dem Betriebserder verbunden
TN-C-S-System	Netzsystem der elektrischen Energieversorgung, Generatorsternpunkt mit Betriebserder verbunden, bis zu den Betriebsmitteln als PEN-Leiter geführt, danach Auftrennung in PE-Leiter sowie Neutralleiter
TN-S-System	Netzsystem der elektrischen Energieversorgung, Generatorsternpunkt mit Betriebserder verbunden, bis zu den Betriebsmitteln vollständige Trennung zwischen PE-Leiter sowie Neutralleiter
TT-System	Netzsystem der elektrischen Energieversorgung, Generatorsternpunkt ist über den Betriebserder geerdet, die Betriebsmittel sind über den Anlagenerder geerdet
V	Volt, SI-Einheit der elektrischen Spannung
VDE	Verband der Elektrotechnik/Elektronik/Informationstechnik e. V.; Vorgängerverein gegründet 1893
VÜN	Vorarlberger Übertragungsnetz GmbH
W	Watt, SI-Einheit der Leistung, beschreibt den Energieumsatz je Zeitspanne
Ω	Ohm, abgeleitete SI-Einheit des elektrischen Widerstandes

Variablenverzeichnis

A	Nennquerschnitt
α	Temperaturkoeffizient
β	Verseilungsfaktor
C	Kapazität
C'	Kapazitätsbelag
G	Leitwert
G'	Ableitungsbelag
I_1	Strom der Phase L1
I_2	Strom der Phase L2
I_3	Strom der Phase L3
I_E	Erdstrom
I_r	Auslösestrom des jeweiligen Kompaktleistungsschalters
t_r	Auslösezeit des jeweiligen Kompaktleistungsschalters

ρ	Spezifischer Widerstand des Leitermaterials
$\rho_{20°}$	Spezifischer Widerstand des Leitermaterials bei 20 °
l	Länge
L	Induktivität
L'	Induktivitätsbelag
R	Widerstand
R'	Widerstandsbelag
R^0	Widerstandswert der Nullsystemkomponente
R^1	Widerstandswert der Mitsystemkomponente
U_{12}	verkettete Spannung zwischen Phase L1 und L2
U_{23}	verkettete Spannung zwischen Phase L2 und L3
U_{31}	verkettete Spannung zwischen Phase L3 und L1
U_{1E}	Leiter-Erde Spannung der Phase L1
U_{2E}	Leiter-Erde Spannung der Phase L2
U_{3E}	Leiter-Erde Spannung der Phase L3
ϑ	Temperatur
ω	Kreisfrequenz
X	Reaktanz
X_L'	Reaktanzbelag

1 Einleitung

1.1 Allgemeine Einführung und Motivation

Aufgrund der zunehmenden Durchdringung der Energiesysteme mit fortwährend intelligenter vernetzen Systemen, sowohl auf Betreiber- als auch auf Kundenseite, ist hier ein Umdenken erforderlich, um einerseits auch in Zukunft Fehler im Netz rechtzeitig erkennen und diese rasch klären zu können, sowie andererseits den zukünftigen Herausforderungen gerecht zu werden. Als gravierendste Herausforderungen, die es in Zukunft zu meistern gilt, zählen hier vor allem der rasant voranschreitende Ausbau der Elektromobilität, Photovoltaiksysteme, Wärmepumpen und viele mehr. Mit dem raschen Fortschreiten der Energiewende[1] stehen vor allem Energieversorgungsunternehmen vor neuartigen Themen wie beispielsweise bidirektionalen Lastflüssen, die ein Umdenken in der Energieversorgung notwendig machen.

Diese Beweggründe haben das Forschungsprojekt „PoSyCo“[2] ins Leben gerufen, welches sich zum Ziel gesetzt hat, die heute bestehenden Netzstrukturen samt zugehöriger konventioneller Schutztechnik („HARDprotection“) um eine zusätzliche „SOFTprotection“ Ebene zu erweitern. Dabei sollen u. a. lokale Überlastungen und Störungen im Netz mit Hilfe neuester Kommunikationstechnologien sowie Analysetools frühzeitig erkannt und mittels Rekonfiguration des Netztes, vermieden werden, um somit das Netz nicht zusätzlich belasten zu müssen. Durch Anwendung dieser Erweiterung des Systems verspricht man sich einerseits eine bessere Absicherung für zukünftige Herausforderungen und andererseits die Ermöglichung neuer Betriebsstrategien.

Das Forschungsprojekt PoSyCo soll ermitteln, ob es zukünftig von Bedeutung sein könnte, eine temporäre Vermaschung des Niederspannungsnetzes in Betracht zu ziehen, um eine Entlastung des Netzes respektive Netzausfälle und die dadurch entstehenden Kosten zu vermindern.

Diese und noch weitere spannende Fragen zum Thema temporäre Rekonfiguration des Niederspannungsnetzes sollen im Zuge dieses Forschungsprojekts mit Hilfe eines Projektkonsortiums aus verschiedenen Projektpartnern, sowohl aus dem Forschungsbereich aber auch aus der Industrie, sowie eines Verteilnetzbetreibers, angestrebt werden.

[1] Beschreibt den Übergang von nicht nachhaltigen, fossilen Energieträgern sowie Kernkraft hin zu erneuerbaren und somit nachhaltigen Energieträgern [31]

[2] **Po**wer **Sy**stem **Co**gnification - Forschungsprojekt [33]

1.2 Stand der Technik

Grundsätzlich geht zum Zeitpunkt der Durchführung dieser Masterarbeit der Trend, auch aufgrund des entsprechenden Kostendrucks, eher in Richtung Entmaschung des Netzes bzw. zu kleineren Maschenweiten. Dieses erhöhte Ausfallrisiko bzw. die erhöhte Fehleranfälligkeit bei nicht vermaschten Systemen im Vergleich zu vermaschten Systemen wird aufgrund Betrachtungen der Wirtschaftlichkeit jedoch in Kauf genommen. [1] In Zukunft wäre also ein System mit höchsten Sicherheitsstandards sowie einer günstigen wirtschaftlichen Ausrichtung erstrebenswert. Zurzeit verfügt das Stromnetz auf Ebene der Niederspannung über eine sogenannte „HARDprotection“ in Form von Leistungsschaltern, Leistungsschutzschaltern sowie Schmelzsicherungen.

PoSyCo beschäftigt sich hier vor allem mit der zusätzlichen Implementierung einer sogenannten SOFTprotection Schicht. Diese soll das Netz mit Hilfe von Kommunikationstechnologiekomponenten sowie Automatisierungstechnik für zukünftige Netzsituationen bestens vorbereiten und somit eine Brücke zu einem Netz der Zukunft, auch „Smart-Grid“[3] genannt, bilden.

1.3 Ziel

Ziel dieser Masterarbeit ist es, einen bestehenden Labordemonstrator um zusätzliche Komponenten im Bereich der Netzwerkkommunikation zu erweitern, um die manuelle Bedienung der im Schaltschrank verbauten Kompaktleistungsschalter um die zusätzliche Möglichkeit einer vollständig automatisierten Ansteuerung zu erweitern. Dafür soll mit Hilfe der Programmiersprache Python ein entsprechender Algorithmus entwickelt werden, welcher diese Ansteuerungsaufgaben übernimmt und eine Integration des bestehenden Algorithmus zur automatisierten Netzrekonfiguration ermöglicht. Ergänzend dazu sollen auch die Messwerte der Leistungsschalter entsprechend ausgelesen und weiterverarbeitet werden können. Zusammengefasst besteht das Kernziel dieser Arbeit in der Entwicklung einer intelligenten Ansteuer- sowie Auswertesoftware für den interaktiven Betrieb des Labordemonstrators im Zuge von weiterführenden Versuchen und Tests sowie für den generellen künftigen Laborbetrieb.

Ergänzend dazu wird zur praxisnahen Verwendung des Labordemonstrators bzw. um zukünftige Versuchsergebnisse realitätsnäher darzustellen ein Erweiterungsmodul in Form eines Niederspannungsleitungsmodells konzeptioniert. Mit Hilfe dieses Moduls, von welchem maximal vier Stück seriell in den bestehenden Aufbau eingebunden werden können, ist es möglich, verschiedenste Niederspannungskabel mit unterschiedlichsten Nennquerschnitten und Kabellängen abzubilden.

[3] Intelligentes Stromnetz welches die Aktionen seiner Nutzer, Erzeuger, Speicher sowie Verbraucher mit Hilfe von kommunikativer Vernetzung integriert um eine effiziente, nachhaltige, wirtschaftliche sowie sichere Energieversorgung zu gewährleisten. [35]

1.4 Methodik

Im Zuge des Forschungsprojekts PoSyCo beschäftigt sich das Institut für Elektrische Anlagen und Netze der Technischen Universität Graz vor allem mit dem Thema der temporären Rekonfiguration des Niederspannungsnetzes. In einer vorangegangenen Masterarbeit am Institut wurde ein Laboraufbau eines vermaschten Niederspannungsnetzes in seiner Grundform gebaut. Mit diesem Grundaufbau konnten einzelne Leistungsschalter manuell mittels Taster geschaltet und so verschiedene Netzstrukturen nachgebildet werden. Dieser bestehende Laboraufbau wurde im Zuge der vorliegenden Masterarbeit mit mehreren Kommunikationskomponenten erweitert. Dies ermöglicht es in weiterer Folge, die Kompaktleistungsschalter mittels des in dieser Arbeit entwickelten externen Algorithmus automatisiert ansteuern zu können. Als Kommunikationsstruktur wird hier der Industrial Ethernet Standard, auch PROFINET genannt, verwendet. Diese Kommunikationsstruktur ermöglicht in weiterer Folge die intelligente Ansteuerung der Leistungsschalter des Labordemonstrators über die Netzwerkschnittstelle. Damit einhergehend sind auch Remote-Zugriffe auf das System möglich und es bieten sich noch realistischere Szenarien für zukünftige Anwendungen. Als Basis der Ansteuerung dient ein digitales Netzmodel des Labordemonstrators in der Netzberechnungssoftware „DIgSILENT PowerFactory“.

Zusätzlich wird eine Machbarkeitsstudie samt Berechnungen, Auslegung sowie Kostenaufstellung von diversen Leitungsmodulen durchgeführt. Maximal vier dieser Leitungsmodule können zusätzlich seitlich an dem Labordemonstrator über sogenannte HEAVICON Steckverbinder in Serie eingebunden werden, um so ein noch realistischeres Abbild eines Netzabschnitts simulieren zu können.

Eingangs galt es sich in die bereits existierende Hardware einzulesen und den bestehenden Aufbau zu verstehen. Darauffolgend wurden Lösungen gesucht, um den Laboraufbau mittels Netzwerkanbindung intelligent zu erweitern, sodass mit verschiedenen Softwarelösungen auf die externe Peripherie (Kompaktleistungsschalter) zugegriffen werden kann. Weiters wurde im Zuge dieser Masterarbeit die Programmiersprache Python in der Entwicklungsumgebung PyCharm verwendet, in welcher der Ansteuerungsalgorithmus entwickelt wurde. Auch der Umgang und das Zusammenspiel mit den verschiedensten Softwarepaketen wie beispielsweise Siemens powerconfig zur Inbetriebnahme, Siemens TIA Portal V16 zur Projektierung sowie die Entwicklungsumgebung PyCharm mit der Programmiersprache Python, aber auch DIgSILENT PowerFactory, wurden bewältigt.

Zur Auslegung bzw. Projektierung des Leitungsmoduls wurde auf bestehende Literatur zurückgegriffen und mit auf die Herstellung von Einphasendrosseln sowie Widerständen spezialisierten Firmen Kontakt aufgenommen. Zudem wurden entsprechende Angebote betreffend die Hardwarekomponenten eingeholt, um so eine belastbare Kalkulation zu gestalten. Anhand dieser Berechnungen sowie der Kostenaufstellung, die mit allen notwendigen Informationen hinterlegt ist, ist in weiterer Folge die Anfertigung solcher Module möglich.

1.5 Ergebnisse, Schlussfolgerungen und Ausblick

Um die automatisierte, netzwerkgebundene Ansteuerung der Leistungsschalter zu ermöglichen, wurde der Aufbau um eine speicherprogrammierbare Steuerung (SPS) der Siemens S7-Reihe erweitert. Da die genannte SPS in diesem Projekt lediglich als Stellglied fungiert, wird hier nicht wie üblich eine SPS-Programmierung durchgeführt, sondern die gesamte Ansteuerung der externen Peripherie (in diesem Fall die Kompaktleistungsschalter) über eine externe Entwicklungsumgebung in der Programmiersprache Python vorgenommen. Die gesamten Status- sowie Steuerbefehle für die Kompaktleitungsschalter werden über ein externes Python Skript verarbeitet, berechnet sowie ausgewertet. Damit besteht die Möglichkeit der Ansteuerung ausschließlich mittels PC und entsprechendem Skript ohne Notwendigkeit jeglicher anderer Software, wie beispielsweise der Siemens-Ansteuerungssoftware. Trotzdem besteht jedoch die Möglichkeit einer zusätzlichen Überwachung der Status- und Steuerbytes, sowie auch der von den Kompaktleistungsschaltern bereitgestellten Messdaten in der Siemens Software TIA Portal V16 und einem erstellten Webserver. Bei etwaigen Fehlern bei der Kommunikation (zB fehlerhafte Befehle am Bus) besteht zudem immer noch die Möglichkeit der manuellen Eingabe der jeweiligen Steuerbefehle an der Vorderseite des Laboraufbaus mittels der entsprechenden Taster sowie auch direkt an den Kompaktleistungsschaltern. Mit der Erweiterung des Laboraufbaus sowie dem entwickelten Ansteuerungsalgorithmus ist es demnach möglich, verschiedenste Szenarien der temporären Vermaschung mittels externer Ansteuerung zu simulieren. Der Algorithmus ermöglicht es, die Leistungsschalter schnell und vor allem zeitgleich, auf Basis der Ergebnisse der Lastflussberechnungen, anzusteuern. Da in der Realität ein Netz bzw. ein Netzabschnitt aus einer Vielzahl an Komponenten besteht wäre es nicht möglich, alle Leistungsschalter zeitgerecht manuell zu schalten. Dies wiederum lässt die Notwendigkeit einer entsprechend automatisierten Ansteuerung der Kompaktleistungsschalter (im konkreten Fall via Python) erahnen. Zudem werden fehlerhafte Schalthandlungen – bereits vor deren Durchführung – vom Ansteuerungsalgorithmus erkannt, was nicht erforderliche (respektive mögliche) Schalthandlungen der Kompaktleistungsschalter verhindert und sich damit positiv zugunsten deren Lebensdauer auswirkt.

Bei der Auslegung der Leitungsmodule wurden verschiedenste Nennquerschnitte für Niederspannungskabel berechnet und unter Berücksichtigung der Kabellänge mögliche Variationen für zukünftige Leitungsmodule zusammengestellt. Hinsichtlich der ermittelten Bauteilwerte, als auch der Werte, die man käuflich erwerben kann, konnten hier verschiedene Möglichkeiten dargelegt werden.

Zukünftig wird sich die Zahl an sogenannten Prosumenten[4], welche nicht nur elektrische Energie aus dem Netz beziehen, sondern auch in dieses rückspeisen, entsprechend erhöhen. Auch bedingt durch die ambitionierten Klimaziele Österreichs bis 2030 [2, 3] wird sich dieser

[4] **Prosument**: Verbraucher (Kon**sument**) der gleichzeitig auch Erzeuger ist (zB Rückeinspeisung einer Photovoltaik Anlagen in das Netz. **Pro**duzent), oftmals (aus dem Englischen) als Prosumer bezeichnet [34]

Faktor noch zunehmend intensivieren. Ein weiterer, jedoch nicht zu vernachlässigender Faktor, sind die steigende Anzahl an Elektrofahrzeugen sowie auch damit unmittelbar verbunden, die zugehörige Ladeinfrastruktur.

All diese Entwicklungen lassen erahnen, dass das österreichische Übertragungsnetz, aber auch das gesamte europäische Verbundnetz in den nächsten Jahren bzw. Jahrzehnten vor enormen Herausforderungen stehen werden, um den steigenden Anforderungen genüge zu leisten. Das Forschungsprojekt PoSyCo betrachtet dabei diese Veränderung und soll erste Konzepte bzw. Gegenmaßnahmen für diese Herausforderungen erarbeiten.

2 Theoretischer Hintergrund

Nachfolgend wird der theoretische elektrotechnische Hintergrund, auf dem diese Arbeit basiert, zusammengefasst. Behandelt werden u. a. die Rolle Österreichs im europäischen Verbundnetz, die Struktur der österreichischen Verteilernetze sowie die verschiedenen, zum Einsatz kommenden, Netzstrukturen. Weiters werden die verschiedenen Systeme nach Art ihrer Erdverbindung gemäß OVE E 8101:2019-01-01 [4] erläutert sowie Fehlerfälle, welche im Niederspannungsnetz auftreten können, beschrieben. Darauffolgend wird auf das Thema der temporären Rekonfiguration sowie auf die Auswirkungen der Elektromobilität auf das Niederspannungsnetz eingegangen. Abgeschlossen wird das Kapitel mit den zukünftigen Herausforderungen bestehender Netzstrukturen.

2.1 Österreich im europäischen Verbundsystem

Unter dem kontinentalen europäischen Verbundsystem UCTE[5], oftmals auch europäisches Verbundnetz genannt, versteht man den Zusammenschluss der einzelnen nationalen Übertragungsnetze. Ausnahmen bilden hier die Länder Großbritannien, Irland und die skandinavischen Länder Schweden, Norwegen sowie Finnland. Auch die drei baltischen Staaten Lettland, Litauen und Estland gehören nicht dem kontinentalen europäischen Verbundnetz UCTE an. Eine entsprechende Zuordnung der einzelnen Länder ist Abbildung 1 zu entnehmen. Alle aufgezählten Verbundnetze weisen zwar dieselbe Netzfrequenz von 50 Hz auf, sind aber asynchron zueinander. Das bedeutet, dass es eine Abweichung bezüglich der Phasenlage bzw. der Netzfrequenz der einzelnen Systeme gibt, weshalb Energie zwischen diesen Netzgebieten nur mittels Gleichstrom, beispielsweise in Form von HGÜ-Hochspannungsgleichstromübertragung, ausgetauscht werden kann. [5]

Seit dem Jahr 2009 wurden alle organisatorischen sowie administrativen Aufgaben der sogenannten ENTSO-E[6] übertragen. Österreichische Mitglieder der ENTSO-E sind die Austrian Power Grid (APG) sowie die Vorarlberger Übertragungsnetze (VÜN). Die Vorarlberger Übertragungsnetze sind aufgrund ihrer geografisch günstigen Lage zu den Nachbarstaaten Deutschland und der Schweiz eine Kooperation mit der APG eingegangen, im Rahmen welcher jedoch die Austrian Power Grid der internationale Ansprech- sowie Koordinationspartner bei Betriebsführungsangelegenheiten für das österreichische Übertragungsnetz ist und somit auch die VÜN vertritt. In Abbildung 2 ist die zentrale Rolle Österreichs im europäischen Strommarkt deutlich zu erkennen. Diese soll das Ausmaß der Vermaschung des europäischen Verbundsystems grafisch verdeutlichen. [5, 6]

[5] **UCTE**…**U**nion for the **C**o-ordination of **T**ransmission of **E**lectricity; de: Union für die Koordinierung des Transports von Elektrizität

[6] **ENTSO-E**…**E**uropean **N**etwork of **T**ransmission **S**ystem **O**perators for **E**lectricity

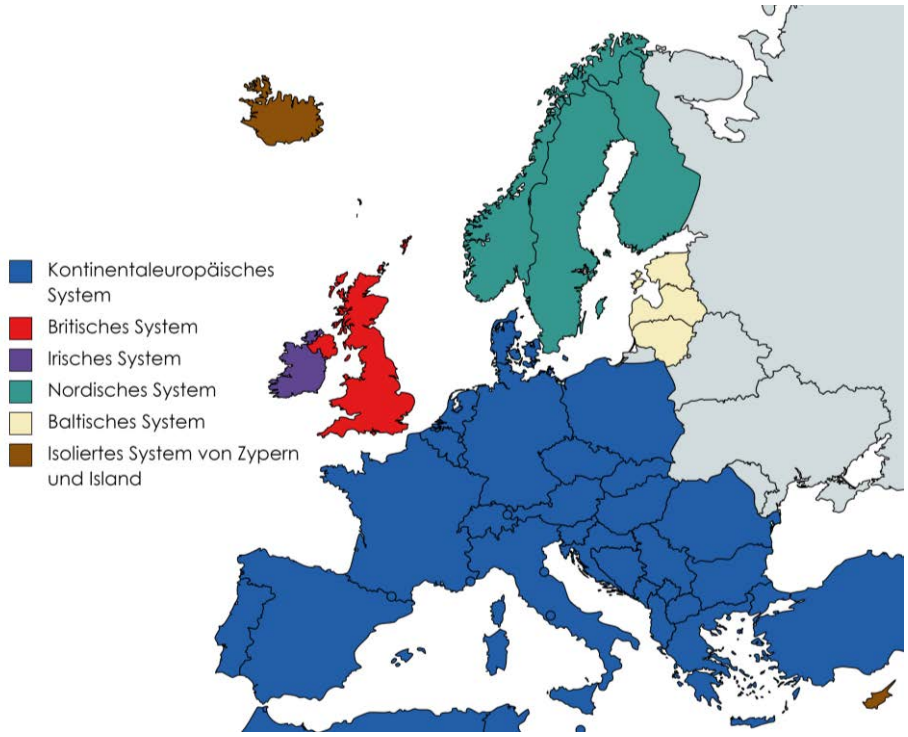


Abbildung 1: Europäisches Verbundnetz auf Basis [7]

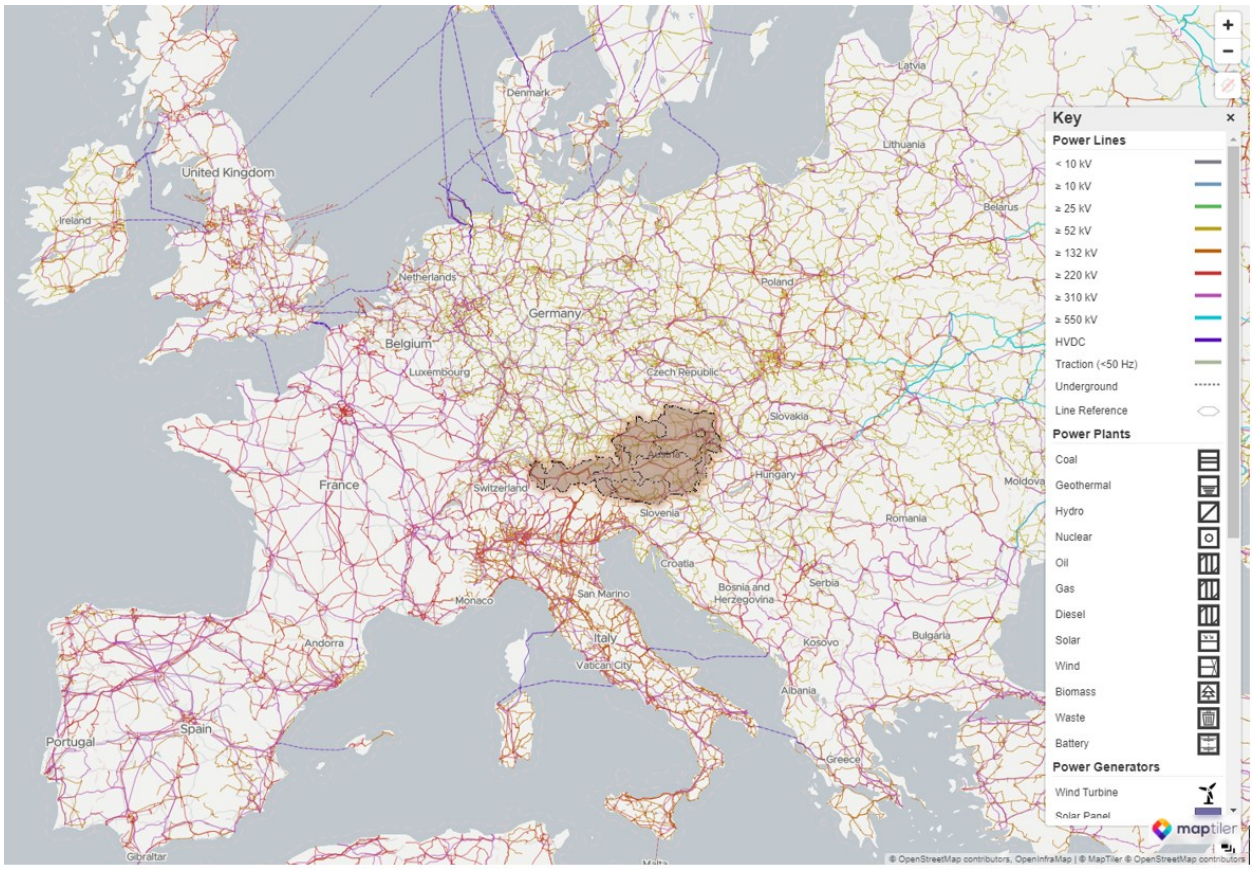


Abbildung 2: Ausmaß der Vermaschung in Europa [8]

2.2 Struktur der österreichischen Verteilernetze

Die APG ist als unabhängiger Übertragungsnetzbetreiber für den reibungslosen Betrieb des Stromnetzes in Österreich, aber auch für den Ausbau des Übertragungsnetzes sowie der grenzüberschreitenden Anbindung zu den Nachbarstaaten in das europäische Übertragungsnetz verantwortlich. Um eine zuverlässige sowie qualitativ hochwertige Stromversorgung in Österreich sicherstellen zu können, bedarf es einem einwandfreien Zusammenspiel auf allen Ebenen. Um den Strom von den Erzeugungsschwerpunkten zu den Kunden zu transportieren, sind sogenannte Übertragungsnetze notwendig. Diese bestehen aus Leitungen mit einer Nennspannung von 220 kV sowie 380 kV und sind somit der Netzebene 1 zuzuordnen, welche sogleich die höchste Spannungsebene darstellt. Auf Netzebene 1 wird Strom mittels Höchstspannung möglichst verlustarm über weite Strecken des Landes sowie auch über die Landesgrenzen hinaus transportiert. Die Netzebene 2 ist eine Zwischenebene und beinhaltet die Umspannung von der Höchstspannung (Netzebene 1) auf die Hochspannung der Netzebene 3, welche mit 110 kV klassifiziert wird. Netzebene 3, also die Hochspannungsebene, dient vor allem der regionalen Verteilung des Stromes. Brücke zwischen Netzebene 3 und Netzebene 5 ist wiederum eine Zwischenebene (Netzebene 4), welche die Umspannung von 110 kV auf die im Mittelspannungsnetz üblichen Werte von 10 kV bis 35 kV sicherstellt. Netzebene 6 dient der Umspannung des Mittelspannungsnetzes, auf das sich in Netzebene 7 befindliche Niederspannungsnetz mit einer Nennspannung von 0,4 kV bzw. 400 V. Abbildung 4 verdeutlicht die Gliederung der sieben beschriebenen Netzebenen. [2]

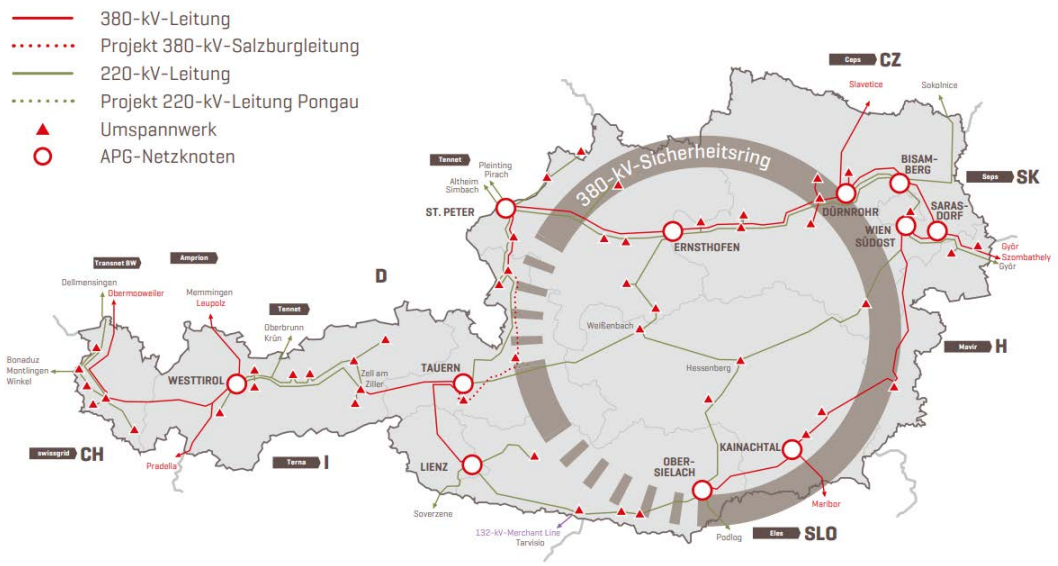


Abbildung 3: Topologie des österreichischen Übertragungsnetzes [2]

In Abbildung 3 ist die Topologie des österreichischen Übertragungsnetzes dargestellt. Wie zu erkennen ist, verfügt Österreich über einen derzeit noch offenen 380–kV–Sicherheitsring. Die Idee dazu gibt es schon seit den 1970er-Jahren um das Stromnetz vor allem für zukünftige Anforderungen zu rüsten. Ziel ist es, diesen Ring in den nächsten Jahren vollständig zu schließen, um so die Versorgungssicherheit weiter zu erhöhen. [2]

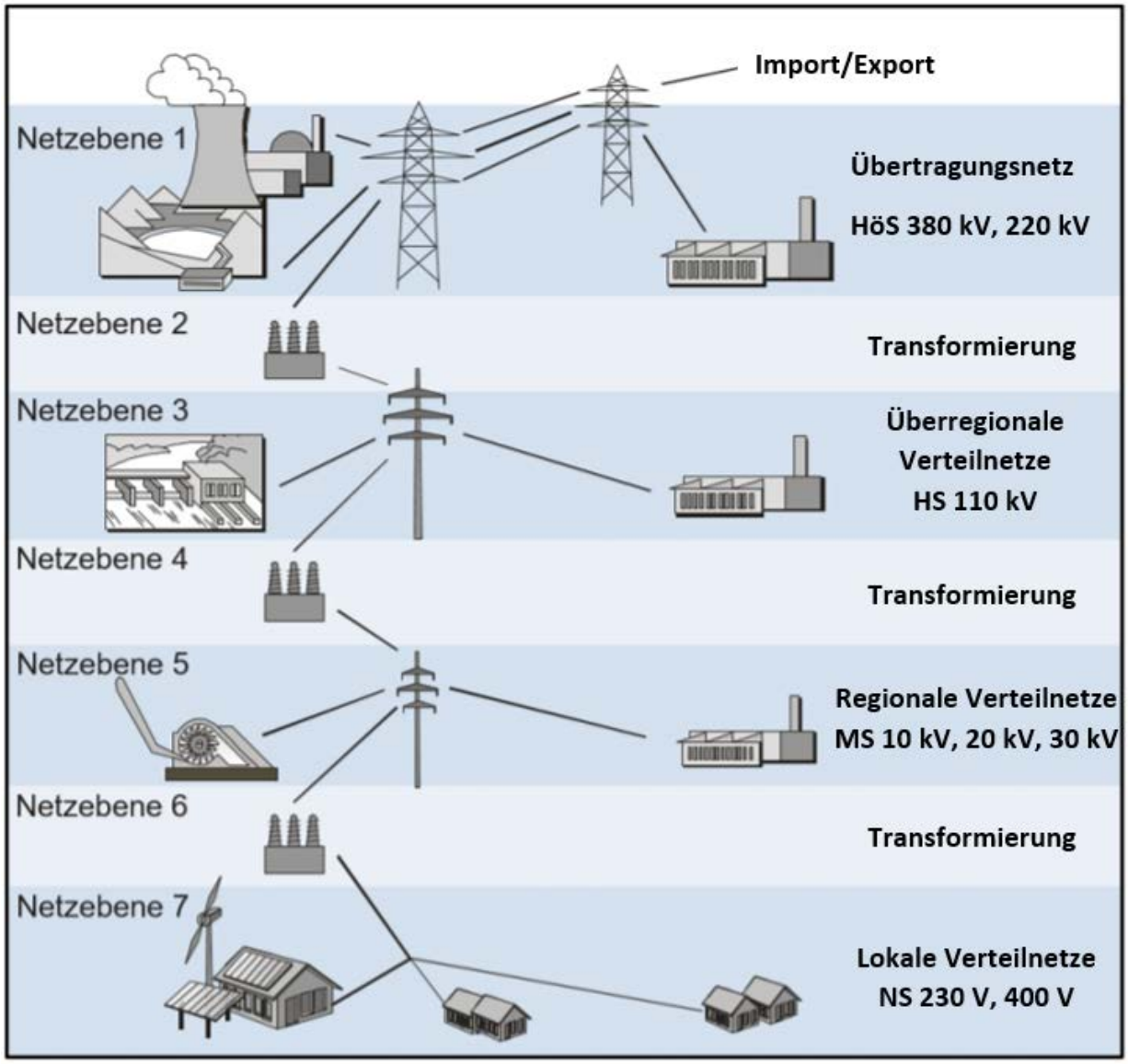


Abbildung 4: Netzebenen auf Basis [9]

2.3 Spannungsmerkmale des Niederspannungsnetzes gemäß ÖVE/ÖNORM EN 50160

Um Missverständnissen bei den in diesem Unterkapitel vorkommenden Begrifflichkeiten vorzubeugen, werden nachfolgend Definitionen gemäß ÖVE/ÖNORM EN 50160:2011-03-01 [10] erläutert.

Die ÖVE/ÖNORM EN 50160 beschreibt Merkmale der Spannung in öffentlichen Elektrizitätsversorgungsnetzen und wird in diesem Unterkapitel näher erläutert, hier insbesondere die Charakteristik bezüglich des Niederspannungsnetzes. Die Norm befasst sich mit den Grenzen beziehungsweise den Werten, innerhalb derer die Spannung an beliebigen Übergabestellen in öffentlichen Elektrizitätsversorgungsnetzen zu erwarten ist. Sie trifft jedoch keinerlei Aussagen über die durchschnittliche Situation wie sie der Endnutzer erwarten kann. [10]

Gemäß ÖVE/ÖNORM EN 50160:2011-03-01 [10] spricht man von Niederspannung bei Spannungen, deren Nenn-Effektivwert $U_{Nenn} \leq 1$ kV ist.

Nennspannung U_n

„Spannung, durch die ein Versorgungsnetz bezeichnet oder identifiziert wird und auf die bestimmte betriebliche Merkmale bezogen werden“ [10]

Übergabestelle

„Stelle in einem öffentlichen Versorgungsnetz, die dafür vorgesehen und vertraglich festgelegt ist, dass an ihr elektrische Energie zwischen den Vertragspartnern ausgetauscht wird“ [10]

Versorgungsspannung

„Effektivwert der Spannung an der Übergabestelle zu einem bestimmten Zeitpunkt, gemessen über ein bestimmtes Intervall“ [10]

Spannungseinbruch

„zeitweiliges Absinken des Effektivwertes der Spannung unter eine festgelegte Anfangseinbruchschwelle an einem bestimmten Punkt des Elektrizitätsversorgungsnetzes“ [10]

Dauer eines Spannungseinbruchs

„Zeit zwischen dem Augenblick, in dem der Effektivwert der Spannung unter die Anfangseinbruchschwelle an einem bestimmten Punkt des Elektrizitätsversorgungsnetzes fällt, und dem Augenblick, in dem er auf die Endeinbruchschwelle ansteigt“ [10]

Spannungsüberhöhung (zeitweilige netzfrequente Überspannung)

„zeitweiliges Ansteigen des Effektivwertes der Spannung über eine festgelegte Anfangsschwelle an einem bestimmten Punkt des Elektrizitätsversorgungsnetzes“ [10]

In der ÖVE/ÖNORM EN 50160 wird zwischen zweierlei Phänomenen unterschieden: Einerseits wird zwischen andauernden Phänomenen, diese sind Abweichungen vom Nennwert, die dauerhaft über eine Zeitspanne auftreten sowie andererseits plötzliche und daher erhebliche Abweichungen von Nennform bzw. der gewünschten Kurvenform, auch Spannungsereignisse genannt, unterschieden. Erstere treten vor allem aufgrund von Lastmustern, Lastschwankungen oder nicht linearen Lasten auf. Zweitere treten meistens durch unvorhersehbare Ereignisse (beispielsweise Fehler) aber auch durch Fremdeinwirkung von außen sowie Wetterereignisse auf. [10]

Für das Niederspannungsnetz gilt eine genormte Nennspannung U_n von 230 V zwischen Außenleitern und Neutralleiter im Vierleiternetz. [10]

Die Nennfrequenz der Versorgungsspannung muss gemäß Norm [4] 50 Hz betragen. Für den normalen Betrieb muss der sogenannte 10-Sekunden-Mittelwert der Grundfrequenz in definierten Bereichen liegen. Diese sind bei Netzen mit synchroner Verbindung zu einem Verbundnetz 50 Hz ± 1 % während 99,5 % des Jahres, beziehungsweise 50 Hz + 4 % / - 6 % während 100 % der Zeit. [10]

Für Netze ohne synchrone Verbindungen, wie es beispielsweise bei Versorgungsnetzen auf Inseln der Fall ist, gelten gemäß Norm andere Bereiche, diese werden im Zuge dieser Arbeit jedoch nicht weiter behandelt.

Weiters gilt, dass unter normalen Betriebsbedingungen, mit Ausnahme von Unterbrechungen, die Versorgungsspannung ± 10 % der Nennspannung nicht überschreiten soll. Um dies zu klassifizieren, werden Prüfungen durchgeführt, bei denen 95 % innerhalb der 10-Minuten-Mittelwerte des Effektivwertes der Versorgungsspannung jedes Wochenintervalls innerhalb ± 10 % der Nennspannung liegen müssen. Zudem müssen alle 10-Minuten-Mittelwerte allenfalls innerhalb eines Bereichs von + 10 % und - 15 % der Nennspannung liegen. [10]

Ein weiterer wesentlicher Punkt sind Einbrüche sowie Überhöhungen der Versorgungsspannung. Typischerweise entstehen Spannungseinbrüche durch Fehler im Versorgungsnetz oder durch Fehler bei Kundenanlagen. Dahingegen entstehen Spannungsüberhöhungen typischerweise durch Schalthandlungen sowie Lasttrennungen. Beide soeben beschriebenen Spannungsabweichungen sind schwer vorherzusagen und daher weitgehend zufälliger Natur. Die jährlich auftretende Anzahl schwankt zudem stark und lässt sich dadurch nur sehr schwer über die Jahre hinweg vergleichen. [10]

2.4 Netzstrukturen

Grundsätzlich wird zwischen Strahlnetzen, Ringnetzen und Maschennetzen unterschieden. Je nach Spannungsebene, den auftretenden Lastdichten, geografischen Konstellationen sowie der zu erfüllenden Versorgungssicherheit können diese angeführten Netzstrukturen eine unterschiedliche Form annehmen. Die drei anfangs angeführten Netzstrukturen bilden die Grundformen, jedoch sind verschiedenste Kombinationen sowie Modifikationen dieser möglich. Weiters sei hier noch zu erwähnen, dass es in jeder Netzstruktur vorgesehene Trennstellen gibt, um gegebenenfalls nötige Wartungsarbeiten durchführen zu können. [11]

2.4.1 Radiale Struktur

Eine radiale Netzstruktur gemäß Abbildung 5 zeichnet sich vor allem durch den simpel gehaltenen Aufbau aus und stellt somit jene dar, welche im Falle eines Fehlers die einfachste Klärung ermöglicht. Ausgehend von einer Transformatorstation, werden Leitungen zu Unterstationen geführt von welchen Stichleitungen (en. feeder) ausgehen, an denen dann die einzelnen Verbraucher angeschlossen sind. Bezeichnend für diese Netzform ist, dass nur von einer Seite aus eingespeist wird. Als Vorteil dieser Netzstruktur ist vor allem der geringe Schutzaufwand, der problemlose Betrieb und die sehr einfache Fehlerlokalisierung anzusehen. Als nachteilig haben sich jedoch die verringerte Betriebssicherheit akzentuiert, denn bei zB einem auftretenden Kurzschluss an einer Stichleitung kommt es zu einer Versorgungsunterbrechung, welche meist im Stundenbereich liegt. Als weiteres Problem hat sich vor allem bei langen Leitungen die Spannungshaltung bei längeren Netzausläufern herausgestellt. Typischer Anwendungsbereich für diese einfachste Form der Netzstruktur sind die Niederspannungsnetze in ländlichen und damit meist dünn besiedelten Gebieten, an denen mit kleineren Lastdichten zu rechnen ist. Auch in Städten mit höherer Lastdichte kommt diese Netzstruktur zum Einsatz, jedoch ist hier die Stichlänge verhältnismäßig kurzgehalten, sodass lediglich ein paar wenige Haushalte an dieser Stichleitung angeschlossen sind. In Zukunft wird es aber auch in ländlich dünn besiedelten Gebieten durch die vermehrte Nutzung von Elektrofahrzeugen und der damit verbundenen Ladeinfrastruktur, hier vor allem im privaten Sektor, zu Problemen der unterbrechungsfreien Energieversorgung mit der Stichstruktur kommen. [11, 12, 13]

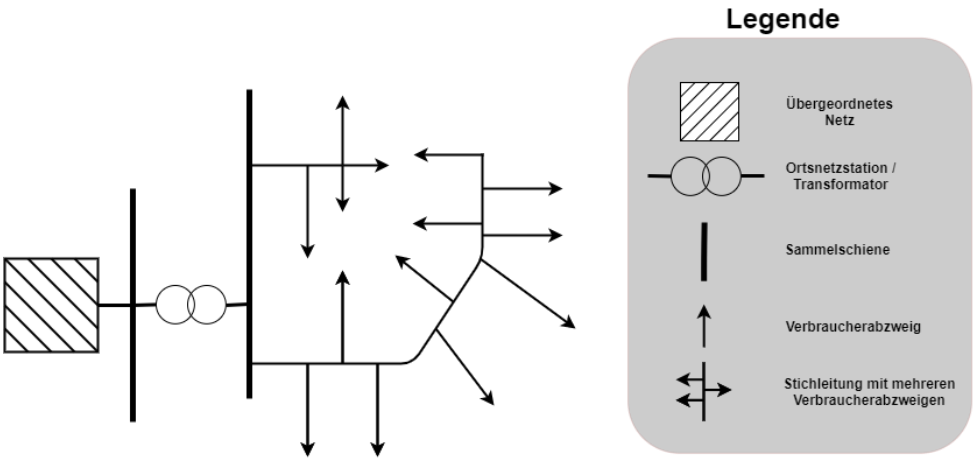


Abbildung 5: **Beispielhafte radiale Netzstruktur**

2.4.2 Ringstruktur

Bei zunehmender Lastdichte in Niederspannungsnetzen wird eine Teilvermaschung durch Bildung von Ringen angestrebt. Besonders in den Mittelspannungsnetzen kommt diese Art der Netzstruktur vermehrt zum Einsatz. Bezeichnend für diese Netzstruktur ist, dass die Verbraucher des Rings von beiden Seiten aus gespeist werden können. Im Normalbetrieb sowie aus Gründen der betrieblichen Einfachheit wird der Ring im Normalfall offen betrieben, d. h. mit offenem Lastschalter oder offener Trennstelle. Bei auftretenden Fehlern, zB Kurzschlüssen an einem Halbring, wird dieser automatisch vom vorgelagerten Netzschutz abgeschaltet. Die angeschlossenen Verbraucher sind dadurch nicht mehr versorgt. Durch gezieltes „Heraustrennen“ des betroffenen Leitungsstücks sowie durch gezieltes Zuschalten der dafür vorgesehenen Querverbindungen können die Verbraucher jedoch wieder vom Halbring, welcher nicht vom Fehler betroffen war, weiterversorgt werden. Demnach ist hierbei eine höhere Betriebssicherheit als bei der einfacheren radialen Struktur gemäß Kapitel 2.4.1 zu verzeichnen. Als weiterer Vorteil ist die bessere Spannungshaltung gegenüber dem Strahlennetz zu sehen. Da jedoch bei dieser Netzstruktur von beiden Seiten aus Kurzschlussströme zur Fehlerstelle zufließen können ist hier ein verbesserter und erhöhter Leitungsschutz erforderlich als dies bei der radialen Struktur notwendig war. [11, 12, 13]

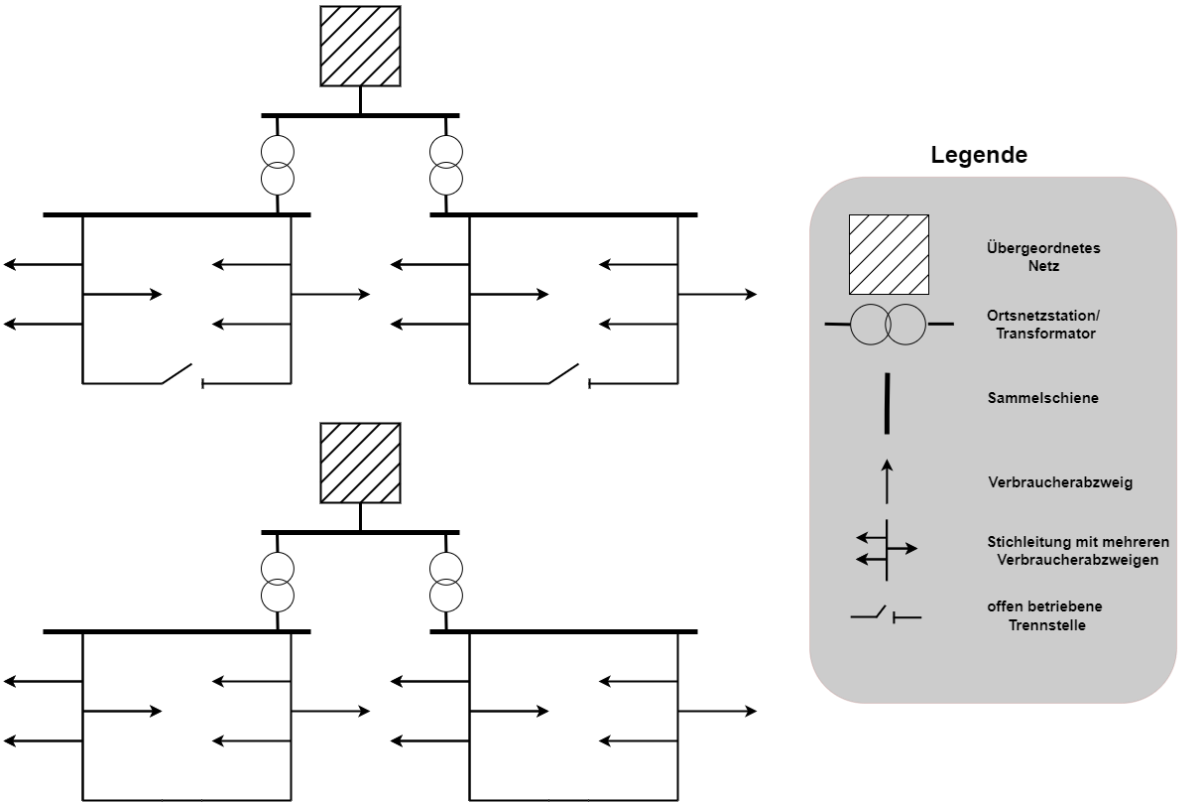


Abbildung 6: Beispielhafte Ringstruktur; oben mit offenem Ring, unten mit geschlossenem Ring betrieben

2.4.3 Vermaschte Struktur

Von einer vermaschten Netzstruktur spricht man, wenn alle zu versorgenden Stationen von mindestens zwei Seiten gespeist werden. Diese Netzstruktur bietet eine erhöhte Versorgungszuverlässigkeit, da in den meisten Fällen jeder Zweig bzw. Abgang mit eigener Sicherung geschützt wird, sodass es im Fehlerfall nur zu, ein kleines Gebiet betreffenden Abschaltungen kommt. Bei Netzen mit großer Ausdehnung wird mit mehreren Transformatoren an jenen Punkten eingespeist, welche die größte Lastdichte aufweisen. Typische Anwendung findet diese Art der Netzstruktur in Hochspannungsnetzen vor allem aus Sicht technisch-wirtschaftlicher Gesichtspunkte. In Nieder-bzw. Mittelspannungsnetzen ist diese Netzstruktur aufgrund der erhöhten Komplexität (vor allem in Bezug auf den konventionellen Schutz wie bspw. Schmelzsicherungen) und des Kostenfaktors eher selten vorzufinden. [11, 12, 13]

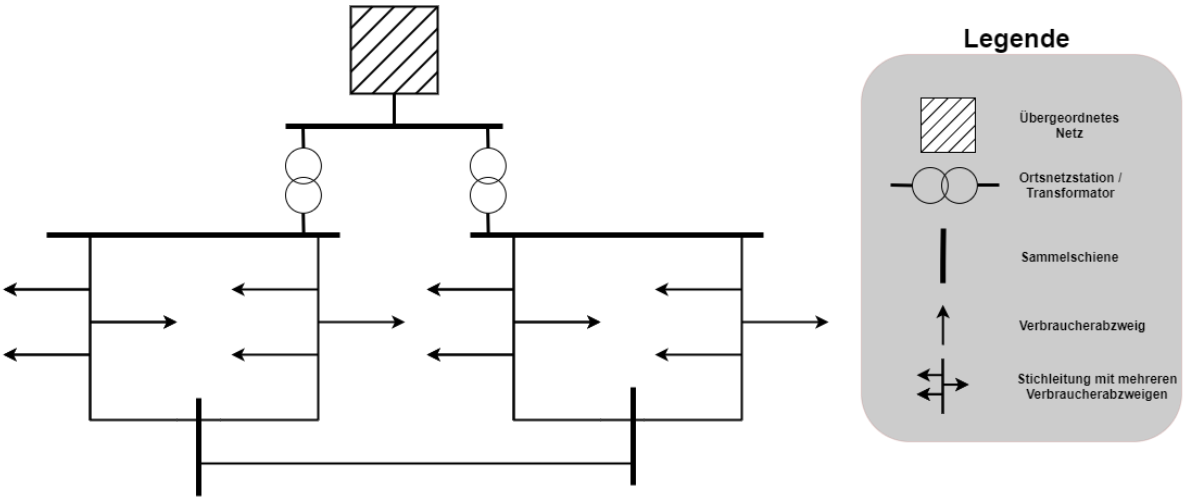


Abbildung 7: Beispielhafte vermaschte Struktur

2.5 Selektivität

Selektivität

„Koordination zwischen den Ansprechkennlinien von zwei oder mehreren Schutzeinrichtungen in der Weise, dass beim Auftreten von Überströmen oder Fehlerströmen zwischen vorgegebenen Grenzwerten die zum Ausschalten innerhalb dieses Bereichs vorgesehene Einrichtung ausschaltet, während die anderen Einrichtungen nicht ansprechen“ [4]

Vereinfacht ausgedrückt beschreibt Selektivität die Eigenschaft eines Schutzsystems, nach Möglichkeit, nur die vom Fehler betroffenen Betriebsmittel abzuschalten. [13]

Diese Vorgangsweise soll im Fehlerfall einen Totalausfall des Systems weitestgehend verhindern. Die Hauptaufgabe des Schutzgerätes ist es, einen Fehler zu erkennen und nur jenen fehlerbehafteten Abschnitt abzuschalten, welcher entsprechend betroffen ist. Selektivität gewährleistet bei hintereinanderliegenden Schutzeinrichtungen eine gestaffelte Abschaltung der Schutzeinrichtungen. Dies ermöglicht es, Fehlerauswirkungen auf das System weitestgehend auf ein Minimum zu reduzieren und schützt so vor Beschädigungen von Betriebsmitteln durch etwaige auftretende Fehler. [4, 13]

Nachfolgend werden Begrifflichkeiten Bezug nehmend auf die Norm OVE E 8101:2019-01-01 [4] erläutert.

Nachfolgende Begrifflichkeiten werden aus Gründen der Vollständigkeit gemäß der normativen Definition zitiert.

Vollselektivität

„Selektivität, bei der nur die Überstrom-Schutzeinrichtung (OCPD[7]) auf der Lastseite bis zum maximalen unbeeinflussten Kurzschlussstrom an deren Anschlusspunkt anspricht“ [4]

Teilselektivität

„Selektivität, bei der die Überstrom-Schutzeinrichtung (OCPD) auf der Lastseite nur bis zu einem Fehlerstrom (dem Selektivitäts-Grenzwertstrom) anspricht, der geringer ist als der maximale unbeeinflusste Kurzschlussstrom an deren Anschlusspunkt“ [4]

[7] **OCPD** **o**ver**c**urrent **p**rotection **d**evice, de. Überstromschutzeinrichtung

2.6 Systeme nach Art der Erdverbindung gemäß OVE E 8101:2019-01-01

Gemäß OVE E 8101:2019-01-01 [4] werden Netzsysteme nach der Art ihrer Erdverbindung in drei verschiedene Übergruppen klassifiziert: TT-, IT- und TN-System. Es lässt sich erkennen, dass die Systeme mittels (mindestens) zwei Buchstaben gekennzeichnet werden, welche sich aus dem Französischen ableiten. Der erste Buchstabe gibt Auskunft über die Beziehung des Stromversorgungssystems zur Erde. Man unterscheidet hier:

T ≡ (terre) Erde; „*direkte Verbindung eines Punkts zur Erde*“ [4]

I ≡ (isolé) isoliert; „entweder alle aktiven Teile von Erde getrennt oder ein Punkt über eine hohe Impedanz mit Erde verbunden“ [4]

Der jeweils zweite Buchstabe gibt Auskunft über die Beziehung der Körper der elektrischen Anlage zur Erde. Man unterscheidet hier:

T ≡ (terre) Erde; „direkte elektrische Verbindung der Körper zur Erde, unabhängig von der etwa bestehenden Erdung eines Punkts des Versorgungssystems“ [4]

N ≡ (neutre) neutral; „direkte elektrische Verbindung der Körper mit dem geerdeten Punkt des Stromversorgungssystems (in Wechselstromsystemen ist der geerdete Punkt des Stromversorgungssystems im Allgemeinen der Sternpunkt oder, falls ein Sternpunkt nicht vorhanden ist, ein Außenleiter)“ [4]

Weitere Buchstaben (falls vorhanden) geben Auskunft über die Anordnung des Neutralleiters und des Schutzleiters. Man unterscheidet hier:

S ≡ (séparé) getrennt; „Schutzfunktion, die durch einen vom Neutralleiter oder vom geerdeten Außenleiter getrennten Leiter (Schutzleiter) vorgesehen wird“ [4]

C ≡ (combiné) kombiniert; „Neutralleiter- und Schutzleiterfunktion kombiniert in einem einzigen Leiter (PEN-Leiter)“ [4]

Die nachfolgenden Kapitel 2.6.1 bis inklusive 2.6.3 stellen die, sich daraus ergebenden Systeme nach Art ihrer Erdverbindung gemäß OVE E 8101:2019-01-01 [4] dar.

2.6.1 TN-System

Im TN-System besteht eine direkte (niederohmige) Verbindung zwischen eines Punktes des Stromversorgungsystems und Erde. Vom Körper der elektrischen Anlage besteht eine durchgehende leitende Verbindung über einen Schutzleiter mit dem direkten Erdverbindungspunkt des Stromversorgungsystems. [14]

Je nach Anordnung des Schutzleiters sowie des Neutralleiters kann eine weitere Unterteilung des TN-Systems in drei Arten vorgenommen werden, die in den nachfolgenden Unterkapiteln erläutert werden.

2.6.1.1 TN-S-System

Von nahezu allen österreichischen Energieversorgungsunternehmen wird das TN-S-System in den Anschlussbedingungen vorgesehen. Dabei wird der grün-gelb gekennzeichnete Schutzleiter, auch PE-Leiter (en. protective earth) genannt, als getrennter Leiter im gesamten System mitgeführt. Dieser Schutzleiter ist bei allen Geräten auf die dafür vorgesehene Erdungsklemme zu verbinden. Der blaue Neutralleiter, auch N-Leiter genannt, wird ebenfalls getrennt verlegt und dient zum Anschluss von einphasigen Betriebsmitteln mit einer Leiter-Erde-Spannung von 230 V. Durch diese Trennung von Schutzleiter und Neutralleiter ist gewährleistet, dass alle metallischen Körper der jeweiligen Geräte des Netzes zusammengeführt werden und somit allesamt auf demselben Potential liegen. In einem fehlerfreien Betriebszustand führt der Schutzleiter keinen Strom. Der Neutralleiter hingegen schon. Kommt es beispielsweise zu einem Kurzschluss zwischen einer der drei Phasen und dem Körper eines Gerätes, so stellt sich ein Stromfluss über den Schutzleiter ein und bewirkt in weiterer Folge das Ansprechen des Schutzorgans. [4, 12, 14]

Die Zuleitung, kommend vom Transformator, ist ein Vierleiterkabel (TN-C-System, siehe Kapitel 2.6.1.2) und wird erst im Haus am jeweiligen Hausanschlusskasten in das vorliegende 5-Leiter-Netz, bestehend aus den drei Phasen L1, L2, L3, sowie Schutzleiter und Neutralleiter, aufgespaltet. Aufgrund des hohen Aufwands dieses Systems wird es nur innerhalb von Gebäuden realisiert. Dieses System wird zudem nicht für öffentliche Verteilernetze in Österreich verwendet. [4, 13, 14] Abbildung 8 verdeutlicht die Ausführung eines solchen TN-S Systems.

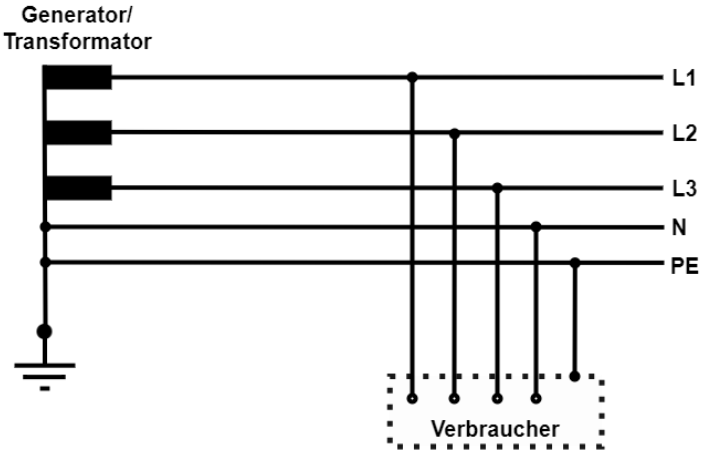


Abbildung 8: TN-S-System

2.6.1.2 TN-C-System

Das TN-C-System ist heutzutage der übliche Standard für Niederspannungsverteilnetze. Bei diesem System wird die Funktion des Neutralleiters (N) und die des Schutzleiters (PE) in einem einzigen Leiter, dem PEN-Leiter kombiniert. Resultierend aus dieser Doppelfunktionalität des PEN-Leiters liegt hierbei schon im Normalbetrieb an den Körpern der geerdeten Geräte eine kleine Spannung an. Grund dafür ist der Strom, der durch den PEN-Leiter fließt und somit aufgrund des Ohm‘schen Gesetzes einen Spannungsabfall zur Folge hat. [4, 12, 14]

Wenn der PEN-Leiter jedoch unterbrochen wird, liegt an leitfähigen Gehäuseteilen, die sich vor der Unterbrechungsstelle befinden, die volle Außenleiterspannung gegen Erde (230 V) an, was eine enorme Gefahrenquelle darstellt und bei Neubauten nicht mehr eingesetzt wird. Dieses System war lange Zeit eine übliche Hausinstallationsvariante, welche aufgrund des verringerten Verkabelungsaufwands entsprechende Kostenvorteile bot. Seit 1975 [15] ist dieses System nur noch mit Querschnitten von mindestens 10 mm² Aluminium zulässig, da sich dadurch eine PEN-Leiter-Unterbrechung auf ein vertretbares Restrisiko senken lässt. Für Altanlagen mit geringem Querschnitt besteht heutzutage Bestandsschutz[8]. In Österreich gilt dieser Bestandsschutz allerdings beispielsweise bei Neuvermietung nicht, weshalb im Falle dessen entsprechend auf das aktuelle System nachgerüstet werden muss. [4, 12, 13, 14]

Abbildung 9 veranschaulicht die Ausführung eines solchen TN-C-Systems.

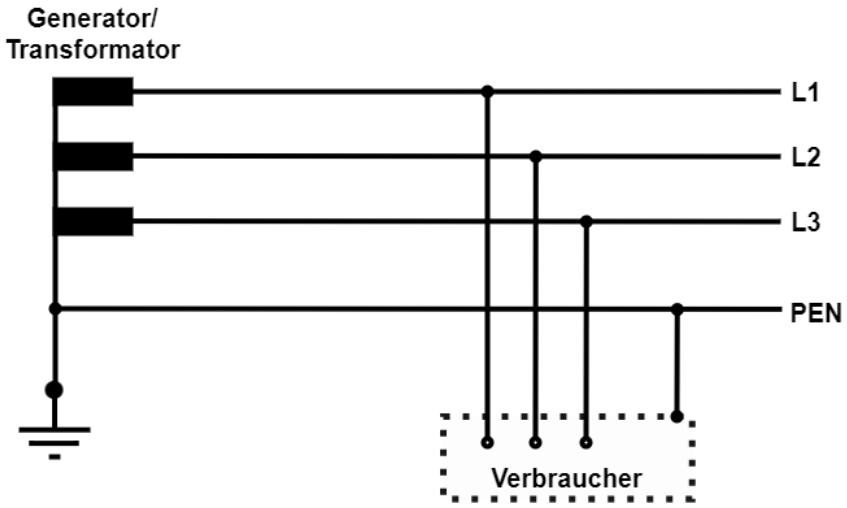


Abbildung 9: TN-C-System

[8] Erlaubt das Weiterbetreiben von Anlagen (auch wenn diese nach früheren Normen errichtet wurden), sofern jedoch keine wesentlichen Änderungen vorgenommen wurden. Wenn die Anlage erweitert wird, müssen die aktuell gültigen Normen eingehalten werden. [32]

2.6.1.3 TN-C-S-System

Dieses System kombiniert ein TN-C-System für das Verteilnetz mit einem TN-S-System in der Endkundenanlage. Der Neutralleiter (N) sowie der Schutzleiter (PE) werden dabei in einem Teil des Systems – dem TN-C-System – kombiniert. Der von dem Transformatorsternpunkt ankommende PEN-Leiter, wird auf Kundenseite in Schutzleiter sowie Neutralleiter aufgegliedert. Dieser Aufteilungspunkt ist kennzeichnend für den Übergang zwischen den beiden Systemen, also TN-C- zu TN-S-System. Ab diesem Übergang zum TN-S-System werden Neutralleiter (N) sowie Schutzleiter (PE) ausdrücklich getrennt. Dieses System bildet die Grundlage der Gebäudeversorgung sowohl in Österreich als auch in Deutschland und der Schweiz. Es bildet somit den Standard für elektrisch versorgte Objekte und Anlagen in Österreich. [4, 13, 14]

Abbildung 10 zeigt die Ausführung eines solchen TN-C-S Systems. Die Auftrennung zwischen N- und PE-Leiter am Übergabepunkt zwischen Verteilnetz und Kundenanlage wird als sogenannte Nullungsverbindung bzw. als Nullungsbügel bezeichnet.

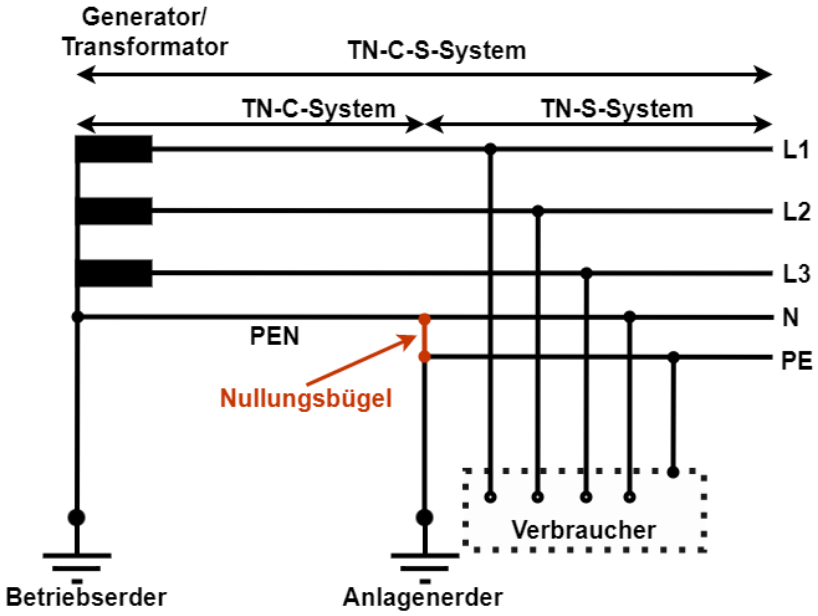


Abbildung 10: TN-C-S-System

2.6.2 TT-System

Im TT-System besteht eine direkte, niederohmige Verbindung eines Punktes des Stromversorgungsystems zu Erde. Die elektrische Anlagen respektive deren Körper sind an den Schutzleiter (PE) angeschlossen, deren Anlagenerdung ist allerdings nicht direkt mit dem Betriebserder des Versorgungssystems verbunden. Diese fehlende Verbindung zwischen dem Betriebserder des Stromversorgungssystems und des Anlagenerders auf der Verbraucherseite bietet den Vorteil, dass nur geringe Ausgleichsströme zwischen beiden Erdern fließen können. Dieses System war ehemals der Standard für öffentliche Verteilernetze in Österreich, ist jedoch auslaufend und wird für neue Verbraucheranlagen gemäß [4] nicht mehr empfohlen. [4, 13, 14]

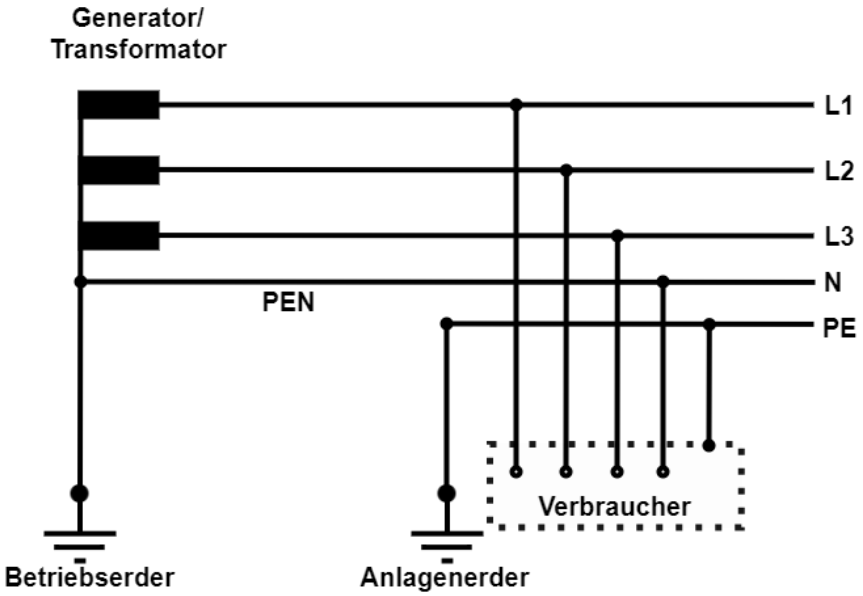


Abbildung 11: TT-System

2.6.3 IT-System

Bei diesem System besteht keine galvanische Verbindung zwischen aktiven Teilen und Erde bzw. eine Verbindung, die eine ausreichend hohe Impedanz aufweist. Diese Netzform wird vor allem bei Netzen mit geringer Ausdehnung angewandt, um somit gegebenenfalls eine Fehlerstelle schneller identifizieren zu können. Die Versorgung kann sowohl über Transformatoren, Generatoren aber auch Batterien erfolgen. Es besitzt gegenüber dem vorher beschriebenen TN- bzw. TT-System eine deutlich bessere Ausfallsicherheit. Aufgrund der Tatsache, dass beim IT-System der Sternpunkt hochohmig geerdet und der Schutzleiter nicht mit dem Sternpunkt verbunden ist, kann es im Fehlerfall zu keinerlei Stromfluss über selbigen kommen. Diese Tatsache begründet auch den oft gebräuchlichen Ausdruck eines isoliert betriebenen Netzes. Im Gegensatz zum TN- bzw. TT-System muss allerdings bereits der erste Fehler gemeldet werden (zB mit Hilfe eines Isolationswächters (en. insulation monitoring device, IMD)), ein potentieller zweiter Fehler kann bekanntlich zu einem ungewollten Stromfluss führen. Die Vorteile des IT-Systems hinsichtlich Verfügbarkeit sind vor allem bei kritischer Infrastruktur wie beispielsweise Operationssälen von Krankenhäusern, in denen Versorgungssicherheit eine wesentliche Rolle spielt, von Relevanz. [4, 13, 16]

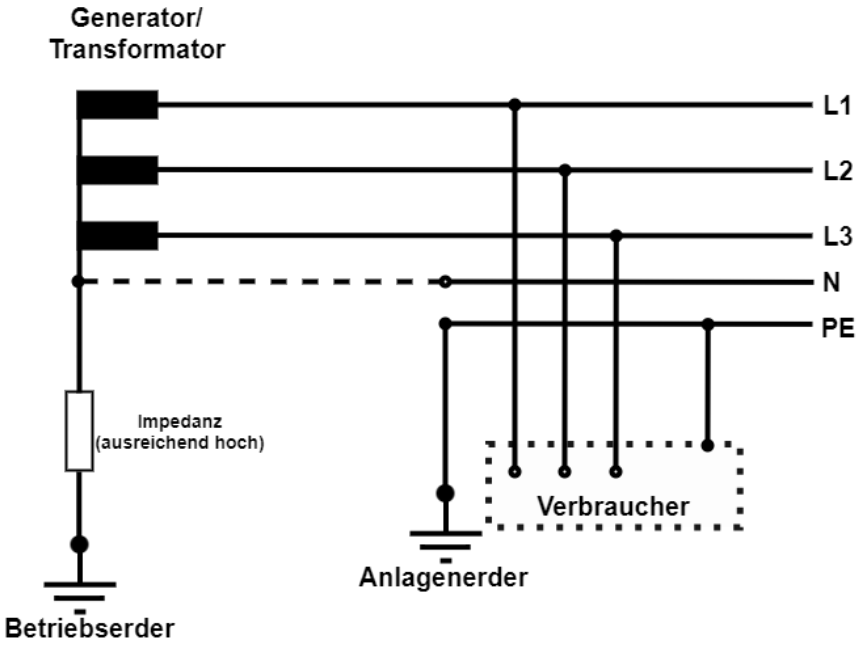


Abbildung 12: IT-System

2.7 Fehlerfälle und deren Auswirkung im Niederspannungsnetz

Grundsätzlich versteht man unter einem Fehler eine ungewollte Änderung des normalen Betriebszustandes. Ein normaler, fehlerfreier Betriebszustand ist durch

- eine ausreichende Spannung,
- eine intakte Isolierung bzw. einen intakten Isolationszustand,
- einen Betriebszustand, der von der Betriebsführung gewollt ist sowie
- intakte Betriebsmittel

gekennzeichnet. [17]

Der mit Abstand am häufigsten auftretende Fehler in den Verteilnetzen ist der einpolige Erdschluss, vgl. hierzu Kapitel 2.7.1. Aus solchen Fehlern entwickeln sich meist mehrpolige Fehlerfälle, genaueres hierzu in den nachfolgenden Unterkapiteln.

2.7.1 Einpoliger Erdkurzschluss

Der einpolige Erdschluss ist mit Abstand der am häufigsten auftretende Fehler. Erklären lässt sich dies dadurch, dass rund 70 % bis hin zu über 90 % der Netzfehler ihren Ursprung als einpoliger Fehler nehmen. Wie bereits aus dem Namen des Fehlers ersichtlich, handelt es sich bei einem einpoligen Erdschluss um eine leitende Verbindung mit einer der drei Phasen gegen Erde. Bei Kabeln beginnt diese Fehlerart meistens mit einem Durchschlag der Isolation eines einzelnen Leiters gegen Erde. Ausgehend von diesem Fehler entwickeln sich meist durch fortschreitende mechanische Zerstörung von Isolationskomponenten mehrpolige Fehlerarten. Hingegen im Falle von Freileitungen können verschiedenste Naturereignisse wie beispielsweise Baumfall, Eis/Schnee, Wind, etc. zu einpoligen Erdschlüssen führen.

Bei Fehlern mit Erdberührung kann es zudem zu gefährlichen Schritt- sowie Berührspannungen kommen. [17, 18]

Abbildung 13 zeigt einen einpoligen Erdschluss der Phase L3 gegen Erde.

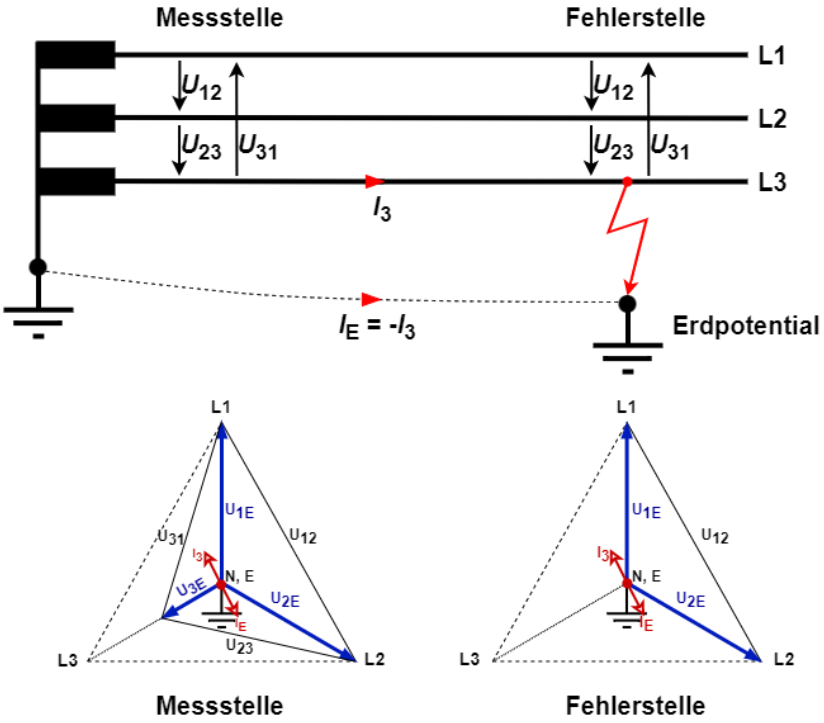


Abbildung 13: Einpoliger Erdschluss Phase L3 gegen Erdpotential – Schaltbild (oben) und Zeigerdiagramm (unten)

Kommt es zu einem einpoligen Erdschluss in einem Netz mit starrer Erdung, so bricht die betroffenen Leiter-Erde-Spannung an der Fehlerstelle auf null zusammen. In Abbildung 13 ist dies grafisch am Beispiel eines einpoligen Erdschlusses von Phase L3 gegen Erde verdeutlicht. Die Spannungen der beiden „gesunden“, d. h. nicht vom Fehler betroffenen Phasen, in diesem Fall Phase L1 und Phase L2, bleiben vollständig erhalten. Errichtet man beispielsweise in gewisser Entfernung zur Fehlerstelle eine Messstelle so wird man feststellen, dass sich die Leiter-Erde-Spannung, die an der Fehlerstelle auf null zusammengebrochen ist, je weiter man sich von der Fehlerstelle entfernt, wieder aufbauen wird. Die betroffene Leiter-Erde-Spannung wird jedoch nicht wieder das Anfangsniveau vor dem Fehler erreichen vgl. hierzu Abbildung 13 (Zeigerdiagramm). Der Erdstrom wird in Folge des Fehlers gleich groß wie der Strom der fehlerbehafteten Phase, steht jedoch in Gegenphase zu diesem. [17, 18]

2.7.2 Zweipoliger Kurzschluss ohne Erdberührung

Diese Fehlerart entsteht, wenn es zu einer leitenden Verbindung zwischen zwei Phasen des Systems kommt. An der Fehlerstelle kommt es zum Zusammenbruch der Dreieck-Spannung zwischen den vom Fehler betroffenen Phasen. Ebenso geht die Leiter-Erde-Spannung der beiden betroffenen Phasen auf die Hälfte des Wertes gegenüber dem fehlerfreien Zustand zurück. Weiters werden sich die beiden Leiter-Erde-Spannungen in Gegenphase zur Leiter-Erde-Spannung des gesunden Leiters drehen. Abbildung 14 veranschaulicht diesen Fehlerfall in Form des zugehörigen Schaltbildes sowie der entsprechenden Zeigerdiagramme. Hier wird ein Fehler zwischen den beiden Phasen L2 sowie L3 dargestellt. Es werden sich weiters zwei Ströme ausbilden, die in Gegenphase zueinanderstehen und gleich groß sind. Es handelt sich hierbei um die beiden Ströme der fehlerbehafteten Phasen. Aufgrund der fehlenden Erdberührung wird sich jedoch in diesem Fall kein Erdstrom ausbilden. [17, 18]

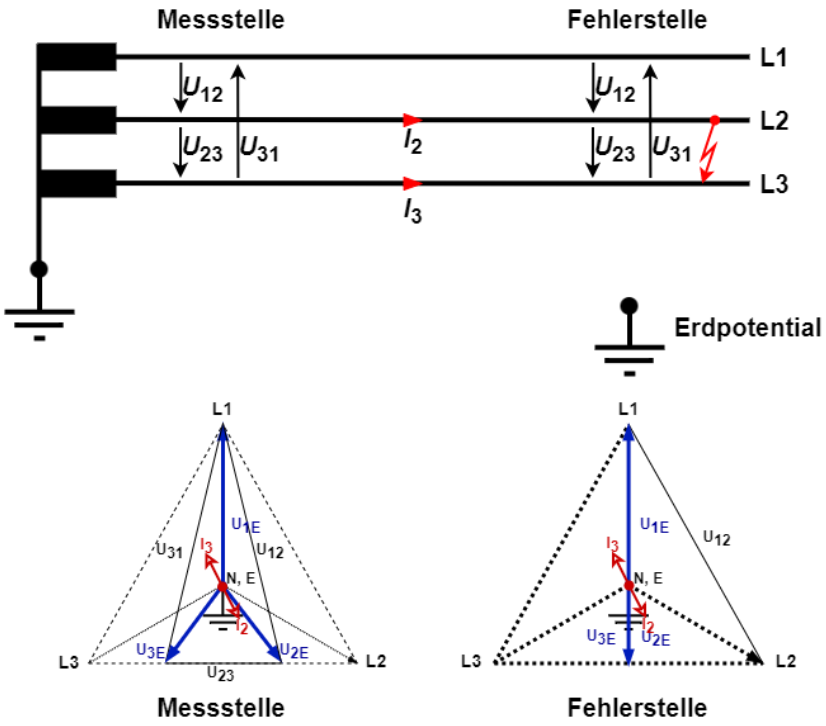


Abbildung 14: Zweipoliger Kurzschluss zwischen Phase L2 und L3, ohne Erdberührung – Schaltbild (oben) und Zeigerdiagramm (unten)

Diese Fehlerart ist in der Praxis sehr selten und entsteht in erster Linie, wenn sich beispielsweise zwei Leiterseile an einer Freileitung durch Windeinfluss berühren. Jedoch ist dieser Fehler aufgrund zahlreicher bautechnischer Vorkehrungen wie beispielsweise Phasenabstandshalter ausgesprochen selten anzutreffen, kann jedoch keinesfalls ausgeschlossen werden. [17, 18]

2.7.3 Zweipoliger Kurzschluss mit Erdberührung im Netz mit starrer Erdung

Diese Fehlerart ist im Vergleich zum Zweipologen Kurzschluss ohne Erdberührung deutlich häufiger anzutreffen, da es sich dabei sozusagen um die nächste Stufe nach dem einpoligen Kurzschluss mit Erdberührung in Richtung eines dreipoligen Fehlers handelt.

Beim zweipoligen Kurzschluss mit Erdberührung werden die beiden Leiter-Erde-Spannungen der betroffenen Phasen an der Fehlerstelle auf null zusammenbrechen. Auch hier bleibt die Leiter-Erde-Spannung der nicht vom Fehler betroffenen Phase vollständig erhalten. Aufgrund der Erdverbindung wird sich jedoch in diesem Fall auch zusätzlich ein Erdstrom zu den beiden Strömen der fehlerbehafteten Phasen ausbilden. Errichtet man wiederum in gewisser Entfernung von der Fehlerstelle eine Messstelle, so wird man feststellen, dass sich die beiden Spannungen der betroffenen Phasen wieder aufbauen werden [17, 18], jedoch wie in Abbildung 15 zu erkennen, nicht auf das Anfangsniveau vor dem Fehlereintritt.

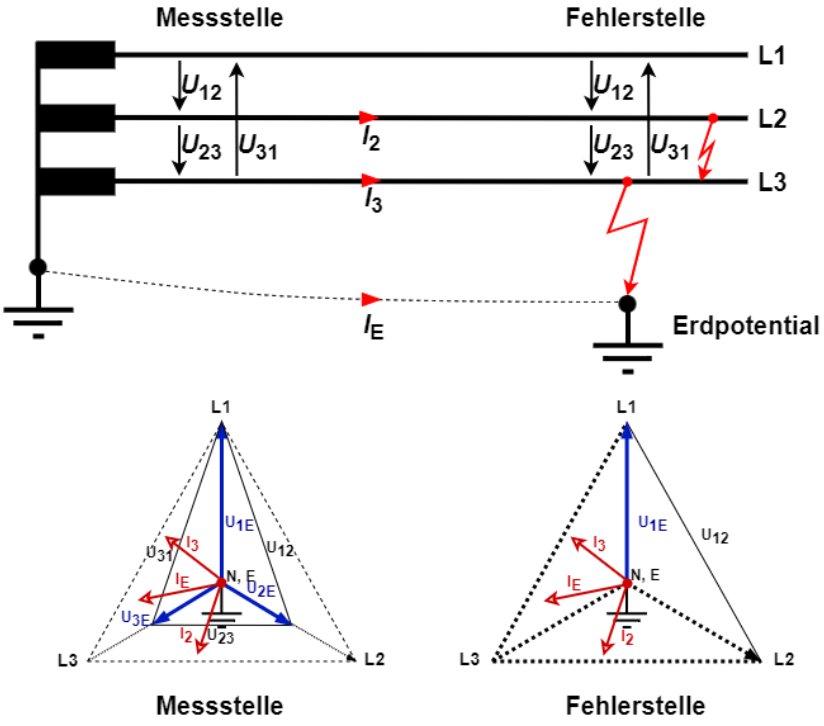


Abbildung 15: Zweipoliger Kurzschluss zwischen Phase L2 und L3, mit Erdberührung – Schaltbild (oben) und Zeigerdiagramm (unten)

2.7.4 Dreipoliger Kurzschluss

Wird ein zweipoliger Kurzschluss nicht rechtzeitig geklärt, so folgt in den meisten Fällen die Weiterentwicklung zu einem dreipoligen Kurzschluss. Das bedeutet, dass eine leitende Verbindung zwischen allen Phasen besteht. Der dreipolige Kurzschluss ist ein symmetrischer Fehler, das bedeutet, dass auch im Fehlerfall die Ströme sowie Spannungen symmetrisch bleiben. Die Höhe der Spannungen kann jedoch eine Abweichung vom fehlerfreien Betrieb erfahren. Bei einem dreipoligen Kurzschluss brechen alle Spannungen an der Fehlerstelle weitgehend vollständig ein, siehe dazu auch Abbildung 16. Als Ausnahme gilt hier lediglich jener Fall, bei welchem hohe Fehlerwiderstände zu Tragen kommen. Die drei Leiterströme werden sich dabei symmetrisch gegenüber den Sternspannungen um den Impedanzwinkel φ_k ausbilden. Der Impedanzwinkel φ_k der Kurzschlussschleife eilt den Strangspannungen nach. Aufgrund des Kurzschlusses beträgt die Höhe der Ströme ein Vielfaches des maximalen Betriebsstromes. Bedingt durch die zahlreichen Überwachungssysteme kommt es nur in Ausnahmefällen zu dieser Fehlerart, welche sogleich den schwersten Fehler und damit die höchste Belastung der Betriebsmittel darstellt. [17, 18]

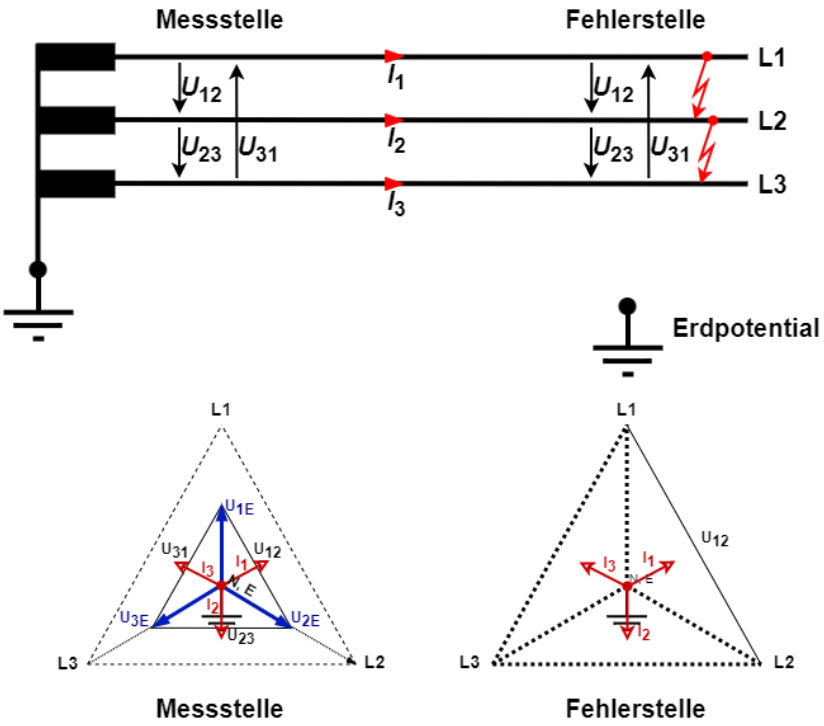


Abbildung 16: Dreipoliger Kurzschluss zwischen Phase L1, L2 sowie L3 – Schaltbild (oben) und Zeigerdiagramm (unten)

2.8 Temporäre Rekonfiguration des Niederspannungsnetzes

Die in Niederspannungsnetzen hauptsächlich vorkommenden radialen bzw. offenen Ring-Strukturen besitzen oftmals mehrere, an dafür vorgesehenen Punkten, Trennstellen. Diese Trennstellen ermöglichen es, den Lastfluss zu verlagern und ganze Abzweige vom übrigen Netz zu trennen. Dies geschieht zurzeit jedoch nur aufgrund von Wartungsarbeiten oder anderer unvorhersehbaren Netzzuständen. [11, 12, 13]

Es wird jedoch davon ausgegangen, dass die Zahl an dezentralen Leistungsquellen sowie Leistungssenken in den Niederspannungsnetzen zukünftig stark zunehmen wird. Als Leistungsquelle gelten hier insbesondere Batteriespeicher oder Photovoltaikanlagen. Als Leistungssenke werden in Zukunft vor allem Ladestationen für Elektrofahrzeuge (en. electric vehicle charging stations, EVCS), Wärmepumpen sowie Klimaanlagen das Netz vor große Herausforderungen stellen. Durch die von der Energiewende getriebenen Ziele der Regierung und den raschen Ausbau der Ladeinfrastruktur für Elektrofahrzeuge wird es in Zukunft zu enormen, lokal bedingten Leistungssenken im Niederspannungsnetz kommen. Ein Problem stellen hier insbesondere Privathaushalte mit eigener Ladestation dar, da diese zunehmend einen höheren Energiebedarf aufweisen. Zurzeit benötigt eine klassische EVCS mit mittlerer Ladeleistung im Höchstbetrieb rund 11 kWp. Der Höchstbetrieb wird bei den meisten Fahrzeugherstellern beim Laden der ersten 80 % des Batteriespeichers benötigt. Dem gegenüber steht der klassische Batteriespeicher mit einer durchschnittlich installierten Kapazität von 5 kWh. Die Ladeleistung von Elektrofahrzeugen nimmt nahezu jährlich um einige kW zu, wobei hingegen die installierte Leistung der Batteriespeicher eher stagniert. Unter anderem mit diesem Problem befasst sich das Konsortium des Projekts PoSyCo in dem Vertreter aus Forschungseinrichtungen, Industrie sowie Netzbetreiber an diesen zukünftigen Fragestellungen forschen. [18]

Betrachtet man die Niederspannungsversorgung mit Kabeln als Leitungstyp, so ist die mit Abstand am häufigsten auftretende Unterbrechungsart der dreipolige Kurzschluss mit PEN Leiter zu beklagen. Dieser tritt mit rund 20 Fehlern pro Jahr pro 100 km vorwiegend im Sommer auf und entsteht in den meisten Fällen fremdverursacht durch zB Grabungsarbeiten mit Baggern im Privat- oder Industriebereich, aber auch durch das Einbringen von Erdspießen in das Erdreich ohne vorherige Bodenkontrolle bzw. Planbeschau durch einen Fachmann. Es kommt in den meisten Fällen vorerst jedoch nur zu einem zweipoligen Kurzschluss zwischen zwei der drei Außenleiter. In weiterer Folge führt dies u. a. durch die hohe thermische Belastung, bedingt durch die beim Kurzschluss auftretenden hohen Kurzschlussströmen, zu einem Isolationsversagen und somit unweigerlich zu einem dreipoligen Kurzschluss. [18]

Aufgrund solcher Versorgungsunterbrechungen, aber insbesondere durch die anfangs erwähnte Zunahme der Energiesenken, kann es in Zukunft von größter Wichtigkeit sein, eine gezielte und geplante temporäre Netzrekonfiguration des Niederspannungsnetzes durchzuführen. Ziel des PoSyCo Forschungsprojektes ist es, die bereits bestehende HARDprotection des Netzes (bestehend aus herkömmlichen Leistungsschaltern, Leitungsschutzschaltern, NH-Sicherungen etc.) um eine Art zusätzliche Erweiterung, der sogenannten SOFTprotection, zu ergänzen. Unter den Begriff SOFTprotection können alle

Maßnahmen zur gezielten Lastflussanpassung des Niederspannungsnetzes angesehen werden, bei denen mittels intelligenter Algorithmen der bestehende Lastfluss betrachtet und – falls notwendig – durch gezielte Schalthandlungen zum Positiven verändert wird. [18]

Hauptaugenmerk dieser Arbeit ist es, einen Algorithmus in der Programmiersprache Python zu entwickeln, der auf Basis der Lastflussdaten anhand der idealen Schalterposition die Kompaktleistungsschalter des Laboraufbaus ansteuern kann, um somit die bestehende Schutztechnik um die sogenannte SOFTprotection zu erweitern. [18]

Dadurch ist es in Zukunft möglich, weitreichende Untersuchungen auf diesem Gebiet anzustellen und hier insbesondere inwieweit eine temporäre Netzrekonfiguration in bestimmten Situationen netzentlastend wirken kann.

2.9 Auswirkung der Elektromobilität auf das Niederspannungsnetz

Im Forschungsbericht „Konzeption von Vehicle to Grid bezogenen Entwicklungsstrategien für österreichische Entscheidungsträger“ [19] wurde eine wirtschaftliche sowie technische Analyse der Auswirkungen der Elektromobilitätsintegration in das bestehende Stromversorgungssystem untersucht. Aus diesem Bericht geht hervor, dass vor allem Ladestrategien im Niederspannungsnetz von größter Bedeutung sind, da es vermehrt bei Privatpersonen zu einer Neuinstallation einer Ladestation bzw. Wallbox für den Heimgebrauch kommen wird. Nachfolgend wird beispielsweise der Unterschied einer Einphasenbelastung und einer symmetrischen Belastung aller drei Phasen näher erläutert. Die weiterführenden Betrachtungen gelten jeweils für den Fall des „ungesteuerten Ladens“. Darunter versteht man den Beginn des Ladevorgangs unmittelbar nach dem Anstecken des Ladesteckverbinders an die Ladeeinrichtung bis hin zur vollständigen Aufladung des Fahrzeugs. Untersucht wurden verschiedene Netzabschnitte, vgl. hierzu. [19]

In Abbildung 17 wird das ungesteuerte Laden (16 A einphasig) für das Netzsegment 0222 [19] gezeigt. Auf der Abszisse sind die verschiedenen Netzknoten, jeweils L1, L2 und L3, des Netzes aufgetragen. Auf der Ordinate ist die Spannung in p.u. (per unit) aufgetragen. In dieser Abbildung wurden alle zu ladenden Elektrofahrzeuge an der gleichen Phase (in diesem Fall Phase L1) angeschlossen was in deutlichen Spannungsänderungen an den jeweiligen Phasen der Netzknoten resultiert. Aufgrund der entsprechenden Bewertung (Betrachtung des zeitlichen Mittels) gemäß ÖVE/ÖNORM EN 50160 [10] besteht jedoch keine Unterspannung obwohl einige wenige Netzknoten einen Wert unter 0,9 p.u. aufweisen. In Orange gehalten wird ein Marktdurchdringungsgrad von 40 % bis zum Jahr 2030 angenommen. Die roten Balken beziehen sich hingegen auf die Annahme eines Marktdurchdringungsgrads an Elektrofahrzeugen von 100 % im Jahr 2050. [19]

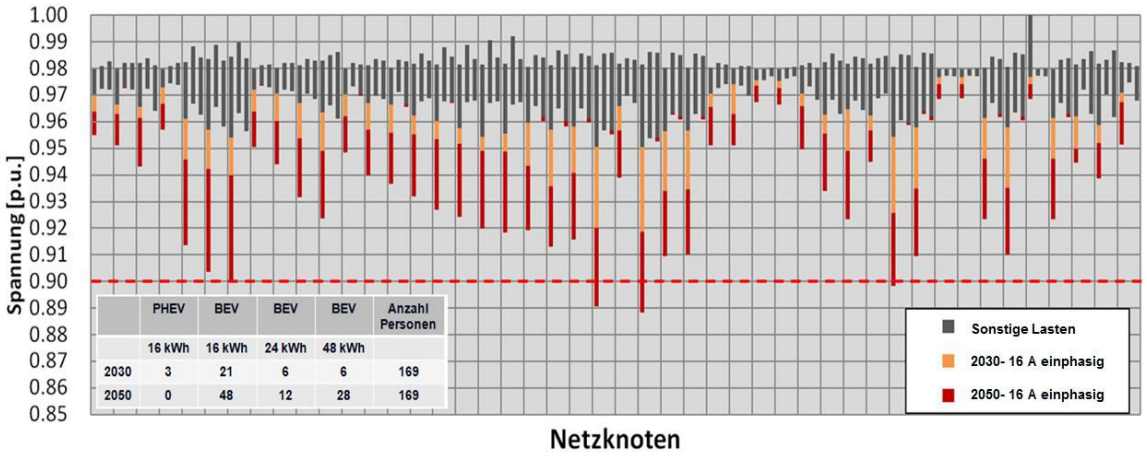


Abbildung 17: Einphasiges Laden [19]

In Abbildung 18 ist selbiges Netz wie in Abbildung 17 mit wiederum ungesteuertem Ladevorgang dargestellt, jedoch mit dem Unterschied, dass hier die zu ladenden Elektrofahrzeuge gleichmäßig über alle Phasen verteilt werden. Der Vergleich der beiden Abbildungen lässt erkennen, dass die Spannungsänderungen im Falle einer dreiphasig-symmetrischen Ladung signifikant geringer sind. In manchen Netzknoten lässt sich hierbei sogar eine Verringerung der Abweichung von der Nennspannung um bis zu 0,05 p.u. erreichen. Somit erreicht man eine deutliche Entlastung der Netzknoten im Vergleich zu jenem Fall, bei dem immer nur an Phase L1 geladen wird. [19]

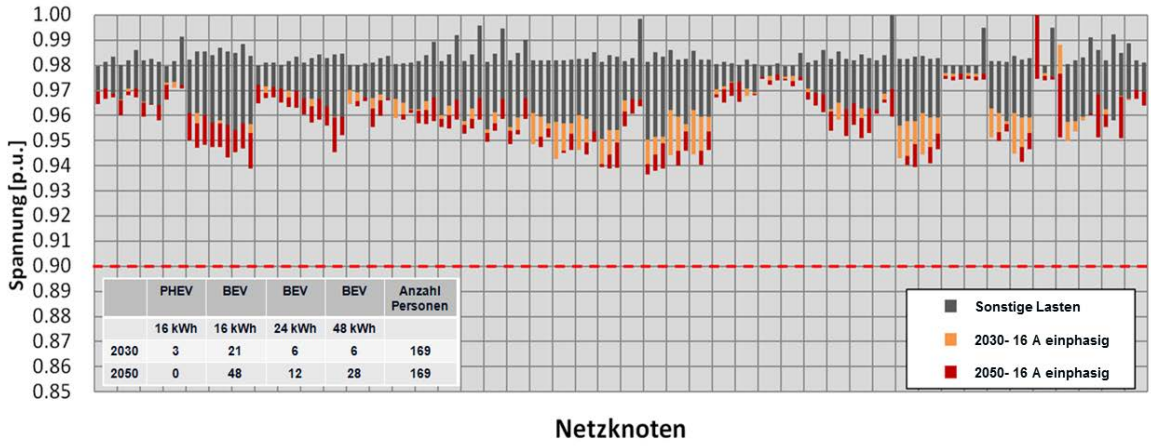


Abbildung 18: Gleichverteiltes Laden auf allen drei Phasen [19]

Anzumerken ist hierbei jedoch, dass diese Testresultate nur mit einer Ladeleistung von rund 3,7 kW durchgeführt wurden. Weiters wird im Bericht dazu geraten, die Ladevorgänge mit einer niedrigen Ladeleistung (beispielsweise 3,7 kW) durchzuführen. Betrachtet man aber die Reichweitensteigerung, die zukünftige Elektrofahrzeuge erfahren werden, so ist klar, dass die verbauten Kapazitäten der Traktionsbatterien der Fahrzeuge ebenso zunehmen werden und so auch eine Migration auf eine höhere Ladeleistung (beispielsweise 11 kW für Privathaushalte) unumgänglich wird. Unter anderem auch aus diesem Grund hat sich das Forschungsprojekt PoSyCo mit einer intelligenten Bewältigung dieses Problems auseinandergesetzt um hier eine zusätzliche Entlastung des Niederspannungsnetzes durch eine automatisierte sowie intelligente Rekonfiguration zu erreichen. [19]

2.10 Zusätzliche Herausforderungen für zukünftige Netzstrukturen

Durch die enorme Zunahme der Prosumenten, vor allem im privaten Sektor, werden die regionalen Verteilnetzbetreiber in Zukunft vor große Aufgaben gestellt. Vor allem die rasante Zunahme im Bereich der Elektromobilität sowie der damit verbundenen Ladeinfrastruktur stellt hier ein besonderes Wagnis dar. Als die heute bestehenden Netzstrukturen errichtet und konzeptioniert wurden, konnte noch nicht damit gerechnet werden, dass eines Tages der Verbrennungsmotor durch den Elektroantrieb abgelöst werden würde.

Auch aufgrund des enormen Kostendrucks – vor allem ausgelöst durch die Strommarktliberalisierung – besteht grundsätzlich zunehmend der Trend zur Entmaschung der Netze hin zu immer kleineren Maschen bzw. Systemen mit geringerer Flächenausdehnung. Das dadurch anwachsende Risiko, welches nicht bzw. weniger vermaschte Systeme im Vergleich zu vollständig vermaschten Systemen mit sich bringen, wird jedoch aufgrund der damit verbundenen Kosteneinsparungspotentiale in Kauf genommen. [1, 11]

3 Automatisierung des Kompaktleistungsschalter-Labordemonstrators

Im Zuge dieser Masterarbeit wird ein bestehender Labordemonstrator, welcher im Zuge von [20] am Institut für elektrische Anlagen und Netze der Technischen Universität Graz entwickelt worden ist, um Komponenten im Bereich der Kommunikations- bzw. Netzwerktechnik erweitert, was eine vollständige Automatisierung dessen ermöglicht. Abbildung 19 zeigt den mehrphasigen Übersichtsplan, welcher im Zuge der vorangegangenen Arbeit [20] entstanden ist.

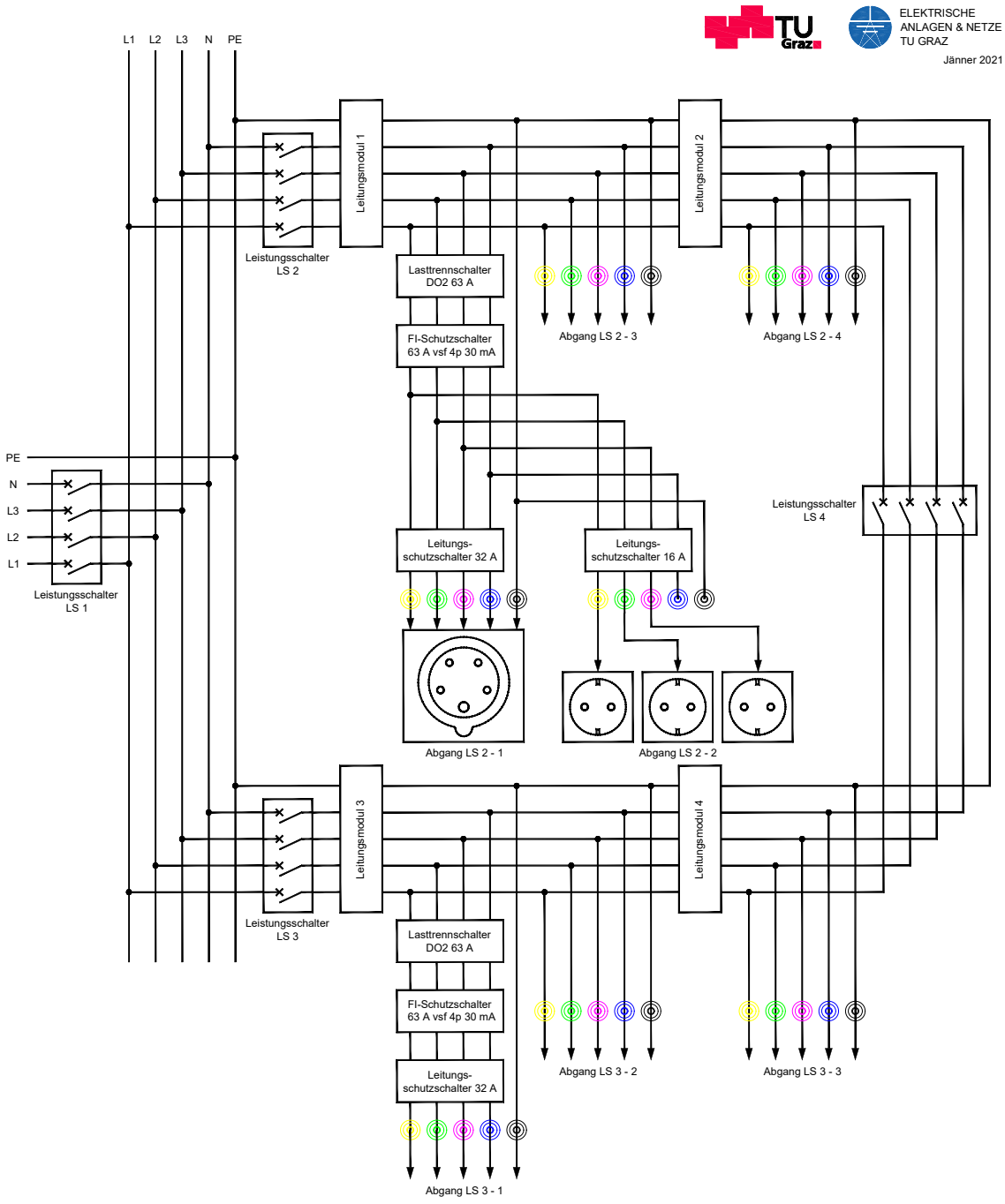


Abbildung 19: Mehrphasiger, schematischer Übersichtsplan [20]

3.1 Allgemeines

Da in der Realität Leistungsschalter in den verschiedenen Netzebenen durchaus über größere Entfernungen verteilt sind, wird im Zuge dieser Arbeit eine Methode entwickelt, um nicht nur eine manuelle, sondern auch eine Ansteuerung mittels Netzwerkschnittstelle zu gewährleisten und damit einen Fernzugriff zu ermöglichen. Der bestehende Laboraufbau wird dazu um geeignete Netzwerkkomponenten erweitert, um eine Kommunikation auf Netzwerkebene bereitzustellen, genaueres dazu in den nachfolgenden Unterkapiteln. Dabei wird die zurzeit aufstrebende Programmiersprache Python verwendet, welche zudem eine vordefinierte Schnittstelle zur Lastflussberechnungssoftware DIgSILENT Powerfactory zur Interaktion der beiden Softwareapplikationen bietet.

Abbildung 20 zeigt ein vereinfachtes Schema der physischen Netzwerkbereitstellung.

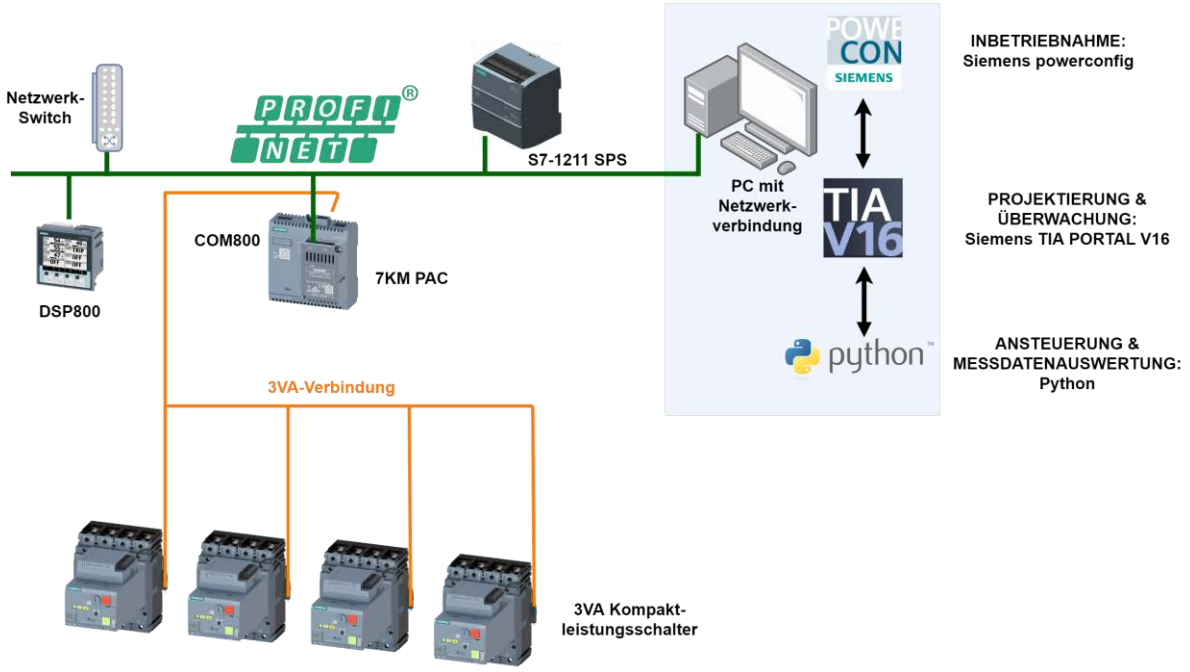


Abbildung 20: Prinzipschema der Netzwerkverbindung bzw. verkabelung des Labordemonstrators mit PROFINET

3.2 Kommunikation & notwendige Komponenten

Um die Leistungsschalter nicht nur manuell, sondern vollständig automatisiert über eine skriptbasierte Anwendung ansteuern zu können, wurde auf das Schema in Abbildung 20 zurückgegriffen. Der bereits bestehende Datenkonzentrator Siemens COM800 wurde um ein entsprechendes Netzwerkerweiterungsmodul ergänzt, näheres dazu ist Kapitel 3.2.1 zu entnehmen. Weiters wurde eine Siemens SPS der S7-Reihe verbaut, Details dazu siehe Kapitel 3.2.2.

Der Kommunikationsaufbau gestaltet sich wie folgt:

Die im Labordemonstrator verbauten Kompaktleistungsschalter der Siemens 3VA-Reihe sind jeweils mit einem Kommunikationsmodul COM060 ausgestattet, welches die Schaltzustände sowie Messwerte der einzelnen Kompaktleistungsschalter erfasst. Von diesem Kommunikationsmodul führt eine Verbindung zu einem T-Abzweig, mit Abschlusswiderstand an den jeweiligen Endpunkten. Als solche gelten hier der Datenkonzentrator und der letzte seriell eingebundene Kompaktleistungsschalter. Ausgehend vom T-Abzweig führt eine sogenannte 3VA-Line zum Datenkonzentrator COM800. Der Datenkonzentrator sorgt dann für die Weiterleitung an ein übergeordnete System. In diesem Fall wurde der Datenkonzentrator zusätzlich mit einem Erweiterungsmodul auf Basis des Industrial Ethernet Standards (auch PROFINET genannt) erweitert, um eine netzwerkgestützte Ansteuerung der Leistungsschalter zu ermöglichen. Zusätzlich wurde eine Siemens SPS S7-1211 verbaut, welche als sogenannter I/O-Supervisor zum erfolgreichen PROFINET-Kommunikationsaufbau erforderlich ist. Um Datenpakete gleichzeitig an das Display (DSP800 an der Schranktür) sowie an das übergeordnete System (I/O-Supervisor) übertragen zu können, ist zusätzlich ein Netzwerk Switch im Labordemonstrator erforderlich, um mehr Zugriffspunkte (Netzwerkanschlüsse) für das sogenannte „Durchrouten“ der Datenpakete zu erhalten. Somit können sowohl Daten der Kompaktleistungsschalter an das Display als auch an die SPS über den Datenkonzentrator respektive das Erweiterungsmodul gleichzeitig weitergeleitet werden.

Der Datenkonzentrator COM800 kann bis zu acht Kompaktleistungsschalter der Siemens 3VA-Reihe verwalten. Bei einem System mit mehr als acht Kompaktleistungsschaltern muss das bestehende System mit weiteren Datenkonzentratoren ausgestattet werden. Maßgebend hierbei ist zu erwähnen, dass jeder Datenkonzentrator, gleich ob COM100 oder COM800, im System mit jeweils dem gleichen Kommunikationsstandard ausgestattet werden muss. Zur Verfügung stehen hier drei verschiedene Typen von Erweiterungsmodulen, mit denen die Datenkonzentratoren versehen werden können. Zur Auswahl stehen die in der Industrie gängigen Kommunikationsstandards „PROFIBUS DP“, „Modbus RTU“ sowie „Switched Ethernet PROFINET“. [21] Eine Vorabrecherche hat ergeben, dass für die netzwerkgestützte Ansteuerung der Kompaktleistungsschalter eine speicherprogrammierbare Steuerung (SPS) mit RJ-45-Netzwerkschnittstelle notwendig ist. Aufgrund dessen ist die Wahl auf das mit selbiger Netzwerkschnittstelle ausgestattete Erweiterungsmodul „Switched Ethernet PROFINET“ gefallen, was somit eine Profinet-Netzwerkkommunikation ermöglicht.

Zudem muss jedes Bauteil, welches über eine Netzwerkschnittstelle neu in das System eingebunden wird, über eine eindeutige Netzwerkkennung verfügen. Diese Netzwerkkennung ist für die erfolgreiche Inbetriebnahme in der Software Siemens TIA Portal zu projektieren sowie an die CPU der speicherprogrammierbaren Steuerung zu übertragen.

3.2.1 Erweiterungsmodul Switched Ethernet Profinet-7KM PAC

Mit der Aufrüstung des Datenkonzentrators um das Erweiterungsmodul „7KM PAC Switched Ethernet PROFINET“ können die Kompaktleistungsschalter über den Netzwerkstandard PROFINET in das sogenannte „Totally Integrated Automation-Portal“ (TIA-Portal) eingebunden werden. Befestigt wird das Erweiterungsmodul mit den an der Rückseite herausragenden Anschlusspins, die sich an der Vorderseite des Datenkonzentrators versenken. Anschließend wird das Modul mit zwei Sicherungsschrauben fix am Datenkonzentrator befestigt, um ein Versehentliches abstecken des Moduls und somit einen Fehlerzustand des Bussystems zu verhindern. An der Oberseite des Moduls befinden sich zwei Netzwerkschnittstellen mit dem Standard RJ-45. Diese sind intern im Modul durchgeroutet, vergleichsweise wie ein Ethernet-Switch, und bieten die Möglichkeit der Datenweiterleitung an übergeordnete Systeme.

Über die zwei bereitgestellten PROFINET-Schnittstellen werden in weiterer Folge auch elektrische Messgrößen der Kompaktleistungsschalter übertragen und im TIA-Portal entsprechend weiterverarbeitet.

Abbildung 21 zeigt das Erweiterungsmodul Switched Ethernet PROFINET.

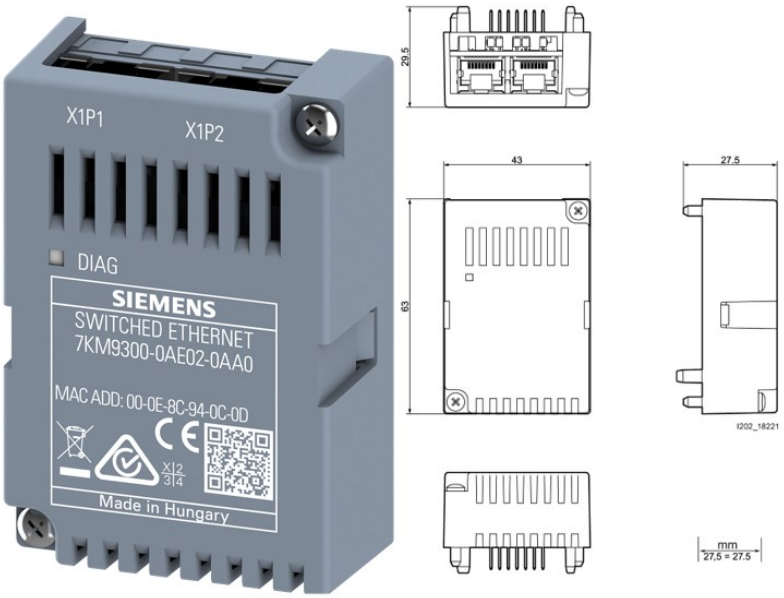
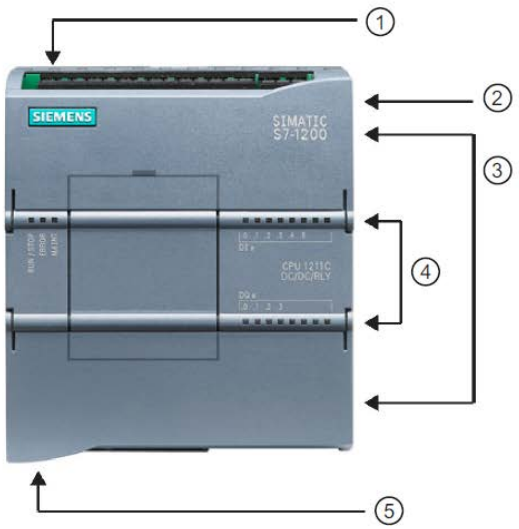


Abbildung 21: Erweiterungsmodul Switched Ethernet PROFINET [22]

3.2.2 SPS Siemens S7-1211

Da für den reibungslosen Kommunikationsaufbau auf Basis von PROFINET ein sogenannter I/O-Supervisor notwendig ist, fiel die Wahl auf eine speicherprogrammierbare Steuerung (SPS), welche diese Funktion übernimmt. Da die Ansteuerung mittels eines externen, auf Python basierenden Skripts stattfindet und so die SPS lediglich als Stellglied fungiert, welches die Eingangs- sowie Ausgangsadressen überwacht, fiel die Wahl auf die kleinste Ausführung der Siemens S7 SPS-Reihe. Im Detail wurde eine Siemens S7-1211 DC/DC/DC (Abbildung 22) verbaut, welche eine Steuerspannung von 24 V DC voraussetzt und selbige auch weiterschleifen kann. Verzichtet wurde hier explizit auf die etwas teurere AC Variante mit integriertem Netzteil, da der bestehende Labordemonstrator bereits über ein integriertes 24 V-Netzteil verfügt. Ergänzend verfügt dieses SPS-Model über sechs integrierte Digitaleingänge, vier Digitalausgänge sowie zwei Analogeingänge und ist zudem nicht erweiterbar, was aber für das Einsatzgebiet in diesem Fall unerheblich ist, da die gesamte Ansteuerung der Kompaktleistungsschalter ohnehin über die integrierte TCP/IP-Netzwerkschnittstelle abgehandelt wird und somit kein Digitaleingang bzw. Digitalausgang verwendet wird. Wie bereits erwähnt, übernimmt der integrierte CPU-Chip die Funktion des PROFINET-I/O-Supervisors, was für den Kommunikationsaufbau unerlässlich ist. Weiters wurde die SPS mit dem Hauptpotentialausgleich des Labordemonstrators verbunden. An der Unterseite der SPS befindet sich ein PROFINET Anschluss zur Datenübertragung. Dieser PROFINET Anschluss wurde mit dem Anschluss des Erweiterungsmoduls Switched Ethernet PROFINET verbunden, um somit die SPS als I/O-Supervisor zu etablieren. Bei Aufbauten, in denen es mehrere Erweiterungsmodule gibt, besteht die Möglichkeit, die PROFINET-Verbindungen mittels der am Erweiterungsmodul befindlichen zweiten PROFINET-Schnittstelle durchrouten zu können. Wichtig hierbei ist, dass jedes Bauteil, welches über eine PROFINET Schnittstelle besitzt, eine eigene und im Netzwerk eindeutige PROFINET-Adresse bzw. einen PROFINET-Namen besitzen muss, vgl. hierzu Kapitel 3.4.2.



① Stromanschluss
② Steckplatz für eine Memory Card unter der oberen Abdeckklappe
③ Steckbarer Klemmenblock für die Anwenderverdrahtung (hinter den Abdeckklappen)
④ Status-LEDs für die integrierten E/A
⑤ PROFINET-Anschluss (auf der Unterseite der CPU)

Abbildung 22: SPS Siemens S7-1211 [23]

3.3 Erstinbetriebnahme über die Software Siemens powerconfig

Mit der Erweiterung um die zusätzlichen Kommunikationskomponenten musste der gesamte Aufbau mittels der Inbetriebnahme-Software Siemens powerconfig neu in Betrieb genommen werden.

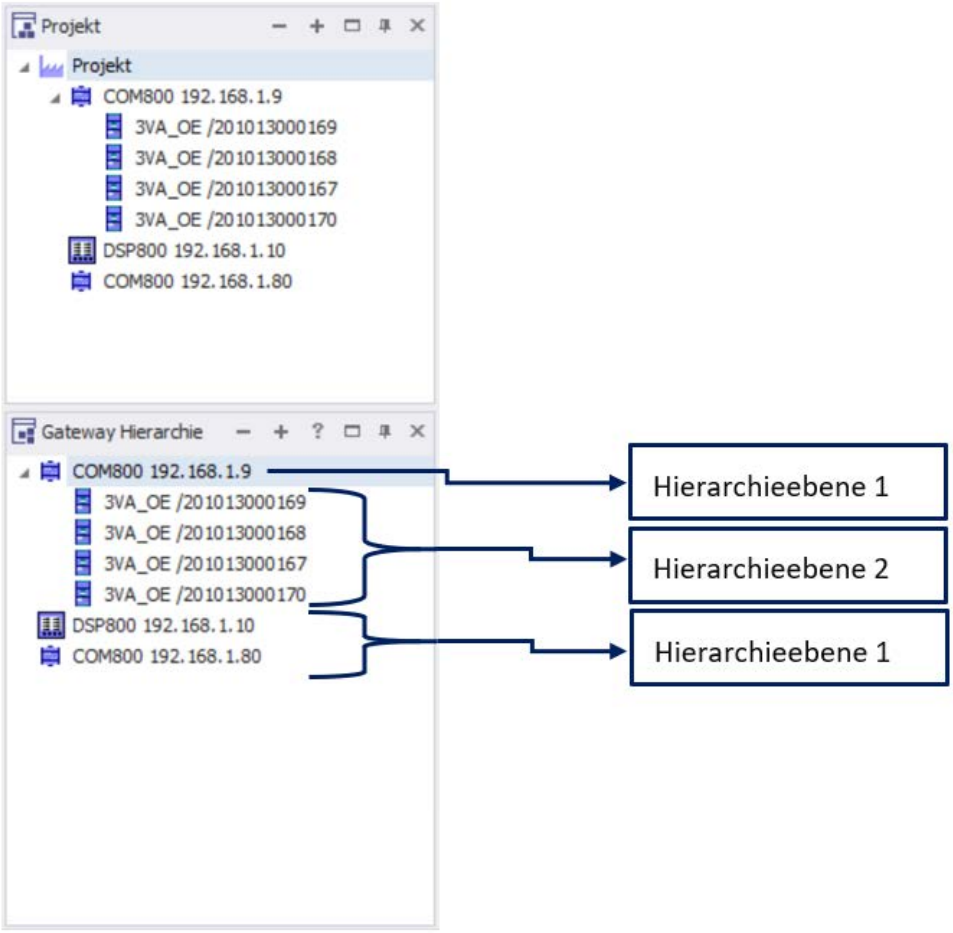


Abbildung 23: Projektstruktur & Gateway-Hierarchie

Abbildung 23 zeigt die neue Projektstruktur sowie den neuen Hierarchiebaum der erweiterten Gerätekonfiguration. In der Gateway-Hierarchie befinden sich die Kompaktleistungsschalter der 3VA-Reihe eine Ebene unter dem Datenkonzentrator COM800 was mit einer Einrückung nach rechts dargestellt wird. Somit bildet der Datenkonzentrator COM800 das Gateway zu den Kompaktleistungsschaltern. Das bedeutet, dass ein Verbindungsaufbau und somit ein Informations- bzw. Datenaustausch zwischen den beiden Komponenten besteht. Die Kompaktleistungsschalter sind über ein Kommunikationsmodul (COM060) mit einem T-Abzweig (mit Abschlusswiderständen an den beiden Endpunkten der Leitung) verbunden, von welchem die sogenannte 3VA-Line als Kommunikations- bzw. Busleitung bis zum Datenkonzentrator verläuft.

Das Display DSP800 sowie das Erweiterungsmodul befinden sich in der Hierarchieebene auf derselben Ebene wie der Datenkonzentrator und besitzen ihrerseits keine darunterliegenden Ebenen. Die Netzmaske (en. subnet mask) ist bei allen Geräten 255.255.255.0 und legt fest, welche IP[9]-Adressen ein Gerät in seinem eigenen Netz, ohne die Hilfe eines Routers, erreichen kann. Hier wären beispielsweise maximal 255 Geräte (Wertebereich der letzten Stelle der Netzmaske verläuft von 1 bis 255) möglich. Wenn das Projekt eine derartige Größe erreicht und mehr als 255 Geräte erforderlich sind, muss die Netzmaske dementsprechend angepasst werden, eine Änderung dieser auf 255.255.0.0 ermöglicht bereits $255 \cdot 255 - 2 = 65534$ verschiedene Geräte. Bei einer solchen Änderung müssen zudem auch die IP-Adressen der Geräte angepasst werden. Die im hier dokumentierten Projekt zugewiesen IP-Adressen sind in Tabelle 1 zusammengefasst.

Tabelle 1: Adressierung

Gerät	Gerätebezeichnung	IP-Adressierung
Datenkonzentrator	COM800	192.168.1.9
Display	DSP800	192.168.1.10
Speicher-programmierbare Steuerung	S7-1211 DC/DC/DC	192.168.1.50
Erweiterungsmodul	7KM PAC Switched Ethernet PROFINET	192.168.1.80
Gateway: 192.168.1.1		
Netzmaske: 255.255.255.0		

[9] IP, kurz für Internet-Protokoll, wird einem Gerät, welches sich in einem Netz befindet, eindeutig zugewiesen und sorgt somit für die Erreichbarkeit sowie die Adressierbarkeit dessen.

3.4 Konfiguration mittels Siemens Software TIA Portal V16

Nach erfolgreicher Inbetriebnahme des neuen Labordemonstrators kann auf die Siemens Software TIA-Portal gewechselt werden. Dabei muss ein neues Projekt angelegt und die entsprechende Siemens SPS (6ES7211-1AE40-0XB0) ausgewählt werden. Danach sind alle PROFINET Gerätenamen und Adressen eindeutig und einmalig jedem Bauteil zuzuordnen und in der Software durchzurouten. Um Zugriff auf die neueste Produktpalette zu erlangen, muss der Hardware Support Katalog (HSP) auf die aktuell gültige Version upgegradet werden. Mit diesem HSP Upgrade wird die Produktpalette des TIA-Portals um die Siemens SENTRON Geräteserie, welche auch den Datenkonzentrator COM800, COM100, sowie die verschiedenen Kompaktleistungsschalter der 3VA-Reihe beinhaltetet, ergänzt.

Nun ist ein exaktes Offlineabbild des Aufbaus zu projektieren, da die Umsetzung in der Software TIA-Portal nur dann funktioniert, wenn dieses Offline-Projektabbild detailgetreu mit dem Online-Projektabbild übereinstimmt. Abbildung 24 zeigt die Projektzusammensetzung mit den 3VA-Kompaktleistungsschaltern sowie dem Datenkonzentrator COM800.

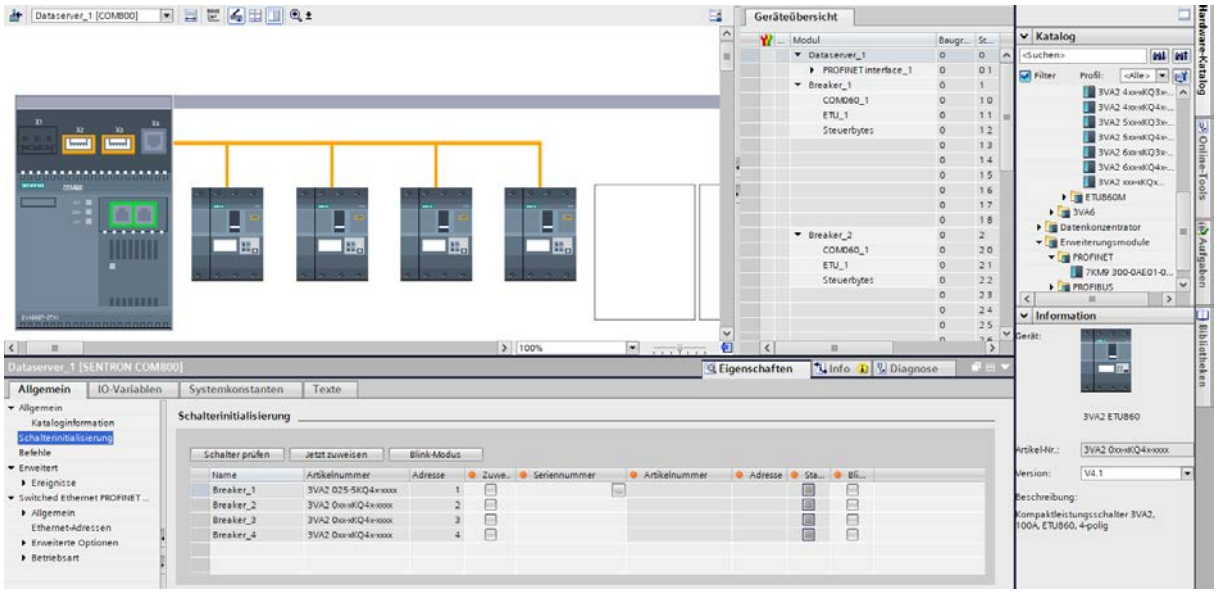


Abbildung 24: Projektierung in TIA Portal V16

Aufgrund eines historisch bedingten Firmwareupdates des Datenkonzentrators ist es zu einer Fehlermeldung im Zuge des Kompilierungsvorganges gekommen, welche weitreichende Folgen mit sich gezogen hat. Die Software TIA-Portal in der Version V16, die zum Zeitpunkt der Projektierung dieses Projekts sogleich die aktuellste Version darstellt, war nur für Datenkonzentratoren mit einer Firmware Version bis V4.3 ausgelegt. Die Firmwareversion des Datenkonzentrators, der hier verwendet wurde, wurde jedoch im Zuge der Vorarbeiten auf V4.4 aktualisiert. Diese scheinbar kleine Abweichung hatte jedoch große Auswirkungen und verhinderte beispielsweise, dass der Datenkonzentrator über das Erweiterungsmodul auf die, in der Hierarchie darunter, befindlichen Kompaktleistungsschalter zugreifen konnte. Zudem bestand das Problem, dass es zu einer Abweichung des Offline- sowie des Online-Projektabbildes kam und somit das erfolgreiche Kompilieren des Projektes aus Sicherheitsgründen softwaretechnisch untersagt wurde.

Konfrontiert mit diesem Problem musste nun eine Behelfslösung gefunden werden, um das Kompilieren zu ermöglichen. Eingehende Recherchen lieferten schlussendlich die Lösung der Problematik mittels einer GSDML[10]-Dateieinbindung in das Projekt, vgl. hierzu Kapitel 3.4.1.

Weiters mussten einige Features, die den Fernzugriff und auch den Zugang über Drittsoftware erlauben, zugelassen werden. Dies ist deshalb erforderlich, da in weiterer Folge mittels der Programmiersprache Python auf die internen Daten der Steuerung zugegriffen werden muss. Dieser notwendige Schritt sollte in der Praxis durch zusätzliche Verschlüsselungsmodule abgesichert werden, um den Zugriff von unbefugten Dritten zu verhindern. Da es sich hierbei um einen an der TU Graz für den Laborbetrieb eingesetzten Aufbau handelt, kann jedoch davon ausgegangen werden, dass auf diese Verschlüsselung verzichtet werden kann. Die Öffnung der CPU, sodass externe Programmiersprachen auf die Steuereinheit zugreifen können, ist in der Praxis aus eben genannten Gründen eher selten anzutreffen, da hier meist die eigens dafür entworfenen Software-Pakete in Verwendung sind. Da sich dieses Projekt jedoch vorrangig mit der Entwicklung bzw. Ausgestaltung neuer Betriebsstrategien in Bezug auf die Netzstruktur beschäftigt, sind die dementsprechenden Vereinfachungen (beispielsweise hinsichtlich der IT-Security) allenfalls zulässig und bieten Steuermöglichkeiten, welche mit der herkömmlichen Software so nicht möglich wären.

[10] GSDML-Generic Station Description Markup Language

3.4.1 Generic Station Description Markup Language (GSDML) - Datei

Da es sich bei PROFINET um einen international gültigen und offenen Kommunikationsstandard handelt ist jeder Hersteller, der ein entsprechend kompatibles Gerät mit einer solchen Schnittstelle herstellt und vertreibt, verpflichtet, eine sogenannte GSDML-Datei (Generic Station Description Markup Language) bereitzustellen. Einige Hersteller stellen diese auf ihren Produktseiten im Internet zur Verfügung, manche erst nach expliziter Anfrage. In dieser GSDML-Datei werden bauteilspezifische wichtige Parameter in einer XML[11]-Datei abgespeichert. Dies ermöglicht es, dass jeder Hersteller mit anderen Bauteilkomponenten von Fremdherstellern arbeiten kann und zumindest die Grundfunktionen verwenden kann. Die Einbindung eines Bauteils mittels GSDML-Datei bietet zwar den Vorteil von Versionsunabhängigkeiten, jedoch auch die Nachteile eines eingeschränkten Funktionsumfangs sowie einer grafisch wenig ansprechenden Aufbereitung. Da in diesem Projekt lediglich die CPU der SPS auf die externe Peripherie vorbereitet werden muss, indem Adressen reserviert werden, stellt dies jedoch kein Problem dar, da die Ansteuerung in weiterer Folge unabhängig von der Software Siemens TIA-Portal über ein Python-Skript abgehandelt wird. In Abbildung 25 ist das Abbild der GSDML-Datei für den COM800 Datenkonzentrator, der mittels demselben Dateiformat in das Projekt eingebunden wird, dargestellt. Vergleicht man Abbildung 24 mit Abbildung 25, wird der grafische Unterschied deutlich.

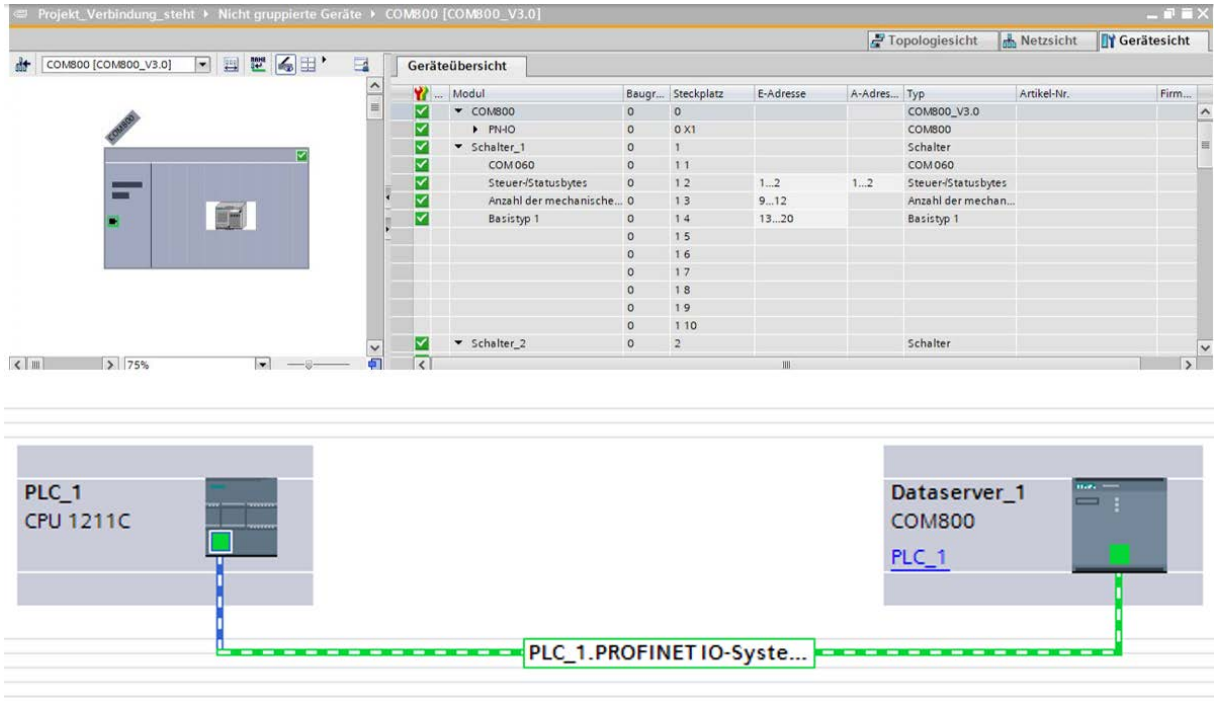


Abbildung 25: Einbindung der GSDML-Datei in TIA-Portal

[11] XML steht abgekürzt für Extensible Markup Language und ist eine Sprache zur Darstellung von Daten in Form von hierarchisch strukturierter Textdateien. Dieses Format erlaubt sowohl eine Interpretation durch Menschen als auch durch Maschinen.

Unter Verwendung dieser GSDML-Datei können die Basistypen der Leistungsschalter ausgewertet werden. Die verwendete GSDML-Datei des COM800 Datenkonzentrators bietet sieben verschiedene Basistypen, welche ausgelesen werden können, die sich jeweils in der Auswahl der Messdaten sowie im Detailierungsgrad unterscheiden. Für das Projekt wird durchgehend der Standardtyp 3 der GSDML-Datei verwendet, da dieser Typ die meisten Messwerte (insgesamt 14 verschiedene Messwerte pro Kompaktleistungsschalter) mit ausreichender Genauigkeit bietet. Abbildung 26 zeigt die im Basistyp 3 zur Verfügung stehenden Messwerte sowie deren Speicherbelegung, Einheit sowie den Variablentyp, welcher für den jeweiligen Messwert zur Verfügung steht.

Byte	Dateninhalt	Format	Einheit	8er ETU	5er ETU
0 … 1	Strom in Phase L_1	U16	A	✓	✓
2 … 3	Strom in Phase L_2	U16	A	✓	✓
4 … 5	Strom in Phase L_3	U16	A	✓	✓
6 … 7	Maximaler Strom in der höchst belasteten Phase	U16	A	✓	✓
8 … 9	Strom im Neutralleiter	U16	A	✓	✓
10 … 11	Spannung L_1-L_2	U16	V	✓	–
12 … 13	Spannung L_2-L_3	U16	V	✓	–
14 … 15	Spannung L_3-L_1	U16	V	✓	–
16 … 17	Spannung L_1-N	U16	V	✓	–
18 … 19	Spannung L_2-N	U16	V	✓	–
20 … 21	Spannung L_3-N	U16	V	✓	–
22 … 23	Durchschnittlicher Powerfaktor in 3 Phasen	U16	Faktor 1000 des tatsächlichen Wertes	✓	–
24 … 25	Wirkenergie Bezug in 3 Phasen	U16	MWh	✓	–
26 … 27	Scheinleistung Σ in 3 Phasen	U16	kVA	✓	–

✓ relevant für
- entfällt

Abbildung 26: Messwerte des Basistyps 3 [21]

Weiters können mittels GSDML-Datei sowohl Status- als auch Steuerbytes der jeweiligen Kompaktleistungsschalter ausgelesen und gegebenenfalls modifiziert werden. Dieses Auslesen ist für die spätere Interaktion und Integration mittels der Programmiersprache Python von größter Wichtigkeit, vgl. Kapitel 3.5. Eine entsprechende Zusammenstellung der Statusbytes sowie der Steuerbytes können Abbildung 27 und Abbildung 28 entnommen werden.

Byte	Bit	Wert	Information zum Kompaktleistungsschalter 3VA	8er ETU	5er ETU	DO[1]	SEO
0	0, 1	0	Trennstellung	■	■	✓	-
		1	Betriebsstellung				
		2	Prüf- / Teststellung				
		3	Nicht vorhanden				
	2, 3	0	Nicht bereit	■	■	■	■
		1	Aus				
		2	Ein				
		3	Hat ausgelöst				
	4	1	Reserviert	-	-	-	-
	5	1	Reserviert				
	6	1	Federspeicher ist gespannt	■	■	■	✓
	7	1	Überlastwarnung liegt an	✓	✓	-	-
1	0	1	Reserviert				
	1	1	Reserviert				
	2	1	Schreibschutz aktiviert	■	■	-	-
	3	1	Reserviert				
	4, 5, 6	0 … 7	Auslösegrund der letzten Auslösung	✓	✓	-	-
		0	Keine Auslösung bzw. letzte Auslösung quittiert				
		1	Überlastauslösung (L)				
		2	Unverzögerter Kurzschluss (I)				
		3	Kurzzeitverzögerter Kurzschluss (sd)				
		4	Erdschluss (G)				
		5	Auslösung durch erweiterte Schutzfunktion				
		6	Überlast im Neutralleiter (N)				
		7	Reserviert				
	7	1	Lastabwurfwarnung	✓	✓	-	-

[1] Einschubeinheit (draw-out unit)
✓ relevant für
■ ist vorhanden
- entfällt

Abbildung 27: Statusbytes der 3VA-Kompaktleistungsschalter [21]

Byte	Bit	Wert	Angestoßene Funktion des Kompaktleistungsschalters 3VA
0	0,1	0 … 3	Schalten des Kompaktleistungsschalters 3VA
		0	Nicht definiert (keine Aktion)
		1	Ausschalten *)
		2	Einschalten *)
		3	Nicht definiert (keine Aktion)
	2	1	Rücksetzen: Letzter Auslösegrund / Quittierung des SEO
	3	1	Reserviert
	4	1	Reserviert
	5	1	Reserviert
	6	1	Reserviert
	7	1	Reserviert
1	0,1	0	Reserviert
		1	Reserviert
		2	Reserviert
		3	Reserviert
	2	1	Löschen der Auslöse- und Ereignisaufzeichnung
	3	1	Zurücksetzen der Minimal- / Maximalwerte
	4	1	Reserviert
	5	1	Reserviert
	6	1	Quittieren der Maintenance-Information
	7	1	Reserviert

*) abhängig vom Einsatz des SEO (Stored Energy Operator)

Abbildung 28: Steuerbytes der 3VA-Kompaktleistungsschalter [21]

3.4.2 S7-1211 CPU-Programm und Vorbereitung für externe Ansteuerung

Um die interne CPU der S7-SPS für die zukünftigen Aufgabenstellungen vorzubereiten, sind in der Siemens Software TIA Portal V16 zahlreiche Einstellungen vorzunehmen, welche in diesem Unterkapitel näher erläutert werden. Die Tatsache, dass die Ansteuerung von Kompaktleistungsschaltern mittels externer Fremdsoftware (hier Python in der Entwicklungsumgebung PyCharm) nicht zu den Standardanwendungsfällen gehört, macht diese Schritte jedoch unumgänglich.

Um einen externen Zugriff mittels eines Skripts in weiterer Folge zu ermöglichen, ist die CPU auf die externe Peripherie entsprechend vorzubereiten. Da hier die SPS als Stellglied dient, müssen im Untermenü der GSDML-Datei die im Projekt bestehenden Kompaktleistungsschalter integriert werden. Dazu sind für jeden Kompaktleistungsschalter sowohl zwei digitale Eingangs- als auch zwei digitale Ausgangsadressen im CPU Speicher zu reservieren und anzulegen. Im nächsten Schritt müssen der bereits erwähnte Basistyp 3, welcher schließlich die Messwerte der Kompaktleistungsschalter beinhaltet, in den CPU Speicher integriert werden. Abbildung 29 zeigt diese Parametrisierung der Adressen in der Software Siemens TIA Portal V16. Notwendig ist dieser Schritt deshalb, da in weiterer Folge über PROFINET, also einer Netzwerkschnittstelle, die Kompaktleistungsschalter geschaltet werden müssen. Dazu ist es erforderlich, auf sogenannte Status- sowie Steuerbytes, vgl. Kapitel 3.4.1, zuzugreifen um entsprechend über die PROFINET-Schnittstelle auf die externe Peripherie zu gelangen. Weiters sind zusätzlich noch zahlreiche andere relevante Messwerte in den CPU-Speicher zu integrieren, um für die nachfolgende Messdatenauswertung auf einen möglichst umfangreichen Messdatensatz zurückgreifen zu können.

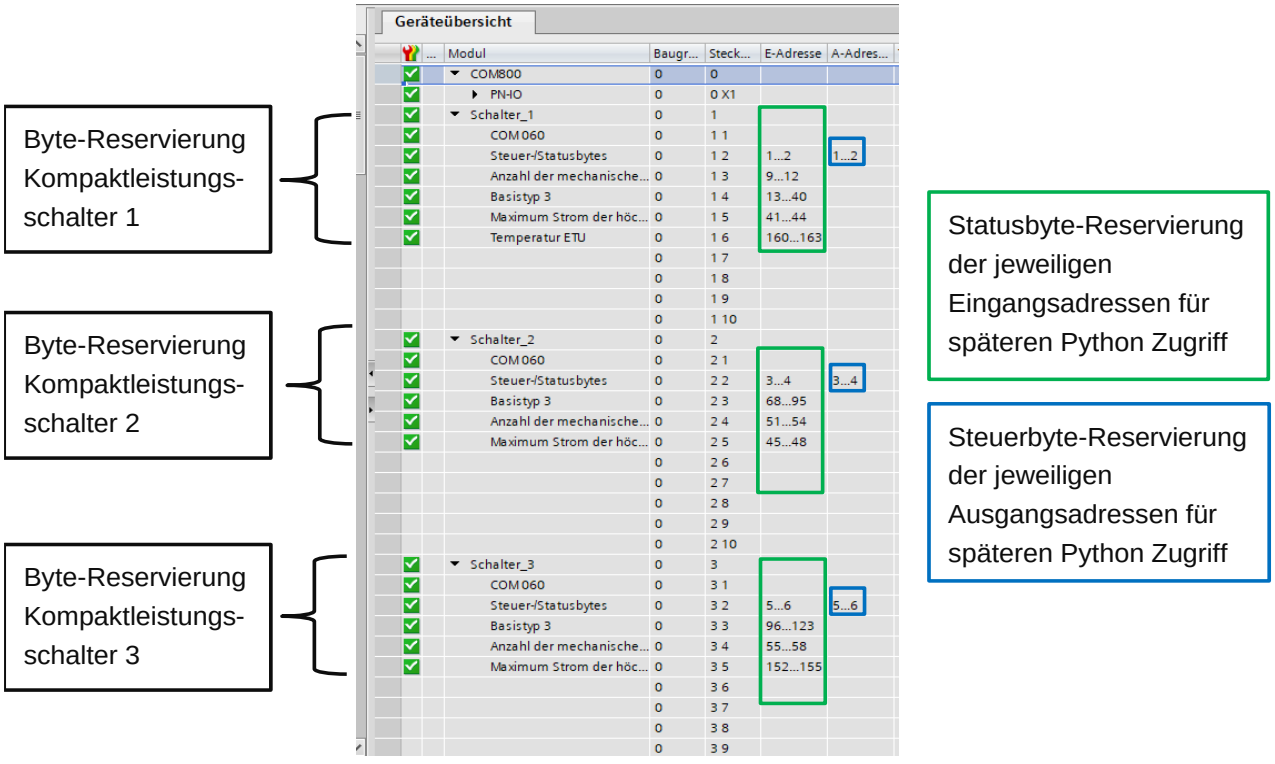


Abbildung 29: GSDML Byte Reservierung in TIA Portal am Beispiel der Kompaktleistungsschalter 1 bis 3

3.4.3 TIA Portal Webserver

Im Zuge dieser Arbeit ist ein Webserver in TIA Portal zu projektieren und freizugegeben. Dies ermöglicht in weiterer Folge, ohne den Besitz einer TIA-Portal-Software-Lizenz auf entsprechend freigegebene Daten zuzugreifen. Dadurch wird über eine dafür anzulegende Netzwerkadresse (IP-Adresse) ein Echtzeitmonitoring der Messdaten mittels eines Browsers ermöglicht.

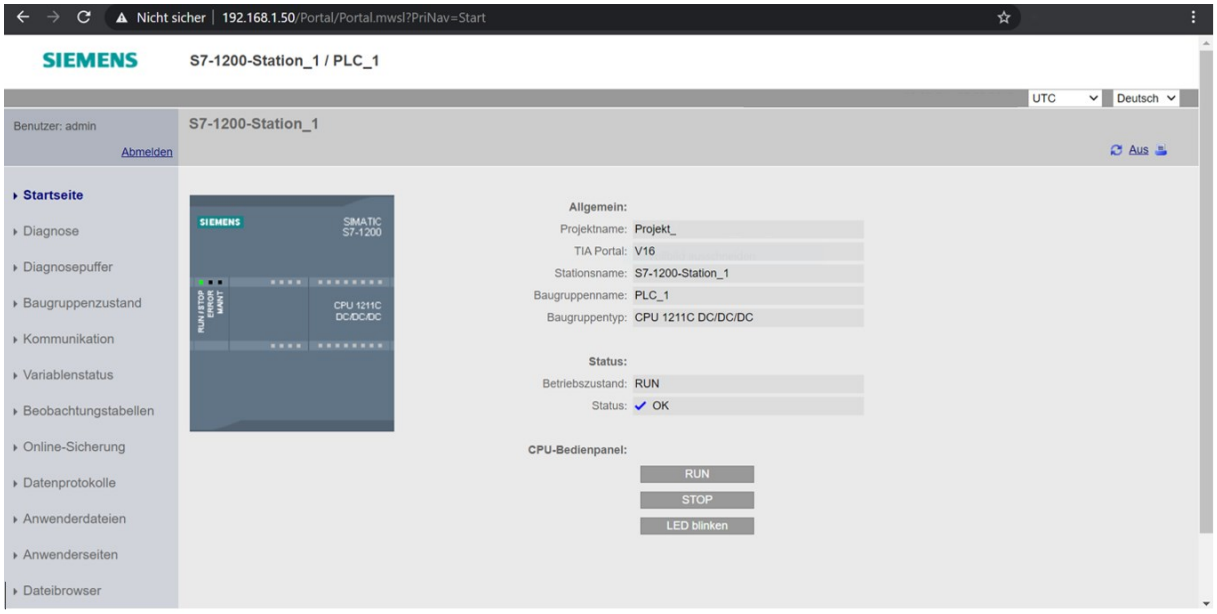


Abbildung 30: Siemens TIA-Portal Webserver-Interface

Mit Hilfe des Webserver-Zugriffs ist es dem/der Benutzer*in möglich, alle relevanten Messdaten der vier Kompaktleistungsschalter sowie die Statusbytes einzusehen. Die Steuerbytes können nach erfolgreicher Passworteingabe ebenso über das Webinterface geändert werden. Das Schalten der Kompaktleistungsschalter ist somit zusätzlich via Webserver möglich. Der vorliegende Webserver soll jedoch nur ein benutzerfreundlicher Zusatz sein, denn das Hauptaugenmerk der Arbeit liegt in der Ansteuerung der Kompaktleistungsschalter über das externe Python-Skript, welches im nachfolgenden Kapitel 3.5 genauer beschrieben wird.

3.5 Python Integration und Beschreibung des Skripts

Nach erfolgreicher Projektierung der Komponenten in der Siemens Software TIA-Portal V16 werden deren Parameter und nach dem Anpassen der externen Zugriffsrechte auf die S7-1211 CPU deren Einstellungen in die SPS geladen sowie kompiliert. Anschließend folgt der externe Verbindungsaufbau zwischen dem in der Programmiersprache Python entwickelten Automatisierungsskript und den Kompaktleistungsschaltern. TIA-Portal erfüllt im aktuellen Schritt nur noch die Aufgabe des Parametermonitorings, d. h. Status- sowie Steuerbytes und ausgewählte Messwerte können dabei in Echtzeit überwacht werden.

Um überhaupt mittels der Programmierschnittstelle Python auf die Siemens S7 SPS Zugriff zu erlangen, ist es notwendig ein Open-Source-Modul in das Skript zu integrieren, vgl. hierzu Kapitel 3.5.2.

Dabei ist in erster Linie eine Netzwerkkommunikation zwischen Python und der SPS herzustellen. Dazu muss sowohl die IP-Adresse der SPS sowie die in TIA-Portal projektierte Rack- bzw. Slotposition in Python definiert werden. Dies kann am Beginn des Codes beobachtet werden, siehe dazu Kapitel 9.2.1 im Anhang.

Am Beginn des Skripts werden IP-Adresse, Slot- bzw. Rackposition definiert und im nächsten Schritt eine Abfrage auf deren Konformität getätigt. Es wird hier zuerst überprüft, ob die SPS eine gültige Verbindung zu Python aufbauen kann. Weiters werden etwaige CPU-Fehlermeldungen ausgelesen, um ein Starten des Codes im Fehlerfall zu vermeiden. Nach erfolgreicher Überprüfung der CPU und deren Parameter werden nun die vorhin in TIA-Portal V16 definierten Ausgänge respektive Steuerbytes der jeweiligen Kompaktleistungsschalter vor dem Schaltprozess ausgelesen. Im nächsten Schritt werden auch die Statusbytes ausgelesen, um die Position der Kompaktleistungsschalter ermitteln zu können. Für jeden Kompaktleistungsschalter werden diese Statusbytes einzeln, in einer dafür vorgesehenen Variable, gespeichert. Nun wird in einem „IF-ELIF-ELSE“ Konstrukt jede mögliche Schalterstellung verarbeitet. Da der hierbei verwendete Labordemonstrator vier Kompaktleistungsschalter beinhaltet, gibt es maximal $2^4 = 16$ verschiedene Möglichkeiten. Die augenblickliche Schalterposition wird entsprechend einer der 16 möglichen Variationen zugeordnet und es wird ein Funktionsaufruf eines weiteren Codes, vgl. hierzu Visualisierungscode Kapitel 9.2.2 im Anhang, zur übersichtlichen Anzeige der aktuellen Schalterposition für den/die Benutzer*in getätigt. Kann keine der 16 Möglichkeiten der aktuellen Schalterposition zugeordnet werden, erfolgt eine Abfrage, ob sich einer der Kompaktleistungsschalter in einer Auslöseposition (en. trip) befindet. Ist dies der Fall, so erfolgt eine dementsprechende Meldung am Bildschirm. Sollte jedoch auch keine der Auslösepositionen zutreffend sein, so wird das Programm aus Sicherheitsgründen mit einem „System-Exit-Befehl“ beendet, ohne eine Schalthandlung durchzuführen. Dies wird damit begründet, dass es sich dabei dann höchstwahrscheinlich um eine fehlerhafte Eingabe eines/einer Benutzer*in handelt und man dann davon ausgehen muss, dass diese Handlung ungewollt stattgefunden hat.

Kann jedoch eine der 16 möglichen Schaltervariationen zugeordnet werden, so wird im nächsten Schritt ein CSV[12]-File, welches sich auf dem ausführenden Computer in einem im Code definierten Programmpfad befindet, geöffnet und es wird in der Zelle B2 der binäre Wert ausgelesen. Der Wert in diesem CSV-File hat seinen Ursprung in einem im Zuge von [24] entwickelten Analysealgorithmus zur Bewertung unterschiedlicher Netzkonfigurationen. Dabei wird die bestmögliche Schalterkombination der Kompaktleistungsschalter auf Basis von sechs Kennzahlen (en. Key Performance Indicators, KPIs) ermittelt.

Nach erfolgreichem Auslesen des binären CSV-File-Wertes wird dieser im Code gespeichert. Im nächsten Schritt wird eine Abfrage getätigt, in der überprüft wird, ob die aktuelle Schalterposition von der besten Schalterposition (anhand des CSV-Files) abweicht. Ist dies nicht der Fall, so befinden sich die Kompaktleistungsschalter bereits in der idealen Schalterposition und es wird – auch um unnötige Schalthandlungen zu vermeiden – keine entsprechende Umschaltung durchgeführt. Weichen die beiden Werte jedoch voneinander ab, so ist klar, dass eine Schalthandlung unumgänglich ist. Bevor die Kompaktleistungsschalter allerdings tatsächlich geschaltet werden, wird im Vorhinein noch eine Abfrage getätigt, ob der neue Schaltzustand sinnvoll ist. Dies bedeutet, dass es mögliche Schalterkombinationen gibt, bei denen es zu einer Nichtversorgung der angeschlossenen Lasten kommen kann. Um sicherzustellen, dass solche Schalthandlungen nicht vorgenommen werden, werden im Code all jene Situationen, die dahingehend nicht sinnvoll sind, berücksichtigt und in weiterer Folge unterbunden. Ist der Schaltzustand jedoch zweckmäßig, so kann mit dem Code fortgefahren werden. Je nach Rekonfiguration werden die Kompaktleistungsschalter dementsprechend geschaltet. Wichtig hierbei ist zu erwähnen, dass nach dem Schalten eine Wartezeit von fünf Sekunden implementiert ist, um das Aufziehen des Motorantriebs (en. Stored Energy Operator, SEO), der sich am Kompaktleistungsschalter befindet, zu ermöglichen. Diese Verzögerungszeit stellt somit sicher, dass der Kompaktleistungsschalter nur dann eingeschaltet werden kann, wenn dieser auch über einen aufgezogenen Federspeicher verfügt, sodass dieser auch wieder in der Lage ist, den Kompaktleistungsschalter mit Hilfe der gespeicherten Federenergie im SEO sicher abschalten zu können. Weiters wurde hier besonders darauf geachtet, dass es nicht zu unnötigen Schalthandlungen kommt sowie zu solchen, die keine Auswirkungen auf das System hätten. Beispielsweise wird jeder Schaltversuch, bei dem der Hauptschalter (LS 1 gemäß Abbildung 19) nach dem Schaltvorgang AUS ist, mit einem „System-Exit-Befehl“ abgefangen. Weiters werden auch jene Schalthandlung, bei denen der Kompaktleistungsschalter LS 1 und LS 4 EIN und Kompaktleistungsschalter LS 2 und LS 3 AUS ist untersagt, da es hier aufgrund der bestehenden Topologie zu keinem Stromfluss kommen würde. Es wird jedoch im Code auch jener Fall bedacht, bei dem man an einer beliebigen Position in der Topologie eine Quelle anschließen könnte. Die entsprechenden Codezeilen sind jedoch in der aktuellen Version auskommentiert, da die Versorgung des Labordemonstrators aktuell ausschließlich über den dafür vorgesehenen Anschluss an der Seite des Schaltschranks erfolgt. Nach erfolgreichem

[12] CSV - (engl. comma-separated values; de. Komma getrennte Werte)

Schalten auf den idealen Schaltzustand werden erneut die Statusbytes der Steuerung ausgelesen, um wiederum diese neue Schalterkonfiguration einer entsprechenden Visualisierung zuordnen zu können. Am Ende werden zusätzlich noch alle Steuerbytes der Kompaktleistungsschalter auf Bytewert null gesetzt, um falschen Steuerbefehlen entgegenzuwirken.

Zusätzlich werden im Code alle Messwerte sowie Schalterparameter des Basistyps 3 (siehe Abbildung 26) in entsprechenden Variablen gespeichert, um diese anschließend in ein externes File schreiben zu können. Das Skript bietet zudem die Möglichkeit, Diagramme der Phasenströme sowie ein Übersichtsdiagramm der höchstbelasteten Phasenströme je Kompaktleistungsschalter in Ampere jeweils vor und nach der Schalthandlung zu generieren. Um den Stromwert der höchstbelasteten Phasen in einer Dezimaldarstellung zu erhalten, ist hier (u. a. aufgrund der Verwendung der GSDML-Datei) auf eine Umrechnung gemäß IEEE 754 [25] zurückzugreifen, siehe dazu Kapitel 3.5.1.

Weiters ist eine Protokollierung sämtlicher Abläufe bzw. Schalthandlungen in Form einer Textdatei im Skript integriert. Zudem besteht am Anfang des Programms die Möglichkeit, bedienungsfreundlich über zahlreiche Variablen den von dem/der Benutzer*in beabsichtigten Output zu generieren. So ist es beispielsweise möglich, zusätzlichen Konsolen-Output zu generieren, welcher für den/die Benutzer*in von Bedeutung sein könnte. Auch besteht dadurch die Möglichkeit, Messwerte in ein externes File zu schreiben sowie auch zahlreiche Abbildungen und Diagramme, die für den/die Benutzer*in von Belangen sein könnten, zu generieren. Bei Ausführung des Codes wird automatisiert ein Ausgabeordner erstellt, welcher ein Protokoll des gesamten Prozesses, Informationsdateien für den/die Benutzer*in sowie 25 verschiedene Abbildungen der Schalthandlung beinhaltetet.

Für das im Zuge dieser Arbeit entwickelte Skript wurde die Programmiersprache Python verwendet, da diese einerseits eine Interaktion mit dem Netzberechnungsprogramm DIgSILENT PowerFatory und andererseits eine Kommunikation zu Simens-S7-Geräten (mittels des entsprechenden Moduls) ermöglicht. Python bietet grundsätzlich einen ähnlichen Funktionsumfang als das Softwarepaket MATLAB, so werden beispielsweise sämtliche Diagramme (im Sinne der Versuchsauswertung) dieser Arbeit mittels Python erstellt. Zudem bietet Python eine große Community und wird ständig weiterentwickelt. Für die Entwicklung des gegenständlichen Algorithmus wurde Python in der Version 3.8 sowie die Entwicklungsumgebung PyCharm 2020.3 (Community Edition) in der „Buildversion“ Jan.27,2021 verwendet. Als Ethernet-Schnittstelle vom Computer (welcher als Plattform für Python resp. PyCharm dient) zum Labordemonstrator wird ein „ASIX AX88179 USB 3.0 to Gigabit Ethernet“ Adapter mit der eingestellten Internetprotokoll-V4-Adresse 192.168.1.30 und der Netzmaske 255.255.255.0 verwendet.

Abbildung 31 zeigt das vereinfachte Ablaufdiagramm des Automatisierungsprozesses wobei zur schematischen Darstellung der wesentlichen Abläufe einzelne Code-Segmente entsprechend zu Blöcken zusammengefasst sind.

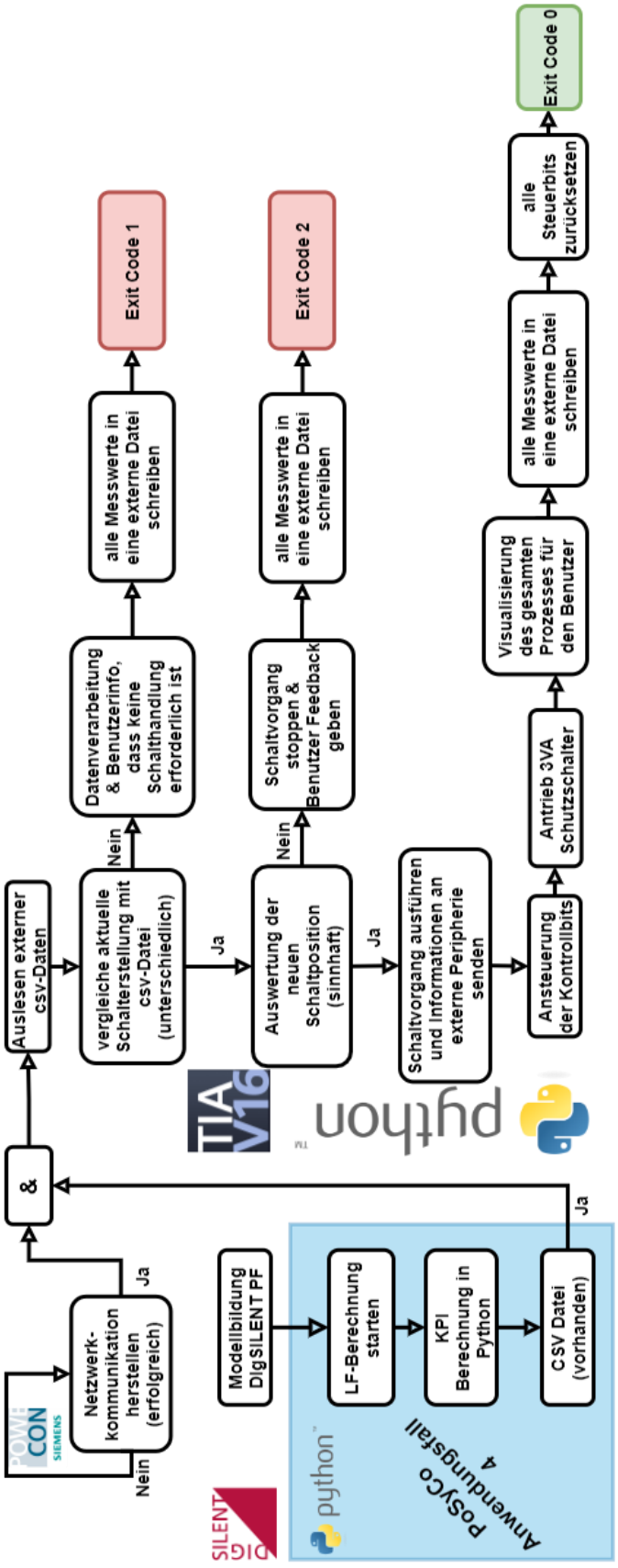


Abbildung 31: Ablaufdiagramm des Automatisierungsprozesses

3.5.1 Umrechnung gemäß IEEE 754

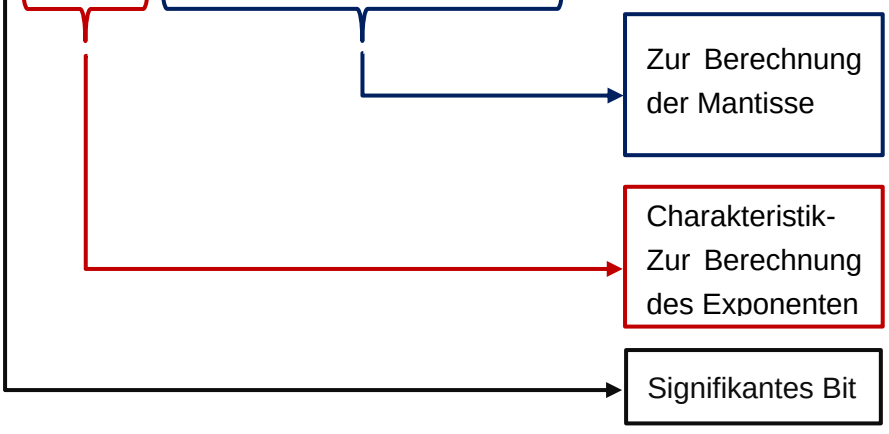
Da die im Basistyp 3 bereitgestellten Messwerte ausschließlich im Ganzzahlenformat vorliegen sind diese auf Basis des IEEE-754-Standards in Dezimaldarstellung umzurechnen. Über die Netzwerkschnittstelle werden mittels des Skripts 4 Bytes, welche den Messwert beinhalten, ausgelesen. Dies liefert vier Integer-Werte, also Werte im Ganzzahlenformat, welche in den zugehörigen Binärwert umzurechnen sind. Für das Beispiel anhand des Integerwertes 64 ergibt sich nachfolgende Umrechnung:

$$64 = \mathbf{0} \cdot 128 + \mathbf{1} \cdot 64 + \mathbf{0} \cdot 32 + \mathbf{0} \cdot 16 + \mathbf{0} \cdot 8 + \mathbf{0} \cdot 4 + \mathbf{0} \cdot 2 + \mathbf{0} \cdot 1 \quad \textbf{(1)}$$

Da der Messwert, welcher über das Netzwerk ausgelesen wird, aus vier Bytes besteht, erhält man demnach viermal 8 Bit. Als nächsten Schritt sind diese vier Zahlen in Bitdarstellung umzurechnen (vgl. hierzu Tabelle 2 sowie Formel (1)) und anschließend aneinanderzureihen. Diese Aneinanderreihung der umgerechneten Werte ergibt somit eine 32 Bit lange Sequenz (vgl. hierzu Tabelle 2). Bei der ersten Stelle dieser Bit-Sequenz handelt es sich um das sogenannte signifikante Bit, oftmals auch nur als „sign-bit“ bezeichnet. Selbiges gibt Auskunft, ob es sich um eine positive oder negative Dezimalzahl handelt. Wenn das erste Bit, also das signifikante Bit, Null ist bedeutet dies, dass es sich um eine positive Gleitkommazahl handelt, beim Wert Eins handelt es sich um eine negative Gleitkommazahl. Im Beispiel in Tabelle 2 ist das signifikante Bit null, was einer positiven Zahl entspricht, vgl. hierzu auch Tabelle 3.

Tabelle 2: Umrechnung gemäß IEEE-754 Standard – Teil 1

Byte	Integer Darstellung	Binär Darstellung
1	64	01000000
2	156	10011100
3	204	11001100
4	204	11001100
Zusammengesetzte Sequenz mit 32 Bit Länge		
01000000100111001100110011001100		



Die nächsten 8 Stellen bilden die Grundlage zur Berechnung des Exponenten, diese Sequenz wird oftmals auch Charakteristik genannt. Die verbleibenden 23 Stellen sind in weiterer Folge zur Berechnung der sogenannten Mantisse notwendig.

Tabelle 3: Umrechnung gemäß IEEE-754 Standard – Teil 2

	Signifikantes Bit	**Charakteristik**	**Zur Berechnung der Mantisse**
	0	**10000001**	**00111001100110011001100**
Interpretation	Positive Gleitkommazahl	Umgerechneter Wert entspricht 129	Wird der Mantisse, beginnend mit 1, angehängt

Der Exponent/Charakteristik errechnet sich, indem man die 8 Bits wertemäßig binär in eine Ganzzahl umrechnet. Unter Zuhilfenahme des Prinzips aus (1) ergibt sich für die Charakteristik 10000001:

$$\mathbf{1} \cdot 128 + \mathbf{0} \cdot 64 + \mathbf{0} \cdot 32 + \mathbf{0} \cdot 16 + \mathbf{0} \cdot 8 + \mathbf{0} \cdot 4 + \mathbf{0} \cdot 2 + \mathbf{1} \cdot 1 = 129 \quad \textbf{(2)}$$

Von diesem Wert wird der fix vorgegebene Wert von 127 subtrahiert, vgl. hierzu auch Tabelle 4. Das Ergebnis des Exponenten beschreibt, wie weit das Komma in der Mantisse nach rechts verschoben werden muss. Da in diesem Beispiel der Wert der Charakteristik 129 entspricht, ist das Ergebnis dieser Subtraktion der Wert 2, was einer Kommaverschiebung von zwei nach rechts entspricht (vgl. hierzu Tabelle 4). Nach Kenntnis der Verschiebung des Kommas ist die Mantisse, immer beginnend mit 1 gefolgt von einem Komma, zu notieren. Daran angehängt werden die letzten 23 Bits. Weiters ist das Komma entsprechend vorigem Schritt nach rechts zu verschieben, sodass sich der Wert "Mantisse Neu“ in Tabelle 4 ergibt.

Tabelle 4: Umrechnung gemäß IEEE-754 Standard – Teil 3

Rechengröße	**Berechnung/Interpretation**	**nächster Schritt**
Exponent	Wert Charakteristik - 127 = 129 - 127 = 2	Mantisse multipliziert mit zehn hoch dem Exponenten; gleichbedeutend mit der Kommaverschiebung in der Mantisse um 2 Stellen nach rechts
Mantisse	1,00111001100110011001100	Komma entsprechend des Exponenten verschieben
Mantisse Neu	100,11100110011001100110	Getrennte Betrachtung und Berechnung
Mantisse Neu_linker Teil	100 binär entspricht 4 in Ganzzahldarstellung	Gleitkommazahl links der Kommastelle
Mantisse Neu_rechter Teil	11100110011001100110 entspricht 0,899	Gleitkommazahl rechts der Kommastelle
Dezimaldarstellung der Zahl = 4,899		

Daraufhin kann die Binärfolge links sowie rechts vom Komma getrennt behandelt werden. Man beginnt mit der Binärfolge links vom Komma wonach sich folgende Beziehung ergibt:

$$\mathbf{0} \cdot 128 + \mathbf{0} \cdot 64 + \mathbf{0} \cdot 32 + \mathbf{0} \cdot 16 + \mathbf{0} \cdot 8 + \mathbf{1} \cdot 4 + \mathbf{0} \cdot 2 + \mathbf{0} \cdot 1 = 4 \quad \textbf{(3)}$$

Das hier beispielhafte Ergebnis 4 entspricht dem Wert vor dem Komma. Die Binärfolge rechts der Kommastelle errechnet sich, indem man wie folgt vorgeht (vgl. hierzu auch Formel (4).

$$11100110011001100 = \sum_{n=1}^{m} b_n \cdot 2^{-n} = \quad \textbf{(4)}$$

$$2^{-1} + 2^{-2} + 2^{-3} + 2^{-6} + 2^{-7} + 2^{-10} + 2^{-11} + 2^{-14} + 2^{-15} + 2^{-18} + 2^{-19} = 0{,}899$$

mit

m = Anzahl der Bits der Mantisse (anhand des gewählten Beispiels: m = 21)

b_n = n-tes Bit der Mantisse

Im letzten Schritt werden beide Werte (links sowie rechts der Kommastelle), die im vorherigen Schritt getrennt wurden, wertemäßig addiert, was sogleich die gesuchte Zahl ergibt.

Mit Hilfe dieser, im Skript implementierten Umrechnung wird eine Anzeige der Phasenströme der höchstbelasteten Phase in Dezimaldarstellung ermöglicht, was u. a. für den in weiterer Folge geplanten Laborbetrieb erforderlich ist.

3.5.2 Open-Source Ethernet Kommunikationsmodul Snap7

Um eine Kommunikation zwischen Python sowie einer SPS der Siemens S7-Produktfamilie zu ermöglichen, wird das Open-Source- und multiplattformbasierte Ethernet Kommunikationsmodul namens „SNAP7“ [26] verwendet. Dieser sogenannte Wrapper[13] beinhaltet notwendige Komponenten, um einen SPS-Zugriff mittels der Programmiersprache Python zu ermöglichen. Er beinhaltet einen Client der auf einen Server, in diesem Fall die physische Siemens SPS S7-1211, zugreift. Auch ein virtueller Server, der eine SPS mit deren Ein- sowie Ausgängen simuliert, ist möglich. Da in diesem Projekt jedoch eine physische SPS verbaut wurde, stellt diese zugleich den Server des Wrappers dar und es erfordert an dieser Stelle keine Implementierung eines virtuellen Servers.

Um den Wrapper in das jeweilige Computersystem zu integrieren ist eine *.DLL[14]- sowie eine *.LIB[15]-Datei direkt in das Verzeichnis des Betriebssystems zu hinterlegen.

Als Nächstes muss ein entsprechendes Modul mittels dem Kommandozeilenbefehl „pip install python-snap7“ im Terminal der Python Entwicklungsumgebung PyCharm installiert werden. Nach erfolgreichem aufsetzten des Wrappers und Integration in die Entwicklungsumgebung kann dieser schließlich verwendet werden und es kann mit der eigentlichen Programmierung begonnen werden. Das gesamte entwickelte Skript ist dem Anhang zu entnehmen (9.2).

[13] oft auch Adapter bzw. Schnittstelle genannt; ist ein Teil einer Software, welcher eine andere Software umgibt

[14] DLL - Dynamic Link Library; de. dynamische Programmbibliothek

[15] LIB - Library Datei; für statische und/oder dynamische Bibliotheken

3.6 Versuchsreihe

Die dokumentierten Versuche sollen grundsätzlich Aufschluss darüber geben, inwiefern der Labordemonstrator künftig im Laborbetrieb eingesetzt werden kann und zudem, wie der entwickelte Code den/der Benutzer*in bei verschiedensten Tests hinsichtlich der temporären Rekonfiguration unterstützen kann. Das Python-Skript ermöglicht es hierbei, zwischen den in diesem Unterkapitel näher erläuterten Szenarien vollständig und automatisiert umzuschalten. Zum Zwecke der Übersichtlichkeit sind im Zuge dieses Dokuments nur fünf der 16 Möglichkeiten näher erläutert, die weiteren Varianten können jedoch mit Hilfe des Python-Skripts ebenso geschaltet sowie ausgewertet werden. Das programmierte Skript erkennt sowohl Szenarienumschaltungen als auch ausgelöste Kompaktleistungsschalter. Mit dem automatisiert für jeden Schaltvorgang erstellen Ausgabeordner verfügt der/die Benutzer*in zudem über zahlreiche grafische Interpretationen sowie Auswertungen des jeweiligen Schaltszenarios.

Um den Ansteuerungs- sowie Analysecode zu testen, wird eine zugehörige Messdatenreihe ermittelt. Alle nachfolgenden Abbildungen stammen direkt aus Python und werden vom Skript automatisiert in einem vordefinierten Ausgabeordner erstellt. Tabelle 5 bietet eine Übersicht der eingestellten Parameter der Leistungsschalter, Tabelle 6 zeigt die Einstellbereiche der verwendeten Siemens 3VA Kompaktleistungsschalter.

Tabelle 5: Eingestellte Parameter der Kompaktleistungsschalter der 3VA- Reihe

Leistungsschalter	**Auslösestrom I_r in A**	**Auslösezeit t_r in s**
LS 1	25	0,5
LS 2	25	0,5
LS 3	13	1
LS 4	15	0,5

Tabelle 6: Einstellbereiche der Komapaktleistungsschalter der 3VA- Reihe

	Auslösestrom I_r in A	**Auslösezeit t_r in s**
Minimum	10	0,5
Maximum	25	25
Vordefinierte Schrittweite	0,5	0,1

Um Übersichtlichkeit zu gewährleisten und Missverständnissen vorzubeugen, werden nachfolgend verschiedene Szenarien definiert und mittels Abbildungen verdeutlicht. Der vierstellige Binärwert repräsentiert die jeweilige Kompaktleistungsschalterposition des Szenarios und wird vom Python-Skript weiterverarbeitet. Dieser ist jeweils von rechts nach links, entsprechend der Binärschreibweise, zu interpretieren. Die äußerst rechte Stelle der vierstelligen Binärdarstellung steht hierbei also für den Kompaktleistungsschalter LS 1, die zweite Stelle von rechts für den Leistungsschalter LS 2 usw. Ein Binärwert von Null, entspricht einem geöffneten Kompaktleistungsschalter, ein Binärwert von Eins einem geschlossenen Kompaktleistungsschaler. Es gibt aufgrund der vier Kompaktleistungsschalter maximal $2^4 = 16$ verschiedene Schaltervariationen, wobei hier jedoch nicht auf alle näher eingegangen wird. Zudem werden jene Schaltpositionen, bei denen sich Kompaktleistungsschalter LS 1 in geöffneter Position befindet, hier nicht behandelt. Die eingezeichneten Lastabgänge in den nachfolgenden Abbildungen der Szenarien werden verwendet, um Lasten in das System zu integrieren. An der Stelle LS 2-2 im oberen Zweig des Systems wurden drei handelsübliche Heizlüfter, an der Stelle LS 3-1 im unteren Zweig des Systems drei große, am Institut für Elektrische Anlagen und Netze der Technischen Universität Graz im Zuge eines Vorprojekts eigens angefertigte Heizlüfter-Kaskaden eingebracht.

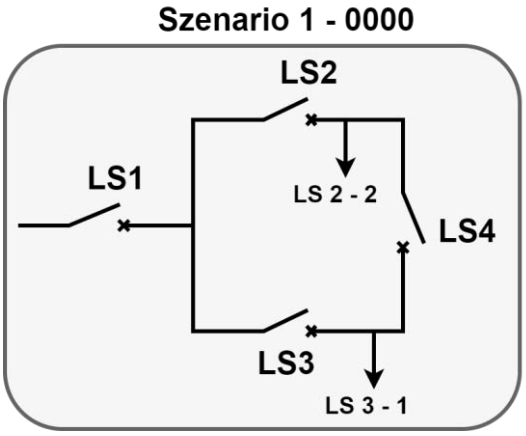


Abbildung 32: Konfiguration Szenario 1

In Abbildung 32 ist das Schema des Laboraufbaus (für den Fall, dass alle vier Kompaktleistungsschalter in der offenen Stellung betrieben werden) zu erkennen. Diese Schalterstellung entspricht Szenario 1.

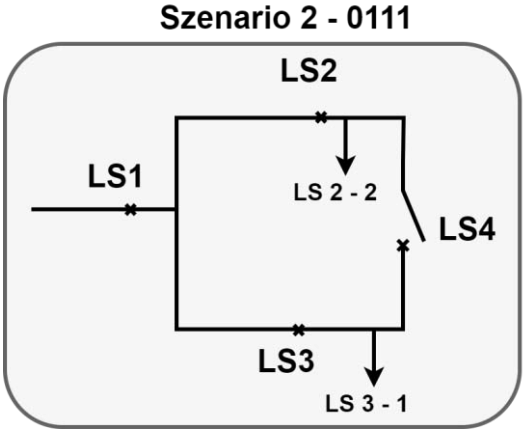


Abbildung 33: Konfiguration Szenario 2

Abbildung 33 zeigt den Aufbau in Szenario 2. Hierbei sind alle Komapktleistungsschalter, mit Ausnahme des vierten Kompaktleistungsschalters-LS4, geschlossen. Dieses Szenario soll zwei kürzere Stichleitungen, oben sowie unten, repräsentieren.

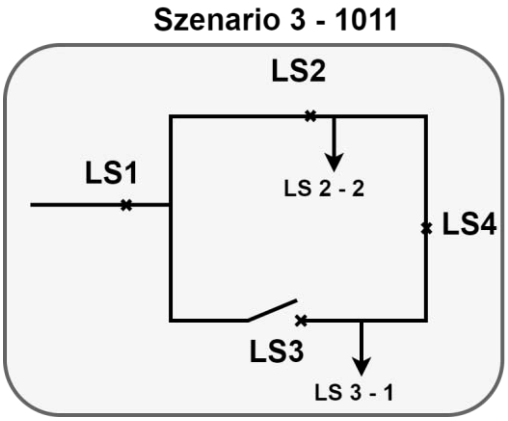


Abbildung 34: Konfiguration Szenario 3

Szenario 3 zeigt den Laboraufbau, wobei hier alle Komapktleistungsschalter, mit Ausnahme des dritten Komapktleistungsschalters-LS3, geschlossen sind. Dieses Szenario soll eine lange Stichleitung, beginnend mit Versorgung des Lastabgangs LS 2-2, gefolgt vom Lastabgang LS 3-1, repräsentieren und ist in nebenstehender Abbildung 34 dargestellt.

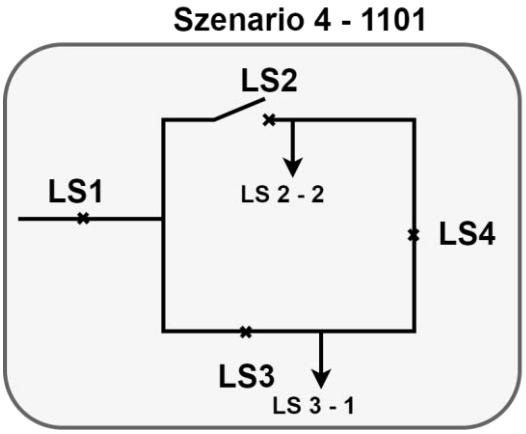


Abbildung 35: Konfiguration Szenario 4

Szenario 4 (Abbildung 35) zeigt den Laboraufbau, wobei hier alle Komapktleistungsschalter, mit Ausnahme des zweiten Kompaktleistungsschalters-LS2, geschlossen sind. Dieses Szenario soll eine lange Stichleitung, beginnend mit Versorgung des Lastabgangs LS 3-1, gefolgt vom Lastabgang LS 2-2, repräsentieren.

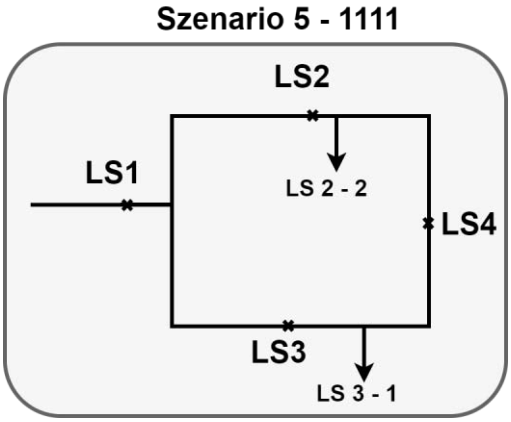


Abbildung 36: Konfiguration Szenario 5

Szenario 5 zeigt den Laboraufbau, wobei hier alle Komapktleistungsschalter geschlossen sind. Dieses Szenario stellt sogleich ein vollständig vermaschtes System für die Versuchsreihe dar und ist in nebenstehender Abbildung 36 dargestellt.

Anhand dieser Szenarien ist eine Versuchsreihe durchgeführt und die entsprechenden Messwerte ausgewertet worden, um den entwickelten Code entsprechend zu testen und dessen Funktionalität verifizieren zu können. In Abbildung 37 bis inklusive Abbildung 43 sind Unterschiede zwischen den einzelnen Szenarien festzustellen. Diese Versuchsreihe soll den

Umfang des Skripts verständlich aufbereiten und zudem auch zeigen, wie der zukünftige Laborbetrieb mit dem Skript unterstützt werden kann.

Alle nachfolgenden Abbildungen sind direkt mit Hilfe des entwickelten Python-Skripts geplottet bzw. automatisiert erstellt und sollen dem/der Benutzer*in eine zusätzliche übersichtliche Darstellungsweise bieten. Neben zahlreichen Diagrammen, die vom Skript generiert werden, wird auch ein Protokoll der jeweiligen Schalthandlung samt Messdaten sowie mehrere Informationsdateien, welche Auskunft über die im Zuge der Code-Iteration erstellten einzelnen Abbildungen vermitteln. Dies geschieht zusätzlich zu den Ansteuerungsbefehlen der Kompaktleistungsschalter, welche dann das Schalten dieser veranlassen.

Abbildung 37 zeigt den maximalen Phasenstrom der höchstbelasteten Phase je Komapaktleistungsschalter vom Übergang des Schaltzustandes in Szenario 1 auf den Schaltzustand in Szenario 2. In selbigem werden mit Hilfe des Labordemonstrators zwei kurze Stichleitungen simuliert, das bedeutet, dass Leistungsschalter LS 4 hierbei geöffnet bleibt.

Es lässt sich erkennen, dass alle Komapaktleistungsschalter mit Ausnahme des geöffneten LS 4 einen Strom führen. Der Strom der höchstbelasteten Phase, welcher am ersten Kompaktleistungsschalter nach dem Schaltvorgang gemessen wird, beträgt 13,3 A. An Leistungsschalter LS 2 lassen sich 5,2 A messtechnisch erfassen, an Leistungsschalter LS 3 8,3 A. Leistungsschalter LS 4 führt, aufgrund des geöffneten Zustandes, keinen Strom. Diese Werte wurden allesamt mittels des zuvor in Kapitel 3.5.1 beschriebenen Umrechnungsalgorithmus in, für den Menschen interpretierbare, Werte konvertiert.

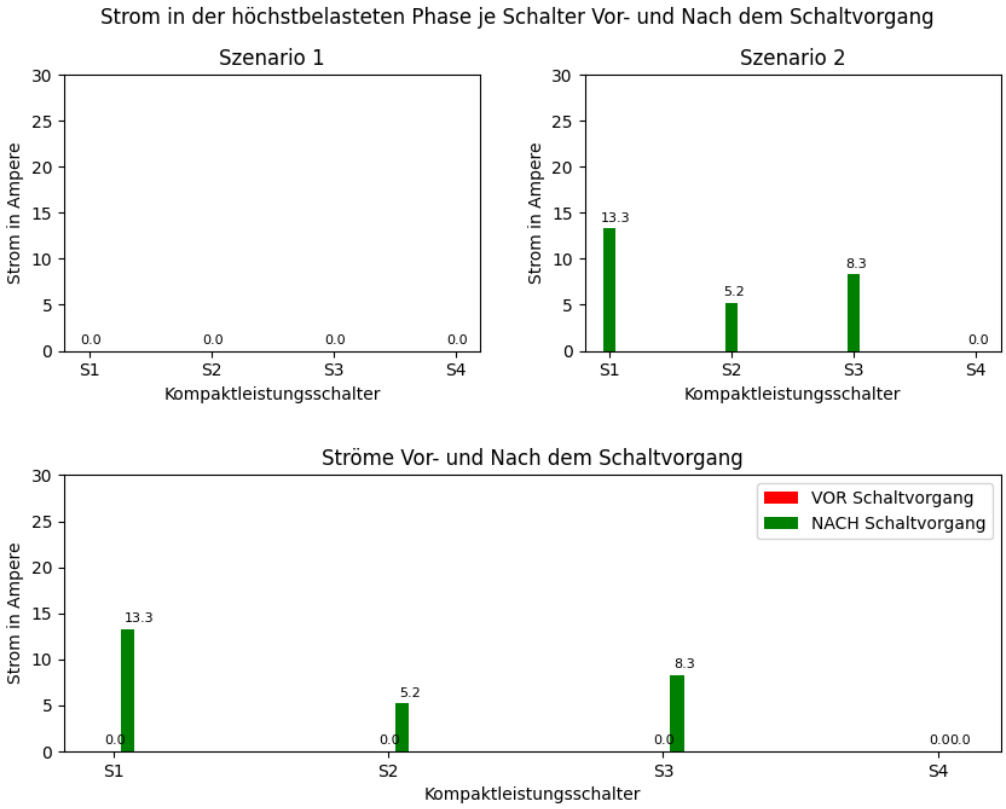


Abbildung 37: Übersichtsgrafik – Umschaltung Szenario 1 auf Szenario 2

Abbildung 38 zeigt den maximalen Phasenstrom der höchstbelasteten Phase je Komapaktleistungsschalter vom Übergang des Schaltzustandes in Szenario 1 auf den Schaltzustand in Szenario 3. Da im Szenario 1 alle Komapaktleistungsschalter geöffnet sind, fließt hier anfangs kein Strom. Szenario 3 soll eine lange Stichleitung, beginnend im oberen Abschnitt des Laboraufbaus simulieren. Der Kompaktleistungsschalter LS 3 ist hierbei geöffnet.

Auch hier ist erkennbar, dass alle Komapaktleistungsschalter mit Ausnahme des geöffneten Kompaktleistungsschalters LS 3 einen Strom führen. In das System fließen 13.4 A in der höchstbelasteten Phase. Da beim Schaltvorgang auf den Schaltzustand in Szenario 3 der Leistungsschalter LS 3 (welcher sich im unteren Abschnitt des Aufbaus befindet) geöffnet ist, fließt hier der gesamte Strom, welcher über Kompaktleistungsschalter LS 1 fließt, auch über LS 2 (welcher sich im Aufbau im oberen Bereich befindet). An LS 4 wird der Strom der höchstbelasteten Phase mit rund 8,3 A gemessen.

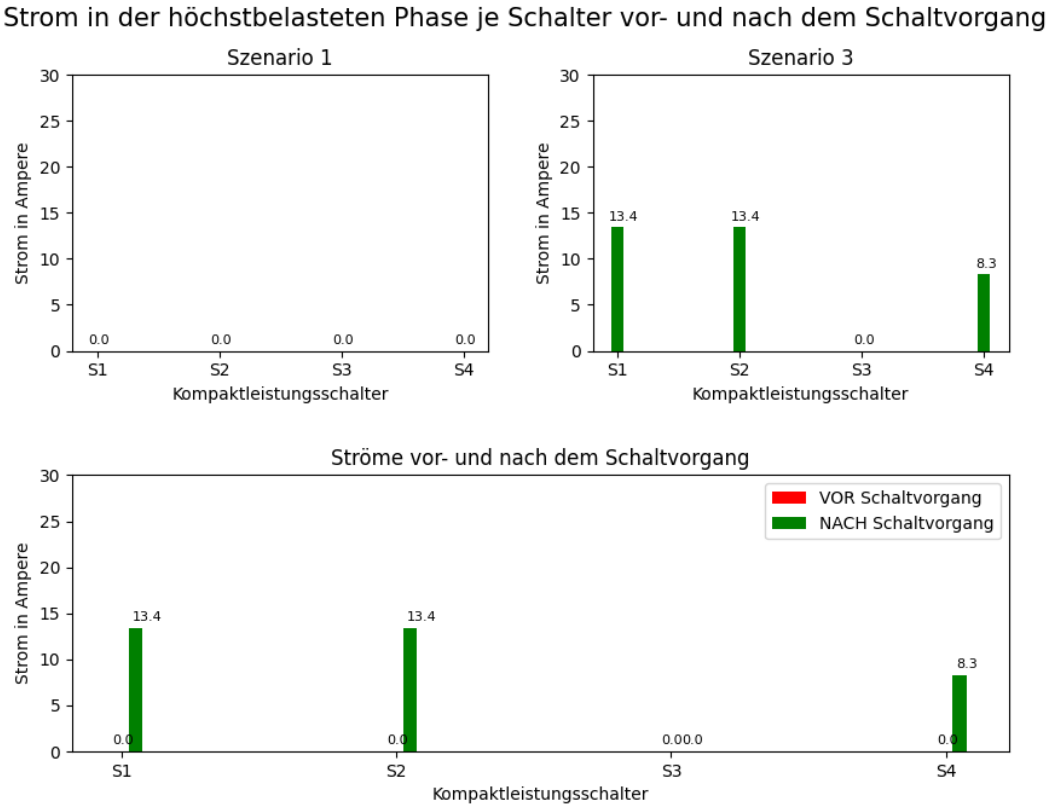


Abbildung 38: Übersichtsgrafik – Umschaltung Szenario 1 auf Szenario 3

Abbildung 39 zeigt den maximalen Phasenstrom der höchstbelasteten Phase je Komapaktleistungsschalter vom Übergang des Schaltzustandes Szenario 1 auf den Schaltzustand in Szenario 5. Auch hier fließt anfangs in Szenario 1 kein Strom. In Szenario 5 wird die vollständige Vermaschung des Laboraufbaus simuliert. Hier lässt sich erkennen, dass nun jeder der Komapaktleistungsschalter einen Strom führt. In das System fließen 13.3 A in der höchstbelasteten Phase. An Kompaktleistungsschalter LS 2 ist ein maximaler Phasenstrom von 8,3 A und am Kompaktleistungsschalter LS 3 ein maximaler Phasenstrom von 6,9 A zu verzeichnen. Über Kompaktleistungsschalter LS 4 fließt ein Ausgleichsstrom in der Höhe von 3,5 A in der höchstbelasteten Phase.

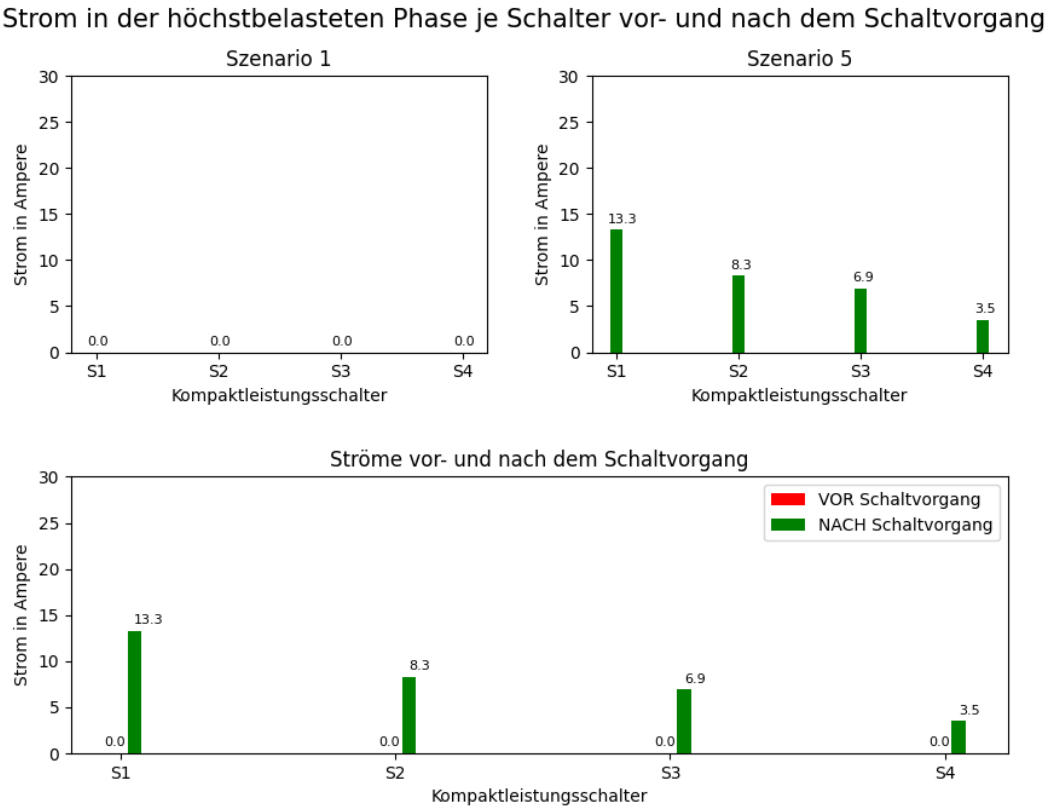


Abbildung 39: Übersichtsgrafik – Umschaltung Szenario 1 auf Szenario 5

Die vorangegangenen Zustandsänderungen gehen immer von einem Ausgangszustand aus, bei dem alle Kompaktleistungsschalter geöffnet waren. Die weiteren Betrachtungen wenden sich hingegen jenen Anwendungsfällen zu, bei denen zwischen den einzelnen Szenarien (ausgenommen Szenario 1) umgeschaltet wird. Dies simuliert zugleich den laufenden Betrieb, wie er auch in der Praxis vorkommen würde.

Als Erstes wird die Schaltstellungsänderung von Szenario 2 auf Szenario 3 genauer betrachtet, siehe dazu Abbildung 40.

Man erkennt durch den Strom von 0 A, dass in Szenario 2 der Leistungsschalter LS 4 geöffnet ist. Vor der Umschaltung besteht das System aus einer Konfiguration aus zwei Stichleitungen. Hier ist zu beobachten, dass sich der Strom (wie in Abbildung 40 ersichtlich) entsprechend dem Lastfluss erwartungsgemäß aufteilt. Nach dem Umschalten auf Szenario 3 ist das System nun als eine lange Stichleitung konfiguriert, beginnend mit der Versorgung der am oberen Abgang befindlichen Lasten und der Versorgung der am unteren Abgang angeschlossenen Lasten. Auch hier ist ersichtlich, dass der Kompaktleistungsschalter LS 3 geöffnet ist, da nun dieser keinen Strom führt. Der in das System fließende Strom teilt sich wieder entsprechend des Lastflusses auf. Diesmal fließt über den Kompaktleistungsschalter LS 2 zugleich der volle Strom, der auch durch den Kompaktleistungsschalter LS 1 fließt. Der Strom der höchstbelasteten Phase, welcher über den Kompaktleistungsschalter LS 4 gemessen wird, ist mit rund 8,2 A zu verzeichnen.

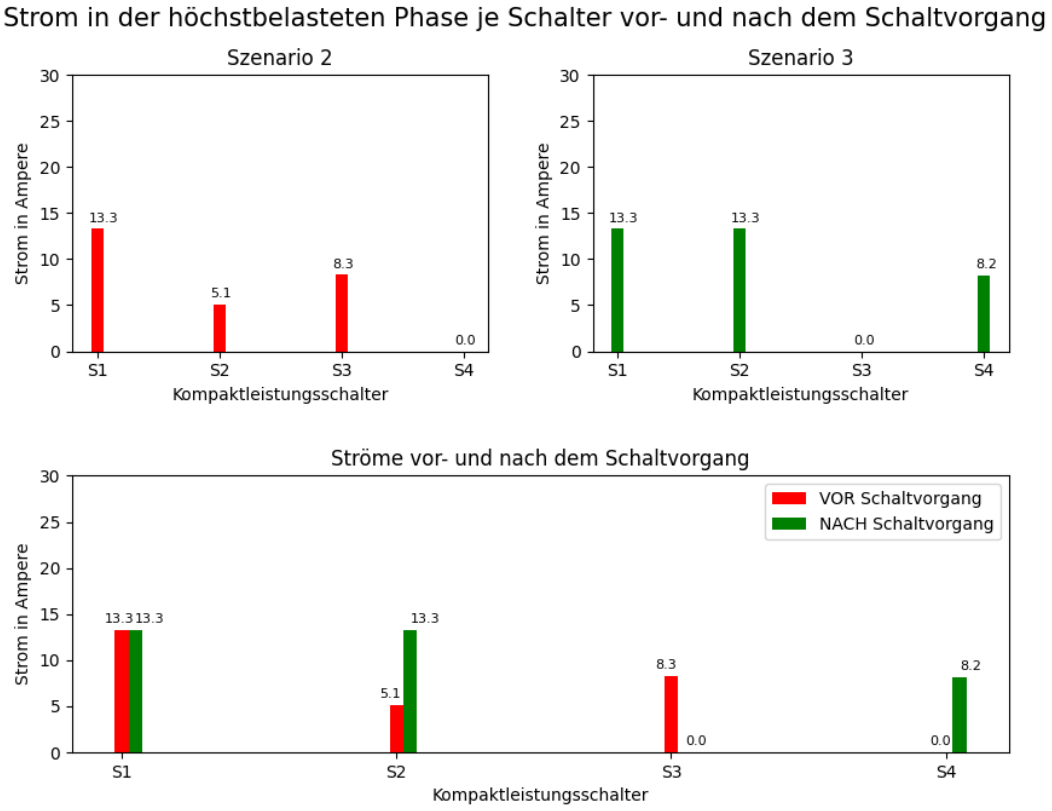


Abbildung 40: Übersichtsgrafik – Umschaltung Szenario 2 auf Szenario 3

In nachfolgender Betrachtung wird das vorangehende Beispiel (Szenario 2 auf Szenario 3) umgekehrt, demnach erfolgt eine Schalthandlung von Szenario 3 auf Szenario 2, siehe Abbildung 41. Das bedeutet, dass von einem langen Stich auf zwei kürzere Stichleitungen umgeschaltet wird. Wie zu erwarten ist, wird sich der zuvor beschriebene Fall umkehren. Trotzdem ist dieser Fall auch wert, untersuch zu werden, da ein unterschiedliches Auslöseverhalten der Kompaktleistungsschalter bei verschiedensten Kombinationen aus Szenarienwechseln auftreten kann.

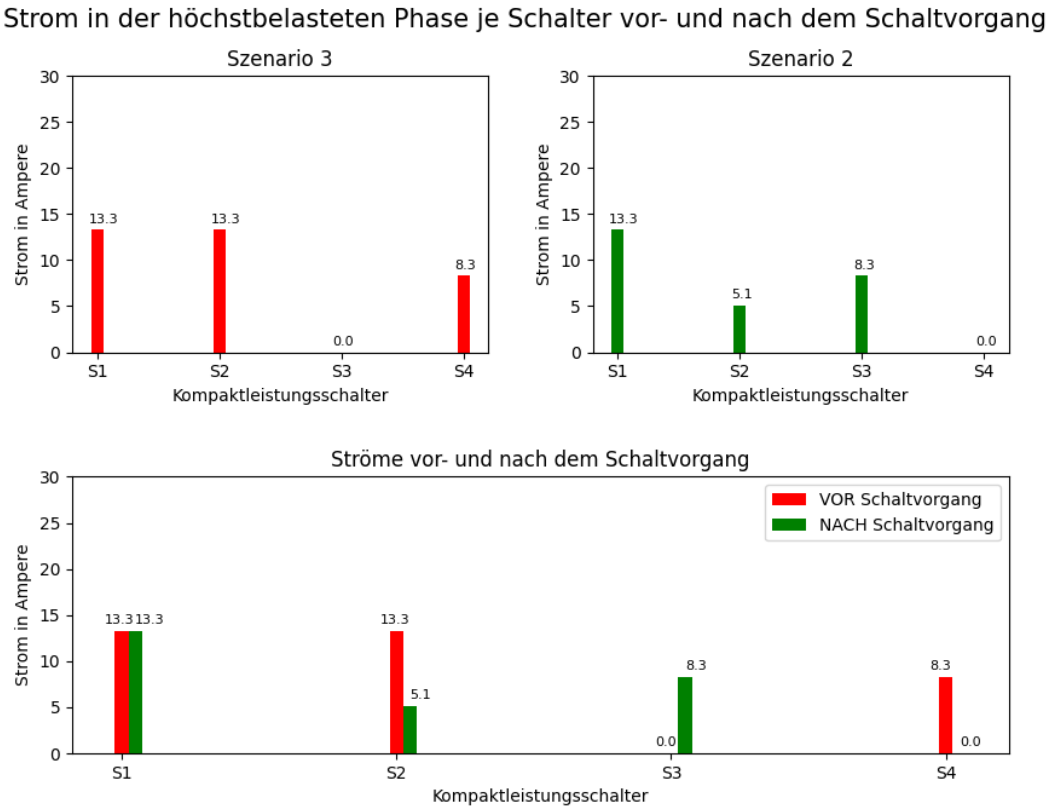


Abbildung 41: Übersichtsgrafik – Umschaltung Szenario 3 auf Szenario 2

In Abbildung 42 ist der Szenarienwechsel von Szenario 2 auf Szenario 5 dargestellt. Diese Umschaltung bildet zugleich den Wechsel eines Systems mit zwei kurzen Stichleitungen auf ein vollständig vermaschtes System ab.

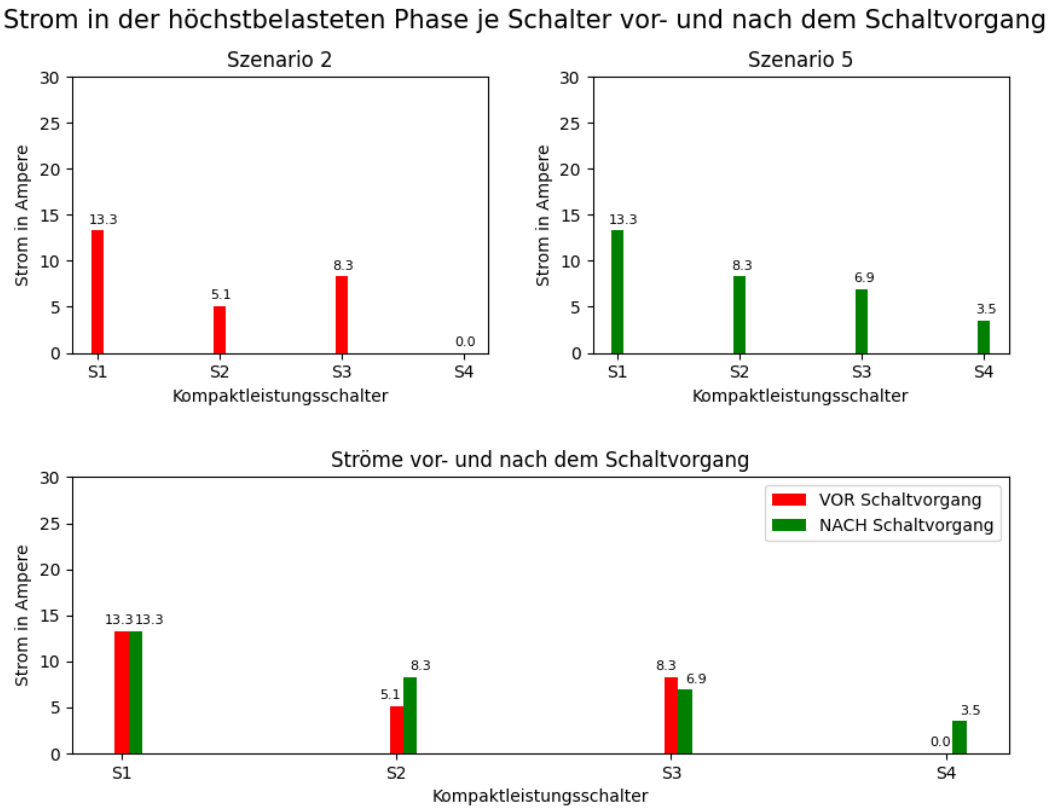


Abbildung 42: Übersichtsgrafik – Umschaltung Szenario 2 auf Szenario 5

In Abbildung 43 ist dabei dieser Fall (Szenario 2 auf Szenario 5) jeweils mit den einzelnen Phasenströmen zu sehen. Aufgrund der Speicherung der Daten stehen diese jeweils nur in einer Integer-Auflösung, d. h. ohne Dezimaldarstellung zur Verfügung. Man kann hier jedoch klar feststellen das sich die gemittelte Summe der drei Phasen der Leistungsschalter LS 2 und LS 3 zu dem Wert von Kompaktleistungsschalter LS 1 summieren.

Es ist zu erkennen, dass alle Werte der Messdaten nur in einer Ganzzahldarstellung verfügbar sind. Dies ist der Datenspeicherung bzw. der Bytereservierung dieser Messdaten geschuldet. Für die Messwerte der jeweiligen Phasenströme werden hier nur jeweils 2 Byte reserviert. Das erste Byte stellt hierbei eine Zählvariable dar, das zweite Byte eine Ganzzahl mit dem maximalen Wert von 255. Die Multiplikation der Zählvariable mit dem maximalen Wert von 255 ergibt folglich summiert mit dem Wert der zweiten Zählvariable (zweites reserviertes Byte) die entsprechenden, in Abbildung 43 ersichtlichen Werte. Aufgrund der gerundeten Werte ist hierbei zusätzlich mit einer Abweichung zu rechnen. Trotzdem ist ein deutlicher Unterschied im Vergleich zu Abbildung 42 zu erkennen. Vor dem Umschaltvorgang (Szenario 2) mit den beiden kürzeren Stichleitungen vermitteln die Messwerte, dass das System vollkommen symmetrisch ist. Dies lässt sich aufgrund der verwendeten Lasten (Heizlüfter) auch

entsprechend erwarten. Nach dem Umschalten in ein vermaschtes System (Szenario 5) ist diese Symmetrie nicht mehr zu erkennen. Wie bereits erwähnt kann es hier aufgrund der Ganzzahlendarstellung zu Abweichungen kommen, jedoch muss man feststellen, dass es hier zu einem unsymmetrischen Verhalten aufgrund minimaler Impedanzabweichungen im Aufbau des Labordemonstrators (einzelne Phasenleiter weisen bspw. aufgrund geringer Längenunterschiede oder unterschiedlicher Kontaktmomente abweichende Impedanzen auf). Dieses Verhalten wurde bereits in der vorangegangenen Arbeit [20] mittels einer Niederohmmessung verifiziert.

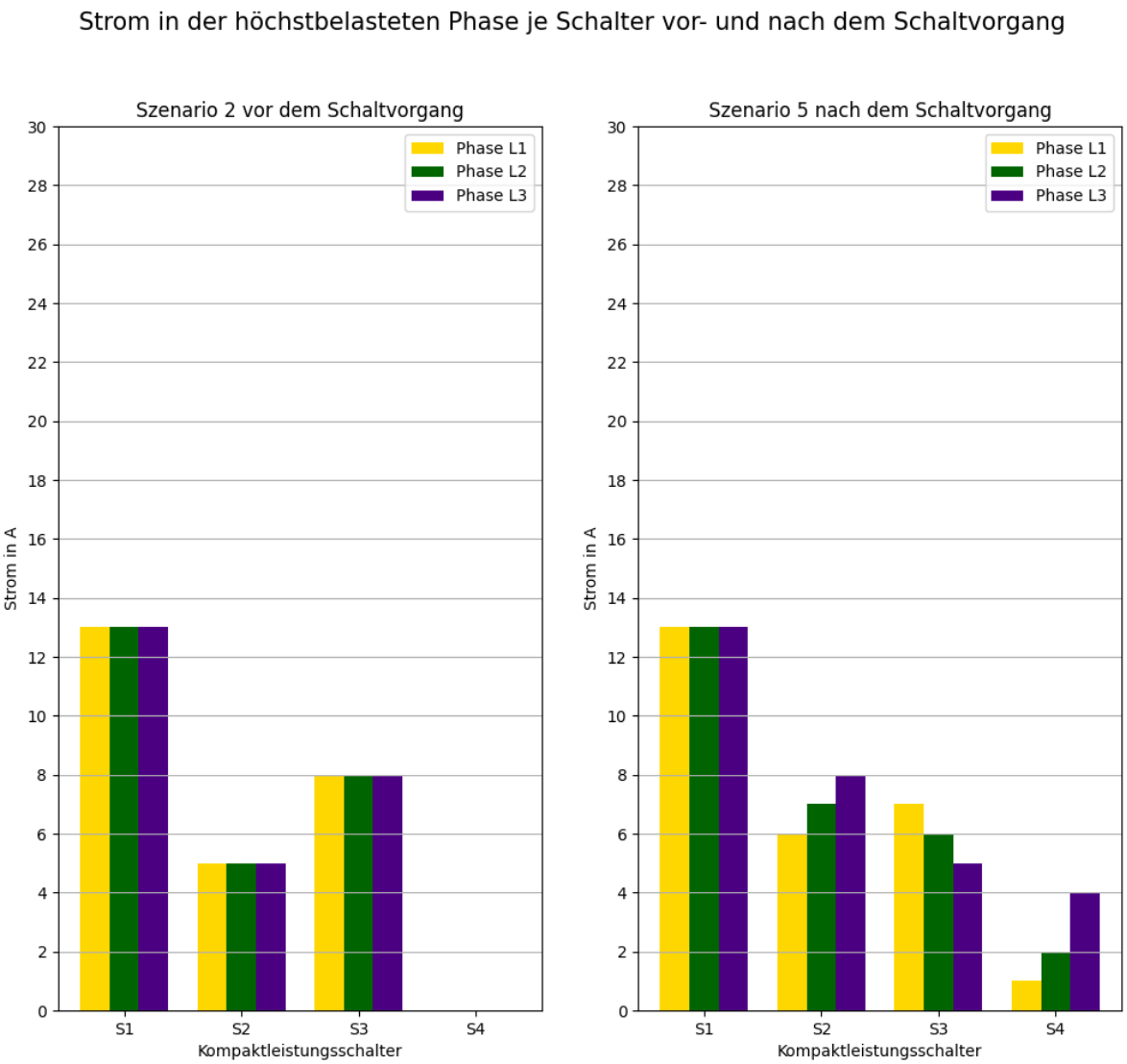


Abbildung 43: Übersichtsgrafik – Phasenströme einzeln; Umschaltung Szenario 2 auf Szenario 5

Abschließend sei noch jener Fall der Umschaltung von Szenario 1 auf Szenario 4 erwähnt, bei dem in der Versuchsreihe einer der Kompaktleistungsschalter auslöste. Dies ist gleichbedeutend mit einer Umschaltung vom Anfangszustand (indem alle Kompaktleistungsschalter geöffnet sind) hin zu einem Zustand, wo das System nur von einer langen Stichleitung, beginnend mit den unteren Lastabgängen, versorgt wird. Hierzu wird der im Aufbau oben befindliche Kompaktleistungsschalter LS 2 geöffnet. Um die am oberen Teil des Aufbaus befindlichen Lastabgänge LS 2-2 versorgen zu können, fließt der gesamte Strom über den Kompaktleistungsschalter LS 3. Wie in Tabelle 5 ersichtlich, handelt es sich hierbei zugleich um jenen Leistungsschalter, mit dem am niedrigsten eingestellten Auslösestrom von 13 A. Kompaktleistungsschalter LS 3 wird zur Auslösung (en. trip) gebracht, wenn der Strom über eine Zeitdauer von mehr als einer Sekunde über dem Wert von 13 A liegt.

Weiters sind auch noch Versuche mit einer Umschaltung von Szenario 2 auf Szenario 4 durchgeführt worden. Auch hier ist erwartungsgemäß beobachten worden, dass der Kompaktleistungsschalter LS 3 ausgelöst hat.

Aufgrund der Auslösung des Kompaktleistungsschalters sind hier keine Messwerte am betroffenen Schalter ermittelt worden, was sich auch in den entsprechend fehlenden Diagrammen widerspiegelt. Dem/der Benutzer*in wird dieser Auslöseprozess jedoch mit Hilfe des Visualisierungsskript, deutlich und entsprechend grafisch am Bildschirm angezeigt. Zudem wird der/die Benutzer*in darüber in Kenntnis gesetzt, wie eine Resetierung des ausgelösten Leistungsschalters durchzuführen ist.

3.7 Netzmodell in DIgSILENT PowerFactory

Im Zuge der Arbeit ist ein Abbild des realen Aufbaus des Labordemonstrators in der Netzberechnungssoftware DIgSILENT PowerFactory erstellt sowie implementiert worden. Dies soll das Auswerten des bestmöglichen Lastflusses für die entsprechende Netztopologie mit Hilfe eines, im Zuge von [24] entwickelten Python-Skripts, ermöglichen. Die Berechnungsergebnisse dieses Skripts sind zudem der Ausgangspunkt der Schalthandlungen, welche vollzogen werden sollen. Siehe dazu auch Kapitel 3.6.

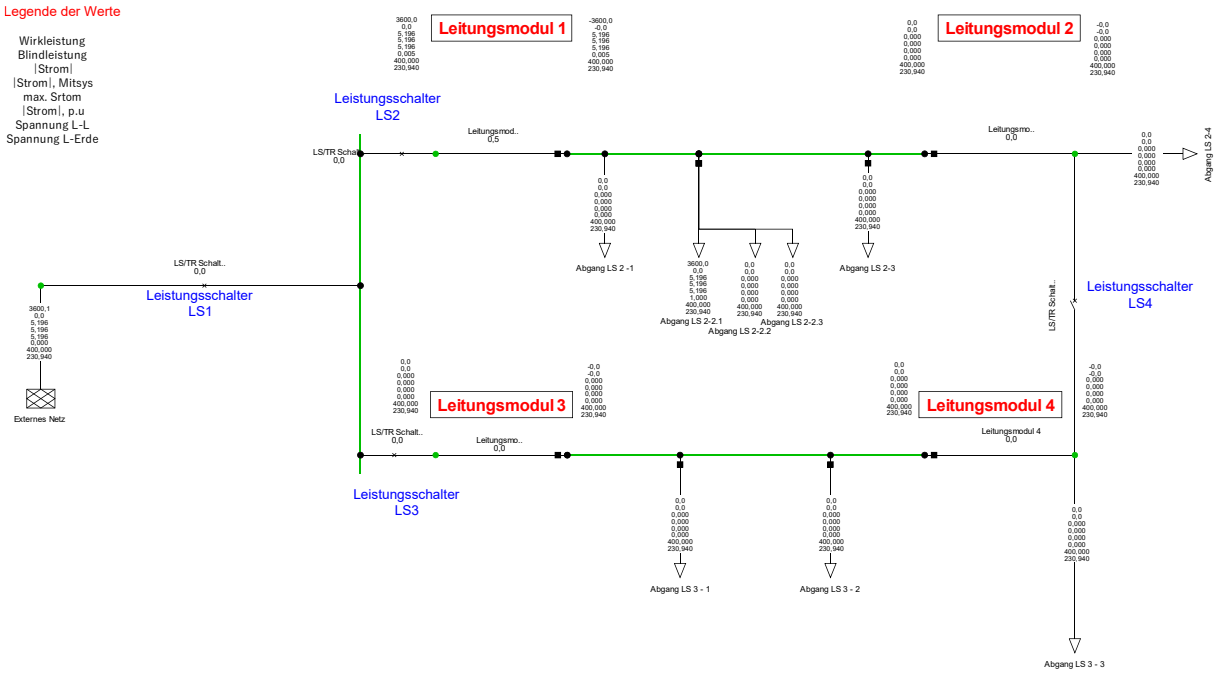


Abbildung 44: DIgSILENT PowerFactory Netzmodell des realen Aufbaus des Labordemonstrators, hier beispielhaft für Szenario 2

Das Modell ist erstellt, parametriert und in Betrieb genommen worden, weitere Tests wurden jedoch nicht mehr im Rahmen der vorliegenden Arbeit durchgeführt und sind daher auch nicht Teil dieses Dokuments.

4 Erweiterung des Laboraufbaus

Um größtmögliche Flexibilität hinsichtlich der Testmöglichkeiten zu gewährleisten, sind im Zuge der Entwicklung vier Erweiterungsmöglichkeiten für Leitungsmodule im Labordemonstrator vorgesehen worden [20], siehe auch Abbildung 19. Mittels sogenannter HEAVYCON-Modular Steckverbinder (Niederspannungs-Leistungssteckverbinder) an der Seite des Laboraufbaus können bis zu vier Leitungsmodule in den bestehenden Aufbau integriert werden. Dies ermöglicht das zusätzliche Simulieren von entsprechenden Kabellängen sowie Nennquerschnitten, um so noch realitätsgetreuere Ergebnisse zu erlangen. Ohne Erweiterung um entsprechende Leitungsmodule befinden sich lediglich vier Leistungsschalter ohne nennenswerte Kabellängen zwischen selbigen in einer angeordneten Ringstruktur. Um dafür eine praxisnahe Lösung zu ermöglichen, kann das Set-up um entsprechende Module in Serie erweitert werden. Nachfolgend sind dahingehend verschiedenste Nennquerschnitte sowie Kabellängen in Form einer Machbarkeitsstudie zusammengestellt. Diese beinhaltet die entsprechend notwendigen Berechnungen sowie eine detaillierte Kostenaufstellung. In Abbildung 45 ist die Seitenansicht des Laboraufbaus mit den dafür vorgesehenen Erweiterungssteckverbindern dargestellt.

Abbildung 45: Seitenansicht Labordemonstrator mit Erweiterungssteckverbinder zur seriellen Einbindung von Leitungsmodulen (rote Kreise)

4.1 Allgemeines

Leitungsmodelle sind eine bewährte Methode, um Leitungselemente nachbilden zu können. Grundsätzlich kann dabei zwischen einem T- sowie einem π-Ersatzschaltbild unterschieden werden (siehe Abbildung 46), wobei für die nachfolgenden Betrachtungen zweiteres herangezogen wurde. Grundsätzlich ist das π-Modell in den meisten Fällen die bevorzugte Variante, da dieses Ersatzschaltbild im Vergleich zum T-Modell, über keinen inneren Knoten verfügt.

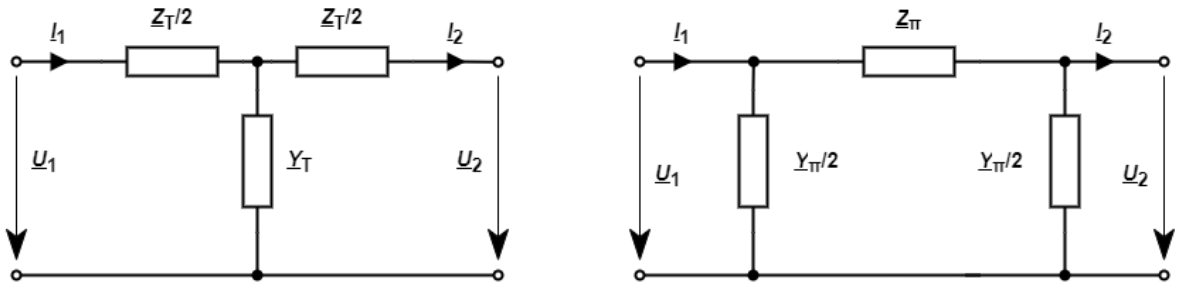


Abbildung 46: Vereinfachte Darstellung der Leitungsersatzschaltbilder – links T, rechts π

Bei einer Frequenz von 50 Hz sind Leitungen (Kabel) bis zu einer Länge von 50 km als elektrisch kurze Leitungen anzusehen und können nach Abbildung 47 modelliert werden. Der größte Teil des engvermaschten europäischen Verbundnetzes besteht aus elektrisch kurzen Leitungen (Freileitung < 250 km, Kabel < 50 km), weshalb diese Annahme untermauert werden kann. [11, 12, 27]

Unter der Annahme, dass es sich um eine elektrisch kurze Leitung handelt, können nachfolgende Beziehungen verwendet werden [12]:

$$\underline{Z}_\pi = (R' + j\omega\, L') \cdot l = R + j\omega\, L \tag{5}$$

$$\underline{Y}_\pi = (G' + j\omega\, C') \cdot l = G + j\omega\, C \tag{6}$$

Die beiden Gleichungen (5) und (6) für eine elektrisch kurze Leitung gelten sowohl für das T- als auch für das π-Ersatzschaltbild. Unter Verwendung dieser lässt sich das π-Ersatzschaltbild detaillierter darstellen, siehe dazu Abbildung 47.

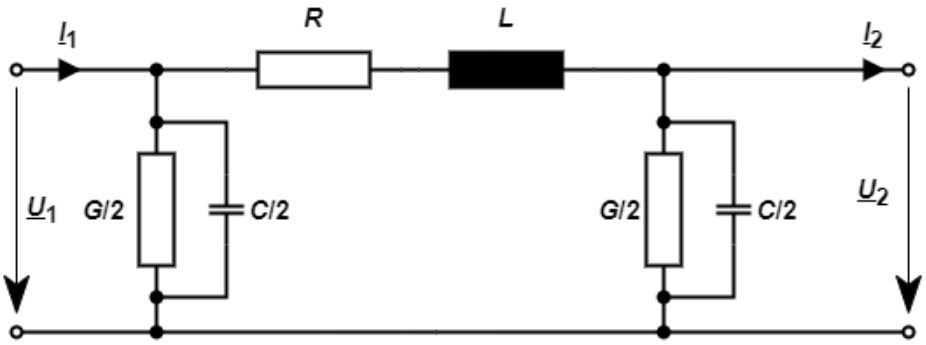


Abbildung 47: Detaildarstellung π-Ersatzschaltbild

4.2 Auslegung und Berechnungen

Da auf niederspannungsebene Kabellängen von unter 50 km üblich sind, kann eine Modellierung nach vorangegangenen Betrachtungen angestrebt werden. Für diese wird ein, in der Praxis weit verbreitetes, 4-Leiter-Niederspannungskabel des Typs NYY-O (Leitermaterial Kupfer, Bemessungsspannung 0,6/1 kV) herangezogen, siehe beispielhaftes Symbolbild in Abbildung 48.

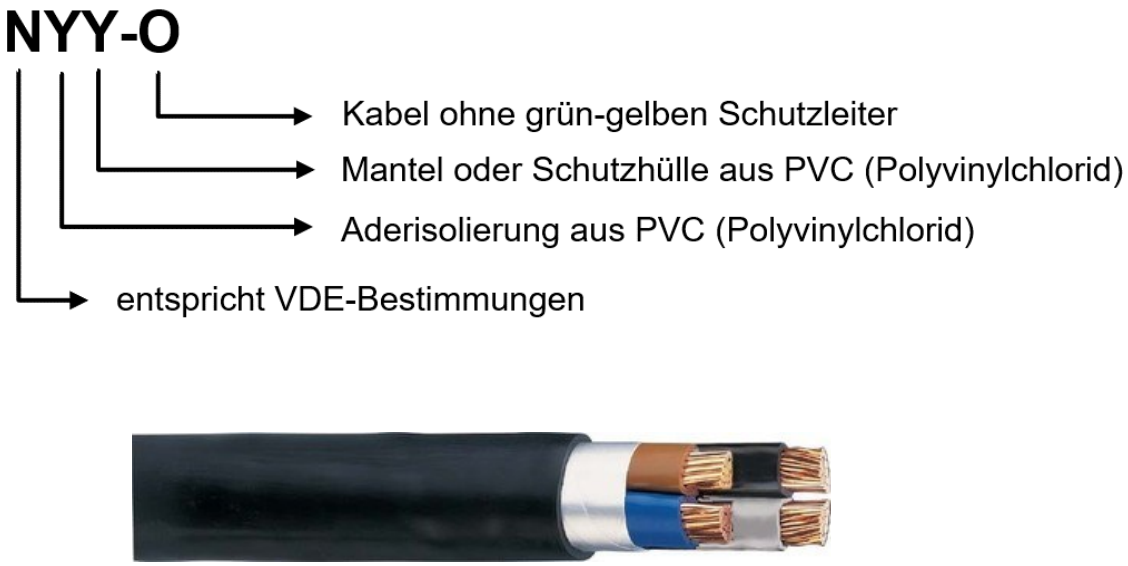


Abbildung 48: Symbolbild eines 4-Leiter-Niederspannungskabel des Typs NYY-O [28]

Im Zuge der Berechnungen werden sowohl die Mit, Gegen- und Nullsystemkomponente ermittelt. Für den Fall der Rückleitung nur über den Neutralleiter (vierter Leiter) beträgt die Nullsystemkomponente jeweils das Vierfache des Wertes der Mit- bzw. Gegensystemkomponente [27], siehe dazu auch nachfolgende Abbildung 49.

$$R^0 = 4 \cdot R^1 \tag{7}$$

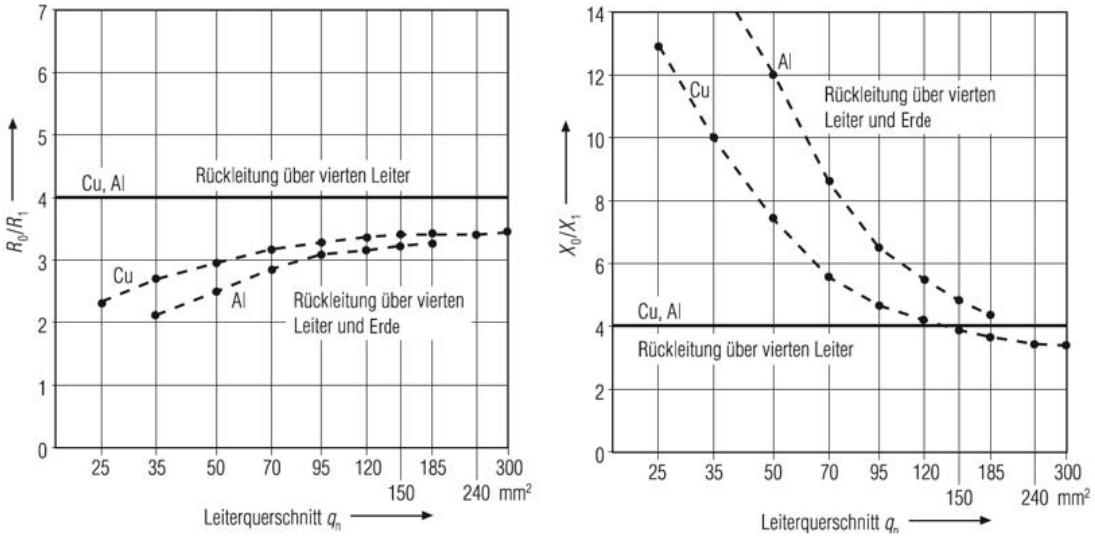


Abbildung 49: Verhältnisse der Widerstands-bzw. Reaktanzwerte im Mit- und Nullsystem [27]

4.2.1 Widerstandsbelag

Der Widerstandsbelag des Modells berechnet sich aus dem spezifischen Widerstand ρ in $\frac{\Omega\,\mathrm{mm}^2}{\mathrm{m}}$ des Leitermaterials, geteilt durch den verwendeten Nennquerschnitt A in mm^2 multipliziert mit dem sogenannten Verseilungsfaktor $\beta \approx 1.07$, siehe Gleichung (4). Letzterer wird nur berücksichtigt, wenn die Drahtlänge größer als die Seillänge ist. Aufgrund der längenmäßig relativ kurzen Ausführung von Niederspannungskabeln, welche im Zuge dieser Arbeit betrachtet werden, kann dieser Faktor jedoch vernachlässigt werden.

$$R' = \frac{\rho}{A} \cdot \beta \tag{8}$$

Bei Temperaturen abweichend von 20 °C muss bei der Berechnung des spezifischen Widerstandes eine entsprechende Korrektur durchgeführt werden, bei einer Temperatur ϑ gilt demnach:

$$\rho = \rho_{20\,^\circ} \cdot [1 + \alpha \cdot (\vartheta - 20\,^\circ)] \tag{9}$$

Für die weiteren Betrachtungen gelten folgende Annahmen:

- Leitermaterial Kupfer,
- Umgebungstemperatur $\vartheta = 22\,^\circ C$,
- spezifischer Widerstand von Kupfer $\rho_{Cu_{20\,^\circ}} = 0{,}017241\ \frac{\Omega\,\mathrm{mm}^2}{m}$ und
- Temperaturkoeffizient von Kupfer $\alpha_{Cu} = 0{,}00393\ \frac{1}{^\circ C}$ [27].

Die bei Leitungen (Kabel) auftretenden Zusatzverluste durch Näheeffekt („proximity effect“) sowie der Stromverdrängungseffekt („skin effect“) können auf Ebene der Niederspannung bis zu einem Nennquerschnitt von 185 mm² vernachlässigt werden [12, 27]. In

Tabelle 7 sind die Berechnungsergebnisse der Widerstände sowie deren Auslegung zusammengefasst.

Die Mitsystemkomponente des Widerstandes ist nachfolgend in Formel (10) zu sehen:

$$R^1 = \frac{\rho}{A} \cdot \beta \cdot l = \frac{\rho_{20°} \cdot [1 + \alpha \cdot (\vartheta - 20°)]}{A} \cdot \beta \cdot l = R' \cdot l \qquad (10)$$

Die Nullsystemkomponente des Widerstandes berechnet sich nach Formel (11) wie folgt:

$$R^0 = 4 \cdot R' \cdot l = 4 \cdot R^1 \qquad (11)$$

Tabelle 7: Werte der Widerstandsberechnungen für verschiedenste Nennquerschnitt-Längenkombinationen

Kabelparameter		**Berechnet**		**Käuflich erwerbbar (Widerstandsnormreihe)**	
Nennquerschnitt	Länge	Mitsystem	Nullsystem	Mitsystem	Nullsystem
A in mm²	*l* in m	R^1 in Ω	R^0 in Ω	R^{1*} in Ω	R^{0*} in Ω
4	108	0,47	1,88	0,47	2,2
4	250	1,09	4,34	1	4,7
4	510	2,22	8,86	2,2	10
6	**162**	**0,47**	**1,88**	**0,47**	**2,2**
10	270	0,47	1,88	0,47	2,2
10	600	1,04	4,17	1	4,7
16	435	0,47	1,89	0,47	2,2

Anmerkung: Die Werte R1* (Mitsystem) sowie R0* (Gegensystem) für den ohmschen Widerstand eines Leiters in einem Niederspannungskabel sind entsprechend den berechneten Werten gerundete Widerstandswerte aus den jeweiligen Widerstandsnormreihen.

In Blau ist jene Kombination zu sehen, welche die Werte für einen Querschnitt von 6 mm² und einer Länge von 162 m abbildet, die für die weiteren Auslegungen herangezogen wird.

4.2.2 Induktivitätsbelag

Eine exakte Berücksichtigung des Induktivitätsbelages ist aufgrund eventuell vorhandener metallischer Mäntel (zB durch eine geschlossene Kabeltrassierung) sowie durch Bewehrungen induzierte Ströme, welche zur Verminderung der Induktivität führen können, schwierig. Somit ist es in den meisten Fällen sinnvoll bzw. praktikabel, die Angaben des Herstellers zu verwenden, da diese Werte unter Laborbedingungen ermittelt wurden. [12, 27]

Da die entsprechenden Reaktanzwerte nur für Querschnitte ab 10 mm² in der Literatur [27] verfügbar sind, ist eine entsprechende Extrapolation zweiten Grades durchzuführen, um Werte für kleinere Querschnitte zu erhalten vgl. Tabelle 8 (grau markiert). Aufgrund dieser Extrapolation kann von Abweichungen der Reaktanz- bzw. Induktivitätswerte bei Querschnitten unter 10 mm² ausgegangen werden. Die entsprechenden Ergebnisse sind in Tabelle 9 zusammengefasst.

Tabelle 8: Nennquerschnitte sowie Reaktanzbeläge X_L‘ im Mitsystem für 0,6-/1-kV-Kabel N(A)YY bei f = 50 Hz aus [27], Werte für 4 und 6 mm² extrapoliert

Nennquerschnitt	Reaktanzbeläge X_L' im Mitsystem
mm²	*Ω/km*
4	0,0970
6	0,0960
10	0,0942
16	0,0892
25	0,0880
35	0,0851
50	0,0848

Anmerkung: Es handelt sich bei den Reaktanzbelägen für die Kabelnennquerschnitte 4 mm² sowie 6 mm² um extrapolierte Werte auf Basis von [27].

Die Reaktanzwerte in Tabelle 9 berechnen sich lt. Formel (12) wie folgt:

$$X = X_L' \cdot l \tag{12}$$

Zu beachten ist hierbei, dass die Reaktanzbeläge in Tabelle 8 aus [27] in Ω/km angegeben sind.

Die Induktivitätswerte aus Tabelle 9 ergeben sich lt. Formel (13) wie folgt:

$$L = \frac{X}{j \cdot \omega} = \frac{X}{j \cdot 2 \cdot \pi \cdot f} \quad \textbf{(13)}$$

Tabelle 9: Reaktanz- sowie Induktivitätswerte für Mit- und Nullsystem für verschiedene Nennquerschnitt-Längenkombinationen

Kabelparameter		Reaktanz X		Induktivität L	
Nennquerschnitt	Länge	Mitsystem	Nullsystem	Mitsystem	Nullsystem
mm²	*m*	Ω	Ω	*mH*	*mH*
4	108	0,0105	0,0419	0,0333	0,1334
4	250	0,0243	0,0970	0,0772	0,3088
4	510	0,0495	0,1979	0,1575	0,6299
6	**162**	**0,0156**	**0,0622**	**0,0495**	**0,1980**
10	270	0,0254	0,1017	0,0810	0,3238
10	600	0,0565	0,2261	0,1799	0,7196
16	435	0,0388	0,1552	0,1235	0,4940

4.2.3 Kapazitätsbelag

Für die exakte Bestimmung der Kapazitätsbeläge von Leitungen (Kabel) auf Ebene der Niederspannung sind Messungen am realen Kabel erforderlich. Aufgrund der geringen Spannung ist jedoch die Bedeutung des Kapazitätsbelags bei Kabel eher als gering einzustufen, weshalb dieser in der Modellierung vernachlässigt werden kann. Erwartet werden dabei Kapazitätsbeläge zwischen 200 nF/km und 400 nF/km bei Kabelausführung. [12, 27]

4.2.4 Ableitungsbelag

Der Ableitungsbelag erfasst Verluste des Dielektrikums zwischen Leiter und Erde. Bei Kabeln liegt die Größenordnung des Ableitungsbelags bei ca. 1 µS/km [27] und kann unter normalen Betriebsverhältnissen vernachlässigt werden. Weiters kommen die dielektrischen Verluste bei einer Kabelausführung erst ab einem Wert über 10 kV zu tragen. [12, 27]

4.2.5 Verwendete Kombination

Im Zuge dieser Arbeit werden Berechnungen für die verschiedensten Nennquerschnitte sowie Leitungslängen eines NYY-O 0,6/1 kV 4-Leiter-Niederspannungskabels durchgeführt und ausgewertet. Als Umgebungstemperatur werden 22 °C angenommen was den Laborbedingungen entspricht. Es werden die Querschnitte 4, 6, 10, 16, 25, 35 sowie 50 mm² berechnet. Für die Querschnitte 4 und 6 mm² wird zudem eine Extrapolation zweiten Grades durchgeführt da es für diese Nennquerschnitte keine Herstellerangaben gibt. Es wird für alle oben genannten Querschnitte sowohl die Mit- als auch Nullsystemkomponente berechnet. Zudem werden Widerstands- und Reaktanzbelag ermittelt und in weiterer Folge durch Einbeziehung der jeweiligen Länge auf den Widerstandswert sowie Reaktanzwert gerechnet. Die Berechnungen sind mittels Excel interaktiv mit Formeln hinterlegt, sodass man bei einer Kabellängenänderung automatisiert die neuen Berechnungsergebnisse abrufen kann. Mit Hilfe dieser Berechnungen werden mehrere Kombinationen ermittelt, bei denen sowohl die Widerstandswerte als auch die Induktivitätswerte in praxistauglichen Wertebereichen vorliegen. Bei den Widerstandswerten muss darauf geachtet werden, dass sowohl Mitsystem als auch Nullsystemberechnungsergebnisse jenen Werten entsprechen, die einer Widerstandsnormreihe genügen. Die Verkleinerung des möglichen Auswahlbereichs aufgrund dieser Einschränkung vermindert die möglichen Modellkombinationen auf unter zehn.

Bei den Werten bezüglich des Induktivitätswertes ist eine Firma kontaktiert worden, die es sich zum Hauptaugenmerk gemacht hat Spezialdrosseln zu fertigen. Es wurden mehrere Werte für Einphasen-Drosseln in offener Ausführung angefragt, das entsprechende Angebot liegt dieser Arbeit im Anhang bei (9.3.1).

Dabei ergibt sich die Kombination „Nennquerschnitt 6 mm² und Kabellänge 162 m“ (in Tabelle 7 und Tabelle 9 jeweils blau markiert) als jene, welche grundsätzlich die in der Praxis vorkommenden Gegebenheiten am besten abbilden würde, weshalb diese in weiterer Folge vollständig samt Kostenaufstellung und möglichem Verteilerbau projektiert wird. Wie bereits in Kapitel 2 beschrieben, kommt es bei langen Netzausläufern zu Problemen hinsichtlich der Spannungshaltung weshalb in der Praxis, wo immer es möglich ist, kurze Leitungslängen bevorzugt werden. Für die ermittelten Kombinationen in Tabelle 7 ist somit die eingangs erwähnte Nennquerschnitt-Längenkombination als die Praxistauglichste anzusehen.

4.3 Kosten Projektierung NYY-O 4x6mm² - 162m

Die Bruttokosten für den Bau eines solchen Leitungsmoduls belaufen sich auf rund 3200 € (Details dazu siehe Kalkulation bzw. Kostenaufstellung im Anhang, Kapitel 9.3.2). Dabei finden jedoch keinerlei Rabatte Anwendung, welche bei einer Bestellung durch eine Bildungseinrichtung wie die TU Graz üblich sind. Weiters ist der genannte Preis entsprechend von den errechneten Widerstands- bzw. Induktivitätswerten abhängig. Zudem ist zu berücksichtigen, dass bei der vorliegenden Kalkulation eines Leitungsmoduls mehrere Materialpositionen enthalten sind, die für den Bau mehrerer Module verwendet werden können. Grund dafür sind beispielsweise die Mindestabnahmemengen bei einigen Montagematerialien sowie die Mindestleitungslängen bei Kabeln und Verdrahtungsleitungen.

Darüber hinaus ist der projektierte Schaltschrank für mindestens drei, bei kompakter Bauweise vier Leitungsmodule ausgelegt.

4.4 Empfehlungen bei einer möglichen Realisierung

Beim Einbau mehrerer Leitungsmodule in einen Schaltschrank sollten diese übersichtlich und getrennt voneinander platziert werden.

Zudem ist beim Einbringen von mehreren Modulen in ein und denselben Schaltschrank die im Betrieb entstehende Wärme zu berücksichtigen. Hier ist gegebenenfalls eine aktive (zB mittels entsprechender Ventilatoren) oder passive (Schlitze bzw. Gitter) Belüftung anzudenken. Zweitere könnte zB durch eine Ausnehmung in der Oberseite (Deckel) des Schaltschranks sichergestellt werden wobei jedoch berücksichtigt werden muss, dass die entstehende Öffnung vor Eindringen von Verunreinigungen jeglicher Form sowie Flüssigkeiten zu schützen ist. Eventuell genügt eine Anhebung der Deckelplatte des Schranks sowie das Einbringen von zusätzlichem Distanzmaterial. Ist dies nicht ausreichend, so kann an der Vorderseite des Schranks eine aktive Entlüftung des Schrankes eingebaut werden. Diese sollte aufgrund des aufsteigenden Wärmestroms möglichst weit oben an der Schrankvorderseite eingebaut werden.

Unter Spannung stehende (aktive) Teile sind durch geeignetes Isoliermaterial abzudecken und damit vor unbeabsichtigter Berührung zu schützten.

Zudem muss bei jeder Querschnittverminderung eine entsprechende Absicherung hinsichtlich des Leitungsschutzes (Überstromschutz) berücksichtigt werden. Dies wird jedoch bereits durch die Projektierung mit einem Sicherungslasttrennschalter seitens des Leitungsmoduls, der Verkabelung im Schrankaufbau mit 16 mm² Einzeladerleitungen sowie der Leitungsmodule mit 6 mm² Einzeladerleitung gewährleistet.

5 Schlussfolgerungen und Ausblick

Sowie sich die Gesellschaft in Etappen bzw. deren Generationen grundlegend im Laufe der Jahre ändert, so ändert sich auch das Stromnetz und hier besonders dessen Anforderungen. Wir befinden uns aktuell in einem Zeitalter, in dem der technische Fortschritt enorm vorangetrieben wird. Dies macht sich zurzeit vor allem beim Thema Umweltschutz sowie „grüner“ Energie bemerkbar. [3] Die Substitution des Verbrennungsmotors durch den Elektromotor ist bereits beschlossen und lässt sich auch mit zahlreichen Zukunftsprognosen, wie beispielsweise Prognosen hinsichtlich des Durchdringungsgrades von Elektrofahrzeugen, untermauern. [19] Dies wird zukünftig aber vor allem das Übertragungsnetz vor außerordentliche Herausforderungen stellen. [2] Die breite Forderung seitens Politik und Gesellschaft dahingehend, dass das neue Mobilitätskonzept (mitunter auch betreffend den Individualverkehr) unter anderem auf batterieelektrischen Fahrzeugen basieren soll, stellt – mit der für die künftige Masse an Elektrofahrzeugen erforderlichen Ladeinfrastruktur – die elektrischen Übertragungs- und Verteilnetze vor neue Aufgaben. [2, 19] Erste Abhilfemaßnahmen könnten beispielsweise das gesteuerte Laden von Elektrofahrzeugen mit sich bringen. Dabei kann der/die Nutzer*in zwar sein/ihr Fahrzeug an der Ladestation zum Laden anstecken, die endgültige Ladefreigabe kommt jedoch erst von einem übergeordneten Energiemanagementsystem bzw. einem entsprechenden Algorithmus. Weitere Entlastungen könnten beispielsweise die Verteilung der Einphasenlasten gleichmäßig auf alle drei Phasen des Netzes sein [19]. Zudem bedarf es im privaten Sektor einer Regelung bezüglich der maximal zulässigen Installationseinheiten an Ladestationen für Elektrofahrzeuge. Diese eben genannten Entlastungen sind heutzutage und wahrscheinlich noch in unmittelbarer Zukunft ausreichend [19] und zweckdienlich, jedoch sei hier erwähnt, dass der Durchdringungsgrad an Elektrofahrzeugen heutzutage im Vergleich zu jenem prognostizierten im Jahre 2030 noch relativ gering ist. In Österreich sind mit Stichtag 31. Mai 2021 etwa 1,1 % aller Fahrzeuge mit einem reinen Elektromotor ausgestattet. [29] Auf Hybridantriebe, folglich Fahrzeuge mit Verbrennungskraftmotor kombiniert mit Elektromotor (und Plug-In-Hybriden ggfs. mit Traktionsbatterie) entfallen etwa 2,1 % der zugelassenen Fahrzeuge in Österreich. [29] Mit zunehmendem Durchdringungsgrad der Elektromobilität steigen folglich auch die privaten Installationen von Ladestationen (Stichwort „Wallbox“), woraufhin die zuvor genannten Maßnahmen in manchen Teilen des elektrischen Verteilnetzes schlussendlich nicht mehr ausreichen könnten und es im Sinne einer Worst-Case-Betrachtung zu Teilausfällen des Netzes kommen könnte, vgl. hierzu Kapitel 2 dieser Arbeit. [18] Auch aufgrund dieser Entwicklungen beschäftigt sich das Forschungsprojekt PoSyCo mit der Frage nach einer zukunftsorientierten und zielführenden Bewerkstelligung dieser Fragestellung. [18] Temporäre Vermaschung im Niederspannungsnetz könnte hierbei ein solcher zielführender Ansatz sein. Jedoch darf man die marktwirtschaftlichen Aspekte keinesfalls außer Acht lassen. Aktuell geht der Trend grundsätzlich eher in Richtung Entmaschung des Netzes bzw. zu kleineren Maschenweiten, das höhere Risiko wird jedoch aus wirtschaftlichen Gründen in Kauf genommen [1]. Es muss hier also ein Kompromiss auf allen Ebenen geschaffen werden, um ein zukunftsfittes Stromnetz im Sinne der Zuverlässigkeit und Versorgungssicherheit zu gewährleisten. Demnach bedarf es noch viel Forschungsarbeit in diesem Themenfeld, wobei

mit den Betrachtungen hinsichtlich der automatisierten Vermaschung von Niederspannungsnetzen im Zuge dieser Arbeit bereits ein dahingehender Beitrag geleistet wird.

Mit der Erweiterung des Labordemonstrators um die Kommunikationskomponenten und den in Kapitel 3 beschriebenen Maßnahmen zur intelligenten, softwaregestützten Ansteuerung der verbauten Kompaktleistungsschalter sowie der zugehörigen Datenverarbeitung besteht nun die Möglichkeit eines vollständig automatisierten Laborbetriebs für weitere Tests und Versuche u. a. im Zuge von [24]. Ermöglicht wird dies durch die gezielte Auswahl der erforderlichen Hardwarekomponenten auf Basis des PROFINET-Kommunikationsstandards, vgl. hierzu Kapitel 3.2. Mit Fertigstellung dieser Arbeit besteht neben der manuellen Betätigung der Kompaktleistungsschalter auch eine weitere Möglichkeit im Sinne der vollständig automatisierten, computergestützten Ansteuerung in Echtzeit. Die gesamten Nachteile einer manuellen Ansteuerung entfallen somit, die Vorteile bleiben zugleich jedoch vollständig erhalten. Dies bedeutet, dass sowohl ungewollte Fehleingaben durch den/die Anwender*in vermieden werden und ermöglicht zudem den Echtzeitbetrieb der Anlage. Auch während des Ausführens der Automatisierung bleibt die Möglichkeit der gleichzeitigen manuellen Ansteuerung vollumfänglich erhalten. Somit verfügt der/die Benutzer*in immer noch über die Möglichkeit des manuellen Eingriffs in das System, wobei dieser durch das dafür in der Programmiersprache Python (in der Entwicklungsumgebung PyCharm) entwickelte Skript erkannt und verarbeitet wird.

Dieses Skript ist zudem noch mit zahlreichen Segmenten versehen, bei denen eine etwaige fehlerhafte Benutzer*inneneingabe jeglicher Art erkannt und verhindert wird. Dies dient vor allem sicherheitstechnischen Aspekten sowie Gründen der Unterbindung von unnötigen Schalthandlungen. Wird ein solches „System-Exit-Segment“ vom Code eingeleitet, bietet es dem/der Benutzer*in zudem eine entsprechende Meldung, warum es zur jeweiligen Beendigung des Codes gekommen ist. Dies ermöglicht wiederum eine entsprechend unmittelbare Interaktion mit dem System in der nächsten Iteration.

Durch die zusätzlich automatisiert generierten Grafiken bzw. Diagramme verfügt der/die Benutzer*in zugleich über die Möglichkeit, optische Vergleiche zwischen einzelnen Szenarien (Schaltstellungen) anzustellen und so Aussagen bezüglich deren Vor- bzw. Nachteile zu treffen. Dahingehend wurde in dieser Arbeit eine kleine Versuchsreiche durchgeführt, um in erster Linie den entwickelten Code zu testen, aber auch um zu zeigen, dass es durchaus Unterschiede hinsichtlich der gewählten Netztopologie betreffend des Lastflusses gibt. Die ausgewerteten Versuche sowie deren grafische Interpretation sind unter (3.6) ersichtlich.

Zukünftig bildet das entwickelte System bzw. das entsprechende Python-Skript die Grundlage für weiterführende Untersuchungen und Analysen der temporären Vermaschung auf Niederspannungsebene am Institut für Elektrische Anlagen und Netze der TU Graz im Zuge des Power System Cognification (PoSyCo) Forschungsprojekts.

Zudem ist für den Betrieb lediglich eine Entwicklungsumgebung für die Programmiersprache Python notwendig. Die Hersteller-Software (Siemens) der verwendeten Kompaktleistungsschalter der Sentron 3VA Geräteserie ist zur Ansteuerung nun nicht mehr

explizit erforderlich, diese kann jedoch zusätzlich zum Parametermonitoring weiter genutzt und eingesetzt werden. Über den freigegebenen Webserver besteht zudem die Möglichkeit, die einzelnen Parameter, wie beispielsweise Messwerte der einzelnen Kompaktleistungsschalter, Temperaturen einzelner Bauteile sowie Anzahl an mechanischen Auslösungen der jeweiligen Kompaktleistungsschalter etc. mittels eines beliebigen Browsers (mittels Eingabe der zugewiesenen IP-Adresse der verwendeten CPU) von jedem Endgerät im entsprechenden Netzwerk einzusehen.

Darüber hinaus legt die in Kapitel 4 dargestellte Erweiterung mittels Leitungsmodulen eine noch praxisnähere Sichtweise dar. Um weiterführende Versuche noch praxisorientierter gestalten zu können, wurde die Möglichkeit geschaffen, am bestehenden Labordemonstrator mittels serieller Ankopplung in Form sogenannter HEAVYCON-Modular Steckverbinder, bis zu vier Leitungsmodule (beispielsweise im Sinne von Leitungsmodellen in Form von Ersatzschaltbildern) einzubinden, vgl. hierzu (4.2). Dies ermöglicht es, verschiedenste Typen von Niederspannungskabeln sowie Leitungslängen zwischen den einzelnen Leistungsschaltern realitätsnahe zu integrieren. Diese Arbeit fasst dabei verschiedenste Kombinationen hinsichtlich Kabelnennquerschnitte und -längen samt deren Auslegung bzw. Berechnung zusammen. Jedoch stellt sich dabei heraus, dass es – aufgrund der am Markt erhältlichen Bauteile – Einschränkungen diesbezüglich gibt. Die in Kapitel 4.2 dargelegten Tabellen zeigen Ergebnisse dieser Berechnungen sowie zugehörige Auslegungen. Aufgrund der Marktsituation werden dabei keine Widerstände kleiner als 470 mΩ verwendet. Auch die Kombination der Werte aus Mit- und Nullsystem reduzieren die Auswahlmöglichkeiten beträchtlich. Aus Einsparungsgründen wurde für die Kostenkalkulation eine Parallelschaltung zweier Widerstände von je 1 Ω berücksichtigt, um auf den kalkulierten Wert von rund 470 mΩ zu gelangen. Grund dafür ist, dass diese Variante der Parallelschaltung Kosteneinsparungspotentiale gegenüber der Variante ohne Parallelschaltung bietet, vgl. hierzu die Kostenaufstellung im Anhang (9.3.2).

Die bereits erwähnten Tabellen in Kapitel 4.2 zeigen mögliche Kombinationen, welche eine praktikable Realisierung abbilden. Die vollständige Projektierung erfolgt hierbei für die am ehesten realitätsgetreueste Paarung aus 6 mm² Nennquerschnitt und 162 m Kabellänge, vgl. hierzu 4.2.5.

Tabelle 9 zeigt die Auswertung der Induktivitätswerte. Die Daten für die jeweiligen Reaktanzbeläge stammen aus [27], jedoch ist hier zu erwähnen, dass die Werte für die Kabelnennquerschnitte 4 mm² und 6 mm² aus einer Extrapolation zweiten Grades stammen. Dieser Aspekt ergibt sich aus dem Grund, dass in der Literatur lediglich Reaktanzbeläge für Nennquerschnitte ≥ 10 mm² angegeben sind. Aus den Reaktanzwerten lassen sich sogleich die Induktivitätswerte berechnen, wobei hierfür ein Angebot eines auf die Fertigung von Widerständen sowie Drosseln spezialisierten Herstellers vorliegt, vgl. hierzu Kapitel 9.3.1 des Anhangs. Es handelt sich hierbei um sogenannte Einphasendrosseln mit einer maximalen Strombelastung von 25 A und einer durchschnittlichen Abweichung von ± 10 %. Der Preis eines solchen Leitungsmoduls wurde in dieser Arbeit mit ca. 3200 € brutto berechnet. Detailinformationen zu Ausführung und Kosten siehe Anhang (9.3).

6 Literaturverzeichnis

[1] Prof. Dr.-Ing. Adolf J. Schwab, Elektroenergiesysteme - Erzeugung, Übertragung und Verteilung elektrischer Energie 5. Auflage, Karlsruhe, Deutschland: Springer Vieweg, 2017.

[2] Austrian Power Grid APG, „Masterplan 2030," Wien, 2013.

[3] „#Mission 2030 - Die österreichische Klima- und Energiestrategie," BUNDESMINISTERIUM FÜR NACHHALTIGKEIT UND TOURISMUS & BUNDESMINISTERIUM FÜR VERKEHR, INNOVATION UND TECHNOLOGIE, Wien, 2018.

[4] OVE Österreichischer Verband für Elektrotechnik, *OVE_E_8101_2019_01_01_de; Elektrische Niederspannungsanlagen,* Wien: OVE Österreichischer Verband für Elektrotechnik, 2019.

[5] entsoe, „https://www.entsoe.eu/," [Online]. Available: https://www.entsoe.eu/news-events/former-associations/#european-transmission-system-operators-etso. [Zugriff am 03 09 2021].

[6] Übertragungsnetz VÜN, [Online]. Available: http://www.vuen.at/de/html/uebertragungsnetz.html. [Zugriff am 16 06 2021].

[7] Krum Gerasimov, Konstantin Gerasimov, Nikolay Nikolaev, „Advanced Tools for Stability Improvement of Interconnected Electric Power Systems," Varna, Bulgaria, 2014.

[8] „Open Infrastructure Map," [Online]. Available: https://openinframap.org/#3.35/48.89/17.01. [Zugriff am 17 6 2021].

[9] Verband Schweizerischer Elektrizitätsunternehmen VSE , „Netznutzungsmodell für das schweizerische Verteilnetz," Verband Schweizerischer Elektrizitätsunternehmen VSE, 2019.

[10] OVE Österreichischer Verband für Elektrotechnik, *OEVE_OENORM_EN_50160_2011_03_01_de_Merkmale der Spannung in öffentlichen Elektrizitätsversorgungsnetzen,* Wien: OVE/Austrian Standards Institute, 2011.

[11] Adolf J. Schwab, *Elektroenergiesysteme-Erzeugung, Transport, Übertragung,* Berlin Heidelberg : Springer-Verlag , 2006.

[12] Valentin Crastan, Elektrische Energieversorgung 1 - Netzelemente, Modellierung, stationäres Verhalten, Bemessung, Schalt- und Schutztechnik 4.Auflage, Evilard in der Schweiz: Springer Vieweg, 2015.

[13] Richard Marenbach; Dieter Nelles; Christian Tuttas, Elektrische Energietechnik - Grundlagen, Energieversorgung, Antriebe und Leistungselektronik, Kronberg, Deutschland: Springer Vieweg, 2013.

[14] Prof. Robert Schürhuber | Institut für Elektrische Anlagen und Netze, „Grundlagen der elektrischen Energiesysteme - Niederspannungssysteme," Graz, 2021.

[15] Ing. Sigurd Seyr, Ing. Günther Rösch, Elektrotechnik - Elektroinstallation, Blitzschutz, Lichttechnik, Wien: Jugend &Volk GmbH, 2006.

[16] Dieter Anke, H.-D. Brüns, B. Deserno, H. Garbe, K.-H. Gonschorek, P. Hansen, J. Luiken ter Haseborg, S. Keim, S. Kohling, K. Rippl, V. Schmidt, H. Singer, Elektromagnetische Verträglichkeit. Grundlagen - Analysen - Maßnahmen, Stuttgart: B.G. Teubner, 1992.

[17] Prof. Herwig Renner; DI Polster; DI Lagler, „Elektrische Energiesysteme Labor," Technische Universität Graz , 2018.

[18] Daniel HERBST, Mike LAGLER, Robert SCHÜRHUBER, Ernst SCHMAUTZER, Lothar FICKERT, Alfred EINFALT, Helfried BRUNNER, Daniel-Leon SCHULTIS, Thomas FRUEHWIRTH, Wolfgang PRUEGGLER, „Zukünftige Anforderungen an Niederspannungsnetze und deren Lösungsansätze am Beispiel des Projekts PoSyCo," Graz, 2020.

[19] Dr. Wolfgang Prüggler,Dipl.-Ing. Rusbeh Rezania,Dipl.-Ing. Markus Litzlbauer,Dipl.-Ing. Andreas Schuster,Dr. Christoph Leitinger,Dr. Maximilian Kloess,Dipl.-Ing. Daniel Burnier De Castro,Dipl.-Ing. Helfried Brunner, MSc,Dipl.-Ing. Thomas Rieder, MBA,Dipl., „V2G-Strategies - Endbericht; NEUE ENERGIEN 2020," Wien, 2013.

[20] Manuel Promberger, *Labordemonstrator zur automatisierten Rekonfiguration im Niederspannungsnetz,* Graz University of Technology, 2021.

[21] Siemens AG, „Kommunikationshandbuch - Sentron Schutzgeräte Kommunikation Kompaktleistungsschalter 3VA IEC/UL Systemhandbuch," Siemens AG, Erlangen Deutschland, 2017.

[22] Siemens 7KMPAC9300, „Siemens Produktsuche," [Online]. Available: https://www.automation.siemens.com/bilddb/search.aspx?searchtoken=68b61d97-a48f-4016-b732-00ab060e80f8. [Zugriff am 28 06 2021].

[23] Siemens S7-1211, „Siemens Produktsuche," [Online]. Available: https://www.automation.siemens.com/bilddb/search.aspx?searchtoken=68b61d97-a48f-4016-b732-00ab060e80f8. [Zugriff am 28 06 2021].

[24] D. Herbst, *Ein Beitrag zu neuen Ansätzen im Niederspannungsschutz (Arbeitstitel), Dissertation,* Technische Universität Graz, laufend.

[25] „Elektronik-Kompendium," [Online]. Available: https://www.elektronik-kompendium.de/sites/dig/1807241.htm. [Zugriff am 29 08 2021].

[26] Gijs Molenaar, Stephan Preeker, „python-snap7 Documentation," 2021.

[27] D. Oeding; B.R. Oswald, *Elektrische Kraftwerke und Netze - 7. Auflage,* Ober-Ramstadt,Hannover: Springer-Verlag Berlin Heidelberg, 2011.

[28] https://www.eibabo.ch, „EIBABO technology store," [Online]. Available: https://www.eibabo.ch/diverse/starkstromkabel-erdkabel-nyy-o-4x-6-re-eca-eb10200105. [Zugriff am 10 08 2021].

[29] Statistik Austria, „Fahrzeugbestand Österreich," 31 05 2021. [Online]. Available: https://www.statistik.at/web_de/statistiken/energie_umwelt_innovation_mobilitaet/verkehr/strasse/kraftfahrzeuge_-_bestand/index.html. [Zugriff am 20 07 2021].

[30] „https://www.energiesystem-forschung.de," [Online]. Available: https://www.energiesystem-forschung.de/energiesystem/energie_transportieren. [Zugriff am 28 07 2021].

[31] „co2online Klimaschutz, der wirkt," [Online]. Available: https://www.co2online.de/klima-schuetzen/energiewende/energiewende-definition-ziele-uebersicht/#c150159. [Zugriff am 29 08 2021].

[32] elektro-plus, „elektro-plus," [Online]. Available: https://www.elektro-plus.com/elektroplanung/bestandsschutz/anpassen-elektrischer-anlagen. [Zugriff am 03 09 2021].

[33] TUGraz; CKI, „tugraz.at," [Online]. Available: https://www.tugraz.at/kooperationen/cki/forschungsprojekte/uebersicht/posyco/. [Zugriff am 04 09 2021].

[34] „wirtschaftslexikon.gabler.de," [Online]. Available: https://wirtschaftslexikon.gabler.de/definition/prosument-54019#:~:text=Prosumer%20(engl.,und%20%22Konsument%22)%20herausgebildet.. [Zugriff am 04 09 2021].

[35] Prof. Robert Schürhuber - Institut für Elektrische Anlagen und Netze an der Technischen Universität Graz, „Planung und Betrieb elektrischer Energiesysteme," Graz, 2019.

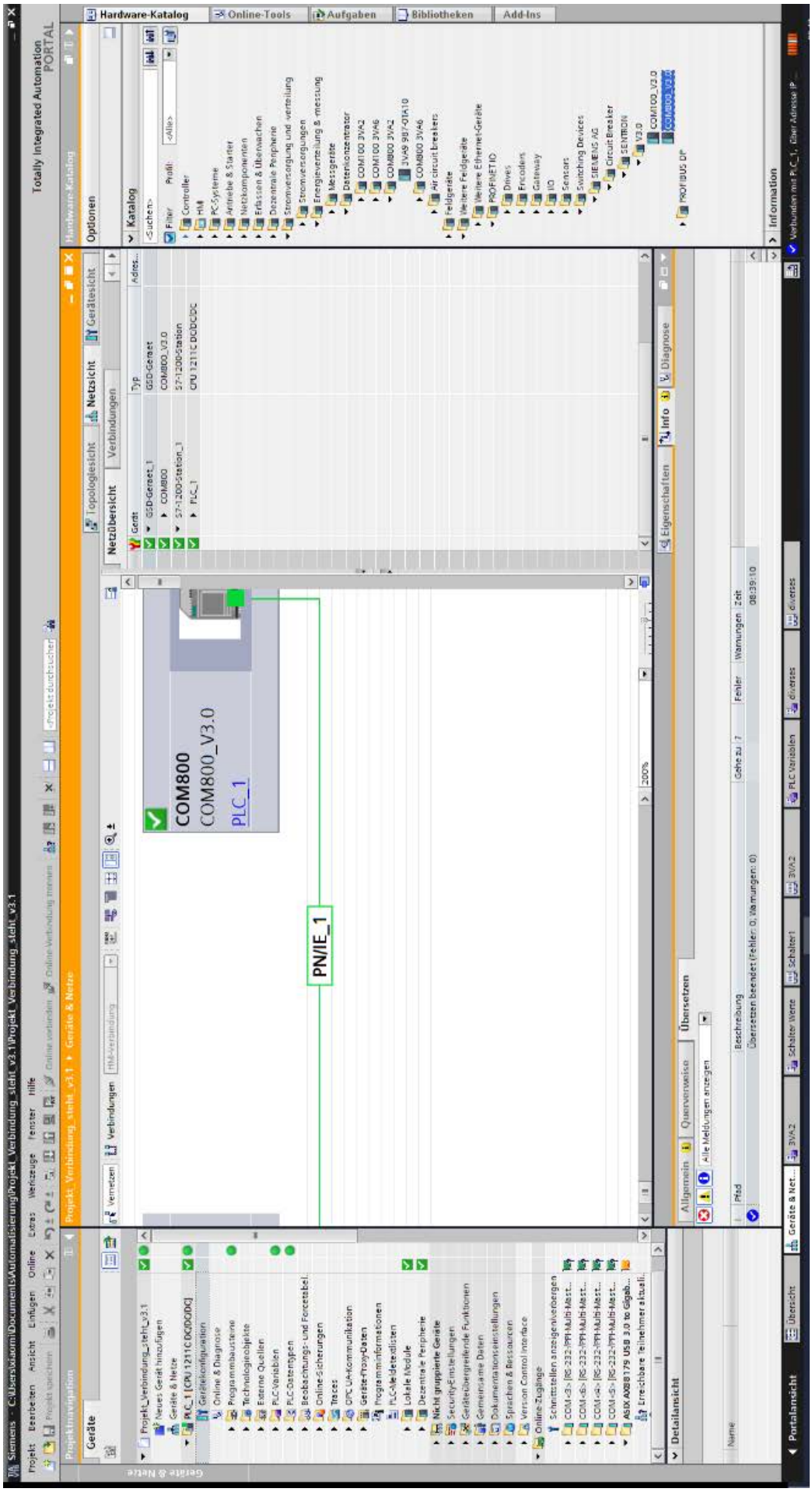
7 Abbildungsverzeichnis

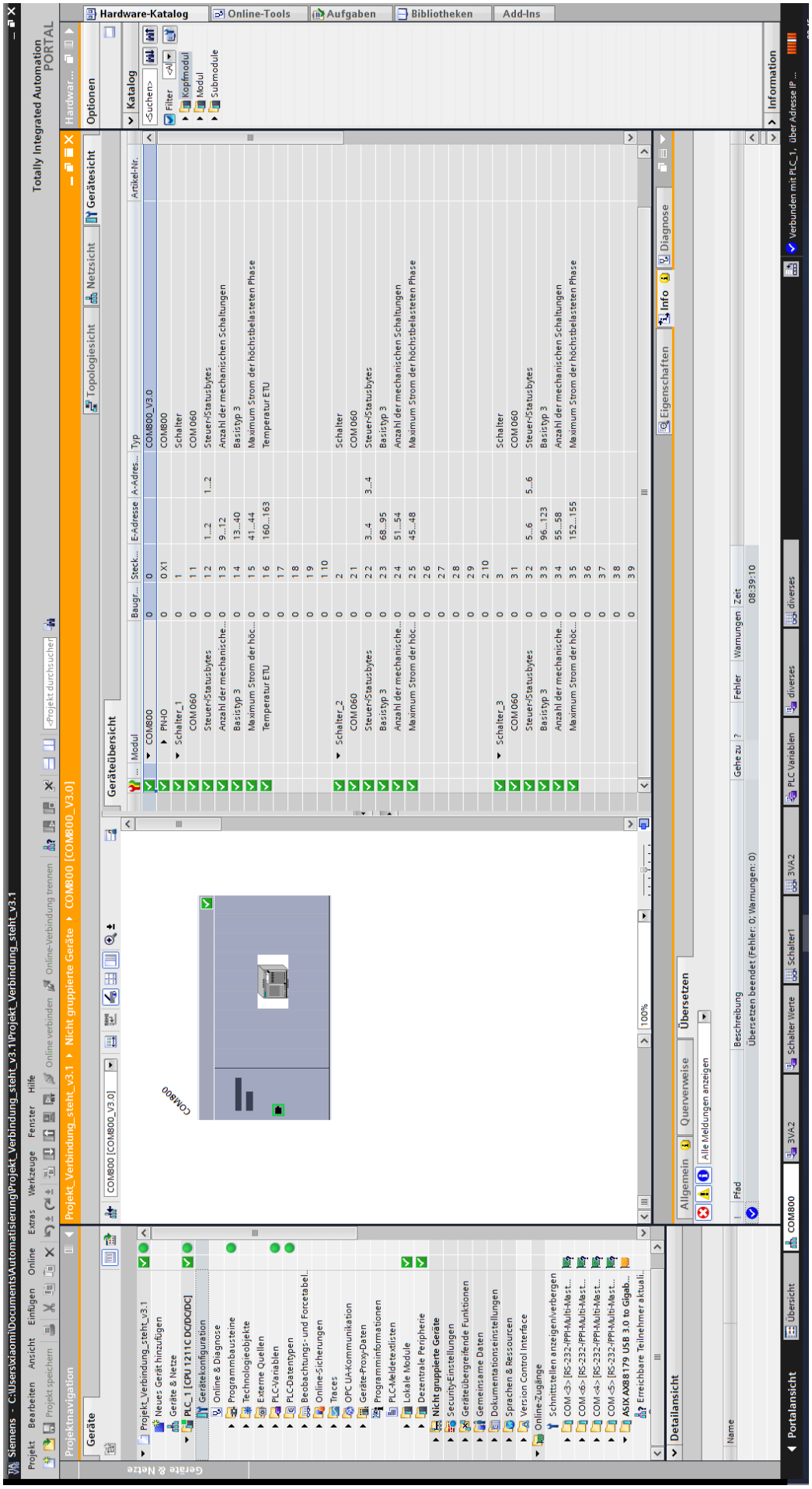
8 Tabellenverzeichnis

9 Anhang

9.1 TIA Portal

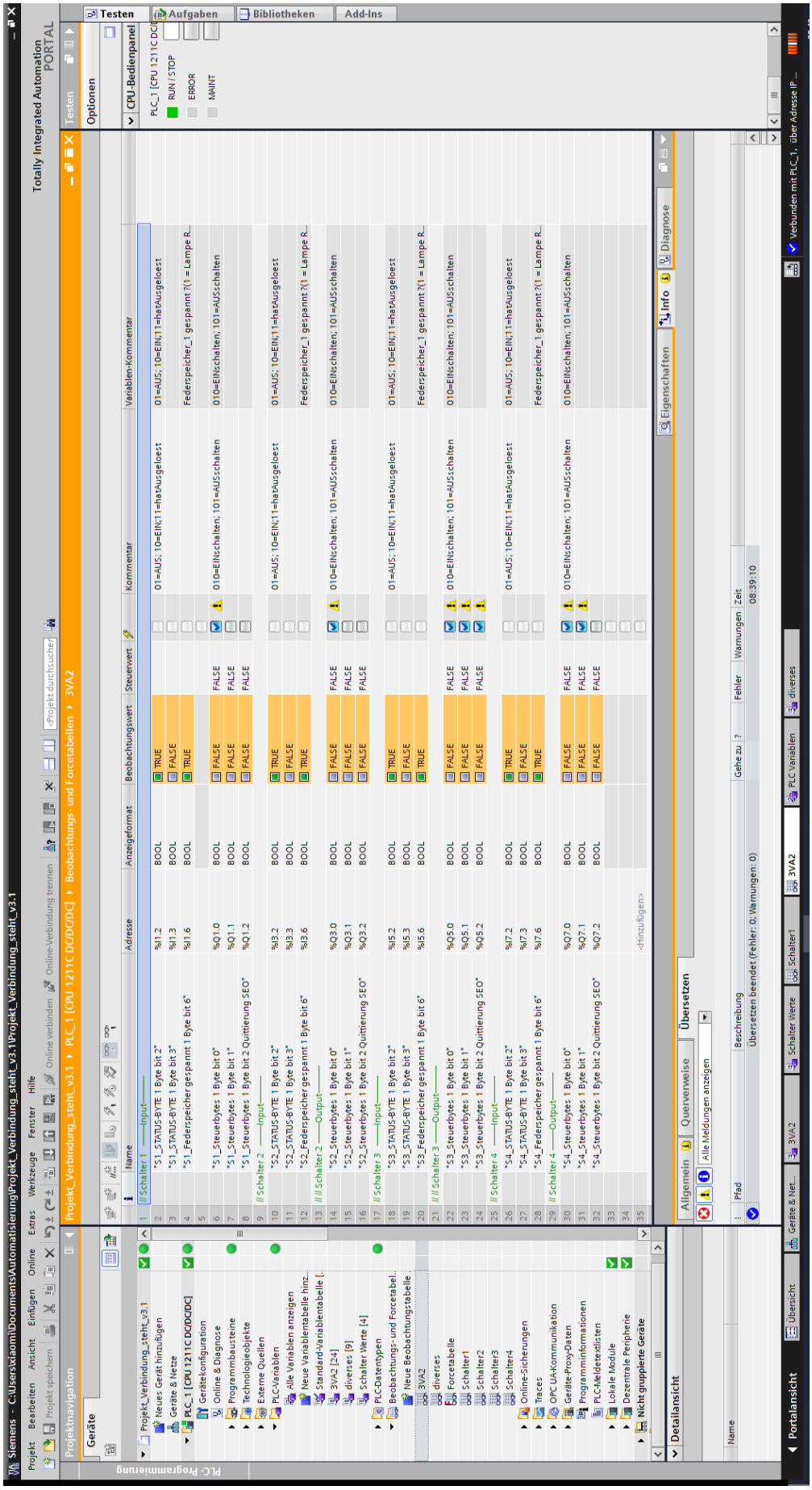
9.1.1 Online Projektierung


Totally Integrated Automation PORTAL
Hardware-Katalog
Online-Tools
Aufgaben
Bibliotheken
Add-Ins
Topologiesicht
Netzsicht
Gerätesicht
Netzübersicht
Verbindungen
COM800
COM800_V3.0
PLC_1
PN/IE_1
Eigenschaften
Info
Diagnose
Allgemein
Querverweise
Übersetzen
Detailansicht
Portalansicht
Übersicht
Geräte & Net...



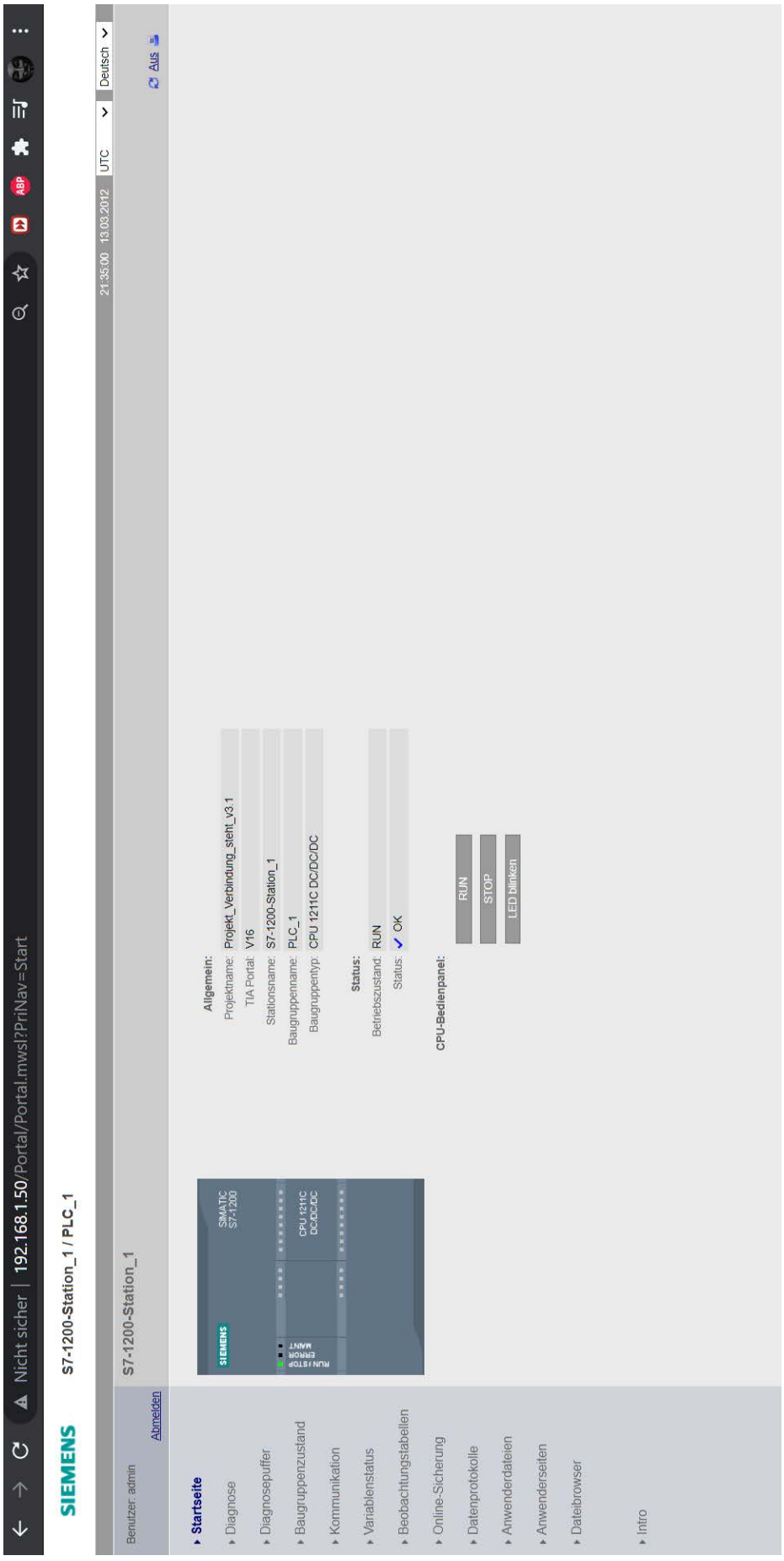
9.1.2 Beobachtungstabellen

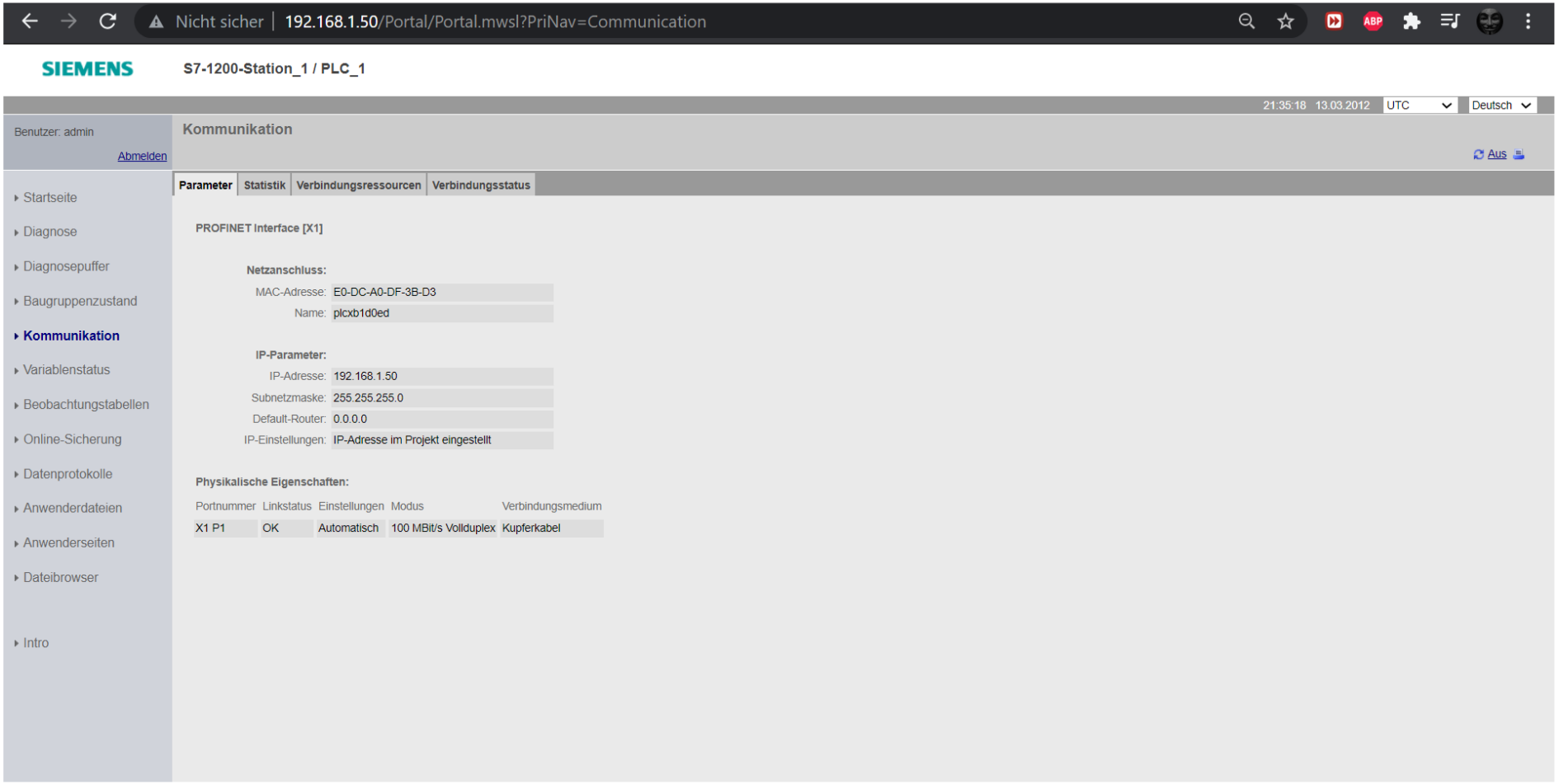
9.1.3 Force Tabellen

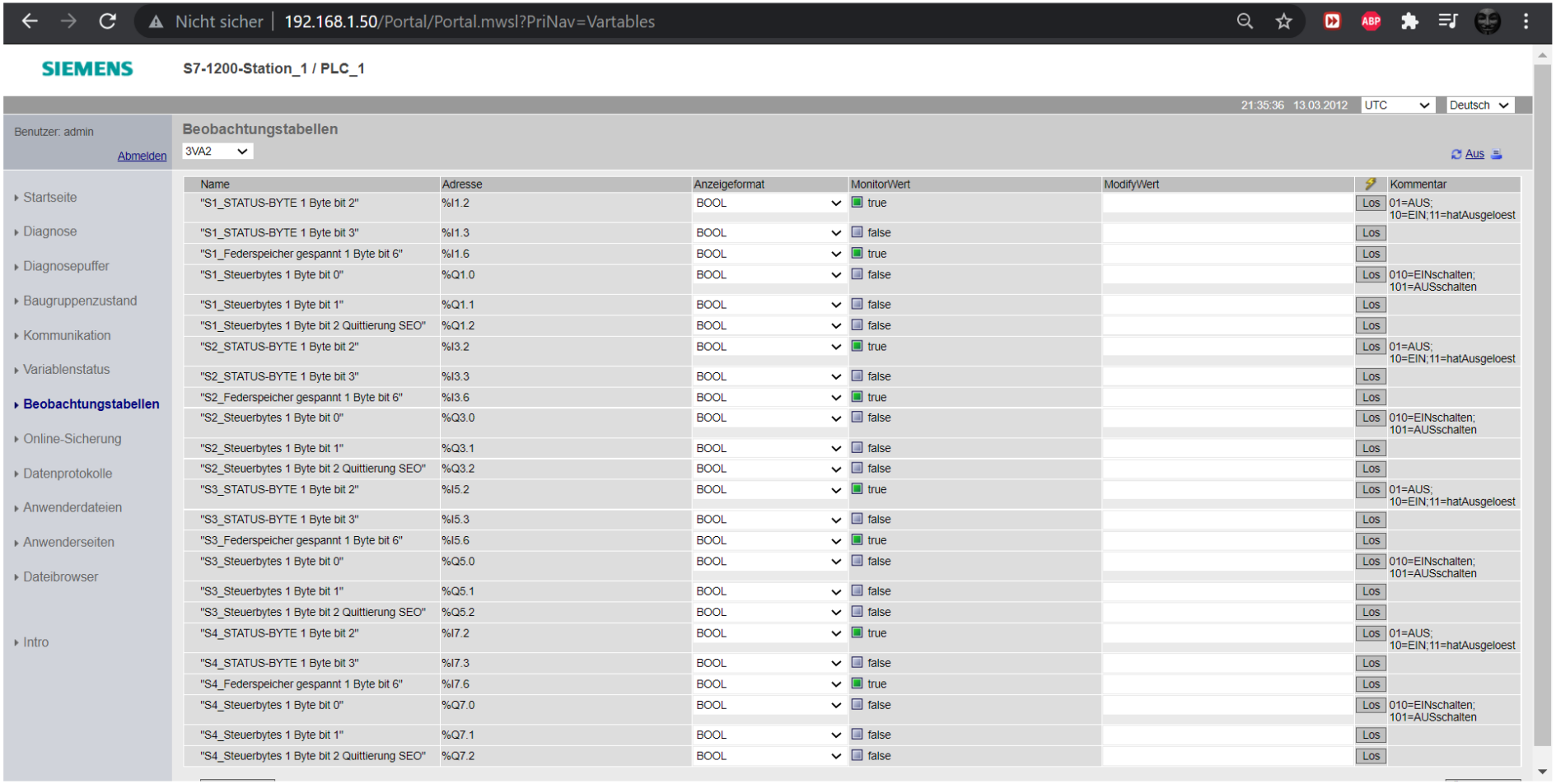



Totally Integrated Automation PORTAL
Projektnavigation
Geräte
Projekt_Verbindung_steht_v3.1
Neues Gerät hinzufügen
Geräte & Netze
PLC_1 [CPU 1211C DC/DC/DC]
Gerätekonfiguration
Online & Diagnose
Programmbausteine
Technologieobjekte
Externe Quellen
PLC-Variablen
Alle Variablen anzeigen
Neue Variablentabelle hinz...
Standard-Variablentabelle [...
3VA2 [24]
diverses [9]
Schalter Werte [4]
PLC-Datentypen
Beobachtungs- und Forcetabel...
Neue Beobachtungstabelle...
3VA2
diverses
Forcetabelle
Schalter1
Schalter2
Schalter3
Schalter4
Online-Sicherungen
Traces
OPC UA-Kommunikation
Geräte-Proxy-Daten
Programminformationen
PLC-Meldetextlisten
Lokale Module
Dezentrale Peripherie
Nicht gruppierte Geräte
Detailansicht
Projekt_Verbindung_steht_v3.1 ▸ PLC_1 [CPU 1211C DC/DC/DC] ▸ Beobachtungs- und Forcetabellen ▸ diverses
Name
Adresse
Anzeigeformat
Beobachtungswert
Steuervert
Kommentar
Variablen-Kommentar
*anzahl mech...
*anzahl mechanis...
*s1_strom höchst...
*s2_strom höchst...
*s3_strom höchst...
*s4_strom höchst...
*s1_temperatur E...
<hinzufügen>
Testen
Aufgaben
Bibliotheken
Add-Ins
Optionen
CPU-Bedienpanel
PLC_1 [CPU 1211C DC/D...
RUN / STOP
ERROR
MAINT
Eigenschaften
Info
Diagnose
Allgemein
Querverweise
Übersetzen
Alle Meldungen anzeigen
Pfad
Beschreibung
Übersetzen beendet (Fehler: 0; Warnungen: 0)
Gehe zu
Fehler
Warnungen
Zeit
08:39:10
Portalansicht
Übersicht
Geräte & Net...
3VA2
Schalter Werte
Schalter1
3VA2
PLC-Variablen
diverses
diverses
Verbunden mit PLC_1, über Adresse IP ...

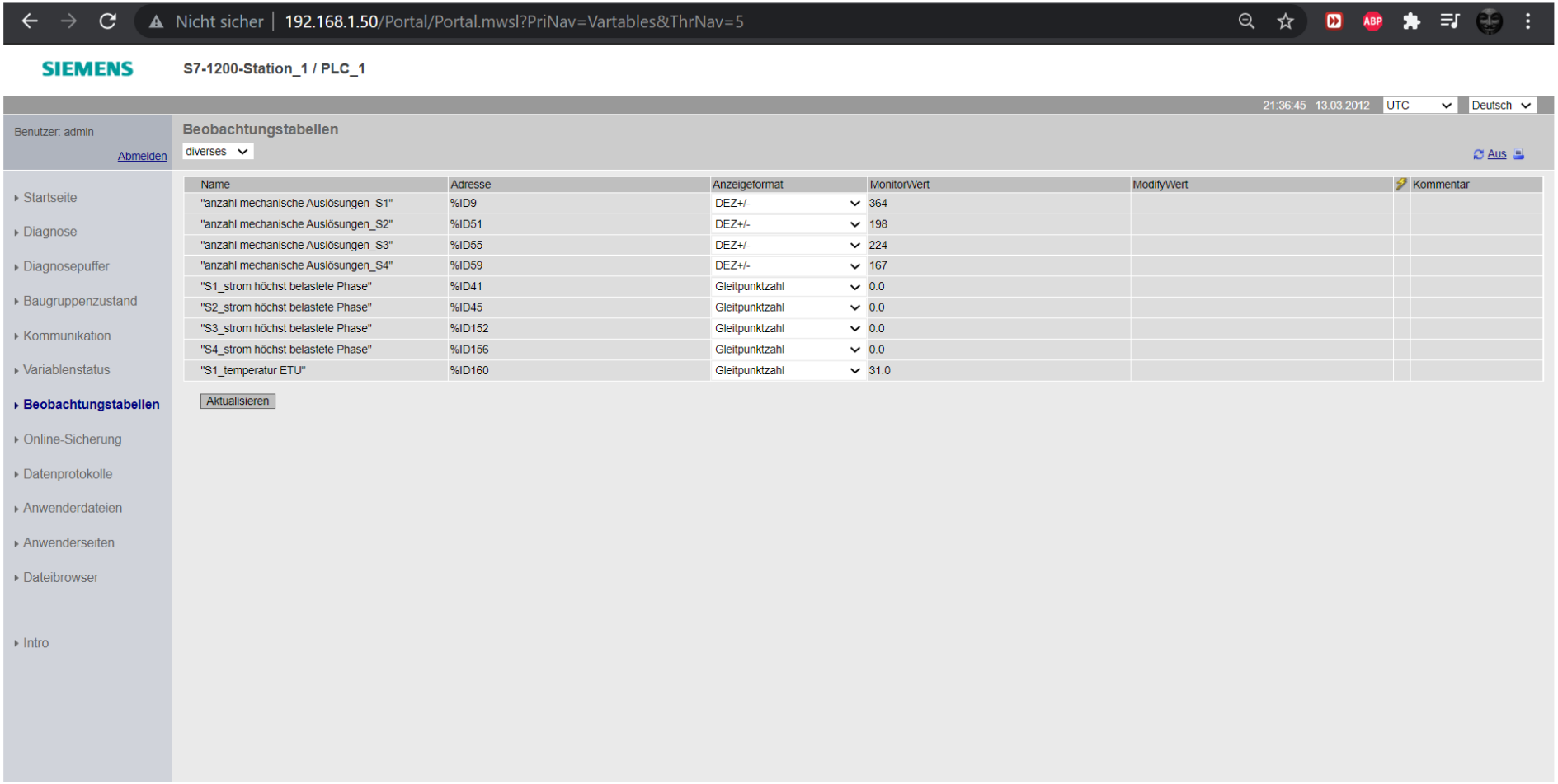
9.1.4 TIA Portal V16 Webserver



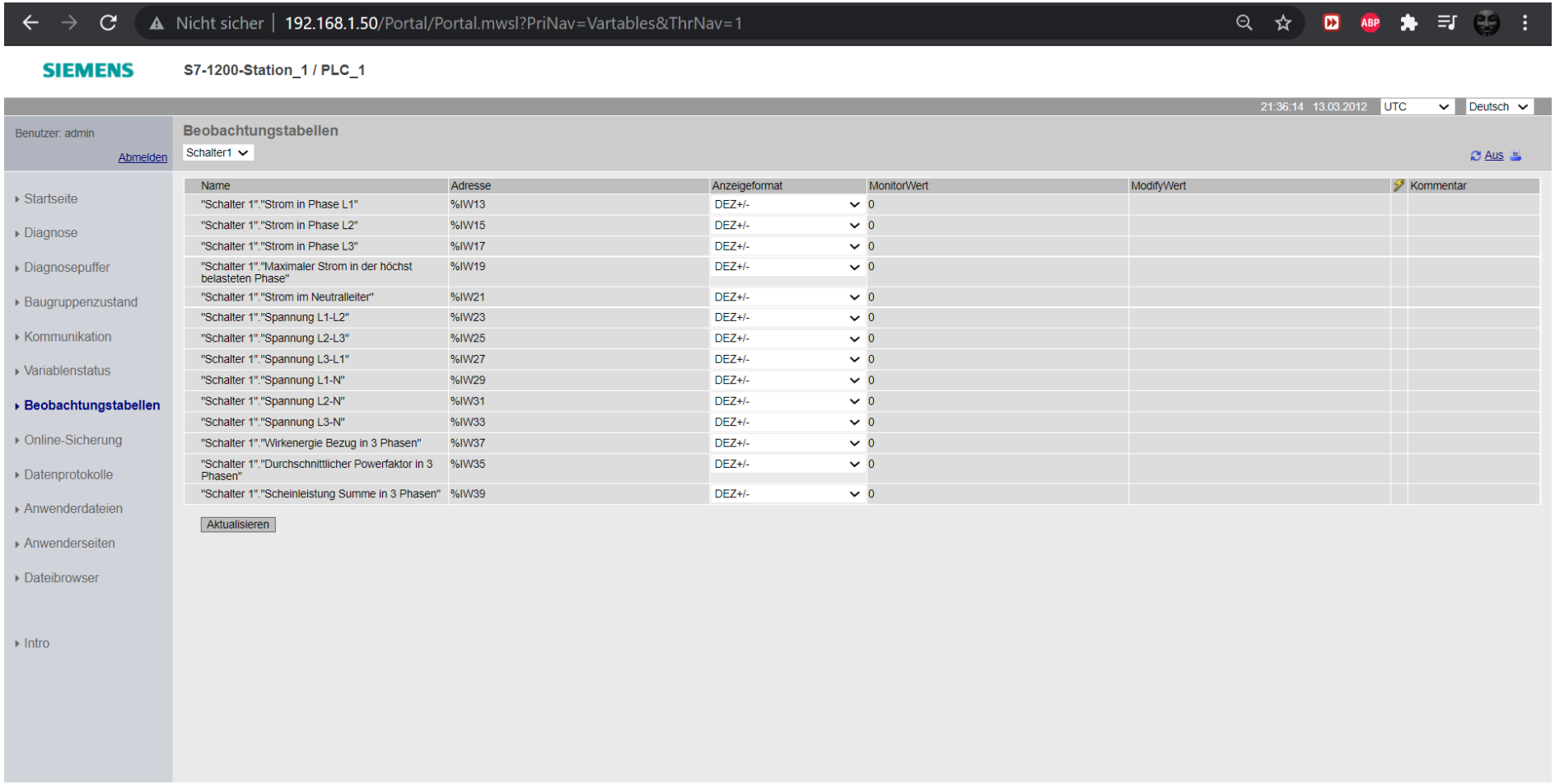

Nicht sicher | 192.168.1.50/Portal/Portal.mwsl?PriNav=Communication
SIEMENS
S7-1200-Station_1 / PLC_1
21:35:18 13.03.2012 UTC Deutsch
Benutzer: admin
Abmelden
Kommunikation
Aus
Startseite
Diagnose
Diagnosepuffer
Baugruppenzustand
Kommunikation
Variablenstatus
Beobachtungstabellen
Online-Sicherung
Datenprotokolle
Anwenderdateien
Anwenderseiten
Dateibrowser
Intro
Parameter
Statistik
Verbindungsressourcen
Verbindungsstatus
PROFINET Interface [X1]
Netzanschluss:
MAC-Adresse: E0-DC-A0-DF-3B-D3
Name: plcxb1d0ed
IP-Parameter:
IP-Adresse: 192.168.1.50
Subnetzmaske: 255.255.255.0
Default-Router: 0.0.0.0
IP-Einstellungen: IP-Adresse im Projekt eingestellt
Physikalische Eigenschaften:
Portnummer Linkstatus Einstellungen Modus Verbindungsmedium
X1 P1 OK Automatisch 100 MBit/s Vollduplex Kupferkabel



Name	Adresse	Anzeigeformat	MonitorWert	ModifyWert		Kommentar
"S1_STATUS-BYTE 1 Byte bit 2"	%I1.2	BOOL	true		Los	01=AUS; 10=EIN;11=hatAusgeloest
"S1_STATUS-BYTE 1 Byte bit 3"	%I1.3	BOOL	false		Los	
"S1_Federspeicher gespannt 1 Byte bit 6"	%I1.6	BOOL	true		Los	
"S1_Steuerbytes 1 Byte bit 0"	%Q1.0	BOOL	false		Los	010=EINschalten; 101=AUSschalten
"S1_Steuerbytes 1 Byte bit 1"	%Q1.1	BOOL	false		Los	
"S1_Steuerbytes 1 Byte bit 2 Quittierung SEO"	%Q1.2	BOOL	false		Los	
"S2_STATUS-BYTE 1 Byte bit 2"	%I3.2	BOOL	true		Los	01=AUS; 10=EIN;11=hatAusgeloest
"S2_STATUS-BYTE 1 Byte bit 3"	%I3.3	BOOL	false		Los	
"S2_Federspeicher gespannt 1 Byte bit 6"	%I3.6	BOOL	true		Los	
"S2_Steuerbytes 1 Byte bit 0"	%Q3.0	BOOL	false		Los	010=EINschalten; 101=AUSschalten
"S2_Steuerbytes 1 Byte bit 1"	%Q3.1	BOOL	false		Los	
"S2_Steuerbytes 1 Byte bit 2 Quittierung SEO"	%Q3.2	BOOL	false		Los	
"S3_STATUS-BYTE 1 Byte bit 2"	%I5.2	BOOL	true		Los	01=AUS; 10=EIN;11=hatAusgeloest
"S3_STATUS-BYTE 1 Byte bit 3"	%I5.3	BOOL	false		Los	
"S3_Federspeicher gespannt 1 Byte bit 6"	%I5.6	BOOL	true		Los	
"S3_Steuerbytes 1 Byte bit 0"	%Q5.0	BOOL	false		Los	010=EINschalten; 101=AUSschalten
"S3_Steuerbytes 1 Byte bit 1"	%Q5.1	BOOL	false		Los	
"S3_Steuerbytes 1 Byte bit 2 Quittierung SEO"	%Q5.2	BOOL	false		Los	
"S4_STATUS-BYTE 1 Byte bit 2"	%I7.2	BOOL	true		Los	01=AUS; 10=EIN;11=hatAusgeloest
"S4_STATUS-BYTE 1 Byte bit 3"	%I7.3	BOOL	false		Los	
"S4_Federspeicher gespannt 1 Byte bit 6"	%I7.6	BOOL	true		Los	
"S4_Steuerbytes 1 Byte bit 0"	%Q7.0	BOOL	false		Los	010=EINschalten; 101=AUSschalten
"S4_Steuerbytes 1 Byte bit 1"	%Q7.1	BOOL	false		Los	
"S4_Steuerbytes 1 Byte bit 2 Quittierung SEO"	%Q7.2	BOOL	false		Los	



Name	Adresse	Anzeigeformat	MonitorWert	ModifyWert	Kommentar
"anzahl mechanische Auslösungen_S1"	%ID9	DEZ+/-	364		
"anzahl mechanische Auslösungen_S2"	%ID51	DEZ+/-	198		
"anzahl mechanische Auslösungen_S3"	%ID55	DEZ+/-	224		
"anzahl mechanische Auslösungen_S4"	%ID59	DEZ+/-	167		
"S1_strom höchst belastete Phase"	%ID41	Gleitpunktzahl	0.0		
"S2_strom höchst belastete Phase"	%ID45	Gleitpunktzahl	0.0		
"S3_strom höchst belastete Phase"	%ID152	Gleitpunktzahl	0.0		
"S4_strom höchst belastete Phase"	%ID156	Gleitpunktzahl	0.0		
"S1_temperatur ETU"	%ID160	Gleitpunktzahl	31.0		



Name	Adresse	Anzeigeformat	MonitorWert	ModifyWert	Kommentar
"Schalter 1"."Strom in Phase L1"	%IW13	DEZ+/-	0		
"Schalter 1"."Strom in Phase L2"	%IW15	DEZ+/-	0		
"Schalter 1"."Strom in Phase L3"	%IW17	DEZ+/-	0		
"Schalter 1"."Maximaler Strom in der höchst belasteten Phase"	%IW19	DEZ+/-	0		
"Schalter 1"."Strom im Neutralleiter"	%IW21	DEZ+/-	0		
"Schalter 1"."Spannung L1-L2"	%IW23	DEZ+/-	0		
"Schalter 1"."Spannung L2-L3"	%IW25	DEZ+/-	0		
"Schalter 1"."Spannung L3-L1"	%IW27	DEZ+/-	0		
"Schalter 1"."Spannung L1-N"	%IW29	DEZ+/-	0		
"Schalter 1"."Spannung L2-N"	%IW31	DEZ+/-	0		
"Schalter 1"."Spannung L3-N"	%IW33	DEZ+/-	0		
"Schalter 1"."Wirkenergie Bezug in 3 Phasen"	%IW37	DEZ+/-	0		
"Schalter 1"."Durchschnittlicher Powerfaktor in 3 Phasen"	%IW35	DEZ+/-	0		
"Schalter 1"."Scheinleistung Summe in 3 Phasen"	%IW39	DEZ+/-	0		

9.2 Python Skript

9.2.1 Ansteuer- bzw. Auswertecode (Hauptcode)

```
"""
+++++++++++++++++++++++++++++++++++++++++++++++++++++++++
| Switching Algorithm - PYTHON-TIA-Portal-3VA2 Switches|
| v19                                                  |
| by Rene GRIESSER                                     |
|                                                      |
| Master Thesis 2021                                   |
|                                                      |
|needs the following files:                            |
|e_output_visualization.py                             |
|IEEE_754_conversion.py                                |
|                                                      |
|  changed lines:  no changes                          |
|                                                      |
|  to ensure that the code is working on every PC...   |
|  make sure to check and modify lines::               |
|  77,304,321,370,387,432,449,494,511,810,1421,1497,   |
|  1544,1558,1744,1778,1782,1804,2068,2099             |
+++++++++++++++++++++++++++++++++++++++++++++++++++++++++
"""


'#-------------------------------------------------------'
'#some information about the file----------------------'
'#-------------------------------------------------------'

# region information
__author__ = "Rene Griesser"
__copyright__ = "Copyright 2021, Griesser"
__credits__ = ["Rene Griesser"]
__license__ = "-"
__version__ = "19"
__maintainer__ = "Rene Griesser"
__email__ = "rene.griesser@student.tugraz.at"
__status__ = "release"
# endregion


'#-------------------------------------------------------'
'#libraries/modules-------------------------------------'
'#-------------------------------------------------------'

import snap7
from snap7.util import *
from e_output_visualization import *
from IEEE_754_conversion import *
import pandas as pd
import time
from datetime import datetime
import sys
import matplotlib.pyplot as plt
import numpy as np
import colorama
from colorama import Fore, Back, Style
colorama.init()


'#-------------------------------------------------------'
'#additional output variable settings------------------'
'#-------------------------------------------------------'
console_output = False
console_output_variables_from_PLC = False
'#-------------------------------------------------------'
'#to be able to generate & save all plots the following '
'#two variables MUST be set TRUE-----------------------'
'#-------------------------------------------------------'
write_values_from_PLC_to_external_file = True
show_plot = False
```

```
'#------------------------------------------------------'
'#external file set up--------------------------------'
'#------------------------------------------------------'
'#if endless report is needed-- change w to a'
'#------------------------------------------------------'

report = open('C:/Users/xiaomi/Desktop/output/report.txt', 'w')


'#------------------------------------------------------'
'#print information-----------------------------------'
'#------------------------------------------------------'
# region print information
print('---------------------------------------')
print(Fore.BLUE + __author__)
print(Fore.BLUE + __copyright__)
print('version:', Fore.RED + __version__)
print('status:', __status__)
print('\033[39m')
print('mail: ', __email__)
print('\033[39m')
print('---------------------------------------')
print()
print()
# endregion


'#------------------------------------------------------'
'#set communication parameters to communicate via ------'
'#PROFINET to PLC1211---------------------------------'
'#------------------------------------------------------'

plc1211 = snap7.client.Client()

plc_ip_address = '192.168.1.50'
TIA_Portal_Rack = 0
TIA_Portal_Slot = 1

plc1211.connect(plc_ip_address, TIA_Portal_Rack, TIA_Portal_Slot)

'#------------------------------------------------------'
'#establish a communication to PLC--------------------'
'#------------------------------------------------------'

plc_connection_status = plc1211.get_connected()
plc_cpu_state = plc1211.get_cpu_state()
plc_last_error = plc1211.get_last_error()
plc_error_text = plc1211.error_text(plc_last_error)


'#------------------------------------------------------'
'#proof for communication to PLC----------------------'
'#------------------------------------------------------'

# region print communication details
print("------------------------")
print("PLC stats---------------")
print("------------------------")
if plc_connection_status == True:
    print(Fore.GREEN + 'PLC is connected: ', plc_connection_status)
    print('\033[39m')
    print('IP-Adress: ', plc_ip_address)
    print('TIA-Portal Rack Position: ', TIA_Portal_Rack)
    print('TIA-Portal Slot Position: ', TIA_Portal_Slot)
else:
    print(Fore.RED + 'PLC is connected: ', plc_connection_status)
    print('\033[39m')
if plc_connection_status == 1:
    print('PLC is in mode: ', plc_cpu_state)
    print('PLC last error code was: ', plc_last_error)
    print('This means PLC is: ', plc_error_text)
    now = datetime.now()
    current_time = now.strftime("%H:%M:%S")
    print("current time is =", current_time)
```

```
    print()
else:
    print("No connection to the given IP-Address: ", plc_ip_address)
    sys.exit('No connection to the given IP-Address')
# endregion


'#-----------------------------------------------------------'
'#read current status of the PLC Outputs before switching---'
'#-----------------------------------------------------------'

bytearray_switch_1and2 = plc1211.ab_read(1, 4)

bytearray_switch_3and4 = plc1211.ab_read(5, 4)

bytearray_merged_switches = bytearray_switch_1and2 + bytearray_switch_3and4


'#-----------------------------------------------------------'
'#read current status of the PLC Inputs before switching----'
'#-----------------------------------------------------------'

input_i_1 = plc1211.eb_read(1, 1)

input_i_3 = plc1211.eb_read(3, 1)

input_i_5 = plc1211.eb_read(5, 1)

input_i_7 = plc1211.eb_read(7, 1)

'#-------------------------------------------------------'
'#print information about the inputs and outputs of PLC-'
'#-------------------------------------------------------'

if console_output == 1:

    print("------------------------")
    print("bytearrays--------------")
    print("------------------------")
    print('bytearray S1+S2 before switching is as follows:    ', bytearray_switch_1and2)
    print('bytearray S3+S4 before switching is as follows:    ', bytearray_switch_3and4)
    print('bytearray PLC outputs before switching is combined as follows: ',
bytearray_merged_switches)
    print()
    print('PLC input1(Switch 1) before switching:    ', input_i_1)
    print('PLC input3(Switch 2) before switching:    ', input_i_3)
    print('PLC input5(Switch 3) before switching:    ', input_i_5)
    print('PLC input7(Switch 4) before switching:    ', input_i_7)
    print()


print("-----------------------------------------")
print("grid configuration before switching-------")
print("-----------------------------------------")


# region if-elif: inputs before switching process for func_print()
if input_i_1 == b'G' and input_i_3 == b'G' and input_i_5 == b'G' and input_i_7 == b'G':
    current_switching_position = func_Print(0)
elif input_i_1 == b'\x0b' and input_i_3 == b'G' and input_i_5 == b'G' and input_i_7 == b'G':
    current_switching_position = func_Print(1)
elif input_i_1 == b'G' and input_i_3 == b'\x0b' and input_i_5 == b'G' and input_i_7 == b'G':
    current_switching_position = func_Print(10)
elif input_i_1 == b'\x0b' and input_i_3 == b'\x0b' and input_i_5 == b'G' and input_i_7 ==
b'G':
    current_switching_position = func_Print(11)
elif input_i_1 == b'G' and input_i_3 == b'G' and input_i_5 == b'\x0b' and input_i_7 == b'G':
    current_switching_position = func_Print(100)
elif input_i_1 == b'\x0b' and input_i_3 == b'G' and input_i_5 == b'\x0b' and input_i_7 ==
b'G':
    current_switching_position = func_Print(101)
elif input_i_1 == b'G' and input_i_3 == b'\x0b' and input_i_5 == b'\x0b' and input_i_7 ==
b'G':
    current_switching_position = func_Print(110)
elif input_i_1 == b'\x0b' and input_i_3 == b'\x0b' and input_i_5 == b'\x0b' and input_i_7 ==
b'G':
    current_switching_position = func_Print(111)
```

```
elif input_i_1 == b'G' and input_i_3 == b'G' and input_i_5 == b'G' and input_i_7 == b'\x0b':
    current_switching_position = func_Print(1000)
elif input_i_1 == b'\x0b' and input_i_3 == b'G' and input_i_5 == b'G' and input_i_7 ==
b'\x0b':
    current_switching_position = func_Print(1001)
elif input_i_1 == b'G' and input_i_3 == b'\x0b' and input_i_5 == b'G' and input_i_7 ==
b'\x0b':
    current_switching_position = func_Print(1010)
elif input_i_1 == b'\x0b' and input_i_3 == b'\x0b' and input_i_5 == b'G' and input_i_7 ==
b'\x0b':
    current_switching_position = func_Print(1011)
elif input_i_1 == b'G' and input_i_3 == b'G' and input_i_5 == b'\x0b' and input_i_7 ==
b'\x0b':
    current_switching_position = func_Print(1100)
elif input_i_1 == b'\x0b' and input_i_3 == b'G' and input_i_5 == b'\x0b' and input_i_7 ==
b'\x0b':
    current_switching_position = func_Print(1101)
elif input_i_1 == b'G' and input_i_3 == b'\x0b' and input_i_5 == b'\x0b' and input_i_7 ==
b'\x0b':
    current_switching_position = func_Print(1110)
elif input_i_1 == b'\x0b' and input_i_3 == b'\x0b' and input_i_5 == b'\x0b' and input_i_7 ==
b'\x0b':
    current_switching_position = func_Print(1111)
elif input_i_1 == b'\x0f' or input_i_3 == b'\x0f' or input_i_5 == b'\x0f' or input_i_7 ==
b'\x0f':
    current_switching_position = func_Print(77)
    print('make sure that no switch is tripped --if any of the switches is tripped(yellow
lamp) reset it (push yellow button first and then the red button of the tripped switch) or do
it via  siemens powerconfig software and repeat the process, if this doesnt work(or no yellow
lamp is on) make this resetation for every switch and restart the program')
    sys.exit('make sure that no switch is tripped --if any of the switches is tripped(yellow
lamp) reset it (push yellow button first and then the red button of the tripped switch) or do
it via  siemens powerconfig software and repeat the process, if this doesnt work(or no yellow
lamp is on) make this resetation for every switch and restart the program')
    # endregion
else:
    current_switching_position = 99
    print('make sure that no switch is tripped --if any of the switches is tripped(yellow
lamp) reset it (push yellow button first and then the red button of the tripped switch) or do
it via  siemens powerconfig software and repeat the process, if this doesnt work(or no yellow
lamp is on) make this resetation for every switch and restart the program')
    sys.exit('make sure that no switch is tripped --if any of the switches is tripped(yellow
lamp) reset it (push yellow button first and then the red button of the tripped switch) or do
it via  siemens powerconfig software and repeat the process, if this doesnt work(or no yellow
lamp is on) make this resetation for every switch and restart the program')
    # endregion
#endregion


'#-------------------------------------------------------'
'#global variables for later output plot generation------'
'#-------------------------------------------------------'

max_phase_current_S1 = 0
max_phase_current_S2 = 0
max_phase_current_S3 = 0
max_phase_current_S4 = 0
max_phase_current_S1_after_switching = 0
max_phase_current_S2_after_switching = 0
max_phase_current_S3_after_switching = 0
max_phase_current_S4_after_switching = 0

'#-------------------------------------------------------'

'#-------------------------------------------------------'
'#read values and print to ex. file if parameter is TRUE-'
'#-------------------------------------------------------'

if write_values_from_PLC_to_external_file == 1:
    k = 1
    l = 1
    print('values before switching', file=report)
    print('values switch 1', file=report)

    values_s1 = plc1211.eb_read(13, 28)

    for i in range(28):
```

```
        if i % 2 == 0:  # even numbers
            j = 0
            if values_s1[i] == 1:
                value = values_s1[i] + (1 * 256)
                j = j + 1
            elif values_s1[i] == 2:
                value = values_s1[i] + (2 * 256)
                j = j + 1
            elif values_s1[i] == 3:
                value = values_s1[i] + (3 * 256)
                j = j + 1
            elif values_s1[i] == 0:
                value = values_s1[i]
                j = j + 1
        if i % 2 != 0:  # odd numbers
            value2 = values_s1[i]
            j = j + 1
            if j == 2:
                if i <= 10:
                    print(value + value2, 'A', file=report)

                    if show_plot == 1 and i <= 6:
                        plt.figure(1)
                        plt.suptitle("Switch 1 -Phase current")
                        plt.subplot(1, 3, k)
                        plt.ylabel("Ampere")
                        plt.ylim([0, 25])
                        plt.bar('phase_ %d' % k, (value + value2), color='red')
                        plt.gcf().set_size_inches(12, 7)

plt.savefig('C:/Users/xiaomi/Desktop/output/plots/Plots_phasenstroeme/figure 1.png')
                        k = k + 1

                    if show_plot == 1 and i > 6 and i <= 10:
                        plt.figure(2)
                        plt.suptitle("Switch 1 - Heavily loaded phase & current in the neutral
conductor")
                        plt.subplot(1, 2, l)
                        plt.ylabel("Ampere")
                        plt.ylim([0, 25])
                        text = 'maximum current in the most heavily loaded phase'
                        if i > 8:
                            text = 'current in the neutral conductor'
                        plt.bar('%s' % text, (value + value2), color="blue")
                        l = l + 1
                        if i > 6 and i <= 8:
                            max_phase_current_S1 = value2
                        plt.gcf().set_size_inches(12, 7)

plt.savefig('C:/Users/xiaomi/Desktop/output/plots/Plots_phasenstroeme/figure 2.png')


                if i > 10 and i < 23:
                    print(value + value2, 'V', file=report)

                if i >= 23:
                    print(value + value2, file=report)

    plt.autoscale()
    plt.draw()

    print('values switch 2', file=report)

    values_s2 = plc1211.eb_read(68, 28)

    k = 1
    l = 1
    for i in range(28):

        if i % 2 == 0:  # even numbers
            j = 0
            if values_s2[i] == 1:
                value = values_s2[i] + 256
                j = j + 1
            elif values_s2[i] == 2:
                value = values_s2[i] + (2 * 256)
                j = j + 1
            elif values_s2[i] == 3:
```

```
                value = values_s2[i] + (3 * 256)
                j = j + 1
            elif values_s2[i] == 0:
                value = values_s2[i]
                j = j + 1
        if i % 2 != 0:  # odd numbers
            value2 = values_s2[i]
            j = j + 1
            if j == 2:
                if i <= 10:
                    print(value + value2, 'A', file=report)

                    if show_plot == 1 and i <= 6:
                        plt.figure(3)
                        plt.suptitle("Switch 2 -Phase current")
                        plt.subplot(1, 3, k)
                        plt.ylabel("Ampere")
                        plt.ylim([0, 25])
                        plt.bar('phase_ %d' % k, (value + value2), color='red')
                        plt.gcf().set_size_inches(12, 7)

plt.savefig('C:/Users/xiaomi/Desktop/output/plots/Plots_phasenstroeme/figure 3.png')
                        k = k + 1

                    if show_plot == 1 and i > 6 and i <=10:
                        plt.figure(4)
                        plt.suptitle("Switch 2 - Heavily loaded phase & current in the neutral
conductor")
                        plt.subplot(1, 2, l)
                        plt.ylabel("Ampere")
                        plt.ylim([0, 25])
                        text = 'maximum current in the most heavily loaded phase'
                        if i > 8:
                            text = 'current in the neutral conductor'
                        plt.bar('%s' % text, (value + value2), color="blue")
                        l = l + 1
                        if i > 6 and i <= 8:
                            max_phase_current_S2 = value2
                        plt.gcf().set_size_inches(12, 7)

plt.savefig('C:/Users/xiaomi/Desktop/output/plots/Plots_phasenstroeme/figure 4.png')

                if i > 10 and i < 23:
                    print(value + value2, 'V', file=report)

                if i >= 23:
                    print(value + value2, file=report)

    print('values switch 3', file=report)

    values_s3 = plc1211.eb_read(96, 28)

    k = 1
    l = 1
    for i in range(28):

        if i % 2 == 0:  # even numbers
            j = 0
            if values_s3[i] == 1:
                value = values_s3[i] + 256
                j = j + 1
            elif values_s3[i] == 2:
                value = values_s3[i] + (2 * 256)
                j = j + 1
            elif values_s3[i] == 3:
                value = values_s3[i] + (3 * 256)
                j = j + 1
            elif values_s3[i] == 0:
                value = values_s3[i]
                j = j + 1
        if i % 2 != 0:  # odd numbers
            value2 = values_s3[i]
            j = j + 1
            if j == 2:
                if i <= 10:
                    print(value + value2, 'A', file=report)

                    if show_plot == 1 and i <= 6:
```

```
                        plt.figure(5)
                        plt.suptitle("Switch 3 -Phase current")
                        plt.subplot(1, 3, k)
                        plt.ylabel("Ampere")
                        plt.ylim([0, 25])
                        plt.bar('phase_ %d' % k, (value + value2), color='red')
                        plt.gcf().set_size_inches(12, 7)

plt.savefig('C:/Users/xiaomi/Desktop/output/plots/Plots_phasenstroeme/figure 5.png')
                        k = k + 1

                    if show_plot == 1 and i > 6 and i <= 10:
                        plt.figure(6)
                        plt.suptitle("Switch 3 - Heavily loaded phase & current in the neutral
conductor")
                        plt.subplot(1, 2, l)
                        plt.ylabel("Ampere")
                        plt.ylim([0, 25])
                        text = 'maximum current in the most heavily loaded phase'
                        if i > 8:
                            text = 'current in the neutral conductor'
                        plt.bar('%s' % text, (value + value2), color="blue")
                        l = l + 1
                        if i > 6 and i <= 8:
                            max_phase_current_S3 = value2
                        plt.gcf().set_size_inches(12, 7)

plt.savefig('C:/Users/xiaomi/Desktop/output/plots/Plots_phasenstroeme/figure 6.png')

                if i > 10 and i < 23:
                    print(value + value2, 'V', file=report)

                if i >= 23:
                    print(value + value2, file=report)

    print('values switch 4', file=report)

    values_s4 = plc1211.eb_read(124, 28)

    k = 1
    l = 1
    for i in range(28):

        if i % 2 == 0:  # even numbers
            j = 0
            if values_s4[i] == 1:
                value = values_s4[i] + 256
                j = j + 1
            elif values_s4[i] == 2:
                value = values_s4[i] + (2 * 256)
                j = j + 1
            elif values_s4[i] == 3:
                value = values_s4[i] + (3 * 256)
                j = j + 1
            elif values_s4[i] == 0:
                value = values_s4[i]
                j = j + 1
        if i % 2 != 0:  # odd numbers
            value2 = values_s4[i]
            j = j + 1
            if j == 2:
                if i <= 10:
                    print(value + value2, 'A', file=report)

                    if show_plot == 1 and i <= 6:
                        plt.figure(7)
                        plt.suptitle("Switch 4 -Phase current")
                        plt.subplot(1, 3, k)
                        plt.ylabel("Ampere")
                        plt.ylim([0, 25])
                        plt.bar('phase_ %d' % k, (value + value2), color='red')
                        plt.gcf().set_size_inches(12, 7)

plt.savefig('C:/Users/xiaomi/Desktop/output/plots/Plots_phasenstroeme/figure 7.png')
                        k = k + 1

                    if show_plot == 1 and i > 6 and i <= 10:
                        plt.figure(8)
```

```
                        plt.suptitle("Switch 4 - Heavily loaded phase & current in the neutral
conductor")
                        plt.subplot(1, 2, l)
                        plt.ylabel("Ampere")
                        plt.ylim([0, 25])
                        text = 'maximum current in the most heavily loaded phase'
                        if i > 8:
                            text = 'current in the neutral conductor'
                        plt.bar('%s' % text, (value + value2), color="blue")
                        l = l + 1
                        if i > 6 and i <= 8:
                            max_phase_current_S4 = value2
                        plt.gcf().set_size_inches(12, 7)

plt.savefig('C:/Users/xiaomi/Desktop/output/plots/Plots_phasenstroeme/figure 8.png')

                if i > 10 and i < 23:
                    print(value + value2, 'V', file=report)

                if i >= 23:
                    print(value + value2, file=report)

    print('number of switching processes', file=report)

    number_sw_process = plc1211.eb_read(9, 4)

    value1 = 0
    for i in range(4):
        if number_sw_process[i] == 0:
            value = number_sw_process[i]
        elif number_sw_process[i] == 1:
            value = number_sw_process[i] + (1 * 256)
        elif number_sw_process[i] == 2:
            value = number_sw_process[i] + (2 * 256)
        elif number_sw_process[i] == 3:
            value = number_sw_process[i] + (3 * 256)
        elif number_sw_process[i] >= 4:
            value = number_sw_process[i]
        value1 = value1 + value

    print('number of switching processes switch 1:', value1, file=report)

    number_sw_process = plc1211.eb_read(51, 4)

    value1 = 0
    for i in range(4):
        if number_sw_process[i] == 0:
            value = number_sw_process[i]
        elif number_sw_process[i] == 1:
            value = number_sw_process[i] + (1 * 256)
        elif number_sw_process[i] == 2:
            value = number_sw_process[i] + (2 * 256)
        elif number_sw_process[i] == 3:
            value = number_sw_process[i] + (3 * 256)
        elif number_sw_process[i] >= 4:
            value = number_sw_process[i]
        value1 = value1 + value

    print('number of switching processes switch 2:', value1, file=report)

    number_sw_process = plc1211.eb_read(55, 4)

    value1 = 0
    for i in range(4):
        if number_sw_process[i] == 0:
            value = number_sw_process[i]
        elif number_sw_process[i] == 1:
            value = number_sw_process[i] + 256
        elif number_sw_process[i] == 2:
            value = number_sw_process[i] + 2 * 256
        elif number_sw_process[i] == 3:
            value = number_sw_process[i] + 3 * 256
        elif number_sw_process[i] >= 4:
            value = number_sw_process[i]
        value1 = value1 + value

    print('number of switching processes switch 3:', value1, file=report)
```

```
    number_sw_process = plc1211.eb_read(59, 4)

    value1 = 0
    for i in range(4):
        if number_sw_process[i] == 0:
            value = number_sw_process[i]
        elif number_sw_process[i] == 1:
            value = number_sw_process[i] + 256
        elif number_sw_process[i] == 2:
            value = number_sw_process[i] + 2 * 256
        elif number_sw_process[i] == 3:
            value = number_sw_process[i] + 3 * 256
        elif number_sw_process[i] >= 4:
            value = number_sw_process[i]
        value1 = value1 + value

    print('number of switching processes switch 4:', value1, file=report)
    print('-------------------------------------------------------------------------------
- ', file=report)


'#-----------------------------------------------------'
'#IEEE-754-FLOAT REPRESENTATION-----------------------'
'#-----------------------------------------------------'


'------------------switch 1----------------------------'
a = 0
b = 0
c = 0
d = 0

values_s1_float = plc1211.eb_read(41, 4)
value_FP1 = 0
for i in range(4):
    value_FP1 += values_s1_float[i]
    a = values_s1_float[0]
    b = values_s1_float[1]
    c = values_s1_float[2]
    d = values_s1_float[3]

bnr1 = bin(a).replace('0b','')
x = bnr1[::-1] #reverse array
while len(x) < 8:
    x += '0'
bnr1 = x[::-1]

bnr2 = bin(b).replace('0b','')
x = bnr2[::-1] #reverse array
while len(x) < 8:
    x += '0'
bnr2 = x[::-1]

bnr3 = bin(c).replace('0b','')
x = bnr3[::-1] #reverse array
while len(x) < 8:
    x += '0'
bnr3 = x[::-1]

bnr4 = bin(d).replace('0b','')
x = bnr4[::-1] #reverse array
while len(x) < 8:
    x += '0'
bnr4 = x[::-1]

bit32value = concat(bnr1, bnr2, bnr3, bnr4)

string1 = str(bit32value) # convert into string for func IEEE

binary2 = int(string1, 2)

FP_value1 = (ieee_754_conversion(binary2))


'------------------switch 2----------------------------'
a = 0
b = 0
c = 0
```

```
d = 0

values_s2_float = plc1211.eb_read(45, 4)
value_FP2 = 0
for i in range(4):
    value_FP2 += values_s2_float[i]
    a = values_s2_float[0]
    b = values_s2_float[1]
    c = values_s2_float[2]
    d = values_s2_float[3]

bnr1 = bin(a).replace('0b','')
x = bnr1[::-1] #reverse array
while len(x) < 8:
    x += '0'
bnr1 = x[::-1]

bnr2 = bin(b).replace('0b','')
x = bnr2[::-1] #reverse array
while len(x) < 8:
    x += '0'
bnr2 = x[::-1]

bnr3 = bin(c).replace('0b','')
x = bnr3[::-1] #reverse array
while len(x) < 8:
    x += '0'
bnr3 = x[::-1]

bnr4 = bin(d).replace('0b','')
x = bnr4[::-1] #reverse array
while len(x) < 8:
    x += '0'
bnr4 = x[::-1]

bit32value = concat(bnr1, bnr2, bnr3, bnr4)

string1 = str(bit32value) # convert into string for func IEEE

binary2 = int(string1, 2)

FP_value2 = (ieee_754_conversion(binary2))


'------------------switch 3----------------------------'
a = 0
b = 0
c = 0
d = 0

values_s3_float = plc1211.eb_read(152, 4)
value_FP3 = 0
for i in range(4):
    value_FP3 += values_s3_float[i]
    a = values_s3_float[0]
    b = values_s3_float[1]
    c = values_s3_float[2]
    d = values_s3_float[3]

bnr1 = bin(a).replace('0b','')
x = bnr1[::-1] #reverse array
while len(x) < 8:
    x += '0'
bnr1 = x[::-1]

bnr2 = bin(b).replace('0b','')
x = bnr2[::-1] #reverse array
while len(x) < 8:
    x += '0'
bnr2 = x[::-1]

bnr3 = bin(c).replace('0b','')
x = bnr3[::-1] #reverse array
while len(x) < 8:
    x += '0'
bnr3 = x[::-1]

bnr4 = bin(d).replace('0b','')
```

```
x = bnr4[::-1] #reverse array
while len(x) < 8:
    x += '0'
bnr4 = x[::-1]

bit32value = concat(bnr1, bnr2, bnr3, bnr4)

string1 = str(bit32value) # convert into string for func IEEE

binary2 = int(string1, 2)

FP_value3 = (ieee_754_conversion(binary2))


'------------------switch 4-----------------------------'
a = 0
b = 0
c = 0
d = 0

values_s4_float = plc1211.eb_read(156, 4)
value_FP4 = 0
for i in range(4):
    value_FP4 += values_s4_float[i]
    a = values_s4_float[0]
    b = values_s4_float[1]
    c = values_s4_float[2]
    d = values_s4_float[3]

bnr1 = bin(a).replace('0b','')
x = bnr1[::-1] #reverse array
while len(x) < 8:
    x += '0'
bnr1 = x[::-1]

bnr2 = bin(b).replace('0b','')
x = bnr2[::-1] #reverse array
while len(x) < 8:
    x += '0'
bnr2 = x[::-1]

bnr3 = bin(c).replace('0b','')
x = bnr3[::-1] #reverse array
while len(x) < 8:
    x += '0'
bnr3 = x[::-1]

bnr4 = bin(d).replace('0b','')
x = bnr4[::-1] #reverse array
while len(x) < 8:
    x += '0'
bnr4 = x[::-1]

bit32value = concat(bnr1, bnr2, bnr3, bnr4)

string1 = str(bit32value) # convert into string for func IEEE

binary2 = int(string1, 2)

FP_value4 = (ieee_754_conversion(binary2))


FP_value1_rounded_BEFORE = FP_value1.__round__(2)
FP_value2_rounded_BEFORE = FP_value2.__round__(2)
FP_value3_rounded_BEFORE = FP_value3.__round__(2)
FP_value4_rounded_BEFORE = FP_value4.__round__(2)
if console_output == 1:
    print('Floating Point variables of max. phase currents follow:')
    print(FP_value1_rounded_BEFORE)
    print(FP_value2_rounded_BEFORE)
    print(FP_value3_rounded_BEFORE)
    print(FP_value4_rounded_BEFORE)


# region read .CSV
'#-----------------------------------------------------'
'#read .csv from external KPI sheet--------------------'
'#-----------------------------------------------------'
```

```
df = pd.read_csv('C:/Users/xiaomi/Desktop/output/switchtable.csv', sep=';')

df = df.loc[:, ['Variation', 'Binary variation']]  # same as print(df.iloc[:,[0,1]]) ; i for
integer

best_switching_positions = df['Binary variation'][0]

if console_output == 1:
    print("best_switching_positions === %4.0f" % best_switching_positions)
# endregion


# region IF ELIF switching process

'#------------------------------------------------------'
'#compare switch positions and modify switch positions--'
'#------------------------------------------------------'

if current_switching_position == best_switching_positions:
    print('current position of the switches is ideal, no need for a switching process')
else:
    print('current position of the switches is not ideal, therefore switching process is
inevitable')

    if best_switching_positions == 0:

        plc1211.ab_write(1, b'\x01\x00\x01\x00')
        time.sleep(1)
        plc1211.ab_write(5, b'\x01\x00\x01\x00')
        '#very important because of the reset of the SEO!!!!'
        time.sleep(5)

        new_bytearray_merged_switches = b'\x01\x00\x01\x00\x01\x00\x01\x00'

    elif best_switching_positions == 1:

        plc1211.ab_write(1, b'\x02\x00\x01\x00')
        time.sleep(1)
        plc1211.ab_write(5, b'\x01\x00\x01\x00')
        time.sleep(5)

        new_bytearray_merged_switches = b'\x02\x00\x01\x00\x01\x00\x01\x00'

    elif best_switching_positions == 10:

        # region main switch S1 on ?
        '#------------------------------------------------------'
        '#main switch S1 on ?--------------------------------'
        '#------------------------------------------------------'
        #plc1211.ab_write(1, b'\x01\x00\x02\x00')
        #time.sleep(1)
        #plc1211.ab_write(5, b'\x01\x00\x01\x00')
        #time.sleep(5)
        new_bytearray_merged_switches = b'\x01\x00\x02\x00\x01\x00\x01\x00'

        if new_bytearray_merged_switches[0] == 1 or new_bytearray_merged_switches[0] == 0:
            print('main switch after switching process turned off')
            print('no switching will be performed')
            sys.exit('CAUTION: for safety reasons switching is not be performed__main switch
OFF after switching')
        # endregion

    elif best_switching_positions == 11:

        plc1211.ab_write(1, b'\x02\x00\x02\x00')
        time.sleep(1)
        plc1211.ab_write(5, b'\x01\x00\x01\x00')
        '#very important because of the reset of the SEO!!!!'
        time.sleep(5)

        new_bytearray_merged_switches = b'\x02\x00\x02\x00\x01\x00\x01\x00'

    elif best_switching_positions == 100:

        # region main switch S1 on ?
        '#------------------------------------------------------'
```

```
        '#main switch S1 on ?---------------------------------'
        '#---------------------------------------------------'
        # plc1211.ab_write(1, b'\x01\x00\x01\x00')
        # time.sleep(1)
        # plc1211.ab_write(5, b'\x02\x00\x01\x00')
        # time.sleep(5)
        new_bytearray_merged_switches = b'\x01\x00\x01\x00\x02\x00\x01\x00'

        if new_bytearray_merged_switches[0] == 1 or new_bytearray_merged_switches[0] == 0:
            print('main switch after switching process turned off')
            print('no switching will be performed')
            sys.exit('CAUTION: for safety reasons switching is not be performed__main switch
OFF after switching ')
        # endregion

    elif best_switching_positions == 101:

        plc1211.ab_write(1, b'\x02\x00\x01\x00')
        time.sleep(1)
        plc1211.ab_write(5, b'\x02\x00\x01\x00')
        '#very important because of the reset of the SEO!!!!'
        time.sleep(5)

        new_bytearray_merged_switches = b'\x02\x00\x01\x00\x02\x00\x01\x00'

    elif best_switching_positions == 110:

        # region main switch S1 on ?
        '#---------------------------------------------------'
        '#main switch S1 on ?---------------------------------'
        '#---------------------------------------------------'
        # plc1211.ab_write(1, b'\x01\x00\x02\x00')
        # time.sleep(1)
        # plc1211.ab_write(5, b'\x02\x00\x01\x00')
        # time.sleep(5)
        new_bytearray_merged_switches = b'\x01\x00\x02\x00\x02\x00\x01\x00'

        if new_bytearray_merged_switches[0] == 1 or new_bytearray_merged_switches[0] == 0:
            print('main switch after switching process turned off')
            print('no switching will be performed')
            sys.exit('CAUTION: for safety reasons switching is not be performed__main switch
OFF after switching ')
        # endregion

    elif best_switching_positions == 111:

        plc1211.ab_write(1, b'\x02\x00\x02\x00')
        time.sleep(1)
        plc1211.ab_write(5, b'\x02\x00\x01\x00')
        '#very important because of the reset of the SEO!!!!'
        time.sleep(5)

        new_bytearray_merged_switches = b'\x02\x00\x02\x00\x02\x00\x01\x00'

    elif best_switching_positions == 1000:

        # region main switch S1 on ?
        '#---------------------------------------------------'
        '#main switch S1 on ?---------------------------------'
        '#---------------------------------------------------'
        # plc1211.ab_write(1, b'\x01\x00\x01\x00')
        # time.sleep(1)
        # plc1211.ab_write(5, b'\x01\x00\x02\x00')
        # time.sleep(5)
        new_bytearray_merged_switches = b'\x01\x00\x01\x00\x01\x00\x02\x00'

        if new_bytearray_merged_switches[0] == 1 or new_bytearray_merged_switches[0] == 0:
            print('main switch after switching process turned off')
            print('no switching will be performed')
            sys.exit('CAUTION: for safety reasons switching is not be performed__main switch
OFF after switching ')
        # endregion

    elif best_switching_positions == 1001:

        # region main switch S1 on ?
        '#---------------------------------------------------'
        '#main switch S1 on ?---------------------------------'
```

```
        '#-----------------------------------------------------'
        # plc1211.ab_write(1, b'\x02\x00\x01\x00')
        # time.sleep(1)
        # plc1211.ab_write(5, b'\x01\x00\x02\x00')
        # time.sleep(5)
        new_bytearray_merged_switches = b'\x02\x00\x01\x00\x01\x00\x02\x00'
        if new_bytearray_merged_switches[0] == 1 or new_bytearray_merged_switches[0] == 0:
            print('main switch after switching process turned off')
            print('no switching will be performed')
            sys.exit(
                'CAUTION: for safety reasons switching is not be performed__main switch ON but
only S4 is ON_means \
                 no departures! ')
        if new_bytearray_merged_switches[6] == 2 and new_bytearray_merged_switches[2] == 1 and
\
                new_bytearray_merged_switches[4] == 1:
            print('no switching will be performed')
            sys.exit('CAUTION_S4: for safety reasons switching is not be performed__main
switch ON but only S4 is \
            ON_means no departures!')
        # endregion

    elif best_switching_positions == 1010:

        # region main switch S1 on ?
        '#-----------------------------------------------------'
        '#main switch S1 on ?-----------------------------------'
        '#-----------------------------------------------------'
        # plc1211.ab_write(1, b'\x02\x00\x01\x00')
        # time.sleep(1)
        # plc1211.ab_write(5, b'\x02\x00\x01\x00')
        # time.sleep(5)
        new_bytearray_merged_switches = b'\x01\x00\x02\x00\x01\x00\x02\x00'

        if new_bytearray_merged_switches[0] == 1 or new_bytearray_merged_switches[0] == 0:
            print('main switch after switching process turned off')
            print('no switching will be performed')
            sys.exit('CAUTION: for safety reasons switching is not be performed__main switch
OFF after switching ')
        # endregion

    elif best_switching_positions == 1011:

        plc1211.ab_write(1, b'\x02\x00\x02\x00')
        time.sleep(1)
        plc1211.ab_write(5, b'\x01\x00\x02\x00')
        '#very important because of the reset of the SEO!!!!'
        time.sleep(5)

        new_bytearray_merged_switches = b'\x02\x00\x02\x00\x01\x00\x02\x00'

    elif best_switching_positions == 1100:

        # region main switch S1 on ?
        '#-----------------------------------------------------'
        '#main switch S1 on ?-----------------------------------'
        '#-----------------------------------------------------'
        # plc1211.ab_write(1, b'\x02\x00\x02\x00')
        # plc1211.ab_write(5, b'\x01\x00\x01\x00')
        # time.sleep(5)
        new_bytearray_merged_switches = b'\x01\x00\x01\x00\x02\x00\x02\x00'

        if new_bytearray_merged_switches[0] == 1 or new_bytearray_merged_switches[0] == 0:
            print('main switch after switching process turned off')
            print('no switching will be performed')
            sys.exit('CAUTION: for safety reasons switching is not be performed__main switch
OFF after switching')
            # endregion

    elif best_switching_positions == 1101:

        plc1211.ab_write(1, b'\x02\x00\x01\x00')
        '#very important to not override the tripping (yellow) lamp!!!!'
        time.sleep(1)
        plc1211.ab_write(5, b'\x02\x00\x02\x00')
        '#very important because of the reset of the SEO!!!!'
        time.sleep(5)
```

```
        new_bytearray_merged_switches = b'\x02\x00\x01\x00\x02\x00\x02\x00'

    elif best_switching_positions == 1110:

        # region main switch S1 on ?
        '#-----------------------------------------------------'
        '#main switch S1 on ?-----------------------------------'
        '#-----------------------------------------------------'
        # plc1211.ab_write(1, b'\x02\x00\x02\x00')
        # time.sleep(1)
        # plc1211.ab_write(5, b'\x02\x00\x01\x00')
        # time.sleep(5)
        new_bytearray_merged_switches = b'\x01\x00\x02\x00\x02\x00\x02\x00'

        if new_bytearray_merged_switches[0] == 1 or new_bytearray_merged_switches[0] == 0:
            print('main switch after switching process turned off')
            print('no switching will be performed')
            sys.exit('CAUTION: for safety reasons switching is not be performed__main switch
OFF after switching ')
            # endregion

    elif best_switching_positions == 1111:

        plc1211.ab_write(1, b'\x02\x00\x02\x00')
        time.sleep(1)
        plc1211.ab_write(5, b'\x02\x00\x02\x00')
        '#very important because of the reset of the SEO!!!!'
        time.sleep(5)

        new_bytearray_merged_switches = b'\x02\x00\x02\x00\x02\x00\x02\x00'

    else:

        print("CAUTION: The given Switching position can't be performed cause of it's unknown
format/value only boolean 0000 to 1111 and all variations of it")
        sys.exit("CAUTION: The given Switching position can't be performed cause of it's
unknown format/value only boolean 0000 to 1111 and all variations of it")
# endregion


# region read inputs from PLC after switching
'#-----------------------------------------------------------'
'#read current status of the PLC Inputs after switching-----'
'#-----------------------------------------------------------'

input_i_1 = plc1211.eb_read(1, 1)

input_i_3 = plc1211.eb_read(3, 1)

input_i_5 = plc1211.eb_read(5, 1)

input_i_7 = plc1211.eb_read(7, 1)
# endregion

if console_output == 1:

    print()
    print('PLC input1(Switch 1) after switching:    ', input_i_1)
    print('PLC input2(Switch 2) after switching:    ', input_i_3)
    print('PLC input3(Switch 3) after switching:    ', input_i_5)
    print('PLC input4(Switch 4) after switching:    ', input_i_7)
    print()

print("-----------------------------------------")
print("grid configuration after switching--------")
print("-----------------------------------------")

# region if-elif: inputs after switching process for func_print()
if input_i_1 == b'G' and input_i_3 == b'G' and input_i_5 == b'G' and input_i_7 == b'G':
    current_switching_position_new = func_Print(0)
elif input_i_1 == b'\x0b' and input_i_3 == b'G' and input_i_5 == b'G' and input_i_7 == b'G':
    current_switching_position_new = func_Print(1)
elif input_i_1 == b'G' and input_i_3 == b'\x0b' and input_i_5 == b'G' and input_i_7 == b'G':
    current_switching_position_new = func_Print(10)
elif input_i_1 == b'\x0b' and input_i_3 == b'\x0b' and input_i_5 == b'G' and input_i_7 ==
b'G':
    current_switching_position_new = func_Print(11)
elif input_i_1 == b'G' and input_i_3 == b'G' and input_i_5 == b'\x0b' and input_i_7 == b'G':
```

```
    current_switching_position_new = func_Print(100)
elif input_i_1 == b'\x0b' and input_i_3 == b'G' and input_i_5 == b'\x0b' and input_i_7 ==
b'G':
    current_switching_position_new = func_Print(101)
elif input_i_1 == b'G' and input_i_3 == b'\x0b' and input_i_5 == b'\x0b' and input_i_7 ==
b'G':
    current_switching_position_new = func_Print(110)
elif input_i_1 == b'\x0b' and input_i_3 == b'\x0b' and input_i_5 == b'\x0b' and input_i_7 ==
b'G':
    current_switching_position_new = func_Print(111)
elif input_i_1 == b'G' and input_i_3 == b'G' and input_i_5 == b'G' and input_i_7 == b'\x0b':
    current_switching_position_new = func_Print(1000)
elif input_i_1 == b'\x0b' and input_i_3 == b'G' and input_i_5 == b'G' and input_i_7 ==
b'\x0b':
    current_switching_position_new = func_Print(1001)
elif input_i_1 == b'G' and input_i_3 == b'\x0b' and input_i_5 == b'G' and input_i_7 ==
b'\x0b':
    current_switching_position_new = func_Print(1010)
elif input_i_1 == b'\x0b' and input_i_3 == b'\x0b' and input_i_5 == b'G' and input_i_7 ==
b'\x0b':
    current_switching_position_new = func_Print(1011)
elif input_i_1 == b'G' and input_i_3 == b'G' and input_i_5 == b'\x0b' and input_i_7 ==
b'\x0b':
    current_switching_position_new = func_Print(1100)
elif input_i_1 == b'\x0b' and input_i_3 == b'G' and input_i_5 == b'\x0b' and input_i_7 ==
b'\x0b':
    current_switching_position_new = func_Print(1101)
elif input_i_1 == b'G' and input_i_3 == b'\x0b' and input_i_5 == b'\x0b' and input_i_7 ==
b'\x0b':
    current_switching_position_new = func_Print(1110)
elif input_i_1 == b'\x0b' and input_i_3 == b'\x0b' and input_i_5 == b'\x0b' and input_i_7 ==
b'\x0b':
    current_switching_position_new = func_Print(1111)
elif input_i_1 == b'\x0f' or input_i_3 == b'\x0f' or input_i_5 == b'\x0f' or input_i_7 ==
b'\x0f':
    current_switching_position = func_Print(77)
    sys.exit('make sure that no switch is tripped --if any of the switches is tripped(yellow
lamp) reset it (push yellow button first and then the red button of the tripped switch) or do
it via  siemens powerconfig software and repeat the process')
    # endregion
else:
    current_switching_position_new = 99
    print('Error current switching positions does not match any given switching positions')
# endregion


# region make a reset of the outputs of the PLC because of otherwise the command for turn
on/off won't work anymore
'#-----------------------------------------------------'
'#RESET very important----------------------------------'
'#-----------------------------------------------------'
plc1211.ab_write(1, b'\x00\x00\x00\x00')
plc1211.ab_write(5, b'\x00\x00\x00\x00')
# endregion


# region Visualize the difference
'#-----------------------------------------------------'
'#Visualization-----------------------------------------'
'#-----------------------------------------------------'
colorama.init()

print(f"{Back.RED}configuration before switching process")
func_Print(current_switching_position)
print(f"{Back.GREEN}configuration after switching process")
func_Print(current_switching_position_new)
colorama.init(autoreset=True)
print()
if current_switching_position == current_switching_position_new:
    print(Fore.BLUE + 'current switching position is ideal, therefore no switching is
performed')

# endregion


'#-----------------------------------------------------'
'#read values and print to console if parameter is TRUE-'
'#-----------------------------------------------------'
```

```
if console_output_variables_from_PLC == 1:

    print()
    print('values switch 1')
    print()

    values_s1 = plc1211.eb_read(13, 28)

    for i in range(28):

        if i % 2 == 0: #even numbers
            j = 0
            if values_s1[i] == 1:
                value = values_s1[i] + 256
                j = j + 1
            elif values_s1[i] == 2:
                value = values_s1[i] + (2 * 256)
                j = j + 1
            elif values_s1[i] == 3:
                value = values_s1[i] + (3 * 256)
                j = j + 1
            elif values_s1[i] == 0:
                value = values_s1[i]
                j = j + 1
        if i % 2 != 0: #odd numbers
            value2 = values_s1[i]
            j = j + 1
            if j == 2:
                if i <= 10:
                    print(value + value2, 'A')
                if i > 10 and i < 23:
                    print(value + value2, 'V')
                if i >= 23:
                    print(value + value2)

    print()
    print('values switch 2')
    print()
    values_s2 = plc1211.eb_read(68, 28)

    for i in range(28):

        if i % 2 == 0: #even numbers
            j = 0
            if values_s2[i] == 1:
                value = values_s2[i] + 256
                j = j + 1
            elif values_s2[i] == 2:
                value = values_s2[i] + (2 * 256)
                j = j + 1
            elif values_s2[i] == 3:
                value = values_s2[i] + (3 * 256)
                j = j + 1
            elif values_s2[i] == 0:
                value = values_s2[i]
                j = j + 1
        if i % 2 != 0: #odd numbers
            value2 = values_s2[i]
            j = j + 1
            if j == 2:
                if i <= 10:
                    print(value + value2, 'A')
                if i > 10 and i < 23:
                    print(value + value2, 'V')
                if i >= 23:
                    print(value + value2)

    print()
    print('values switch 3')
    print()

    values_s3 = plc1211.eb_read(96, 28)

    for i in range(28):

        if i % 2 == 0: #even numbers
            j = 0
```

```
            if values_s3[i] == 1:
                value = values_s3[i] + 256
                j = j + 1
            elif values_s3[i] == 2:
                value = values_s3[i] + (2 * 256)
                j = j + 1
            elif values_s3[i] == 3:
                value = values_s3[i] + (3 * 256)
                j = j + 1
            elif values_s3[i] == 0:
                value = values_s3[i]
                j = j + 1
        if i % 2 != 0: #odd numbers
            value2 = values_s3[i]
            j = j + 1
            if j == 2:
                if i <= 10:
                    print(value + value2, 'A')
                if i > 10 and i < 23:
                    print(value + value2, 'V')
                if i >= 23:
                    print(value + value2)

    print()
    print('values switch 4')
    print()
    values_s4 = plc1211.eb_read(124, 28)

    for i in range(28):

        if i % 2 == 0: #even numbers
            j = 0
            if values_s4[i] == 1:
                value = values_s4[i] + 256
                j = j + 1
            elif values_s4[i] == 2:
                value = values_s4[i] + (2 * 256)
                j = j + 1
            elif values_s4[i] == 3:
                value = values_s4[i] + (3 * 256)
                j = j + 1
            elif values_s4[i] == 0:
                value = values_s4[i]
                j = j + 1
        if i % 2 != 0: #odd numbers
            value2 = values_s4[i]
            j = j + 1
            if j == 2:
                if i <= 10:
                    print(value + value2, 'A')
                if i > 10 and i < 23:
                    print(value + value2, 'V')
                if i >= 23:
                    print(value + value2)

    print()
    print('number of switching processes')
    print()

    number_sw_process = plc1211.eb_read(9, 4)

    value1 = 0
    for i in range(4):
        if number_sw_process[i] == 0:
            value = number_sw_process[i]
        elif number_sw_process[i] == 1:
            value = number_sw_process[i] + 256
        elif number_sw_process[i] == 2:
            value = number_sw_process[i] + 2*256
        elif number_sw_process[i] == 3:
            value = number_sw_process[i] + 3*256
        elif number_sw_process[i] >= 4:
            value = number_sw_process[i]
        value1 = value1 + value

    print('number of switching processes switch 1:', value1)

    number_sw_process = plc1211.eb_read(51, 4)
```

```
    value1 = 0
    for i in range(4):
        if number_sw_process[i] == 0:
            value = number_sw_process[i]
        elif number_sw_process[i] == 1:
            value = number_sw_process[i] + 256
        elif number_sw_process[i] == 2:
            value = number_sw_process[i] + 2*256
        elif number_sw_process[i] == 3:
            value = number_sw_process[i] + 3*256
        elif number_sw_process[i] >= 4:
            value = number_sw_process[i]
        value1 = value1 + value

    print('number of switching processes switch 2:', value1)

    number_sw_process = plc1211.eb_read(55, 4)

    value1 = 0
    for i in range(4):
        if number_sw_process[i] == 0:
            value = number_sw_process[i]
        elif number_sw_process[i] == 1:
            value = number_sw_process[i] + 256
        elif number_sw_process[i] == 2:
            value = number_sw_process[i] + 2*256
        elif number_sw_process[i] == 3:
            value = number_sw_process[i] + 3*256
        elif number_sw_process[i] >= 4:
            value = number_sw_process[i]
        value1 = value1 + value

    print('number of switching processes switch 3:', value1)

    number_sw_process = plc1211.eb_read(59, 4)

    value1 = 0
    for i in range(4):
        if number_sw_process[i] == 0:
            value = number_sw_process[i]
        elif number_sw_process[i] == 1:
            value = number_sw_process[i] + 256
        elif number_sw_process[i] == 2:
            value = number_sw_process[i] + 2*256
        elif number_sw_process[i] == 3:
            value = number_sw_process[i] + 3*256
        elif number_sw_process[i] >= 4:
            value = number_sw_process[i]
        value1 = value1 + value

    print('number of switching processes switch 4:', value1)

'#-------------------------------------------------------'
'#read values and print to ex. file if parameter is TRUE-'
'#-------------------------------------------------------'

if write_values_from_PLC_to_external_file == 1:

    print('values after switching', file=report)
    print('values switch 1', file=report)

    values_s1 = plc1211.eb_read(13, 28)

    k = 1
    l = 1
    for i in range(28):

        if i % 2 == 0:  # even numbers
            j = 0
            if values_s1[i] == 1:
                value = values_s1[i] + 256
                j = j + 1
            elif values_s1[i] == 2:
                value = values_s1[i] + (2 * 256)
                j = j + 1
            elif values_s1[i] == 3:
                value = values_s1[i] + (3 * 256)
```

```
                j = j + 1
            elif values_s1[i] == 0:
                value = values_s1[i]
                j = j + 1
        if i % 2 != 0:  # odd numbers
            value2 = values_s1[i]
            j = j + 1
            if j == 2:
                if i <= 10:
                    print(value + value2, 'A', file=report)

                    if show_plot == 1 and i <= 6:
                        plt.figure(11)
                        plt.suptitle("Switch 1 - AFTER SWITCHING -Phase current")
                        plt.subplot(1, 3, k)
                        plt.ylabel("Ampere")
                        plt.ylim([0, 25])
                        plt.bar('phase_ %d' % k, (value + value2), color='red')
                        plt.gcf().set_size_inches(12, 7)

plt.savefig('C:/Users/xiaomi/Desktop/output/plots/Plots_phasenstroeme/figure 11.png')
                        k = k + 1

                    if show_plot == 1 and i > 6 and i <= 10:
                        plt.figure(12)
                        plt.suptitle("Switch 1 - AFTER SWITCHING - Heavily loaded phase &
current in the neutral conductor")
                        plt.subplot(1, 2, l)
                        plt.ylabel("Ampere")
                        plt.ylim([0, 25])
                        text = 'maximum current in the most heavily loaded phase'
                        if i > 8:
                            text = 'current in the neutral conductor'
                        plt.bar('%s' % text, (value + value2), color="blue")
                        plt.gcf().set_size_inches(12, 7)

plt.savefig('C:/Users/xiaomi/Desktop/output/plots/Plots_phasenstroeme/figure 12.png')
                        l = l + 1
                        if i > 6 and i <= 8:
                            max_phase_current_S1_after_switching = value2

                if i > 10 and i < 23:
                    print(value + value2, 'V', file=report)

                if i >= 23:
                    print(value + value2, file=report)

    print('values switch 2', file=report)

    values_s2 = plc1211.eb_read(68, 28)

    k = 1
    l = 1
    for i in range(28):

        if i % 2 == 0:  # even numbers
            j = 0
            if values_s2[i] == 1:
                value = values_s2[i] + 256
                j = j + 1
            elif values_s2[i] == 2:
                value = values_s2[i] + (2 * 256)
                j = j + 1
            elif values_s2[i] == 3:
                value = values_s2[i] + (3 * 256)
                j = j + 1
            elif values_s2[i] == 0:
                value = values_s2[i]
                j = j + 1
        if i % 2 != 0:  # odd numbers
            value2 = values_s2[i]
            j = j + 1
            if j == 2:
                if i <= 10:
                    print(value + value2, 'A', file=report)

                    if show_plot == 1 and i <= 6:
                        plt.figure(13)
```

```
                        plt.suptitle("Switch 2 - AFTER SWITCHING -Phase current")
                        plt.subplot(1, 3, k)
                        plt.ylabel("Ampere")
                        plt.ylim([0, 25])
                        plt.bar('phase_ %d' % k, (value + value2), color='red')
                        plt.gcf().set_size_inches(12, 7)

plt.savefig('C:/Users/xiaomi/Desktop/output/plots/Plots_phasenstroeme/figure 13.png')
                        k = k + 1

                    if show_plot == 1 and i > 6 and i <= 10:
                        plt.figure(14)
                        plt.suptitle("Switch 2 - AFTER SWITCHING - Heavily loaded phase &
current in the neutral conductor")
                        plt.subplot(1, 2, l)
                        plt.ylabel("Ampere")
                        plt.ylim([0, 25])
                        text = 'maximum current in the most heavily loaded phase'
                        if i > 8:
                            text = 'current in the neutral conductor'
                        plt.bar('%s' % text, (value + value2), color="blue")
                        plt.gcf().set_size_inches(12, 7)

plt.savefig('C:/Users/xiaomi/Desktop/output/plots/Plots_phasenstroeme/figure 14.png')
                        l = l + 1
                        if i > 6 and i <= 8:
                            max_phase_current_S2_after_switching = value2
                if i > 10 and i < 23:
                    print(value + value2, 'V', file=report)

                if i >= 23:
                    print(value + value2, file=report)

    print('values switch 3', file=report)

    values_s3 = plc1211.eb_read(96, 28)

    k = 1
    l = 1
    for i in range(28):

        if i % 2 == 0:  # even numbers
            j = 0
            if values_s3[i] == 1:
                value = values_s3[i] + 256
                j = j + 1
            elif values_s3[i] == 2:
                value = values_s3[i] + (2 * 256)
                j = j + 1
            elif values_s3[i] == 3:
                value = values_s3[i] + (3 * 256)
                j = j + 1
            elif values_s3[i] == 0:
                value = values_s3[i]
                j = j + 1
        if i % 2 != 0:  # odd numbers
            value2 = values_s3[i]
            j = j + 1
            if j == 2:
                if i <= 10:
                    print(value + value2, 'A', file=report)

                    if show_plot == 1 and i <= 6:
                        plt.figure(15)
                        plt.suptitle("Switch 3 - AFTER SWITCHING -Phase current")
                        plt.subplot(1, 3, k)
                        plt.ylabel("Ampere")
                        plt.ylim([0, 25])
                        plt.bar('phase_ %d' % k, (value + value2), color='red')
                        plt.gcf().set_size_inches(12, 7)

plt.savefig('C:/Users/xiaomi/Desktop/output/plots/Plots_phasenstroeme/figure 15.png')
                        k = k + 1

                    if show_plot == 1 and i > 6 and i <= 10:
                        plt.figure(16)
                        plt.suptitle("Switch 3 - AFTER SWITCHING - Heavily loaded phase &
current in the neutral conductor")
```

```
                        plt.subplot(1, 2, l)
                        plt.ylabel("Ampere")
                        plt.ylim([0, 25])
                        text = 'maximum current in the most heavily loaded phase'
                        if i > 8:
                            text = 'current in the neutral conductor'
                        plt.bar('%s' % text, (value + value2), color="blue")
                        plt.gcf().set_size_inches(12, 7)

plt.savefig('C:/Users/xiaomi/Desktop/output/plots/Plots_phasenstroeme/figure 16.png')
                        l = l + 1
                        if i > 6 and i <= 8:
                            max_phase_current_S3_after_switching = value2
                if i > 10 and i < 23:
                    print(value + value2, 'V', file=report)

                if i >= 23:
                    print(value + value2, file=report)

    print('values switch 4', file=report)

    values_s4 = plc1211.eb_read(124, 28)

    k = 1
    l = 1
    for i in range(28):

        if i % 2 == 0:  # even numbers
            j = 0
            if values_s4[i] == 1:
                value = values_s4[i] + 256
                j = j + 1
            elif values_s4[i] == 2:
                value = values_s4[i] + (2 * 256)
                j = j + 1
            elif values_s4[i] == 3:
                value = values_s4[i] + (3 * 256)
                j = j + 1
            elif values_s4[i] == 0:
                value = values_s4[i]
                j = j + 1
        if i % 2 != 0:  # odd numbers
            value2 = values_s4[i]
            j = j + 1
            if j == 2:
                if i <= 10:
                    print(value + value2, 'A', file=report)

                    if show_plot == 1 and i <= 6:
                        plt.figure(17)
                        plt.suptitle("Switch 4 - AFTER SWITCHING -Phase current")
                        plt.subplot(1, 3, k)
                        plt.ylabel("Ampere")
                        plt.ylim([0, 25])
                        plt.bar('phase_ %d' % k, (value + value2), color='red')
                        plt.gcf().set_size_inches(12, 7)

plt.savefig('C:/Users/xiaomi/Desktop/output/plots/Plots_phasenstroeme/figure 17.png')
                        k = k + 1

                    if show_plot == 1 and i > 6 and i <= 10:
                        plt.figure(18)
                        plt.suptitle("Switch 4 - AFTER SWITCHING - Heavily loaded phase &
current in the neutral conductor")
                        plt.subplot(1, 2, l)
                        plt.ylabel("Ampere")
                        plt.ylim([0, 25])
                        text = 'maximum current in the most heavily loaded phase'
                        if i > 8:
                            text = 'current in the neutral conductor'
                        plt.bar('%s' % text, (value + value2), color="blue")
                        plt.gcf().set_size_inches(12, 7)

plt.savefig('C:/Users/xiaomi/Desktop/output/plots/Plots_phasenstroeme/figure 18.png')
                        l = l + 1
                        if i > 6 and i <= 8:
                            max_phase_current_S4_after_switching = value2
                if i > 10 and i < 23:
```

```
                print(value + value2, 'V', file=report)

            if i >= 23:
                print(value + value2, file=report)

print('number of switching processes', file=report)

number_sw_process = plc1211.eb_read(9, 4)

value1 = 0
for i in range(4):
    if number_sw_process[i] == 0:
        value = number_sw_process[i]
    elif number_sw_process[i] == 1:
        value = number_sw_process[i] + 256
    elif number_sw_process[i] == 2:
        value = number_sw_process[i] + 2 * 256
    elif number_sw_process[i] == 3:
        value = number_sw_process[i] + 3 * 256
    elif number_sw_process[i] >= 4:
        value = number_sw_process[i]
    value1 = value1 + value

print('number of switching processes switch 1:', value1, file=report)

number_sw_process = plc1211.eb_read(51, 4)

value1 = 0
for i in range(4):
    if number_sw_process[i] == 0:
        value = number_sw_process[i]
    elif number_sw_process[i] == 1:
        value = number_sw_process[i] + 256
    elif number_sw_process[i] == 2:
        value = number_sw_process[i] + 2 * 256
    elif number_sw_process[i] == 3:
        value = number_sw_process[i] + 3 * 256
    elif number_sw_process[i] >= 4:
        value = number_sw_process[i]
    value1 = value1 + value

print('number of switching processes switch 2:', value1, file=report)

number_sw_process = plc1211.eb_read(55, 4)

value1 = 0
for i in range(4):
    if number_sw_process[i] == 0:
        value = number_sw_process[i]
    elif number_sw_process[i] == 1:
        value = number_sw_process[i] + 256
    elif number_sw_process[i] == 2:
        value = number_sw_process[i] + 2 * 256
    elif number_sw_process[i] == 3:
        value = number_sw_process[i] + 3 * 256
    elif number_sw_process[i] >= 4:
        value = number_sw_process[i]
    value1 = value1 + value

print('number of switching processes switch 3:', value1, file=report)

number_sw_process = plc1211.eb_read(59, 4)

value1 = 0
for i in range(4):
    if number_sw_process[i] == 0:
        value = number_sw_process[i]
    elif number_sw_process[i] == 1:
        value = number_sw_process[i] + 256
    elif number_sw_process[i] == 2:
        value = number_sw_process[i] + 2 * 256
    elif number_sw_process[i] == 3:
        value = number_sw_process[i] + 3 * 256
    elif number_sw_process[i] >= 4:
        value = number_sw_process[i]
    value1 = value1 + value

print('number of switching processes switch 4:', value1, file=report)
```

```
'#-------------------------------------------------------'
'#close external file ---------------------------------'
'#-------------------------------------------------------'

report.close()

'#-----------------------------------------------------------'
'#plot the figures at the end so that the code can continue-'
'#-----------------------------------------------------------'

if show_plot == 1:
    plt.tight_layout(pad=0.4, w_pad=0.5, h_pad=1.0)
    plt.show()



'#------------------before switching----------------------'
if show_plot == 1:
    fig, ax = plt.subplots()

    max_phase_currents_circuit_breakers = ('S1', 'S2', 'S3', 'S4')
    phase_current_data = [max_phase_current_S1, max_phase_current_S2, max_phase_current_S3,
max_phase_current_S4]
    y_pos = np.arange(len(max_phase_currents_circuit_breakers))
    couleur = ['#2e42d3']

    plt.title('max.phase current of circuit breakers BEFORE switching procedure')
    plt.ylabel('Current in Ampere')
    plt.xlabel("Circuit Breakers")

    heading = plt.bar(y_pos, phase_current_data, align='center', alpha=0.6, color=couleur)

    ax.set_xticks(range(len(max_phase_currents_circuit_breakers)))
    ax.set_xticklabels(max_phase_currents_circuit_breakers, rotation='horizontal')


    def label_on_top(rects):

        for rect in rects:
         height = rect.get_height()
         ax.text(rect.get_x() + rect.get_width()/2., height + 0, '%.1f' % float(height),
ha='center', va='bottom')

    label_on_top(heading)
    plt.ylim(0, 30)
    '-------------'
    plt.gcf().set_size_inches(7, 7)
    plt.savefig('C:/Users/xiaomi/Desktop/output/plots/vorher_naher/max.phase current of
circuit breakers BEFORE switching procedure {0}.png'.format(0))


'#------------------after switching----------------------'
if show_plot == 1:

    fig, ax = plt.subplots()

    max_phase_currents_circuit_breakers_after_switching = ('S1', 'S2', 'S3', 'S4')
    phase_current_data_after_switching = [max_phase_current_S1_after_switching,
max_phase_current_S2_after_switching, max_phase_current_S3_after_switching,
max_phase_current_S4_after_switching]
    y_pos = np.arange(len(max_phase_currents_circuit_breakers_after_switching))
    couleur = ['#2e42d3']

    plt.title('max.phase current of circuit breakers AFTER switching procedure')
    plt.ylabel('Current in Ampere')
    plt.xlabel("Circuit Breakers")


    heading = plt.bar(y_pos, phase_current_data_after_switching, align='center', alpha=0.6,
color=couleur)

    ax.set_xticks(range(len(max_phase_currents_circuit_breakers_after_switching)))
    ax.set_xticklabels(max_phase_currents_circuit_breakers_after_switching,
rotation='horizontal')


    def label_on_top(rects):
```

```
        for rect in rects:
         height = rect.get_height()
         ax.text(rect.get_x() + rect.get_width()/2., height + 0, '%.1f' % float(height),
ha='center', va='bottom')

    label_on_top(heading)
    plt.ylim(0, 30)
    '-------------'
    plt.gcf().set_size_inches(7, 7)
    plt.savefig('C:/Users/xiaomi/Desktop/output/plots/vorher_naher/max.phase current of
circuit breakers AFTER switching procedure {0}.png'.format(1))
    #plt.show()


datafile = open('C:/Users/xiaomi/Desktop/output/variables description for report file.txt',
'w')
datafile.write("\r ------------------------------")
datafile.write("\r| Master Thesis 2021           |")
datafile.write("\r| by Rene GRIESSER             |")
datafile.write("\r ------------------------------\n")
datafile.write("\rBasistype 3 measurement data- reference to the report.txt file\n\n")
datafile.write("\rvalue 1 -- current in Phase L1")
datafile.write("\rvalue 2 -- current in Phase L2")
datafile.write("\rvalue 3 -- current in Phase L3")
datafile.write("\rvalue 4 -- current in the heavily loaded phase")
datafile.write("\rvalue 5 -- current in neutral conductor")
datafile.write("\rvalue 6 -- Voltage L1-L2")
datafile.write("\rvalue 7 -- Voltage L3-L2")
datafile.write("\rvalue 8 -- Voltage L3-L1")
datafile.write("\rvalue 9 -- Voltage L1-N")
datafile.write("\rvalue 10 -- Voltage L2-N")
datafile.write("\rvalue 11 -- Voltage L3-N")
datafile.write("\rvalue 12 -- averafe powerfactor")
datafile.write("\rvalue 13 -- effective energy reference")
datafile.write("\rvalue 14 -- apparent power")
datafile.close()

datafile = open('C:/Users/xiaomi/Desktop/output/figure description.txt', 'w')
datafile.write("\r ------------------------------")
datafile.write("\r| Master Thesis 2021           |")
datafile.write("\r| by Rene GRIESSER             |")
datafile.write("\r ------------------------------\n")
datafile.write("\rfigure description\n\n")
datafile.write("\rfigure 1 -- Switch 1 _ all 3 phase currents BEFORE switching")
datafile.write("\rfigure 2 -- Switch 1 _ max.current in the heavily loaded phase + current in
the neutral conductor BEFORE switching")
datafile.write("\rfigure 3 -- Switch 2 _ all 3 phase currents BEFORE switching")
datafile.write("\rfigure 4 -- Switch 2 _ max.current in the heavily loaded phase + current in
the neutral conductor BEFORE switching")
datafile.write("\rfigure 5 -- Switch 3 _ all 3 phase currents BEFORE switching")
datafile.write("\rfigure 6 -- Switch 3 _ max.current in the heavily loaded phase + current in
the neutral conductor BEFORE switching")
datafile.write("\rfigure 7 -- Switch 4 _ all 3 phase currents BEFORE switching")
datafile.write("\rfigure 8 -- Switch 4 _ max.current in the heavily loaded phase + current in
the neutral conductor BEFORE switching")
datafile.write("\rfigure 11 -- Switch 1 _ all 3 phase currents AFTER switching")
datafile.write("\rfigure 12 -- Switch 1 _ max.current in the heavily loaded phase + current in
the neutral conductor AFTER switching")
datafile.write("\rfigure 13 -- Switch 2 _ all 3 phase currents AFTER switching")
datafile.write("\rfigure 14 -- Switch 2 _ max.current in the heavily loaded phase + current in
the neutral conductor AFTER switching")
datafile.write("\rfigure 15 -- Switch 3 _ all 3 phase currents AFTER switching")
datafile.write("\rfigure 16 -- Switch 3 _ max.current in the heavily loaded phase + current in
the neutral conductor AFTER switching")
datafile.write("\rfigure 17 -- Switch 4 _ all 3 phase currents AFTER switching")
datafile.write("\rfigure 18 -- Switch 4 _ max.current in the heavily loaded phase + current in
the neutral conductor AFTER switching")
datafile.close()

#
# time.sleep(1)
```

```
'#-----------------------------------------------------'
'#IEEE-754-FLOAT REPRESENTATION-----------------------'
'#-----------------------------------------------------'


'------------------switch 1----------------------------'
a = 0
b = 0
c = 0
d = 0

values_s1_float = plc1211.eb_read(41, 4)
value_FP1 = 0
for i in range(4):
    value_FP1 += values_s1_float[i]
    a = values_s1_float[0]
    b = values_s1_float[1]
    c = values_s1_float[2]
    d = values_s1_float[3]

bnr1 = bin(a).replace('0b','')
x = bnr1[::-1] #reverse array
while len(x) < 8:
    x += '0'
bnr1 = x[::-1]

bnr2 = bin(b).replace('0b','')
x = bnr2[::-1] #reverse array
while len(x) < 8:
    x += '0'
bnr2 = x[::-1]

bnr3 = bin(c).replace('0b','')
x = bnr3[::-1] #reverse array
while len(x) < 8:
    x += '0'
bnr3 = x[::-1]

bnr4 = bin(d).replace('0b','')
x = bnr4[::-1] #reverse array
while len(x) < 8:
    x += '0'
bnr4 = x[::-1]

bit32value = concat(bnr1, bnr2, bnr3, bnr4)

string1 = str(bit32value) # convert into string for func IEEE

binary2 = int(string1, 2)

FP_value1 = (ieee_754_conversion(binary2))


'------------------switch 2----------------------------'
a = 0
b = 0
c = 0
d = 0

values_s2_float = plc1211.eb_read(45, 4)
value_FP2 = 0
for i in range(4):
    value_FP2 += values_s2_float[i]
    a = values_s2_float[0]
    b = values_s2_float[1]
    c = values_s2_float[2]
    d = values_s2_float[3]

bnr1 = bin(a).replace('0b','')
x = bnr1[::-1] #reverse array
while len(x) < 8:
    x += '0'
bnr1 = x[::-1]

bnr2 = bin(b).replace('0b','')
x = bnr2[::-1] #reverse array
while len(x) < 8:
    x += '0'
```

```
bnr2 = x[::-1]

bnr3 = bin(c).replace('0b','')
x = bnr3[::-1] #reverse array
while len(x) < 8:
    x += '0'
bnr3 = x[::-1]

bnr4 = bin(d).replace('0b','')
x = bnr4[::-1] #reverse array
while len(x) < 8:
    x += '0'
bnr4 = x[::-1]

bit32value = concat(bnr1, bnr2, bnr3, bnr4)

string1 = str(bit32value) # convert into string for func IEEE

binary2 = int(string1, 2)

FP_value2 = (ieee_754_conversion(binary2))


'------------------switch 3-----------------------------'
a = 0
b = 0
c = 0
d = 0

values_s3_float = plc1211.eb_read(152, 4)
value_FP3 = 0
for i in range(4):
    value_FP3 += values_s3_float[i]
    a = values_s3_float[0]
    b = values_s3_float[1]
    c = values_s3_float[2]
    d = values_s3_float[3]

bnr1 = bin(a).replace('0b','')
x = bnr1[::-1] #reverse array
while len(x) < 8:
    x += '0'
bnr1 = x[::-1]

bnr2 = bin(b).replace('0b','')
x = bnr2[::-1] #reverse array
while len(x) < 8:
    x += '0'
bnr2 = x[::-1]

bnr3 = bin(c).replace('0b','')
x = bnr3[::-1] #reverse array
while len(x) < 8:
    x += '0'
bnr3 = x[::-1]

bnr4 = bin(d).replace('0b','')
x = bnr4[::-1] #reverse array
while len(x) < 8:
    x += '0'
bnr4 = x[::-1]

bit32value = concat(bnr1, bnr2, bnr3, bnr4)

string1 = str(bit32value) # convert into string for func IEEE

binary2 = int(string1, 2)

FP_value3 = (ieee_754_conversion(binary2))


'------------------switch 4-----------------------------'
a = 0
b = 0
c = 0
d = 0

values_s4_float = plc1211.eb_read(156, 4)
```

```
value_FP4 = 0
for i in range(4):
    value_FP4 += values_s4_float[i]
    a = values_s4_float[0]
    b = values_s4_float[1]
    c = values_s4_float[2]
    d = values_s4_float[3]

bnr1 = bin(a).replace('0b','')
x = bnr1[::-1] #reverse array
while len(x) < 8:
    x += '0'
bnr1 = x[::-1]

bnr2 = bin(b).replace('0b','')
x = bnr2[::-1] #reverse array
while len(x) < 8:
    x += '0'
bnr2 = x[::-1]

bnr3 = bin(c).replace('0b','')
x = bnr3[::-1] #reverse array
while len(x) < 8:
    x += '0'
bnr3 = x[::-1]

bnr4 = bin(d).replace('0b','')
x = bnr4[::-1] #reverse array
while len(x) < 8:
    x += '0'
bnr4 = x[::-1]

bit32value = concat(bnr1, bnr2, bnr3, bnr4)

string1 = str(bit32value) # convert into string for func IEEE

binary2 = int(string1, 2)

FP_value4 = (ieee_754_conversion(binary2))


FP_value1_rounded = FP_value1.__round__(2)
FP_value2_rounded = FP_value2.__round__(2)
FP_value3_rounded = FP_value3.__round__(2)
FP_value4_rounded = FP_value4.__round__(2)
if console_output == 1:
    print(FP_value1_rounded)
    print(FP_value2_rounded)
    print(FP_value3_rounded)
    print(FP_value4_rounded)

'#------------------BEFORE switching--------------------'
if show_plot == 1:
    fig, ax = plt.subplots()

    max_phase_currents_circuit_breakers_FP = ('S1', 'S2', 'S3', 'S4')
    phase_current_data_FP = [FP_value1_rounded_BEFORE, FP_value2_rounded_BEFORE,
FP_value3_rounded_BEFORE, FP_value4_rounded_BEFORE]
    y_pos = np.arange(len(max_phase_currents_circuit_breakers_FP))
    couleur = ['#2e42d3']

    plt.title('max.phase current of circuit breakers BEFORE switching procedure in Ampere')
    plt.ylabel('Current in Ampere')
    plt.xlabel("Circuit Breakers")

    heading = plt.bar(y_pos, phase_current_data_FP, align='center', alpha=0.6, color=couleur)

    ax.set_xticks(range(len(max_phase_currents_circuit_breakers_FP)))
    ax.set_xticklabels(max_phase_currents_circuit_breakers_FP, rotation='horizontal')


    def label_on_top(rects):

        for rect in rects:
            height = rect.get_height()
            ax.text(rect.get_x() + rect.get_width() / 2., height + 0, '%.1f' % float(height),
ha='center', va='bottom')
```

```
    label_on_top(heading)
    plt.ylim(0, 30)
    '-------------'
    plt.gcf().set_size_inches(10, 7)
    plt.savefig('C:/Users/xiaomi/Desktop/output/plots/vorher_naher/Floating POINT_max.phase
current of circuit breakers BEFORE switching procedure {0}.png'.format(0))

'#------------------FP_AFTER switching--------------------'
if show_plot == 1:
    fig, ax = plt.subplots()

    max_phase_currents_circuit_breakers_FP = ('S1', 'S2', 'S3', 'S4')
    phase_current_data_FP = [FP_value1_rounded, FP_value2_rounded, FP_value3_rounded,
FP_value4_rounded]
    y_pos = np.arange(len(max_phase_currents_circuit_breakers_FP))
    couleur = ['#2e42d3']

    plt.title('max.phase current of circuit breakers AFTER switching procedure in Ampere')
    plt.ylabel('Current in Ampere')
    plt.xlabel("Circuit Breakers")

    heading = plt.bar(y_pos, phase_current_data_FP, align='center', alpha=0.6, color=couleur)

    ax.set_xticks(range(len(max_phase_currents_circuit_breakers_FP)))
    ax.set_xticklabels(max_phase_currents_circuit_breakers_FP, rotation='horizontal')


    def label_on_top(rects):

        for rect in rects:
         height = rect.get_height()
         ax.text(rect.get_x() + rect.get_width()/2., height + 0, '%.1f' % float(height),
ha='center', va='bottom')

    label_on_top(heading)
    plt.ylim(0, 30)
    '-------------'
    plt.gcf().set_size_inches(10, 7)
    plt.savefig('C:/Users/xiaomi/Desktop/output/plots/vorher_naher/Floating POINT_max.phase
current of circuit breakers AFTER switching procedure {0}.png'.format(0))
    plt.show()

'#------------------additional output in German-----------------'

if show_plot == 1:

    breakers = np.array(["S1", "S2", "S3", "S4"])
    y = np.array([FP_value1_rounded_BEFORE, FP_value2_rounded_BEFORE,
FP_value3_rounded_BEFORE, FP_value4_rounded_BEFORE])
    plt.suptitle("Strom in der höchstbelasteten Phase je Schalter vor- und nach dem
Schaltvorgang", fontsize = 15)

    plt.subplot(2, 2, 1)
    plt.ylim(0, 30)
    plt.title('Szenario X')
    plt.ylabel('Strom in Ampere')
    plt.xlabel("Kompaktleistungsschalter")

    plt.bar(breakers, y, color="red", width = 0.1)
    for index, data in enumerate(y):
        plt.text(x=index - 0.07, y=data + 0.75, s=f"{data}", fontdict=dict(fontsize=8))

    y = np.array([FP_value1_rounded, FP_value2_rounded, FP_value3_rounded, FP_value4_rounded])
    plt.subplot(2,2,3)
    plt.ylim(0, 30)
    plt.title('Szenario Y')
    plt.ylabel('Strom in Ampere')
    plt.xlabel("Kompaktleistungsschalter")

    plt.bar(breakers, y, color="green", width = 0.1)
    for index, data in enumerate(y):
        plt.text(x=index - 0.07, y=data + 0.75, s=f"{data}", fontdict=dict(fontsize=8))
    plt.gcf().set_size_inches(10, 7)

    plt.subplot(1, 2, 2)

    Kompaktleistungsschaler = ['S1', 'S2', 'S3', 'S4']
```

```
    Schaltvorgang = ['VOR', 'NACH']
    pos = np.arange(len(Kompaktleistungsschaler))
    bar_width = 0.05
    daten_vor_schaltvorgang = [FP_value1_rounded_BEFORE, FP_value2_rounded_BEFORE,
FP_value3_rounded_BEFORE, FP_value4_rounded_BEFORE]
    daten_nach_schaltvorgang = [FP_value1_rounded, FP_value2_rounded, FP_value3_rounded,
FP_value4_rounded]
    plt.tight_layout()
    plt.bar(pos, daten_vor_schaltvorgang, bar_width, color='red')
    for index, data in enumerate(daten_vor_schaltvorgang):
        plt.text(x=index - 0.035, y=data + 0.25, s=f"{data}", fontdict=dict(fontsize=8))
    plt.bar(pos + bar_width, daten_nach_schaltvorgang, bar_width, color='green')
    for index, data in enumerate(daten_nach_schaltvorgang):
        plt.text(x=index + 0.02, y=data + 0.25, s=f"{data}", fontdict=dict(fontsize=8))
    plt.xticks(pos, Kompaktleistungsschaler)
    plt.title('Ströme vor- und nach dem Schaltvorgang')
    plt.xlabel('Kompaktleistungsschalter')
    plt.ylabel('Strom in Ampere')
    plt.ylim(0, 30)

    plt.legend(Schaltvorgang, loc=1)
    plt.gcf().set_size_inches(12, 7)
    plt.subplots_adjust(left=0.125,
                        bottom=0.1,
                        right=0.9,
                        top=0.9,
                        wspace=0.25,
                        hspace=0.45)
    plt.gcf().set_size_inches(10, 7)
    plt.savefig('C:/Users/xiaomi/Desktop/output/plots/vorher_naher/Uebersichtsgrafik
{0}.png'.format(1))
    plt.show()



    breakers = np.array(["S1", "S2", "S3", "S4"])
    y = np.array([FP_value1_rounded_BEFORE, FP_value2_rounded_BEFORE,
FP_value3_rounded_BEFORE, FP_value4_rounded_BEFORE])
    plt.suptitle("Strom in der höchstbelasteten Phase je Schalter vor- und nach dem
Schaltvorgang", fontsize = 15)

    plt.subplot(2, 2, 1)
    plt.ylim(0, 30)
    plt.title('Szenario X')
    plt.ylabel('Strom in Ampere')
    plt.xlabel("Kompaktleistungsschalter")

    plt.bar(breakers, y, color="red", width = 0.1)
    for index, data in enumerate(y):
        plt.text(x=index - 0.07, y=data + 0.75, s=f"{data}", fontdict=dict(fontsize=8))

    y = np.array([FP_value1_rounded, FP_value2_rounded, FP_value3_rounded, FP_value4_rounded])
    plt.subplot(2,2,2)
    plt.ylim(0, 30)
    plt.title('Szenario Y')
    plt.ylabel('Strom in Ampere')
    plt.xlabel("Kompaktleistungsschalter")
    plt.bar(breakers, y, color="green", width = 0.1)
    for index, data in enumerate(y):
        plt.text(x=index - 0.07, y=data + 0.75, s=f"{data}", fontdict=dict(fontsize=8))
    plt.gcf().set_size_inches(10, 7)
    plt.subplot(2, 1, 2)
    Kompaktleistungsschaler = ['S1', 'S2', 'S3', 'S4']
    Schaltvorgang = ['VOR Schaltvorgang', 'NACH Schaltvorgang']
    pos = np.arange(len(Kompaktleistungsschaler))
    bar_width = 0.05
    daten_vor_schaltvorgang = [FP_value1_rounded_BEFORE, FP_value2_rounded_BEFORE,
FP_value3_rounded_BEFORE, FP_value4_rounded_BEFORE]
    daten_nach_schaltvorgang = [FP_value1_rounded, FP_value2_rounded, FP_value3_rounded,
FP_value4_rounded]
    plt.tight_layout()
    plt.bar(pos, daten_vor_schaltvorgang, bar_width, color='red')
    for index, data in enumerate(daten_vor_schaltvorgang):
        plt.text(x=index - 0.06, y=data + 0.75, s=f"{data}", fontdict=dict(fontsize=8))
    plt.bar(pos + bar_width, daten_nach_schaltvorgang, bar_width, color='green')
    for index, data in enumerate(daten_nach_schaltvorgang):
        plt.text(x=index + 0.05, y=data + 0.75, s=f"{data}", fontdict=dict(fontsize=8))
    plt.xticks(pos, Kompaktleistungsschaler)
```

```
    plt.title('Ströme vor- und nach dem Schaltvorgang')
    plt.xlabel('Kompaktleistungsschalter')
    plt.ylabel('Strom in Ampere')
    plt.ylim(0, 30)
    plt.legend(Schaltvorgang, loc=1)
    plt.gcf().set_size_inches(12, 7)
    plt.subplots_adjust(left=0.125,
                        bottom=0.1,
                        right=0.9,
                        top=0.9,
                        wspace=0.25,
                        hspace=0.45)
    plt.gcf().set_size_inches(10, 7)
    plt.savefig('C:/Users/xiaomi/Desktop/output/plots/vorher_naher/Uebersichtsgrafik
{0}.png'.format(2))
    plt.show()

'#-------------------------------------------------------'
'#-------------------------------------------------------'
'#-------------------------------------------------------'
'#-------------------------------------------------------'
"""
+++++++++++++++++++++++++++++++++++++++++++++++++++++++++
|                                                       |
| Switching Algorithm - PYTHON-TIA-Portal-3VA2 Switches|
| v19                                                   |
| by Rene GRIESSER                                      |
| Master Thesis 2021                                    |
|                                                       |
+++++++++++++++++++++++++++++++++++++++++++++++++++++++++
"""
```

9.2.2 Visualisierungscode (Funktionsaufruf im Hauptcode)

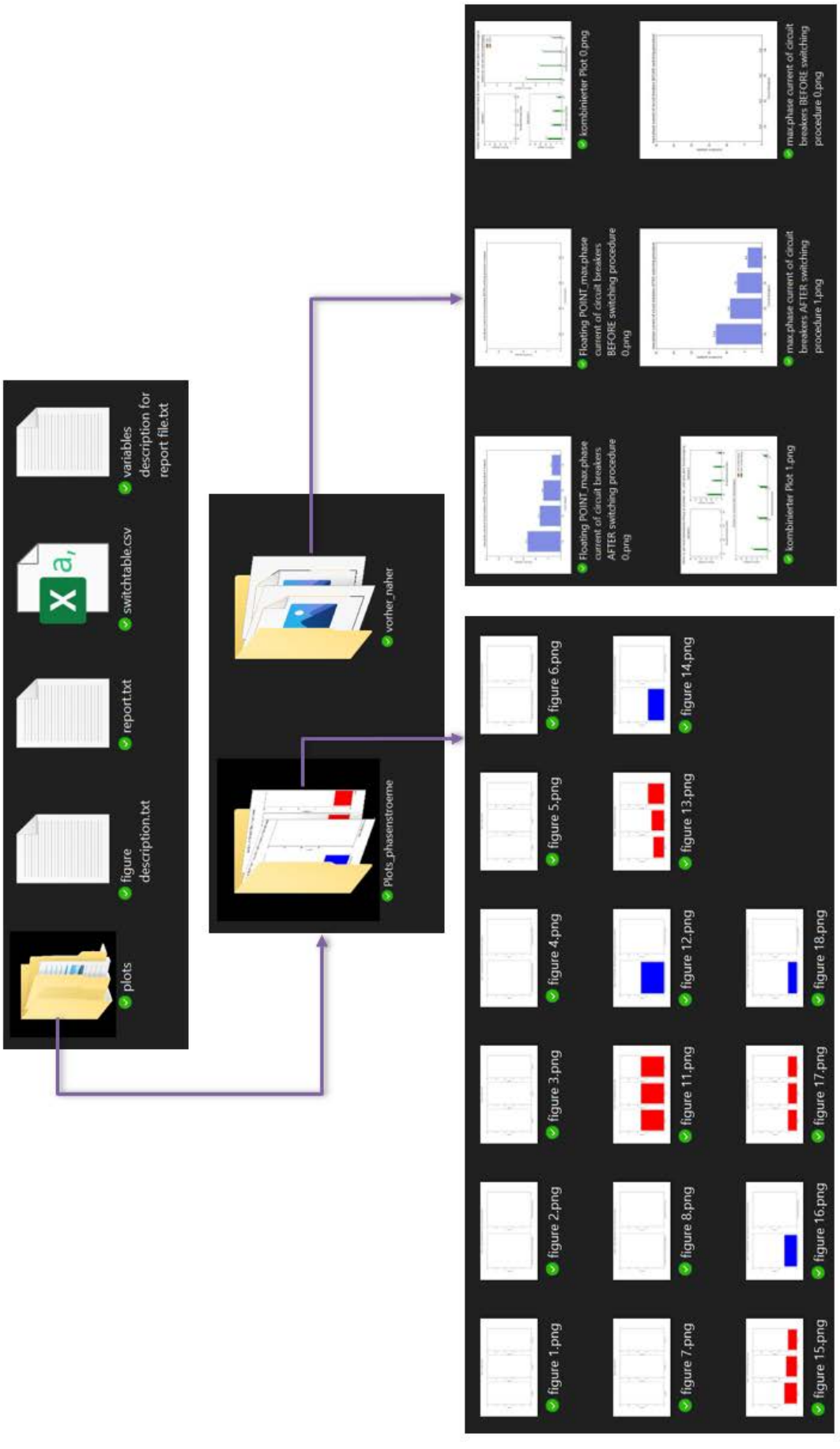
```
"""
+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
| Switching Algorithm - output visualization                |
| v7                                                        |
| by Rene GRIESSER                                          |
| Master Thesis 2021                                        |
|                                                           |
+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
"""

def func_Print(Schalterposition):

    a = Schalterposition
    if a == 0:
        print('0 = 0 0 0 0')
        print("""                    ----/----
                     |       |
                  -/-|       -
                     |       |
                     ----/----          """)
    elif a == 1:
        print('1 = 0 0 0 1')
        print("""                    ----/----
                     |       |
                  ---|       -
                     |       |
                     ----/----          """)
    elif a == 10:
        print('2 = 0 0 1 0')
        print("""                    ---------
                     |       |
                  -/-|       -
                     |       |
                     ----/----          """)
    elif a == 11:
        print('3 = 0 0 1 1')
        print("""                    ---------
                     |       |
                  ---|       -
                     |       |
                     ----/----          """)
    elif a == 100:
        print('4 = 0 1 0 0')
        print("""                    ----/----
                     |       |
                  -/-|       -
                     |       |
                     ---------          """)
    elif a == 101:
        print('5 = 0 1 0 1')
        print("""                    ----/----
                     |       |
                  ---|       -
                     |       |
                     ---------          """)
    elif a == 110:
        print('6 = 0 1 1 0')
        print("""                    ---------
                     |       |
                  -/-|       -
                     |       |
                     ---------          """)
    elif a == 111:
        print('7 = 0 1 1 1')
        print("""                    ---------
                     |       |
                  ---|       /
                     |       |
                     ---------          """)
    elif a == 1000:
        print('8 = 1 0 0 0')
        print("""                    ----/----
                     |       |
                  -/-|       |
```

```
                    |       |
                    ----/----         """)
    elif a == 1001:
        print('9 = 1 0 0 1')
        print("""                    ----/----
                    |       |
                ---|        |
                    |       |
                    ----/----         """)
    elif a == 1010:
        print('10 = 1 0 1 0')
        print("""                    ---------
                    |       |
                -/-|        |
                    |       |
                    ----/----         """)
    elif a == 1011:
        print('11 = 1 0 1  1')
        print("""                    ---------
                    |       |
                ---|        |
                    |       |
                    ----/----         """)
    elif a == 1100:
        print('12 = 1 1 0 0 ')
        print("""                    ----/----
                    |       |
                -/-|        |
                    |       |
                    ---------         """)
    elif a == 1101:
        print('13 = 1 1 0 1')
        print("""                    ----/----
                    |       |
                ---|        |
                    |       |
                    ---------         """)
    elif a == 1110:
        print('14 = 1 1 1 0')
        print("""                    ---------
                    |       |
                -/-|        |
                    |       |
                    ---------         """)
    elif a == 1111:
        print('15 = 1 1 1 1')
        print("""                    ---------
                    |       |
                ---|        |
                    |       |
                    ---------         """)
    elif a == 77:
        print("""
 _________  _______     _____  _______  _______  ________  _____
|  _   _  ||_   __ \   |_   _||_   __ \|_   __ \|_   __  ||_   _|.
|_/ | | \_|  | |__) |    | |    | |__) | | |__) | | |_ \_|  | | `. \
    | |      |  __ /     | |    |  ___/  |  ___/  |  _| _   | |  | |
   _| |_    _| |  \ \_  _| |_  _| |_    _| |_    _| |__/ | _| |_.' /
  |_____|  |____| |___||_____||_____|  |_____|  |________||_____.'

""")
    else:
        print("CAUTION: Wrong input parameter! There is no matching switching position! Must
be 0000 to 1111 and all binary combination within this range")

    return Schalterposition


"""
+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
| Switching Algorithm - output visualization                |
| v7                                                        |
| by Rene GRIESSER                                          |
| Master Thesis 2021                                        |
|                                                           |
+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
"""
```

9.2.3 Python Skript Ausgabe

```
C:\Users\reneg\PycharmProjects\AI\venv\Scripts\python.exe C:/Users/reneg/PycharmProjects/AI/snapy.py
--------------------------------------
Rene Griesser
Copyright 2021, Griesser
version: 7.3
status: under coding

mail:  rene.griesser@student.tugraz.at

--------------------------------------



----------------------
PLC stats---------------
----------------------
PLC is connected:  True

IP-Adress:  192.168.1.50
TIA-Portal Rack Position:  0
TIA-Portal Slot Position:  1
PLC is in mode:  S7CpuStatusRun
PLC last error code was:  0
This means PLC is:  OK
current time is = 09:38:23

-----------------------------------------
grid configuration before switching-------
-----------------------------------------
12 = 1 1 0 0
                    ----/----
                    |       |
                 -/-|       |
                    |       |
                    ---------
current position of the switches is not ideal, therefore switching process is inevitable
-----------------------------------------
grid configuration after switching--------
-----------------------------------------
15 = 1 1 1 1
                    ---------
                    |       |
                 ---|       |
                    |       |
                    ---------
configuration before switching process
12 = 1 1 0 0
                        ----/----
                        |       |
                     -/-|       |
                        |       |
                        ---------
configuration after switching process
15 = 1 1 1 1
                        ---------
                        |       |
                     ---|       |
                        |       |
                        ---------


Process finished with exit code 0
```

9.2.4 Programmierter Output Folder (Dateien und Struktur)



9.2.4.1 Externe Daten-Files

9.2.4.1.1 Description Files

figure description.txt

```
 -------------------------------
| Master Thesis 2021            |
| by Rene GRIESSER              |
 -------------------------------

figure description


figure 1 -- Switch 1 _ all 3 phase currents BEFORE switching
figure 2 -- Switch 1 _ max.current in the heavily loaded phase + current in the neutral conductor BEFORE switching
figure 3 -- Switch 2 _ all 3 phase currents BEFORE switching
figure 4 -- Switch 2 _ max.current in the heavily loaded phase + current in the neutral conductor BEFORE switching
figure 5 -- Switch 3 _ all 3 phase currents BEFORE switching
figure 6 -- Switch 3 _ max.current in the heavily loaded phase + current in the neutral conductor BEFORE switching
figure 7 -- Switch 4 _ all 3 phase currents BEFORE switching
figure 8 -- Switch 4 _ max.current in the heavily loaded phase + current in the neutral conductor BEFORE switching
figure 11 -- Switch 1 _ all 3 phase currents AFTER switching
figure 12 -- Switch 1 _ max.current in the heavily loaded phase + current in the neutral conductor AFTER switching
figure 13 -- Switch 2 _ all 3 phase currents AFTER switching
figure 14 -- Switch 2 _ max.current in the heavily loaded phase + current in the neutral conductor AFTER switching
figure 15 -- Switch 3 _ all 3 phase currents AFTER switching
figure 16 -- Switch 3 _ max.current in the heavily loaded phase + current in the neutral conductor AFTER switching
figure 17 -- Switch 4 _ all 3 phase currents AFTER switching
figure 18 -- Switch 4 _ max.current in the heavily loaded phase + current in the neutral conductor AFTER switching
```

variables description for report file.txt

```
 -------------------------------
| Master Thesis 2021            |
| by Rene GRIESSER              |
 -------------------------------

Basistype 3 measurement data- reference to the report.txt file


value 1 -- current in Phase L1
value 2 -- current in Phase L2
value 3 -- current in Phase L3
value 4 -- current in the heavily loaded phase
value 5 -- current in neutral conductor
value 6 -- Voltage L1-L2
value 7 -- Voltage L3-L2
value 8 -- Voltage L3-L1
value 9 -- Voltage L1-N
value 10 -- Voltage L2-N
value 11 -- Voltage L3-N
value 12 -- averafe powerfactor
value 13 -- effective energy reference
value 14 -- apparent power
```

9.2.4.1.2 Run Report

report.txt

```
values before switching
values switch 1
0 A
0 A
0 A
0 A
0 A
0 V
0 V
0 V
0 V
0 V
0 V
0
0
0
values switch 2
0 A
0 A
0 A
0 A
0 A
0 V
0 V
0 V
0 V
0 V
0 V
0
0
0
values switch 3
0 A
0 A
0 A
0 A
0 A
0 V
0 V
0 V
0 V
0 V
0 V
0
0
0
values switch 4
0 A
0 A
0 A
0 A
0 A
0 V
0 V
0 V
0 V
0 V
0 V
0
0
0
number of switching processes
number of switching processes switch 1: 411
number of switching processes switch 2: 281
number of switching processes switch 3: 311
number of switching processes switch 4: 214
------------------------------------------------
```

```
------------------------------------------------
values after switching
values switch 1
13 A
13 A
13 A
13 A
0 A
392 V
392 V
393 V
224 V
226 V
226 V
993
0
9
values switch 2
6 A
7 A
9 A
9 A
0 A
392 V
392 V
394 V
225 V
226 V
226 V
993
0
5
values switch 3
7 A
6 A
5 A
7 A
0 A
394 V
391 V
394 V
225 V
226 V
226 V
993
0
4
values switch 4
1 A
2 A
4 A
4 A
0 A
392 V
392 V
394 V
225 V
226 V
226 V
983
0
2
number of switching processes
number of switching processes switch 1: 412
number of switching processes switch 2: 282
number of switching processes switch 3: 312
number of switching processes switch 4: 215
```

9.3 Leitungsmodul

9.3.1 Angebot RUSA GmbH

Rusa GmbH **Schumanngasse 36, 1180 A-Wien**
Transformatoren · Widerstände · Drosseln · Expressreparaturen
office@rusatrafo.at **Tel:** +43 1 405 33 85 ARA-Lizenznr: 5195
www.rusatrafo.at **Fax:** +43 1 405 33 85 10 UID-Nr: ATU 50788009

RUSA GmbH, Schumanngasse 36, 1180 Wien
Rene Griesser
rene.griesser@outlook.com
Technische Universität Graz

Kunden Nr.:	29998
Bearbeiter:	Fr.Mag.Rusa
Bestellnr.:	tel 8.3.21
Lieferdatum:	10.03.2021
Datum:	10.03.2021

Angebot Nr. 2021045

Sehr geehrter Herr Griesser,

Bezugnehmend auf Ihre Anfrage freuen wir uns, Ihnen freibleibend und unverbindlich unter Zugrundelegung der allgemeinen Geschäftsbedingungen (AGB) das akutalisierte Angebot wie folgt anbieten zu können:

Pos	Menge		Text	Einzelpreis EUR	Gesamtpreis EUR
1	3	Stk.	Einphasen-Drossel E19 49 µH ±10% 25 A offene Ausführung, Schutzart IP00 Anschlüsse mit Kabelschuhen Isolationsklasse E, getränkt Abmessung L x B x H: ca. 55 x 55 x 92 mm Gewicht: 0,8 kg	105,13	315,39
2	1	Stk.	Einphasen-Drossel E22 198 µH ±10% 25 A offene Ausführung, Schutzart IP00 Anschlüsse mit Kabelschuhen Isolationsklasse E, getränkt Abmessung L x B x H: ca. 65 x 57 x 101 mm Gewicht: 1,10 kg	124,35	124,35
A	3	Stk.	Alternativposition Luftspule 49 µH ±5% 25 A offene Ausführung, Schutzart IP00 Anschlüsse mit Kabelschuhen Isolationsklasse E, getränkt Gewicht: ca. 0,9 kg gewickelt auf HP Zylinder Innendurchmesser ca. 36 mm Aussendurchmesser ca. 69 mm Gesamtgröße mit rechteckigem Flansch: 100x100x40 vakuumgetränkt	95,00	(285,00)
A	1	Stk.	Alternativposition Luftspule 198 µH ±5% 25 A offene Ausführung, Schutzart IP00 Anschlüsse mit Kabelschuhen	112,00	(112,00)
Zwischensumme					439,74

Garantieansprüche werden nur bis zur Höhe des gelieferten Wertes akzeptiert. Folgekosten oder Vermögensschäden jeglicher Art werden nicht akzeptiert. Firmenbuch Nr. FN 203665h beim Handelsgericht Wien. Bis zur vollständigen Bezahlung der Faktura bleibt die Ware in unserem Eigentum. Zahlbar und klagbar in Wien. Versand ab Lager auf Gefahr des Empfängers. Es gelten ausschließlich unsere Geschäftsbedingungen, zu finden auf https://www.rusa-transformatoren.at/agb

Rusa GmbH **Schumanngasse 36, 1180 A-Wien**
Transformatoren · Widerstände · Drosseln · Expressreparaturen
office@rusatrafo.at **Tel:** +43 1 405 33 85 ARA-Lizenznr: 5195
www.rusatrafo.at **Fax:** +43 1 405 33 85 10 UID-Nr: ATU 50788009

Rene Griesser
rene.griesser@outlook.com
Technische Universität Graz

Kunden Nr.:	29998
Bearbeiter:	Fr.Mag.Rusa
Bestellnr.:	tel 8.3.21
Lieferdatum:	10.03.2021
Datum:	10.03.2021

Pos	Menge		Text	Einzelpreis EUR	Gesamtpreis EUR
Übertrag					439,74
			Isolationsklasse E, getränkt Gewicht: ca. 1,5 kg gewickelt auf HP Zylinder Innendurchmesser ca. 36 mm Aussendurchmesser ca. 76 mm Gesamtlänge ca. 60 mm Gesamtgröße mit rechteckigem Flansch: 100x100x60 vakuumgetränkt		
Zwischensumme					439,74
zzgl. Porto und Verpackung DPD					12,00
Gesamt Netto					451,74
zzgl. 20,00 % USt. auf				439,74	87,95
zzgl. 20,00 % USt. auf Nebenleistungen				12,00	2,40
Gesamtbetrag					**542,09**

Zahlbar netto bei Rechnungserhalt

Preise inkl. aller Materialzuschläge
Lieferzeit: ca. 2-3 Wochen
Lieferbedingungen: unverpackt, ab Werk
Zahlungsart: Vorauskasse
CA BA: IBAN – AT26 1100 0093 8356 5000 BIC – BKAUATWW BLZ 11000 Kto 09383565000 Lieferdatum–Rechnungsdatum Unsere UID Nr ATU50788009

Aufgrund der unvorhersehbaren Entwicklung des Coronavirus und daraus folgenden politischen Maßnahmen, können wir die Einhaltung unserer Angebote und Lieferzeiten nicht mehr garantieren. Wir werden weiterhin alles in unserer Macht stehende tun, um Ihre Bestellungen fristgerecht zu erfüllen.

Vielen Dank für Ihre Anfrage, wir würden uns über einen Auftrag freuen und verbleiben
mit freundlichen Grüßen
Mag. Marion Rusa

9.3.2 Kostenkalkulation 4x6mm² NYY-O 162 m

Pos.	Bezeichnung	Anzahl	Netto Einzelpreis	Netto Gesamtpreis
1	Widerstand Mitsystem 470 mΩ ; 1kW ; jeweils 2x 1 Ω parallel	6,00	€ 40,60	€ 243,59
2	Widerstand Nullsystem 2,2 Ω ; 1kW	1,00	€ 46,84	€ 46,84
3	Induktivität Mitsystem 0,0495 mH RUSA Anfertigung	3,00	€ 105,13	€ 315,39
4	Induktivität Nullsystem 0,1985 mH RUSA Anfertigung	1,00	€ 124,35	€ 124,35
5	Ringkabelschuh 6 mm² 100 stk	1,00	€ 32,70	€ 32,70
6	Einzeladerleitung 6 mm² - Schwarz 100 m	1,00	€ 56,50	€ 56,50
7	Einzeladerleitung 6 mm² - Blau 100 m	1,00	€ 56,50	€ 56,50
8	Einzeladerleitung 6 mm² - Gelb/Grün 100 m	1,00	€ 56,50	€ 56,50
9	Phoenix Anschlussklemme bis 6 mm² Grau	6,00	€ 1,66	€ 9,96
10	Phoenix Anschlussklemme bis 6 mm² Blau	2,00	€ 1,66	€ 3,32
11	Phoenix Anschlussklemme bis 6 mm² Gelb/Grün	2,00	€ 7,59	€ 15,18
12	Phoenix Kontaktabdeckung	8,00	€ 0,86	€ 6,88
13	Phoenix Kontaktabdeckung TWIN	2,00	€ 0,88	€ 1,76
14	Schaltschrank Rittal 1800x600x400 für mind. 3 Module	1,00	€ 593,52	€ 593,52
15	Hutschiene 500 mm	2,00	€ 4,94	€ 9,88
16	Verdrahtungskanal 40 mm breit 1000 mm lang	4,00	€ 23,64	€ 94,56
17	Verbindungskabel 5x6 mm² Schrank zu Schrankverbindung 50 m f. 4 Modulverb. a 4 m [Durchm. 19 mm]	1,00	€ 383,58	€ 383,58
18	Verbindungsschrauben M6x25 mm - 50 stk	1,00	€ 10,35	€ 10,35
19	Muttern M6 250 stk	1,00	€ 8,10	€ 8,10
20	Beilagen M6 250 stk	1,00	€ 4,52	€ 4,52
21	Selbstschneiderschrauben für Hutschienenmontage 100 stk	1,00	€ 11,62	€ 11,62
22	Adernendhülsen bis 6 mm² für Anschluss an Klemmblock 100 stk	1,00	€ 7,20	€ 7,20
23	Sicherungslasttrennschalter 32 A Einphasig	5,00	€ 6,47	€ 32,35
24	diverses Kleinmaterial + unvorhergesehene Montagematerialien(Kleinmaterial, Schrumpfschlauch etc.)	1,00	€ 200,00	€ 200,00
25	Verbinderstecker Heavycon Gehäuse mit 2 Kabelabgängen B24- Tüllenseite TS	2,00	€ 7,49	€ 14,98
26	Verbinderstecker Heavycon Kabelverschraubung M32 für Kabel von 11 bis 21 mm Durchmesser f. TS	4,00	€ 2,45	€ 9,80
27	Verbinderstecker Heavycon Modulträgerrahmen B24 Tüllenseite	2,00	€ 6,63	€ 13,26
28	Verbinderstecker Heavycon Kontakteinsatzmodul Male Tüllenseite	10,00	€ 13,00	€ 130,00
29	Verbinderstecker Heavycon Kontakteinsatzmodul Blindeinsatz Tüllenseite mind. 5.stk Abnahmemenge	5,00	€ 0,88	€ 4,40
30	Verbinderstecker Heavycon Modulträgerrahmen B24 Anbauseite	1,00	€ 6,63	€ 6,63
31	Verbinderstecker Heavycon Kontakteinsatzmodul Female Anbauseite	5,00	€ 13,00	€ 65,00
32	Verbinderstecker Heavycon Gehäuseverriegelung für Anbauseite	1,00	€ 6,85	€ 6,85
33	Durchgangsklemme TWIN	10,00	€ 5,70	€ 57,00
			Summe Netto:	€ 2.633,07
			Summe Brutto:	€ 3.159,68