BACHELORARBEIT

Untersuchung unterschiedlicher Referenzdatensätze im Energiebereich

durchgeführt am
Studiengang Informationstechnik & System–Management
an der
Fachhochschule Salzburg GmbH

vorgelegt von

Carlo Bellucci, Anna-Maria Oberluggauer, Maximilian Tschuchnig

Studiengangsleiter: FH-Prof. DI Dr. Gerhard Jöchtl

Betreuer: Mag.(FH) DI Christian Peuker, BSc

Salzburg, Februar 2017

Eidesstattliche Erklärung

Wir versichern an Eides statt, dass wir die vorliegende Bachelorarbeit ohne unzulässige fremde Hilfe und ohne Benutzung anderer als der angegebenen Quellen und Hilfsmittel angefertigt und alle aus ungedruckten Quellen, gedruckter Literatur oder aus dem Internet im Wortlaut oder im wesentlichen Inhalt übernommenen Formulierungen und Konzepte gemäß den Richtlinien wissenschaftlicher Arbeiten zitiert, bzw. mit genauer Quellenangabe kenntlich gemacht haben. Diese Arbeit wurde in gleicher oder ähnlicher Form weder im In- noch im Ausland in irgendeiner Form als Prüfungsarbeit vorgelegt und stimmt mit der durch die Begutachter/Begutachterinnen beurteilten Arbeit überein.

Salzburg, am 15.02.2017

1410555010

Carlo Bellucci — Matrikelnummer

Oberluggauer Anna-Maria

1410555077

Anna-Maria Oberluggauer — Matrikelnummer

1410555083

Maximilian Ernst Tschuchnig — Matrikelnummer

Kurzzusammenfassung

Auf Grund der Energiewende und der Einführung von Smart-Grids ergeben sich verschiedene Anwendungsfälle, wie Non intrusive load monitoring und Lastprofilmodellierung, welche Energiereferenzdaten benötigen. Energiereferenzdatensätze sind Energiedaten, welche auf Basis von Smart-Metern erstellt werden. Im Zuge dieser Arbeit wurden, auf Grund von Lokalität, Frequenz und Aufnahmedauer, vier Datensätze, ADRES, GREEND, REDD und UK-DALE ausgesucht, analysiert und miteinander verglichen. Da diese Daten unterschiedliche Formate aufweisen, werden die einzelnen Referenzdatensätze auf ein einheitliches Format gebracht, um ein Vergleichen der Datensätze zu ermöglichen. Deshalb wurde ein gemeinsamer Parser programmiert, welcher die verschiedenen Referenzdatensätze benutzerspezifisch einliest und in ein einheitliches Format bringt. Diese Daten werden in eine SQL-Datenbank gespeichert, damit, für die verschiedenen Anwendungsfälle, ein einfacher Zugriff auf die Daten gewährleistet wird.

Abstract

Due to the Energy Transition and the introduction of Smart-Grids, several new applications, that use energy reference data, emerged. Those applications would be, for example load profile modelling and non intrusive load monitoring. An energy reference dataset is composed of energy data, based on smart meter data. As part of this paper, four different reference data sets, ADRES, GREEND, REDD and UK-DALE, were examined with the main criteria being locality, frequency and sample duration. Due to the different formats of the datasets, parsing into one common dataset is needed to allow for a comparison. Therefore, a parser was developed, which enables user specific input and the conversion into a common format. The parsed data then gets stored into a SQL database, further increasing the comfortable access of the data for the newly emerged applications.

Inhaltsverzeichnis

Inhaltsverzeichnis **IV**

Abkürzungsverzeichnis **V**

Abbildungsverzeichnis **VI**

Tabellenverzeichnis **VII**

1 Einleitung **1**
1.1 Motivation und Aufgabenstellung 1
1.2 Aufbau und Kapitelübersicht 2

2 Energiereferenzdaten **3**
2.1 Definition von Energiereferenzdaten 3
2.2 Anwendungsfälle von Referenzdaten 3
2.3 Betrachtung ausgewählter Referenzdatensätze 6
2.3.1 Autonomes Dezentrales Regeneratives Energie-System 6
2.3.2 GREEND Electrical ENergy Dataset 8
2.3.3 Reference Energy Disaggregation DataSet 9
2.3.4 UK Domestic Appliance-Level Electricity 11
2.4 Evaluierung 12
2.4.1 Vorteile eines gemeinsamen Parsers 12
2.4.2 Anforderungen an einen gemeinsamen Parser 13

3 Praktischer Teil **14**
3.1 Beschreibung der Aufgabenstellung 14
3.2 Analyse der Daten 14
3.3 Implementierung 19
3.3.1 Vorbereitung der Daten 19
3.3.2 Parser 20
3.3.3 Datenbank 23
3.4 Tests 24
3.5 Erkenntnisse 26

4 Zusammenfassung/Ausblick **29**

Literatur **I**

A Anhang **III**

Abkürzungsverzeichnis

ADRES Autonome Dezentrale Regenerative Energie-Systeme

GREEND GREEND Electrical ENergy Dataset

GUI Graphical User Interface

ID Kennung

ILM Intrusive Load Monitoring

MAC Media-Access-Control

MATLAB MATrix LABoratory

MIT Massachusetts Institute of Technology

NILM Nonintrusive Load Monitoring

NIOM Nonintrusive Occupancy Monitoring

OOP Objektorientierte Programmierung

REDD Reference Energy Disaggregation Data Set

SQL Structured Query Language

UI User Interface

UK-DALE United Kingdom Domestic Appliance-Level Electricity Dataset

UNIX Uniplexed Information and Computing System

Abbildungsverzeichnis

2.1 Einschaltwarscheinlichkeit Kaffeemaschine Haushalt 0 9

3.1 UK-Dale Datenvorbereitung 19
3.2 ADRES Datenvorbereitung 20
3.3 Datenbankmodell SQL Power Architect 23

Tabellenverzeichnis

3.1 Auszug Spannungswerte [1] 15
3.2 Auszug Leistungs-/Wirkleistungswerte [1] 15
3.3 Auszug Haushalt 0, 10.12.2013 [2] 16
3.4 Auszug hochfrequente Stromwerte Haushalt 3 [3] 17
3.5 Auszug niederfrequente Leistungswerte Haushalt 1, Kühlschrank [3] . . 17
3.6 Auszug Stromwerte Haushalt 1, Woche 32, Datei 118 [4] 18
3.7 Auszug Spannungswerte Haushalt 1, Woche 32, Datei 118 [4] 18
3.8 Auszug Leistungswerte Haushalt 1, TV [4] 18

1 Einleitung

1.1 Motivation und Aufgabenstellung

Die Umstellung von konventioneller auf alternative Energieerzeugung bewirkt unbeeinflussbare Stromschwankungen im Energiemarkt. Diese Stromschwankungen treten auf, da sich Energieerzeugung wie zum Beispiel Photovoltaik oder Windkraft nicht steuern lassen. Des Weiteren wird der Energiemarkt, durch private Energieerzeuger, immer stärker dezentralisiert. Eine breitflächige Umstellung von konventioneller auf alternative Energieerzeugung bringt also auch Nachteile. Der größte Vorteil konventioneller Energien ist, jederzeit schnell Strom erzeugen zu können. Diese, beinahe augenblickliche, Energiebereitschaft ist durch alternative Energien nicht mehr gegeben [5].

Vor einer solchen Umstellung wird die benötigte Energie also von den Verbrauchern vorgegeben, welche daraufhin von Energieerzeugern abgedeckt wird. Da dies aufgrund der eben beschriebenen Probleme von alternativen Energien nicht mehr ausreichend möglich ist, sind so genannte Smart-Grids notwendig. Smart-Grids repräsentieren Stromnetze, welche aus intelligenten Verbrauchern, Verteilern und Erzeugern bestehen. Um Smart-Grids effizient betreiben zu können, werden Daten über den Energieverbrauch im Netz benötigt. Dies resultiert jedoch in dem Problem, dass die Verbrauchsdaten, laut EU Recht, nicht in Echtzeit verwendet werden dürfen (auch nicht anonymisiert) [6].

Referenzdaten werden erstellt, um spezifische Verbrauchergruppen (zum Beispiel Haushalte), ohne rechtliche Probleme, repräsentieren zu können. Dies wurde bereits einige Male durchgeführt und resultierte in diversen Modellen [2][7][4]. Da Referenzdatensets jedoch für einen gewissen Zweck erzeugt wurden und unterschiedlich aufgebaut sind, ist eine einheitliche Bearbeitung sehr kompliziert. Die Hauptaufgabe der Arbeit ist es daher, sich mit einigen bestimmten Referenzdaten auseinanderzusetzen und diese in ein vorgegebenes, einheitliches Format zu bringen.

In dieser Arbeit werden daher unterschiedliche Referenzdatensätze untersucht und ein Programm erstellt, welches Referenzdaten in das vorgegebene Format bringt. Dadurch werden die Daten vergleichbar und einfacher zu bearbeiten. In Zukunft kann dieses Programm auf andere Referenzdatensätze erweitert werden, was den Nutzen des Programms noch weiter erhöht.

1.2 Aufbau und Kapitelübersicht

Der erste Teil der vorliegenden Bachelorarbeit befasst sich mit dem Begriff Referenzdaten im Zusammenhang mit elektrischen Netzen. Es wird vor allem auf die Zwecke und Anwendungsfälle dieser Referenzdaten eingegangen. Zusätzlich wird der grundlegende Aufbau von Referenzdaten erörtert.

Nach der allgemeinen Behandlung der Referenzdaten wird genauer auf die Eigenschaften, welche in der Arbeit priorisiert werden, eingegangen. Die gewählten Referenzdaten werden anschließend auf diese Eigenschaften untersucht.

Abschließend werden die Referenzdatensätze analysiert und verglichen. Diese Gegenüberstellung soll zeigen, welche der Eigenschaften implementiert sind und wie sie sich für gewisse Anwendungsfälle eignen.

Der zweite Teil befasst sich unter anderem mit den Vorteilen einer Umwandlung der Daten in ein allgemeines Format. Des Weiteren wird die praktische Aufgabenstellung beschrieben und die gegebenen Referenzdaten auf Möglichkeiten der Umwandlung untersucht. Danach wird die Implementierung der Umwandlung der Daten in ein gemeinsames Referenzdatenformat untersucht. Um die Implementierung zu überprüfen, werden Tests erstellt, welche die Verwendbarkeit der umgewandelten Daten kontrollieren.

Abschließend werden die wesentlichen Erkenntnisse dieser Arbeit zusammengefasst dargestellt. Zukunftsperspektiven und Verbesserungsvorschläge werden ebenso vorgestellt.

2 Energiereferenzdaten

Der theoretische Teil dieser Arbeit befasst sich mit Energiereferenzdaten und deren Zweck. Des Weiteren werden Anwendungsfälle dieser Referenzdaten, wie zum Beispiel Nonintrusive Load Monitoring (NILM), Lastprofilmodellierung und so weiter, beschrieben. Im weiteren Verlauf dieses Kapitels werden die ausgewählten Datensätze genauer erläutert und deren Anwendungsbereiche dargestellt. Abschließend werden noch die Vorteile eines gemeinsamen Parsers geschildert.

2.1 Definition von Energiereferenzdaten

Energiereferenzdaten sind große Mengen an Daten, welche unter anderem von einer großen Anzahl von Smart Metern aufgezeichnet werden. Um den gesetzlichen Bedingungen zu entsprechen, werden die Daten anonymisiert, bevor diese veröffentlicht werden. Referenzdaten dienen vor allem als Vergleichsdaten.

Energiereferenzdaten werden beispielsweise erstellt, um den durchschnittlichen Energieverbrauch zu ermitteln. Dies ist vor allem hilfreich, da immer mehr auf konventionelle, erneuerbare Energie, wie Windenergie und Solarenergie, umgestiegen wird und diese bestmöglich aus lokalen Energieerzeugern verwendet werden soll [1]. Diese Energieerzeugung hängt von den Umwelteinflüssen ab und kann somit nicht vom Menschen beeinflusst werden. Anhand dieser Energiereferenzdaten ist es aber möglich, den Verbrauch eines Landes oder ähnliches vorherzusagen und zum richtigen Zeitpunkt die Haushalte mit Energie versorgen zu können. Auch die steigende Dezentralisierung dieser alternativen Energieerzeuger macht das Wissen über den Energieverbrauch unumgänglich.

Referenzdaten liegen in unterschiedlichen Datenformaten vor, sehen Sie dazu Abschnitt 3.2. Mit Hilfe solcher Energiereferenzdaten können smarte Energienetze, mit erneuerbarer Energieversorgung, entwickelt werden. Diese Systeme stellen eine Kommunikationsplattform zwischen den Energieerzeugern und den Energieverbrauchern dar. Die smarten Energienetze können, anhand des Wissens, welcher Haushalt wie viel Energie durchschnittlich benötigt, die Haushalte gezielt mit Energie versorgen.

2.2 Anwendungsfälle von Referenzdaten

Referenzdaten werden nicht zufällig aufgenommen, sondern sollen für einen bestimmten Zweck verwendet werden. Der Anwendungsfall gibt sowohl die Dauer und die Anzahl

der betroffenen Haushalte des Referenzdatensatzes, als auch die aufzunehmenden Daten und deren Genauigkeit vor.

Non-intrusive load monitoring

NILM ist eine Methode, mit der versucht wird, aus den gemessenen Daten eines gesamten Hauses, den Verbrauch einzelner Haushaltsgeräte zu bestimmen. Der NILM-Prozess analysiert die Strom- und Spannungsänderungen des Hauses. Die Methode wird als non-intrusive bezeichnet, da kein Eindringen in das Haus notwendig ist. Es wird auch als eine kostengünstige Alternative zu Intrusive Load Monitoring (ILM) Techniken verwendet, da nicht für jedes Haushaltsgerät ein eigener Sensor benötigt wird. Die aufgenommenen Daten werden dann mit Modellen von Haushaltsgeräten verglichen. Dadurch ist es möglich, herauszufinden, welche Geräte wann ein- und ausgeschalten wurden. Da beim Aufzeichnen der Daten allerdings nicht verhindert werden kann, dass auch Rauschen mit aufgezeichnet wird, können nicht immer alle Geräte genau bestimmt werden. Die Verwendung von NILM Systemen macht es möglich, Konsumenten, ohne Eingriffe in ihren Haushalt, Vorschläge zur Senkung ihres Energieverbrauches zu machen. Die Tatsache, dass NILM ohne Eindringen in das Haus implementiert werden kann, führt allerdings, in Bezug auf die Privatsphäre der Bewohner, zu Bedenken. Dadurch, dass kein Eingriff in den Haushalt nötig ist, könnten, ohne das Wissen der Bewohner, Daten aufgezeichnet und gesammelt werden [2][8].

Abflachung des Spitzenstrombedarfs in Smart Homes

Die Abflachung des Spitzenstrombedarfs eines Haushalts führt zu einem konstanteren Strombedarf. Da Stromspitzen einen ungleichmäßigen Effekt auf das Netz haben, wird durch deren Abflachung, auch der Verlust, der durch die Stromverteilung entsteht, verringert. Dies kann durch die Verwaltung der Hintergrundlast des Hauses geschehen. Während Stromspitzen auftreten, können zum Beispiel nicht verwendete Geräte abgeschaltet werden, wodurch die Spitze abgeflacht wird. Das Problem bei dieser Aufgabe ist, dass diese Verwaltung die Bewohner des Hauses in ihren Tagesabläufen nicht beeinflussen darf. Um herauszufinden, wann am besten Geräte an- und abgeschaltet werden können, um Stromspitzen abzuflachen und ohne die Bewohner zu behindern, werden Referenzdaten benötigt [9].

Lastprofilmodellierung

Ein weiterer Anwendungsfall von Referenzdaten ist das Erstellen eines Lastprofils, um bessere Voraussagen treffen zu können. Im Vergleich zum vorher erwähnten Anwendungsfall ist für die Lastprofilmodellierung auch die Gebäudeart und Struktur wichtig, da diese ebenfalls in das Modell miteinbezogen werden sollten. Für die Erstellung eines akkuraten Lastprofils werden allerdings nicht nur die Daten der Verbraucher, sondern auch Energieerzeugerdaten benötigt. Da die Energieerzeuger selbst bereits die benötigten Daten aufzeichnen, müssen diese nicht extra durch langwierige Messungen erfasst werden, sondern können direkt bei den Erzeugern beantragt werden. Soll das Modell für eine Region mit unterschiedlichen Jahreszeiten erstellt werden, ist es auch für diesen Anwendungsfall nötig, genügend Daten in jeder Jahreszeit zu sammeln. Lastprofile sind besonders für Energieversorger wichtig, da diese jederzeit die benötigte Leistung liefern sollten. Durch die Verwendung von Lastprofilen können die Versorger eine gute Einschätzung über den Verbrauch zu einer bestimmten Zeit treffen. Dadurch lässt sich bereits im Voraus ein Plan für die Energieversorgung erstellen. Nach der Fertigstellung des Lastprofils könnten diese Daten auch verwendet werden, um eine Simulation zu erstellen, mit der zukünftige Ereignisse vorausgesagt werden [1][10].

Wissenschaftliche Analyse

Für viele Anwendungsfälle werden Daten über einen kurzen Zeitraum, für einen speziellen Zweck aufgezeichnet. Das führt dazu, dass Aspekte die erst nach längerer Datenaufzeichnung offensichtlich werden, nicht erkannt werden. Außerdem müssen, durch die Spezialisierung der gesammelten Datensätze, für neue Projekte oft neue Daten aufgenommen werden. Dies hat dazu geführt, dass inzwischen Datensätze erstellt werden, die bewusst versuchen, viele Daten zu sammeln, damit diese für mehrere Projekte verwendet werden können. Daten, die für wissenschaftliche Analysen aufgenommen werden, werden über einen längeren Zeitraum, mit einem langlebigen System aufgezeichnet. Da dies ein andauernder Prozess ist, in dem so wenig wie möglich eingegriffen wird, wird von Anfang an ein breites Spektrum an Informationen aufgezeichnet. Auch wenn sie am Beginn uninteressant erscheinen, kann auf diese Daten an einem späteren Zeitpunkt zurückgegriffen werden. Die Aufbewahrung und Verwaltung dieser Daten ist besonders wichtig, da für diesen Anwendungsfall eine sehr große Menge an Informationen gesammelt wird [7].

2.3 Betrachtung ausgewählter Referenzdatensätze

Da die Hauptaufgabe dieser Arbeit darin besteht, die Daten anderer Referenzdatensätze in das verwendete Zieldatensatzformat umzuwandeln, wurden Referenzdatensätze ausgewählt, welche logisch gut in dieses Format passen. Darum waren folgende Dateneigenschaften ausschlaggebend: Aufnahmefrequenz der Messwerte, regionale Unterschiede, Dauer der Aufnahme, saisonale Unterschiede und die Anzahl der gemessenen Haushalte. Aufgrund dieser Anforderungen wurden Referenzdaten aus Europa bevorzugt, welche eine hohe Frequenz aufweisen und über eine lange Zeitperiode zu unterschiedlichen Saisonen aufgenommen worden sind.

Wie in dem vorherigen Abschnitt erwähnt, gibt es Energiereferenzdaten für eine große Anzahl an Themen und Aufgabenstellungen. Aufgrund beschränkter Ressourcen ist es in dieser Arbeit nicht möglich auf alle Referenzdaten einzugehen. Daher liegt der Fokus auf den folgenden Referenzdatensätzen: UK-Dale [4], REDD [3], ADRES [1] und GREEND [2]. Dies Datensätze werden in den folgenden Abschnitten genauer erläutert.

2.3.1 Autonomes Dezentrales Regeneratives Energie-System

Hintergrund

Das Konzept Autonome Dezentrale Regenerative Energie-Systeme (ADRES) wurde erstellt, um ein System zu entwickeln, welches darauf ausgelegt ist, die Energieressourcen der Umgebung zu verwenden. Es sollte also bestmöglich auf den Import von Energie verzichtet werden und somit die Energieverbraucher mit regionaler Energie wie Wind, Solarthermie, Geothermie, Biomasse, Photovoltaik und Wasserkraft versorgen. Da die regionalen Ressourcen begrenzt sind, sollte unter anderem ein intelligentes Energienetz erstellt werden, welches die Energieressourcen zwischen Erzeuger und Verbraucher effizient aufteilt. Des Weiteren soll durch das intelligente Energienetz der Energieverbrauch reduziert werden, weshalb die Daten bezüglich Geräte, Gebäudeart, Anzahl der Personen pro Haushalt und so weiter, analysiert wurden. Auf Grund der Tatsache, dass die Erzeugung erneuerbarer Energie nicht beeinflusst werden kann, ist es deshalb auch wichtig, ein System zur Speicherung der überschüssigen Energie zu erstellen. Hierbei ist es von großer Bedeutung, die Energieerzeugung von Wind, Wasserkraft und viele mehr, prognostizieren zu können, weshalb Energieerzeugungsdaten aufgezeichnet wurden [1].

In diesem Projekt konnte nicht standardmäßig das H0-Lastprofil verwendet werden, da Informationen bezüglich der einzelnen Geräte in das ADRES-Konzept mit einbezogen

werden. H0 ist das Standardlastprofil für Haushalte, um den Stromverbrauch dieser Haushalte zu prognostizieren. Aus diesem Grund wurde ein neues Lastprofil erstellt, welches aber mit dem Standardlastprofil H0 korrelliert. Anhand des Wissens über die Haushaltsgröße und Gebäudeart der gemessenen Haushalte wurde ein neues ADRES-Lastprofil modelliert. Auch bei welchem Wochentag und in welcher Saison die Haushalte aufgezeichnet wurden, wurden in das Lastprofil miteinbezogen [1] [11].

Inhalt

Für diesen Datensatz wurde der Verbrauch von rund 40 Haushalten, darunter 21 Einfamilienhäuser, acht Mehrfamilienhäuser und zehn Wohnungen, über einen Zeitraum von zwei Wochen im Sommer und im Winter aufgezeichnet. Es wurde neben dem Gesamtverbrauch der Haushalte, auch der Verbrauch spezifischer Geräte ermittelt und aufgezeichnet. Die daraus resultierenden Daten haben eine Auflösung von einer Sekunde und beinhalten die elektrische Wirkleistung, Strom, Spannung und den Energieverbrauch dieser Haushalte [1].

Erreichte Ziele

Mit Hilfe des ADRES-Konzepts konnten, sowohl Windkraft als auch Wasserkraft erfolgreich prognostiziert werden. Bei der Vorhersage der Windkraft ist im oberen und unteren Erzeugungsbereich eine leichte Überschätzung ersichtlich. Ansonsten stimmt die Prognose mit der tatsächlich erzeugten Energie ziemlich gut überein. Auch bei der Vorhersage für Photovoltaik sind teilweise leichte Überschätzungen sichtbar, sie stimmen aber oftmals mit der Erzeugung überein [1].

Außerdem wurde untersucht, wie und wie viel Energieverbrauch in Österreich reduziert werden kann. Diese Untersuchung passiert auf MATrix LABoratory (MATLAB)-Simulationen. Dabei wurde eine deutliche Reduktion des Energieverbrauchs ermittelt, indem die Haushalte alte Geräte, durch Geräte mit zumindest dem Energiestandard A+, austauschen. Der Energiestandard eines Gerätes legt fest, wie viel Energie dieses Gerät verbraucht [12]. Die überschüssige Energie sollte gespeichert werden, wo der Entschluss gefasst wurde, dass die Energiespeicher nicht zu groß sein dürfen, da trotz der hohen Kosten, der Nutzen nicht stark ansteigt [1].

Abschließend ist zu sagen, dass es theoretisch möglich ist, ein autonomes System, welches auf lokale, erneuerbare Ressourcen zurückgreift, zu verwenden. Dies ist auf Grund

der hohen Kosten aber nicht wirtschaftlich [1].

2.3.2 GREEND Electrical ENergy Dataset

Hintergrund

Der GREEND Electrical ENergy Dataset (GREEND) Datensatz wurde erstellt, um potentielle Stromschwankungen des Energienetzes, durch intelligenten Betrieb von Verbrauchern, auszugleichen. Diese Stromschwankungen treten durch die steigende dezentralisierte Energieerzeugung und dem verstärkten Einsatz von alternativen Energien auf. Des Weiteren wird durch GREEND die Auswirkungen dieser Steuerung auf den Alltag untersucht. Dies wird durch den Vergleich der Daten von GREEND mit den unterschiedlichen Steuermöglichkeiten der Endverbraucher ermöglicht. Das GREEND Verbrauchsdatenset beinhaltet öffentlich zugänglichen Daten über den Energieverbrauch ganzer Haushalte und einzelner Stromkreise dieser Haushalte. Dadurch können die beschriebenen Steuermöglichkeiten so angenehm wie möglich für den Verbraucher gehalten werden [2].

Inhalt

Um die zuvor genannten Ziele zu erreichen, wurden neun unterschiedliche Haushalte über einen Zeitraum von einem Jahr aufgenommen. Die Haushalte selbst wurden drei bis sechs Monate lang gemessen. In jedem dieser Haushalte wurden jeweils acht interne Schaltkreise gemessen. Die Frequenz, mit welcher gemessen wurde, beträgt ein Herz. Um eine große Anzahl an Use Cases abzubilden, wurden die gemessenen Stromkreise und Haushalte so unterschiedlich wie möglich gestaltet [2].

Erreichte Ziele

Der GREEND Datensatz ermöglicht es erfolgreich, Nonintrusive Occupancy Monitoring (NIOM) durchzuführen. NIOM beschreibt ein Verfahren, um die Benutzungszeiten und die Flexibilität dieser Benutzungszeiten von Geräten eines Haushaltes festzustellen. GREEND erlaubt es also, herauszufinden, ob und inwiefern es möglich ist, Geräte zu unterschiedlichen Zeitpunkten einzuschalten, ohne zu große Störungen im Alltag hervorzurufen.

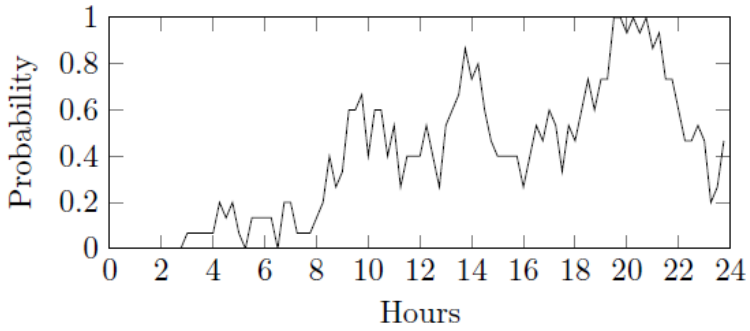

Abbildung 2.1: Einschaltwarscheinlichkeit Kaffeemaschine Haushalt 0

Umgesetzt wird dieses Prinzip als Histogramm mit der Einschaltwahrscheinlichkeit von bestimmten Geräten. Ein weiteres erreichtes Ziel von GREEND ist die anteilhafte Darstellung des Energieverbrauchs pro Gerät, gemessen am gesamten Haushalt. Dadurch und durch die erfolgreiche Implementierung von NIOM ist es GREEND möglich, höchst effizient NILM durchzuführen (vgl. Abschnitt 2.2). GREEND ist also ein freies Datenset, welches in Österreich und Italien zur Simulation in Smart Grid Werkzeugen verwendet werden kann [2].

2.3.3 Reference Energy Disaggregation DataSet

Hintergrund

Erneuerbare Energie zählt zu den wichtigsten Forschungsgebieten der Welt. Viele Probleme und Aufgabenstellungen in diesem Forschungsbereich sind grundsätzlich Probleme, die mit Datenanalysen und Vorhersagen zusammenhängen. Deshalb sind für dieses Forschungsgebiet Aufgaben wie Data Mining und maschinelles Lernen von hoher Relevanz. Trotz deren Bedeutsamkeit gibt es im Vergleich zu anderen Forschungsgebieten in diesen Bereichen nur wenig Fortschritte. Dieser Mangel an Fortschritt liegt zum Teil daran, dass es nur wenige, für Wissenschaftler zugängliche, veröffentlichte Daten gibt. Aus diesem Grund wurde bei der Erstellung des Reference Energy Disaggregation Data Set (REDD), einem Datenset des Massachusetts Institute of Technology (MIT), darauf geachtet, dass sich die Daten für die Anwendungsfälle von NILM eignen. Eine wichtige Anforderung an REDD, die durch den Fokus auf NILM entsteht, ist die Verteilung der Häuser, von denen Daten aufgezeichnet werden. Dies ist wichtig, da ein NILM-Algorithmus, der perfekt für ein bestimmtes Haus funktioniert, unter Umständen für den Energieverbrauch anderer Häuser keine sinnvollen Ergebnisse liefert. Bei der Auf-

zeichnung von REDD wurden deshalb unterschiedliche Häuser in einer größeren Region gewählt. Da die Aufzeichnung der Daten außerdem über einen Zeitraum von mehreren Monaten stattgefunden hat, wurden Daten aus mehreren Jahreszeiten aufgezeichnet. Dadurch können bei Versuchen, die mit diesen Daten durchgeführt werden, auch genau die Daten gewählt werden, die für den Versuch benötigt werden [3].

Inhalt

REDD besteht aus Daten von sechs verschiedenen Häusern in der Region von Boston. Die Daten der Häuser wurden in 2011 über einen Bereich von mehreren Monaten aufgezeichnet. REDD beinhaltet sowohl Informationen über den Energieverbrauch der gesamten Haushalte, als auch von einzelnen Geräten und Steckdosen. Da REDD speziell für die Aufgaben von NILM (vgl. Abschnitt 2.2) aufgenommen wurde, wurden Steckdosen bevorzugt, an denen unterschiedliche Geräte verwendet werden. Um die Verarbeitung der Daten zu erleichtern, wurden diese Steckdosen in kleinere Unterkategorien eingeteilt (zum Beispiel Küchengeräte). Da die Frequenz, mit der die Daten aufgezeichnet werden, für NILM eine wichtige Rolle spielt, wurden die Daten der gesamten Haushalte mit einer Frequenz von 15 kHz aufgezeichnet. Die Aufzeichnung der Daten für die einzelnen Steckdosen erfolgte mit einer Frequenz von 1 Hz. Externe Daten (zum Beispiel Wetter und Jahreszeit) sind für einige Herangehensweisen an NILM ebenfalls von Bedeutung. Da alle Daten die von REDD aufgezeichnet wurden, allerdings auch mit einem UTC Zeitstempel versehen sind, können diese externen Informationen jederzeit herausgefunden werden [3].

Erreichte Ziele

Der REDD Datensatz wurde erstellt um Wissenschaftlern, die in den Bereichen Data Mining und maschinelles Lernen forschen, einen öffentlichen und leicht zugänglichen Datensatz zu bieten. Dieses Ziel wurde insofern erreicht, dass REDD, zum Zeitpunkt der Veröffentlichung, einer der ersten öffentlichen Datensätze mit dem Fokus auf NILM Forschung war. Vom MIT selbst durchgeführte Versuche zeigen, dass sich REDD gut für diesen Zweck eignet und bei Versuchen mit einfachen NILM Algorithmen akzeptable Ergebnisse liefert. Durch REDD wurden auch die Erstellung anderer Datensätze wie zum Beispiel das United Kingdom Domestic Appliance-Level Electricity Dataset (UK-DALE) [4], inspiriert. Diese, später erstellten, Datensätze orientieren sich zum Teil auch an der Struktur und der Frequenz der REDD Daten [3].

2.3.4 UK Domestic Appliance-Level Electricity

Hintergrund

Der Datensatz UK-DALE wurde mit dem Ziel aufgenommen, Vergleichswerte von Geräten in Haushalten aufnehmen zu können, ohne in den Haushalten Sensoren verbauen zu müssen. Dadurch können Haushalte, kostengünstig den Verbauch der eigenen Geräte nachverfolgen. Das Wissen über den Verbrauch einzelner Geräte ist wichtig, da Haushalte ohne Vergleichswerte zwei bis drei Mal mehr Energie verbrauchen als notwendig [4]. Um den Verbrauch von Haushalten darzustellen hat Großbritannien, sowie auch eine Großzahl anderer europäischer Länder, mit dem großflächigen Ausbau von Smart-Metern, in Haushalten und Unternehmen, begonnen. Studien zeigen, dass das Wissen über den Haushaltsverbrauch alleine jedoch noch keinen großen Erfolg bringt. Es ist weit effizienter zu wissen, welche Geräte welchen Anteil des Verbrauches ausmachen [13]. Darum wird, zusätzlich zu dem Smart-Meter Ausbau, noch ein Referenzdatenset benötigt. Dies resultierte im UK-DALE Referenzdatensatz.

Inhalt

Um als Referenzdatensatz für die beschriebene Anwendung verwendet werden zu können, wurden vier Haushalte mit 44,1 kHZ (später auf 16 kHZ umgerechnet um Speicherplatz zu sparen und um mit REDD Programme kompatibel zu sein) gemessen. Jeder dieser Haushalte hatte zwischen fünf und 55 interne Messkreise, welche unterschiedliche Geräte, mit einer Frequenz von $\frac{1}{6}$ Hz, maßen. Die Aufnahmezeit betrug, beim Haushalt, welcher am längsten gemessen wurde, 499 Tage. Dies sichert die Aussagekraft des Datensatzes über die Jahreszeiten hinweg. Damit der Datensatz großflächig in Großbritannien eingesetzt werden kann, wurde er in Großbritannien, mit einer großen demografischen Verteilung der Bewohner, aufgenommen. Um Speicherplatz zu sparen, wurde der Datensatz in das freie, verlustlose Datenformat .flac umgewandelt [4].

UK-DALE ist nicht der erste Datensatz welcher versucht, die Verwendung von NILM zu ermöglichen. Der zuvor besprochene Datensatz REDD war zur Erstellung von UK-DALE bereits vorhanden und beinhaltet ebenfalls Daten, welche NILM (vgl. Abschnitt 2.2) zulassen. Für die geplanten Anwendungsfälle war er jedoch unbrauchbar, da er in Boston aufgenommen wurde [3]. Die Lage, in welchen Datensätze aufgenommen werden, ist durchaus wichtig, da diese die verwendeten Geräte, das Klima und kulturelle Faktoren beeinflusst. Da für Großbritannien, zur Zeit der Erstellung UK-DALEs,

jedoch noch kein Datensatz, welcher NILM zulässt, zur Verfügung stand, musste dieser neue Datensatz aufgenommen werden.

Erreichte Ziele

UK-DALE ermöglicht eine erfolgreiche Verwendung von NILM in Großbritannien. Die angestrebten 16 kHz wurden erreicht und auch die Gerätedaten wurden erfolgreich gemessen. Die gemessenen Daten von UK-Dale zeigen einerseits erfolgreich die gesamte Strom- und Spannungsaufnahme der gesamten Haushalte und andererseits die Disaggregationsdaten der einzelnen Geräte eben dieser Haushalte.

UK-DALE bietet ein freies Datenformat, welches für NILM in Großbritannien verwendet werden kann. Es hat eine sehr hohe Auflösung und ist der Verwendung ähnlich wie REDD. Dies ermöglicht die Einbindung von UK-DALE Daten in REDD Software. Die Daten von UK-DALE können außerdem für die Modellierung elektrischer Netze und der Untersuchung von Smart-Geräten auf Stromnetze verwendet werden [4].

2.4 Evaluierung

In folgendem Teil der Arbeit, werden die Vorteile eines gemeinsamen Parsers, für unterschiedliche Referenzdatensätze, erläutert. Außerdem wird erklärt, wofür ein gemeinsamer Parser benötigt wird.

2.4.1 Vorteile eines gemeinsamen Parsers

Referenzdatensätze werden meist für sehr spezifische Zwecke erstellt. Das führt dazu, dass für die Erstellung der unterschiedlichen Datensätze, verschiedene Anforderungen, im Vordergrund stehen. Deshalb sind die Formate der Datensätze darauf spezialisiert, deren Anforderungen so gut wie möglich zu erfüllen. Durch diese Vorgehensweise entsteht eine große Menge an sinnvollen Daten, die allerdings nicht ohne weiteres miteinander verglichen werden können.

Für Anwendungen wie zum Beispiel NILM und wissenschaftliche Analysen, ist es von Vorteil, viele Daten zur Verfügung zu haben. Auch wenn Datensätze nicht speziell für diese Aufgaben erstellt wurden, könnten die Daten trotzdem von Nutzen sein. Da die Datensätze allerdings in unterschiedlichen Formaten vorliegen, ist es kompliziert deren Daten miteinander zu vergleichen. Ein gemeinsamer Parser wird benötigt, um die verschiedenen Datensätze in ein einheitliches Format zu bringen. Das Ziel dabei ist

es, das Vergleichen und Weiterverarbeiten von Daten zu vereinfachen.

Solange kein gemeinsamer Parser für Datensätze existiert, muss für das Bearbeiten jedes Datensatzes ein extra Parser erstellt werden. Das führt dazu, dass bei Änderungen am gewünschten Ausgangsformat alle verwendeten Parser angepasst werden müssen. Durch das Verwenden eines gemeinsamen Parsers wird verhindert, dass für jeden einzelnen Datensatz, ein zusätzlicher Parser benötigt wird, wodurch weitere Komplexität vermieden wird. Dadurch wird auch der Aufwand, für das Updaten und Warten verringert. Sollte das Ausgangsformat der Datensätze verändert werden, so muss dieses nur an einer Stelle im gemeinsamen Parser verändert werden. Außerdem wird damit sichergestellt, dass das Ausgangsformat, für alle zu parsenden Datensätze einheitlich ist. Soll ein Datensatz geparst werden, der nicht vom Parser unterstützt wird, muss kein neuer Parser erstellt werden, sondern nur der neue Datensatz zum gemeinsamen Parser hinzugefügt werden.

2.4.2 Anforderungen an einen gemeinsamen Parser

Da der Parser als gemeinsamer Parser für mehrere Datensätze dient, ist seine Erweiterbarkeit eine wichtige Anforderung. Er muss die Möglichkeit bieten neue Datensätze einbinden zu können, ohne das bereits vorhandene Programm umzuschreiben. Außerdem muss der Parser dem Benutzer die Auswahl der zu parsenden Daten ermöglichen, da nicht in jedem Datensatz die gleichen Daten aufgezeichnet sind. Es muss möglich sein nur bestimmte, vom Benutzer gewählte, Datentypen aus allen Daten herauszufiltern (zum Beispiel nur Leistungswerte). Des Weiteren muss auch die Auswahl der Geräte, von denen Daten herausgefiltert werden sollen, dem Benutzer überlassen sein. Da Referenzdatensätze sehr umfangreich sind und oft nur Daten aus einem gewissen Zeitraum interessant sind, soll der Benutzer diesen auch auswählen können. Zusätzlich soll es dem Benutzer auch erlaubt sein das Intervall, in dem die Daten verwendet werden, zu bestimmen. Je nach eingegebenem Intervall muss der Parser dann entscheiden, welche Werte verwendet und welche verworfen werden.

3 Praktischer Teil

In diesem Kapitel wird die Aufgabenstellung der gesamten Arbeit beschrieben. Auch wird die Umsetzung für die Erstellung eines Parsers genauer erläutert. Dazu zählen eine genaue Analyse und Vorbereitung der ausgewählten Referenzdaten. Des Weiteren befasst sich dieses Kapitel mit der Implementierung eines gemeinsamen Parsers. Es wird außerdem beschrieben, wie der erstellte Code überprüft wurde, um Fehler bestmöglich ausschließen zu können. Zum Abschluss werden die Erkenntnisse dieses praktischen Teils dargelegt.

3.1 Beschreibung der Aufgabenstellung

Auf Grund der Tatsache, dass die Referenzdaten in verschiedenen Formaten vorliegen, ist ein direktes Vergleichen nicht möglich. Um die Daten vergleichen zu können, müssen die zu vergleichenden Daten, durch einen gemeinsamen Parser in das gleiche Format gebracht werden. Vergleichbare Referenzdaten werden vor allem für smarte Energienetze immer mehr gebraucht.

Ziel dieser Arbeit ist es, einen Parser zu programmieren, welcher die in Abschnitt 2.3 genannten Referenzdaten richtig einliest und parst. Hierbei ist es wichtig, dass für diese Referenzdaten nicht jeweils ein eigener Parser programmiert wird, sondern ein kompakter, gemeinsamer Parser entwickelt wird. Dabei soll letztendlich der zukünftige Nutzer dieses Parsers die Möglichkeit haben, die Parameter bezüglich des zu parsenden Datensatzes, der gewünschten Frequenz, der Anzahl der Haushalte, der enthaltenen Daten und vielem mehr einzugeben.

Um das Ergebnis, so genau wie möglich, an die Bedürfnisse und Vorhaben des Benutzers anpassen zu können. Diese Daten sollen daraufhin automatisch gefiltert in die Datenbank gespeichert werden.

3.2 Analyse der Daten

ADRES

Der Datensatz ADRES [1] steht in einer .mat Datei zur Verfügung. Darin sind zwei verschiedene Tabellen enthalten. Die erste Tabelle enthält Spannungsdaten und die zweite Tabelle enthält Leistungs- und Wirkleistungsdaten.

Werden die Daten aus der Spannungstabelle genauer analysiert (vgl. Tabelle 3.1), kann festgestellt werden, dass die Daten in einem Wertebereich von 220 bis 250 V liegen. Es sind insgesamt 1209600 Zeilen und 90 Spalten an Daten vorhanden.

Tabelle 3.1: Auszug Spannungswerte [1]

225,17000	223,67999	222,89999	233,26666	237,10834
225,17999	223,70000	222,91000	233,13800	237,10400
225,14999	223,70000	222,89999	233,20799	237,11200
225,02000	223,64999	222,89000	233,27600	237,12801
225,17999	223,70000	222,89999	233,34000	237,04800
225,23000	223,69000	222,89000	233,26601	237,02400

Die Tabelle, welche aus Leistungs- und Wirkleistungsdaten besteht (vgl. Tabelle 3.2), enthält gleich viele Daten. Die ungeraden Spalten wurden mit den geraden Spalten verglichen. Daraus resultierte, dass die Daten in den ungeraden Spalten immer größer sind, als die in den geraden Spalten. Deshalb sind die Daten in den ungeraden Spalten die Leistung und die Daten in den anderen Spalten die Wirkleistung. Es kommen auch öfters negative Werte vor, was sich darauf zurückschließen lässt, dass die erzeugte Leistung nicht immer zur Gänze verbraucht wurde. Bezüglich des Wertebereichs können keine Rückschlüsse gezogen werden, da die Werte sowohl im negativen als auch im positiven Bereich ein großes Spektrum vertreten.

Tabelle 3.2: Auszug Leistungs-/Wirkleistungswerte [1]

40,200001	-14,670000	22,150000	3,1889999	141,50000
41,500000	-14,980000	22,180000	3,5290000	140,70000
40,799999	-15,120000	22,040001	3,4080000	141
40,900002	-14,660000	22,219999	2,9560001	141,10001
40,700001	-14,860000	22,120001	3,1480000	141
41,500000	-14,770000	22,120001	3,0030000	140,39999

Grundsätzlich kann gesagt werden, dass die einzelnen Werte, wenn sie spaltenweise gelesen werden, sich kaum voneinander unterscheiden. Deshalb kann davon ausgegangen werden, dass pro Spalte ein Haushalt dargestellt wird. Leider wurden diese Tabellen seitens von ADRES [1] nicht ausreichend dokumentiert. Aus diesem Grund wurden die oben genannten Merkmale im Laufe dieser Arbeit ausgearbeitet.

GREEND

Beim Datensatz GREEND [2] gibt es pro Haushalt einen eigenen Ordner, in welchem sich .csv Dateien befinden. Es wurden insgesamt acht Haushalte aufgenommen, woraus sich die Ordner building0 bis building7 ergeben. Die enthaltenen .csv Dateien bestehen zuerst aus einem Zeitstempel, die darauffolgenden Werte sind Wirkleistungsdaten (vgl. Tabelle 3.3). Der verwendete Zeitstempel liegt im Uniplexed Information and Computing System (UNIX) Zeitformat vor. Die weiteren Spaltenüberschriften sind die Media-Access-Control (MAC) Adressen der einzelnen Geräte.

Im Vergleich zum ADRES-Datensatz [1] liegt der Wertebereich der Wirkleistungsdaten im positiven Bereich. Teilweise sind die Werte 0, was darauf zurückzuführen ist, dass das angeschlossene Gerät momentan keine Energie braucht. Es fällt auf, dass oftmals Werte „NULL“ sind, was dadurch entstehen kann, dass dieses Gerät derzeit nicht erreichbar ist. Dies kann dadurch entstehen, dass das Gerät nicht angeschlossen ist, oder dass zurzeit kein Verbindungsaufbau möglich ist.

Tabelle 3.3: Auszug Haushalt 0, 10.12.2013 [2]

timestamp	000D6F0002906FA7	000D6F0002907BA2	000D6F0002907BC8
1386632839.373578	0.0	0.0	2.2675751981909307
1386632840.374721	0.0	0.0	0.0
1386632841.832325	0.0	NULL	0.0
1386632842.874697	0.0	NULL	0.0
1386632843.915006	0.0	NULL	0.0

Es wurden pro Haushalt unterschiedlich viele Werte aufgenommen. Die Anzahl der aufgenommenen Werte schwankt zwischen knapp 800 und mehr als 80000 Werte. Auffällig ist, dass die Haushalte nicht immer an den gleichen Tagen gemessen wurden. Auch wurden die Daten meist nicht durchgängig aufgenommen.

REDD

Die Daten des REDD-Datensatzes [3] liegen in .dat Dateien vor. Für diesen Datensatz wurden sechs Haushalte aufgezeichnet. Dabei werden zwei verschiedene Arten von Daten unterschieden. Es gibt hochfrequente und niederfrequente Daten.

Die hochfrequenten Daten (vgl. Tabelle 3.4) wurden nur für die Haushalte drei und fünf aufgezeichnet. Hierbei gibt es pro Haushalt drei verschiedene Dateien. Zwei dieser Da-

teien enthalten zuerst einen Zeitstempel und danach die Anzahl der Perioden für diesen Verlauf (Signal). Darauf folgen die einzelnen Stromwerte. In der anderen Datei sind, nach Zeitstempel und Anzahl der Perioden, die Spannungswerte aufgelistet. Die Strom- und Spannungsdaten sind wie bei ADRES [1] im positiven und negativen Bereich. Auch hier lassen sich die negativen Daten darauf zurückzuführen, dass mehr Energie erzeugt als verbraucht wurde. Insgesamt werden pro Zeitstempel 275 Dezimalwerte für Strom beziehungsweise Spannung aufgezeichnet.

Tabelle 3.4: Auszug hochfrequente Stromwerte Haushalt 3 [3]

Zeitstempel	Anzahl der Perioden	Stromwert 1	Stromwert 2	Stromwert 3
1302922280.014060	1782.000000	-0.017231	-0.017167	-0.017059
1302922309.800943	1782.000000	-0.017288	-0.017300	-0.017320
1302930686.027452	2760.000000	0.002427	0.007893	0.020264
1302930732.154724	2065.000000	0.004701	0.008471	0.022125
1302930766.559476	206.000000	0.002921	0.006746	0.021267
1302930770.002607	258.000000	0.008324	0.011263	0.022454

Für die niederfrequenten Daten (Tabelle 3.5) wurden alle sechs Haushalte aufgenommen. Hier gibt es pro Haushalt einen Ordner mit einer unterschiedlichen Anzahl von .dat Dateien. Diese Dateien enthalten zuerst einen Zeitstempel und darauffolgend Leistungswerte für die verschiedenen Geräte, welche durch einzelne Nummern im Dateinamen gekennzeichnet sind. Der Wertebereich liegt im positiven Bereich.

Tabelle 3.5: Auszug niederfrequente Leistungswerte Haushalt 1, Kühlschrank [3]

Zeitstempel	Leistung
1304287970	5.00
1304287974	6.00
1304287977	2.00
1304287981	404.00
1304287989	293.00
1304287993	474.00

Sowohl die hochfrequenten als auch die niederfrequenten Daten haben einen Zeitstempel im UNIX- Zeitformat. Des Weiteren ist bei den niederfrequenten Daten eine genaue Aufschlüsselung, bezüglich Haushalt und Gerät, möglich.

UK-DALE

Bei UK-DALE wurden 16 kHz-Daten aufgezeichnet. Außerdem hat UK-DALE eigene Daten, welche für die verschiedenen Geräte aufgenommen wurden. Die 16 kHz-Daten befinden sich in einem Ordner mit der Ordnerstruktur Haushalt, Jahr, Woche. Darin befinden sich mehrere .flac Dateien. UK-Dale hat den Zeitstempel für die 16 kHz-Daten nicht direkt in den Daten sondern im Dateinamen der jeweiligen .flac Datei.

Flac ist ein Dateiformat zur verlustlosen Komprimierung. Um Daten aus solchen Dateien lesen zu können, wurden diese in eine Datei ohne Dateiendung umgewandelt. Daraus entstanden zwei verschiedene Dateien. Eine Datei enthält Spannungswerte (vgl. Tabelle 3.6), die andere Stromwerte (vgl. Tabelle 3.7). Darin befinden sich hexadezimale Werte, welche mit einem Hex-Editor in dezimale Werte umgewandelt werden können. Diese Werte werden hintereinander angezeigt, das heißt es sind keine Spalten vorhanden.

Tabelle 3.6: Auszug Stromwerte Haushalt 1, Woche 32, Datei 118 [4]

-0,308205452	-0,542049944	-0,505493234	-0,740459251	0,303316102	0,1697919

Tabelle 3.7: Auszug Spannungswerte Haushalt 1, Woche 32, Datei 118 [4]

-37,63286841	-223,4090734	-333,7527632	-330,4802193	-158,7298423	40,53375025

Die Daten, welche einzelnen Geräten zugeordnet werden können, werden in die jeweiligen Haushalte unterteilt. In den Haushalts-Ordnern befinden sich .dat Dateien, welche nach den Geräten benannt wurden. Welche Nummer welchem Gerät zugeordnet werden kann, steht in der labels.dat Datei. Die .dat Dateien der Geräte beinhalten Leistungsdaten (Tabelle 3.8).

Tabelle 3.8: Auszug Leistungswerte Haushalt 1, TV [4]

Zeitstempel	Leistung
1352500098	98
1352500104	100
1352500110	99
1352500116	96
1352500122	100
1352500128	99

Sowohl die Strom- als auch die Spannungswerte liegen im positiven und negativen Bereich. Beim Strom sind die positiven und negativen Werte auf Wechselstrom zurückzuführen. Die Leistungswerte der spezifischen Geräte liegen im positiven Wertebereich.

Abschließend kann gesagt werden, dass die vier verschiedenen Referenzdatensätze, mit Ausnahme von UK-Dale [4] und REED [3], total unterschiedlich sind. Die Datensätze UK-DALE [4] und REDD [3] haben eine zusätzliche Unterteilung der Daten bezüglich der unterschiedlichen Geräte. Bei den anderen Datensätzen ist dies nicht möglich. Auch die Anzahl der gemessenen Haushalte varriiert, wobei teilweise nicht einmal durchgehend Daten aufgenommen wurden. ADRES ist der einzige Datensatz ohne Timestamp.

3.3 Implementierung

In diesem Teil der Arbeit wird sowohl die Implementierung des Parsers als auch der Datenbank und deren Funktionen genauer beschrieben. Außerdem wird darauf eingegangen, in welcher Weise bestimmte Datensätze vor-bearbeitet werden müssen, um eine Verwendung des Parsers zu ermöglichen.

3.3.1 Vorbereitung der Daten

Um die verschiedenen Datenformate in den Parser einlesen zu können, war eine gewisse Vorarbeit notwendig. Das .flac Datenformat von UK-DALE ist eine Art der verlustlosen Komprimierung, welche nicht so einfach durch einen C# Reader gelesen werden kann. Daher muss dieses Datenformat, vor der eigentlichen Bearbeitung, dekomprimiert werden.

```
avconv -i "$inputfile" -filter_complex \
'channelsplit=channel_layout=2[FL][FR]' \
-map '[FL]' -f f64le "tempvolts" \
-map '[FR]' -f f64le "tempamps" > \
/dev/null 2>&1
```

Dieser Codeabschnitt beginnt ab einem bestimmten Ordner, rekursiv alle .flac Dateien in die respektive Spannungs- und Stromdateien zu entpacken. Der vordere Teil der .flac Dateien, welcher dem Zeitstempel entspricht, wird beibehalten.

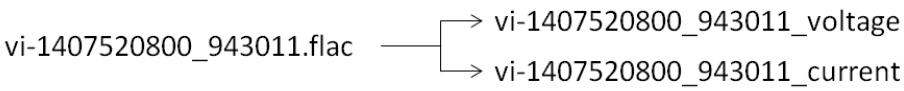

Abbildung 3.1: UK-Dale Datenvorbereitung

Die resultierenden Dateien beinhalten Hex Werte von Strom und Spannung und sind damit einfach lesbar [14]. Des Weiteren kann der Zeitstempel aus dem Dateinamen

entnommen werden, was die Verarbeitung der Daten im Parser weiter vereinfacht.

Die ADRES .mat Daten wären rein theoretisch lesbar, durch die unangenehme Struktur wurden diese vor der eigentlichen Anwendung jedoch in .csv Daten umgewandelt. Diese Umwandlung geschieht in MATLAB mit folgendem Code.

```
1 FileData = load('ADRES_Daten.mat');
2 dlmwrite('ADRES_Daten_U.csv', FileData.Data.U, 'delimiter',
     ',', 'precision', 9);
3 dlmwrite('ADRES_Daten_PQ.csv', FileData.Data.PQ, 'delimiter',
      ',', 'precision', 9);
```

Der Code lädt die ADRES-Daten mit dem Namen ADRES_Daten.mat. Diese .mat Datei besteht aus einer Struktur, welches jeweils aus Wirk– und Blindleistungswerten und den Spannungswerten bestehen. Diese Werte werden der Reihe nach in zwei unterschiedliche .csv Dateien gespeichert.

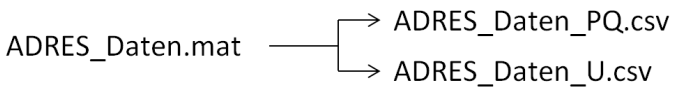

Abbildung 3.2: ADRES Datenvorbereitung

Das Ergebnis dieser Umwandlung ermöglicht eine sehr einfache Handhabung der Daten. Die GREEND und REDD Daten liegen standardmäßig als .csv bzw. als .dat Datei vor [3] [2]. Eine Vorbereitung der Daten ist, durch die einfache Lesbarkeit, daher nicht notwendig.

3.3.2 Parser

Um sicherzustellen, dass der Parser für alle verwendeten Datensätze gleich funktioniert und auch die Möglichkeit bietet neue Datensätze problemlos zu implementieren, wurde eine Vererbungsstruktur für den Aufbau des Parsers verwendet. Es wurde eine abstrakte Klasse namens IDataHandler erstellt, die danach in allen Handlern für die einzelnen Datensätze implementiert wurde. Dadurch wurde die minimale Funktionalität der Handler, die für das Parsen der Datensätze zuständig sind, festgelegt. Die Umsetzung dieser Funktionen kann allerdings für jeden Datensatzhandler unterschiedlich gestaltet werden.

```
1 abstract class IDataHandler
```

```
2  {
3      protected String Path;
4      protected List<TypeEnum> types;
5      protected List<ApplicationEnum> application;
6
7      protected DateTime starttime;
8      protected DateTime endtime;
9      protected double frequenzy;
10
11     public abstract void ParseDirectory();
12     protected abstract void processFile(String path);
13     protected abstract void filterByApplication();
14     protected abstract void filterByTime(DateTime minDT,
           DateTime maxDT);
15     protected abstract void filterByType();
16     protected abstract void fillDBWithMeters();
17
18     public void setPath(String _path){ this.Path = _path; }
19 }
```

Der IDataHandler enthält mehrere Variablen, die den Konstruktoren der Unterklassen mitgegeben werden. Dabei musste darauf geachtet werden, dass die Reihenfolge der Parameter für die Konstruktoren in allen Klassen übereinstimmt, da diese sich nicht von der Superklasse vererben lässt. Diese mitgegebenen Werte werden im weiteren Verlauf für das Einlesen der Daten und deren Filterung benötigt. Für das Filtern nach Typen und Applikationen werden Listen mit Enums verwendet, da in diesen Fällen auch mehrere Elemente herausgefiltert werden können. Hier ist zu beachten, dass bei der Implementierung eines neuen Datensatzes eventuell Geräte verwendet werden, die noch nicht vom Parser unterstützt werden. Diese müssen extra in das ApplicationEnum hinzugefügt werden. Die fillDBWithMeters Funktion wird dazu verwendet, neue Smart Meters in die Datenbank einzufügen. Die Kennung (ID), die diesen Metern zugewiesen wird, kann daraufhin mit den jeweils zutreffenden Datensätzen verknüpft werden. Da die verwendete Datenbank zusätzliche Informationen über die eingetragenen Smart Meter benötigt, die in dieser Arbeit nicht alle zugänglich waren, wurden diese mit Default Werten initialisiert. Um die Aufrufe der einzelnen Handler konsistent zu gestalten, wird als Pfad der Datensätze immer der des obersten Ordners in der Ordnerstruktur des Datensatzes verwendet. Dieser Pfad wird dann in der ParseDirectory Funktion weiterverarbeitet. Durch diese Vorgehensweise wurde sichergestellt, dass jeder DataHandler auf die gleiche Weise aufgerufen werden kann. In folgendem Beispiel werden aus dem GREEND Datensatz alle enthaltenen Daten von Fernseher und Radios

am 06.12.2013 zwischen der Uhrzeit von 23:50:00 und 23:59:59 herausgefiltert.

```
DatabaseDriver myDatabase = new DatabaseDriver();
List<ApplicationEnum> tempApplicationList = new List<
   ApplicationEnum>();
tempApplicationList.Add(ApplicationEnum.TV);
tempApplicationList.Add(ApplicationEnum.Radio);
IDataHandler handler = new GREENDDataHandler(myDatabase, new
   DateTime(2013, 12, 06, 23, 50, 00), new DateTime(2013, 12,
    06, 23, 59, 59), tempApplicationList, null, 1);

handler.setPath(@"E:\Lastprofile\GREEND_0-1_311014");
handler.ParseDirectory();
```

Nach der Erstellung eines Datenbanktreibers wird der DataHandler erstellt (hier GREEND). Dieser wird mit einer Liste zur Filterung von Geräten, einem Datums- und Zeitintervall, einer Frequenz und einer Liste der Datentypen (hier null) befüllt. Der erstellten und mitgegebene Datenbank Treiber wird für die Verbindung mit der Datenbank benötigt. Im nächsten Teil der Arbeit wird darauf noch genauer eingegangen. Die setPath Funktion des Handlers wird dazu verwendet, den Pfad zum Datensatz anzugeben. Das eigentliche Parsen und Speichern in die Datenbank startet mit der ParseDirectory Funktion und wird dann je nach DataHandler individuell ausgeführt.

Der grundsätzliche Aufbau jeder DataHandler-Klasse ist zuerst den gegebenen Ordner, rekursiv zu durchlaufen. In diesem rekursiven Durchlauf sollen daraufhin, alle passenden Dateien durch den Parser in die Datenbank geschrieben werden. Um diese Aufgabe zu vereinfachen, wurde für jede Datenbanktabelle, eine Klasse erstellt. Diese Klassen stellen also die Tabellen in der Datenbank dar. Dies ermöglicht es einen Eintrag zu bearbeiten und nach erfolgreicher Bearbeitung, diesen direkt in die Datenbank einzufügen. Um dies zu ermöglichen, erben alle Datenklassen von IData, welche die Funktion GetData vererbt. Diese Funktion nimmt die Werte des Datenelements und übersetzt sie in, für die Datenbank lesbare, Strings.

```
powerValue[i] = voltValues[i] * ampValues[i];
MeterData temp = new MeterData(this.housesList[
   householdNumber - 1].Item1);
temp.SetTimestamp(realTimestamp.ToString());
temp.SetPowerP1(powerValue[i]);
temp.SetVoltage(voltValues[i]);
myDatabase.InsertMeterDataIntoDb(temp);
```

In diesem Beispiel (aus UKDaleDataHandler) kann man die Verwendung einer solchen Datenklasse beobachten. Die Klasse MeterData repräsentiert die Messaufnahme eines bestimmten Geräts zu einer bestimmten Zeit. Der Parameter, welcher dem Konstruktor mitgegeben wird, ist die ID des Smart Meters, welcher diesem Datensatz zugewiesen ist. Diese ID ist, im genannten Beispiel, durch die Nummer des Haushalts gegeben, welche aus dem Ordnernamen herausgefunden werden kann. Nach dem instanziieren des Objekts kann man die verwendeten Werte über Setter setzten und über den Datenbanktreiber einfügen. Dieses Prinzip funktioniert mit allen Datenklassen, welche von IData erben.

3.3.3 Datenbank

Um die Datenbank zu implementieren, musste diese erst, nach den Vorgaben des Zielformats, modelliert werden. Dies geschah im Structured Query Language (SQL) Power Architect.

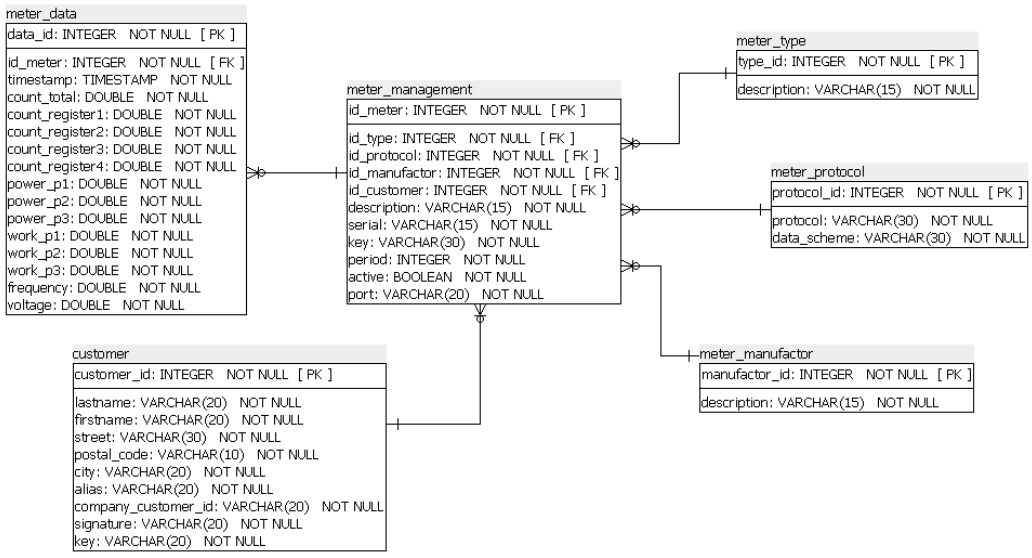

Abbildung 3.3: Datenbankmodell SQL Power Architect

Aus der modellierten Datei wurde daraufhin der SQL Code erzeugt. Daraufhin wurde dieser SQL Code als Query in der Service Based Database ausgeführt. Der Zugriff auf die Datenbank über den Parser wird durch die DatabaseDriver Klasse realisiert. Diese Klasse stellt verschiedene Funktionen zur Verfügung. Um die Verbindung auf- und abzubauen, werden die Funktionen ConnectToDB und CloseConnection verwendet.

```
connection = new System.Data.SqlClient.SqlConnection();
connection.ConnectionString = "Data Source =(LocalDB)\\
   MSSQLLocalDB; AttachDbFilename = Pfad\\MyDatabase.mdf;
   Integrated Security = True";
try
{
    connection.Open();
}
catch (Exception e)
{
    Console.WriteLine(e.Message);
}
```

Die Funktion ConnectToDB muss, um die Funktion des Parsers bereitzustellen, angepasst werden. Um die korrekte Verbindung sicherzustellen, muss der Pfad zur Service Based Database, im ConnetionString, korrigiert werden. Der Datenbanktreiber stellt außerdem eine Menge an Debugfunktionen dar, welche einen Dump pro Datenbanktabelle zulassen. Dies ist für die Entwicklung neuer Anwendungen von Bedeutung. Eine weitere Gruppe an Funktionen, welche für Entwicklungszwecke interessant sind, sind die DeleteAll-Funktionen. Diese löschen den Inhalt einer gesamten Tabelle und sind für jede Tabelle implementiert. Des Weiteren gibt es je Tabelle die Funktion, eine einzigartige ID dieser Tabelle herauszufinden. Diese ID wird zufällig aus dem gesamten int32 Bereich erzeugt und von der Funktion zurückgegeben, wenn die ID in der Tabelle noch nicht vorkommt. Die wohl wichtigsten Funktionen des Datenbanktreibers sind die Insert-Funktionen. Diese erlauben es dem Verwender, wie im vorherigen Abschnitt beschrieben, ein Datenelement zu übergeben und dieses direkt in die Datenbank zu schreiben.

Bsp: GREENDDataHandler siehe Anhang

Wie man sehen kann, ist die Verwendung der Funktion sehr einfach gestaltet. Wie bei allen Datenbankfunktionen, muss als erstes eine Verbindung hergestellt werden. Danach wird ein SQL Befehl mit den korrekt umgewandelten Daten befüllt und ausgeführt. Wie alle wichtigen Funktionen im Parser wird jede Insert-Funktion durch Exception Handling abgesichert.

3.4 Tests

Die Tests, die im folgenden Teil der Arbeit beschrieben werden, wurden alle auf mehreren Rechnern durchgeführt, um sicherzustellen, dass der Parser auch auf unterschiedli-

chen Maschinen richtig funktioniert. Da keine speziellen Hardwareanforderungen für die Aufgabestellung dieser Arbeit notwendig sind, können die hier beschriebenen Tests auf jedem beliebigen Rechner, mit mindestens 4 GB Arbeitsspeicher, durchgeführt werden.

Da die, für diese Arbeit ausgewählten Datensätze, teilweise in problematisch zu parsenden Formaten vorliegen, wurde der erste Test sofort nach der Vorverarbeitung dieser Datensätze durchgeführt. Dieser Test war ein einfacher Vergleich der Daten vor und nach deren Verarbeitung, um sicherzugehen, dass während der Verarbeitung der Daten keine Fehler aufgetreten sind.

Dadurch, dass die Programmierung des Parsers in mehrere einzelne Phasen aufgeteilt wurde, wurden in jeder Phase mehrere Tests durchgeführt und erst nach Sicherstellung der Funktionalität mit der nächsten Phase begonnen, damit nicht mit einem fehlerhaften Code weitergearbeitet wird.

Die erste Phase der Programmierung beschäftigte sich nur mit dem Navigieren in der Ordnerstruktur der Datensätze und dem Einlesen der benötigten Daten aus den Datensätzen. Um die späteren Tests zum Einlesen der Daten zu erleichtern, wurde anfangs nur die Navigation der Ordnerstruktur ausprogrammiert. Um zu prüfen, ob das Programm wirklich alle benötigten Files der Datensätze erreicht, wurden die Filepfade, die das Programm erreicht manuell mit denen aus der Ordnerstruktur verglichen. Die Tests wurden als bestanden bezeichnet, wenn alle Files aus der Liste und keine zusätzlichen gefunden wurden. Nachdem die automatische Navigation in den Datensätzen funktionsfähig war, wurde mit dem Einlesen der Daten begonnen. Aufgrund der Größe der Datensätze wurden für die Tests nicht die gesamten Datensätze, sondern nur die ersten 500 Werte aus jedem benötigten File der Datensätze verwendet. Es wurde zuerst für jeden Datensatz einzeln geprüft, ob das Einlesen der Daten korrekt funktioniert, indem die eingelesenen Daten ausgegeben und mit den Daten aus dem Originalfile des Datensatzes verglichen wurden. Nachdem die Funktionen zum Einlesen der einzelnen Datensätze separat getestet wurden, wurden die Funktionen in ein einzelnes Programm zusammengeführt. In dem dadurch entstandenen Programm wurden danach, als weitere Tests, sowohl unterschiedliche Kombinationen der Datensätze als auch alle Datensätze gleichzeitig eingelesen.

An diesem Punkt der Arbeit war das exakte Datenbankformat, auf welches die Daten geparst werden sollten, noch nicht bekannt. Deshalb wurde zuerst an den Filtern für die eingelesenen Daten gearbeitet. Ähnlich wie beim Einlesen der Daten, wurden auch in dieser Phase die ersten Tests separat für einzelne Filter der Datensätze durchgeführt.

Da das korrekte Einlesen der Daten bereits getestet wurde, wurden für die Application und Datentyp Filter manuell überprüft ob wirklich nur Daten von Geräte und Datentypen, die dem Filter übergeben wurden, eingelesen werden. Für die Tests der Zeitfilter wurden mehrere Zeitspannen die im Bereich des zu testenden Datensatzes ausgewählt. Um zu testen ob die gefilterten Daten wirklich im gewählten Zeitbereich liegen wurden darauf geschaut, dass die Werte direkt vor dem ersten und nach dem letzten gefilterten Wert nicht mehr innerhalb des ausgewählten Filterbereiches liegen konnten. Um die Frequenzfilter zu testen, wurde eine passende Frequenz ausgewählt und die ersten zwanzig Werte die mit dieser Frequenz übereinstimmen in einer Liste gespeichert. Die vom Programm herausgefilterten Werte wurden danach mit jenen aus der Liste verglichen. Um Fehler leichter zu erkennen, wurde bei jedem, nicht übereinstimmenden Wert eine Fehlermeldung ausgegeben. Der Filter wurde als funktionsfähig erachtet, wenn bei der Durchführung keine Fehlermeldungen ausgegeben wurden. Der nächste Test bestand darin Filterkombinationen für einen einzelnen Datensatz zu testen, um sicherzustellen, dass die Filter untereinander keine Probleme verursachen. Danach wurden die einzelnen Filter erneut dem Hauptprogramm hinzugefügt. Der nächste Schritt bestand aus mehreren Test Kombinationen unterschiedlicher Datensätze und Filteroptionen und einem Versuch, bei dem alle Datensätze mit allen Filteroptionen gleichzeitig getestet wurden.

Die letzte Phase der Programmierung bestand aus der Implementierung der Datenbank. Da an diesem Punkt bereits alles andere auf Funktionstauglichkeit getestet wurde, musste nach der Einbindung der Datenbank nur noch überprüft werden, ob die Daten auch wirklich in die Datenbank gespeichert werden. Dies wurde sichergestellt indem alle Daten die, in die Datenbank gespeichert wurden, während des Speichervorgangs ausgegeben wurden und danach alle Daten aus der Datenbank ausgelesen wurden. Bei Übereinstimmung der gespeicherten und ausgelesenen Daten wurde die Datenbankanbindung als erfolgreich angesehen.

3.5 Erkenntnisse

In diesem Teil der Arbeit wird darauf eingegangen, welche Schwierigkeiten bei der beschriebenen Implementierungsherangehensweise aufgetreten sind und wie diese bewältigt wurden. Die erste Hürde war es, ein einfach erweiterbares und verwendbares Softwaredesign zu wählen. Um dies zu erreichen, wurde stark auf die Objektorientierte Programmierung (OOP)s-Prinzipien aufgebaut. Während der Polymorphismus eine hohe Wiederverwendbarkeit implementierte, half die starke Kapselung bei der Teamar-

beit. Diese erleichterte das Entwickeln durch mehrere Personen enorm. Gemeinsam mit Abstraktion und Vererbung wird es also ermöglicht, gleichzeitig und ohne Namensgebungsprobleme, zu programmieren.

Da der UK-Dale Datensatz dem REDD Datensatz nachempfunden wurde, war es eine angenehme Erkenntnis zu erfahren, dass die beiden Datenformate ähnlich aufgebaut sind. Die Erkenntnis, dass die Disaggregationsdaten der beiden Datensätze beinahe ident sind, hat die Entwicklungszeit deutlich verkürzt (siehe Abschnitt 2.3.3 bis 2.3.4).

Das erste Problem, welches aufgetreten ist, sind die enorm großen Dateien von REDD und UK-Dale (selbst nach der Umwandlung). Um diese zu bewältigen und um die Arbeitsspeicheranforderung möglicher Verwender nicht ins Unzumutbare zu treiben, wurden diese Daten gepuffert eingelesen. Dadurch wird der verwendete Arbeitsspeicher minimiert.

Ein weiteres Problem, welches bei der Entwicklung des UK-Dale Handlers aufgetreten ist, war das Strom und Spannung in zwei externen Files, ohne zuweisenden Zeitstempel abgelegt waren. Da Strom nicht in unserem Datenbankmodell vorkommt, musste dieser, mit der Formel $P = U * I$ in Leistung umgerechnet werden. Um dieses Problem zu lösen, wurden zwei Filereader gleichzeitig gestartet. Da der Zeitstempel zum Startzeitpunkt des Lesens einer Datei aus dem Dateinamen ausgelesen werden konnte, war es möglich die Zeit über einen internen Zähler und dem Wissen über die verwendete Frequenz (16 kHz) herauszufinden. Dies ermöglichte es, die Leistung aus, der nun zusammenfassbaren Spannung und dem Strom zu errechnen.

Die Datenbankanbindung resultierte in einer Menge an Problemen. Unter anderem muss bei der Verwendung einer SQL Verbindung immer darauf geachtet werden, dass diese korrekt auf- und abgebaut wird. Ansonsten läuft man in das Problem, das eine .mdf Datenbank immer nur eine gewisse Anzahl an Verbindungen aufrechterhalten kann. Um dieses Problem zu minimieren, implementieren sämtliche Datenbankfunktionen die Verbindungsauf– und Abbaufunktionen bereits selbst. Ein weiteres Problem der Datenbank befasst sich mit dem SQLCommand und dem SQLDataReader. Diese Klassen dürfen jeweils nur einmal instanziiert werden, was häufig zu Problemen führte. Diese Probleme wurden durch die Verwendung des using Ausdrucks behoben. Dadurch wird ein Objekt, nach der Verwendung sofort wieder zerstört, was Duplikate verhindert und das Programmieren vereinfacht.

Aktuell kann mit dem Programm, über die Konsole, ausgewählt werden, welche Datensätze eingelesen werden sollten. Des Weiteren kann man einen Filterzeitraum, eine

Filterfrequenz und eine Liste mit Gerät– und Anwendungsfiltern auf die auszulesenden Daten anwenden. Diese Daten werden daraufhin in das beschriebene Datenbankmodell umgewandelt (siehe Abschnitt 4.3.3), gefiltert und der Reihe nach eingefügt. Der Vorgang des Einfügens ist die noch größte Schwachstelle des Programms. Dadurch dass jede SQL Querry einzeln ausgeführt werden muss, dauert dies sehr lange. Sinnvoller wäre es, eine Transaktion zu starten, und dieser Transaktion mehrere Querries hinzuzufügen. Wenn nun die Transaktion durchgeführt wird, muss nur einmal eine Verbindung hergestellt und die Daten übertragen werden. Dies wäre ein wichtiger Ansatzpunkt für eine höhere User Acceptance.

In Zukunft wäre es gut vorstellbar, in diesen Parser noch weitere Datensätze einzubauen um die Effektivität weiter zu erhöhen. Ein weiteres empfehlenswertes Upgrades wären, die Implementierung eines, nach User Interface (UI) Richtlinien erstellten, Graphical User Interface (GUI)s.

4 Zusammenfassung/Ausblick

Für diese Arbeit wurden vier verschiedene Referenzdaten, UK-DALE [4], REDD [3], ADRES [1] und GREEND [2] ausgewählt. Sie wurden deshalb ausgewählt, weil sie im europäsichen Raum und in Österreich aufgenommen wurden und eine bestimmte Auflösung haben. Bei der Gegenüberstellung der Referenzdaten, wurde erkannt, dass die einzelnen Energiereferenzdatensätze unterschiedliche Formate haben. Dies ist darauf zurückzuführen, dass jeder Datensatz für einen anderen Anwendungsfall erstellt wurde. So wurden zum Beispiel REDD [3] und UK-DALE [4] aufgenommen, um NILM (vgl. Abschnitt 2.2) zu ermöglichen. ADRES [1] hingegen, wurde erstellt, um den Energieverbrauch der Haushalte bestmöglich vorherzusagen und so auf heimische Ressourcen zurückgreifen zu können. Dies hatte die Erstellung eines neuen Lastprofils zur Folge.

Auf Grund der Tatsache, dass die Energiereferenzdaten in unterschiedlichen Formaten vorliegen und somit ein Vergleich dieser kompliziert und aufwendig ist, wurde ein gemeinsamer Parser erstellt. Um den programmierten Parser verwenden zu können, müssen, die Datensätze UK-DALE [4] und ADRES [1] vorbearbeitet werden (vgl. Abschnitt 3.3.1). Für den Parser wurde ein Interface erstellt, um die Wiederverwendbarkeit für weitere Referenzdatensätze zu gewährleisten. Grundlegende Funktionen des Parsers sind zunächst das Einlesen und das Ausgeben der Daten in eine Datenbank. Es wurden auch Funktionen für das Filtern der Daten bezüglich Zeitraum, Art der Daten, also Strom, Spannung und so weiter, und Geräte implementiert. Auf Grund dessen, dass zum Beispiel ADRES [1] keine Unterteilung in Geräte und keinen Zeitstempel hat, konnten die jeweiligen Filterfunktionen nicht implementiert werden. Sollte der Benutzer jedoch trotzdem versuchen diesen Datensatz zu laden und eine gerätespezifische Ausgabe wollen, wird nichts in die Datenbank geschrieben und eine Fehlermeldung angezeigt (vgl. Abschnitt 3.3.2 und 3.3.3). Während der Programmierung des Parsers wurde festgestellt, dass die Daten für das Einlesen zu groß waren. Deshalb wurden die Daten gepuffert eingelesen.

Momentan können die vier ausgewählten Referenzdatensätze erfolgreich eingelesen, benutzerspezifisch gefiltert und in eine Datenbank gespeichert werden. Das Filtern passiert aktuell noch über die Konsole, was in Zukunft auch über eine grafische Oberfläche möglich sein wird. Des Weiteren können, durch die Verwendung eines Interfaces, einfach weitere Energiereferenzdatensätze eingebaut werden.

Literaturverzeichnis

[1] A. Einfalt *et al.*, „ADRES-Concept – Konzeptentwicklung für ADRES - Autonome Dezentrale Regenerative EnergieSysteme," TU Wien and Austiran Institute of Technology and Austian Power Grid, Tech. Rep., 2012.

[2] A. Monacchi *et al.*, „GREEND: An Energy Consumption Dataset of Households in Italy and Austria," in *2014 IEEE International Conference on Smart Grid Communications (SmartGridComm)*, Venice, Italy, November 2014, S. 511–516.

[3] J. Z. Kolter und M. J. Johnson, „REDD: A Public Data Set for Energy Disaggregation Research," Massachusetts Institute of Technology Cambridge, Tech. Rep., 2011.

[4] J. Kelly und W. Knottenbelt, „UK-DALE: A dataset recording UK Domestic Appliance-Level Electricity demand and whole-house demand," Department of Computing, Imperial College London, Tech. Rep., 2011.

[5] A. Azarpour *et al.*, „A Review on the Drawbacks of Renewable Energy as a Promising Energy Source of the Future," *Arabian Journal for Science and Engineering*, Vol. 38, no. 2, S. 317–328, 2012.

[6] K. Hänsch und L. Atienza Serna, „Directive 95/46/EC of the European Parliament and of the Council of 24 October 1995 on the protection of individuals with regard to the processing of personal data and on the free movement of such data," *Official Journal of the European Communities*, Vol. 281, S. 31–50, 1995.

[7] S. Barker *et al.*, „Smart*: An Open Data Set and Tools for Enabling Research in Sustainable Homes Sean," University of Massachusetts Amherst, Tech. Rep., 2012.

[8] E. J. Aladesanmi und K. A. Folly, „Overview of non-intrusive load monitoring and identification techniques," *IFAC-PapersOnLine*, Vol. 48, Nr. 30, S. 415–420, December 2015, Online erhältlich unter http://www.sciencedirect.com/science/article/pii/S2405896315030566; abgerufen am 29. November 2016.

[9] S. Barker *et al.*, „SmartCap: Flattening peak electricity demand in smart homes," in *2012 IEEE International Conference on Pervasive Computing and Communications*, Lugano, Switzerland, March 2012, S. 67–75.

[10] R. Paschotta, „RP-Energie-Lexikon," https://www.energie-lexikon.info/lastprofil.html (26.03.2016), 2013, Abgerufen am 29. November 2016.

[11] M. Hinterstocker *et al.*, „Bewertung der aktuellen Standardlastprofile österreichs und Analyse zunkünftiger Anpassungsmöglichkeiten im Strommarkt," in *13. Symposium Energieinnovation*, München, Deutschland, Februar 2014.

[12] Österreichische Energieagentur – Austrian Energy Agency, „EU-Energielabel: Handel erzielt EU-Durchschnitt, in einigen Kategorien großer Nachholbedarf," https://www.energyagency.at/aktuelles-presse/news/detail-archiv/artikel/handel-erzielt-eu-durchschnitt-in-einigen-kategorien-grosser-nachholbedarf.html, Abgerufen am 06. Februar 2017.

[13] J. Seryak und K. Kissock, „Occupancy and Behavioral Aspects on Residential Energy Use," *Proceedings of the Solar conference*, S. 717–722, 2003.

[14] A. Unterweger und D. Engel, „Lossless Compression of High-Frequency Voltage and Current Data in Smart Grids," Salzburg University of Applied Sciences, Josef Ressel Center for User-Centric Smart Grid Privacy, Security and Control, Tech. Rep., 2016.

A Anhang

GREENDDataHandler - InsertMeterDataIntoDb

GREENDDataHandler:

```
myDatabase.InsertMeterDataIntoDb(temp);
```

DatabaseDriver:

```
public void InsertMeterDataIntoDb(MeterData newData)
{
    this.ConnectToDB();
    try
    {
        String sqlCommand = @"INSERT INTO meter_data VALUES (
           @p1, @p2, @p3, @p4, @p5, @p6, @p7, @p8, @p9, @p10,
            @p11, @p12, @p13, @p14, @p15, @p16)";
        using (SqlCommand cmd = new SqlCommand(sqlCommand,
           connection))
        {
        String[] data = newData.GetData();
        cmd.Parameters.AddWithValue("@p1", Int32.Parse(data
           [0]));
        cmd.Parameters.AddWithValue("@p2", Int32.Parse(data
           [1]));
        cmd.Parameters.AddWithValue("@p3", this.
           UnixTimeStampToDateTime(Convert.ToDouble(data[2]))
           );
        cmd.Parameters.AddWithValue("@p4", Convert.ToDouble(
           data[3]));
        cmd.Parameters.AddWithValue("@p5", Convert.ToDouble(
           data[4]));
        cmd.Parameters.AddWithValue("@p6", Convert.ToDouble(
           data[5]));
        cmd.Parameters.AddWithValue("@p7", Convert.ToDouble(
           data[6]));
        cmd.Parameters.AddWithValue("@p8", Convert.ToDouble(
           data[7]));
        cmd.Parameters.AddWithValue("@p9", Convert.ToDouble(
           data[8]));
        cmd.Parameters.AddWithValue("@p10", Convert.ToDouble(
           data[9]));
```

```
20          cmd.Parameters.AddWithValue("@p11", Convert.ToDouble(
               data[10]));
21          cmd.Parameters.AddWithValue("@p12", Convert.ToDouble(
               data[11]));
22          cmd.Parameters.AddWithValue("@p13", Convert.ToDouble(
               data[12]));
23          cmd.Parameters.AddWithValue("@p14", Convert.ToDouble(
               data[13]));
24          cmd.Parameters.AddWithValue("@p15", Convert.ToDouble(
               data[14]));
25          cmd.Parameters.AddWithValue("@p16", Convert.ToDouble(
               data[15]));
26          cmd.ExecuteNonQuery();
27          }
28      }
29      catch (Exception e)
30      {
31          Console.WriteLine(e.Message);
32      }
33      this.CloseConnection();
34  }
```