Masterarbeit

im Studiengang Erneuerbare Energien der Fakultät für
Anlagen, Energie- und Maschinensysteme an der TH Köln

von

Fabian Zickel

Matrikel-Nr. 11141449

Prototypische Entwicklung einer Hard- & Softwareplattform zur prädiktiven Regelung und Anbindung an ein open-source Energiemanagementsystem

Prototypical development of a hardware & software platform for predictive control and connection to an open-source energy management system

Erstgutachter:

Prof. Dr.-Ing. Thorsten Schneiders

Zweitgutachter:

Dipl.-Ing. Michael Pauli

Technology
Arts Sciences
TH Köln

manage

EGM 27/21

Ausgabe: 07.10.2021

Abgabe: 17.02.2022

Kontakt: f.zickel@gmx.net

Prototypische Entwicklung einer Hard- & Softwareplattform zur prädiktiven Regelung und Anbindung an ein open-source Energiemanagementsystem

Prototypical development of a hardware & software platform for predictive control and connection to an open-source energy management system

Registrier-Nr. EGM27/21 von

Fabian Zickel

Fakultät für Anlagen, Energie- & Maschinensysteme

Abgabe: 17.02.2022

Erstgutachter: Prof. Dr.-Ing. Thorsten Schneiders

Zweitgutachter: Dipl.-Ing. Michael Pauli

Keywords: lückenlose Energiedatenmessung, prototypische Hard- und Softwareplattform, Energieeffizienzsteigerung, *Azure IoT*, modellprädiktive Regelung

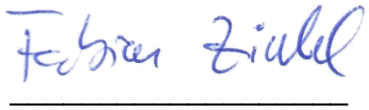
Kurzfassung

Diese Abschlussarbeit befasst sich mit der prototypischen Entwicklung einer Hard- und Softwareplattform, die als Basis für ein potenzielles Projekt der Firma *manageE* gedacht ist. Dieses Projekt soll durch den Einsatz einer modellprädiktiven Regelung Energieeffizienzen in Unternehmen steigern, indem auf Basis ermittelter Energie- und Betriebsdaten Prognosen erstellt werden. Die entwickelte Hard- und Softwareplattform soll dabei nicht nur die Schnittstelle zu den Anlagen und Steuerungen des Betriebs bilden, sondern auch die Rückgabe von Sollwerten an diese für deren Regelung realisieren. Eine Marktrecherche zeigt, dass aktuelle Datenlogger die hohen Anforderungen, die an die hier konzipierte Lösung gestellt werden, nicht erfüllen. Wird das in dem angedachten Projekt umzusetzende Gesamtsystem für die Energiedatenaufnahme und -analyse mit marktüblichen Systemen verglichen, zeigt sich, dass auch diese auf einem geringeren technischen Stand sind und weniger Funktionalitäten bieten. In einer ausführlichen Hardwarerecherche wurde ein geeigneter Computer definiert und ausgewählt. Unter Verwendung der Automatisierungssoftware *CODESYS*, *Microsoft Azure IoT* und *Python* wurden folgende Softwarekomponenten im Rahmen der Arbeit entwickelt: Ein Programm, das Messdaten von *Modbus-TCP-Slaves* abruft, an einen lokalen *SQL*-Server sendet und zusätzlich als *.csv*-Dateien abspeichert, und ein *Python*-Skript, das diese Dateien in *Blockblobs* umwandelt und in einen *Azure IoT Hub* hochlädt. Eine abschließende Validierung der Arbeitsergebnisse mithilfe eines Versuchsaufbaus, der die Datenaufnahme einer Energiesenke und deren Regelung simulierte, war erfolgreich.

I EIGENSTÄNDIGKEITSERKLÄRUNG

Hiermit erkläre ich, dass ich die vorliegende Arbeit eigenständig und ohne fremde Hilfe angefertigt habe. Textpassagen, die wörtlich oder dem Sinn nach auf Publikationen oder Vorträgen anderer Autoren beruhen, sind als solche kenntlich gemacht. Die Arbeit wurde bisher keiner anderen Prüfungsbehörde vorgelegt und auch noch nicht veröffentlicht.

Köln, 17.02.2022



Fabian Zickel (Unterschrift)

II KURZFASSUNG

Diese Abschlussarbeit befasst sich mit der prototypischen Entwicklung einer Hard- und Softwareplattform, die als Basis für ein potenzielles Projekt der Firma *manageE* gedacht ist. Dieses Projekt soll durch den Einsatz einer modellprädiktiven Regelung Energieeffizienzen in Unternehmen steigern, indem auf Basis ermittelter Energie- und Betriebsdaten Prognosen erstellt werden. Die entwickelte Hard- und Softwareplattform soll dabei nicht nur die Schnittstelle zu den Anlagen und Steuerungen des Betriebs bilden, sondern auch die Rückgabe von Sollwerten an diese für deren Regelung realisieren. Eine Marktrecherche zeigt, dass aktuelle Datenlogger die hohen Anforderungen, die an die hier konzipierte Lösung gestellt werden, nicht erfüllen. Wird das in dem angedachten Projekt umzusetzende Gesamtsystem für die Energiedatenaufnahme und -analyse mit marktüblichen Systemen verglichen, zeigt sich, dass auch diese auf einem geringeren technischen Stand sind und weniger Funktionalitäten bieten. In einer ausführlichen Hardwarerecherche wurde ein geeigneter Computer definiert und ausgewählt. Unter Verwendung der Automatisierungssoftware *CODESYS*, *Microsoft Azure IoT* und *Python* wurden folgende Softwarekomponenten im Rahmen der Arbeit entwickelt: Ein Programm, das Messdaten von *Modbus-TCP-Slaves* abruft, an einen lokalen *SQL*-Server sendet und zusätzlich als *.csv*-Dateien abspeichert, und ein *Python*-Skript, das diese Dateien in *Blockblobs* umwandelt und in einen *Azure IoT Hub* hochlädt. Eine abschließende Validierung der Arbeitsergebnisse mithilfe eines Versuchsaufbaus, der die Datenaufnahme einer Energiesenke und deren Regelung simulierte, war erfolgreich.

Schlüsselwörter: lückenlose Energiedatenmessung, prototypische Hard- und Softwareplattform, Energieeffizienzsteigerung, *Azure IoT*, modellprädiktive Regelung

III ABSTRACT

This thesis deals with the prototypical development of a hardware and software platform, which is intended as a basis for a potential project of the company *manageE*. The project's goal is to increase energy efficiency in companies by using a model-predictive control system that generates forecasts based on gathered energy and operating data. The developed hardware and software platform shall not only serve as an interface to the company's plants and controls, but also implement the return of setpoints to control them. Market research shows that current data loggers do not meet the high requirements that are placed on the solution designed here. When comparing the presented overall system for energy data recording and analysis to commercially available systems, it becomes apparent that these are also at a lower technical level and offer fewer functionalities. In a detailed hardware research, a suitable computer was defined and selected. Using *CODESYS* automation software, *Microsoft Azure IoT* and *Python*, the following software components were developed: A program that retrieves measurement data from *Modbus-TCP slaves*, sends it to a local *SQL* server and additionally stores it in *.csv* files, and a *Python* script that converts these files into block blobs and uploads them to an *Azure IoT Hub*. A final validation of the work results using an experimental setup that simulated the data collection of an energy sink and its control was successful.

Keywords: consistent energy data measurement, prototypical hardware and software platform, enhancement of energy efficiency, *Azure IoT*, model predictive control

IV INHALTSVERZEICHNIS

I EIGENSTÄNDIGKEITSERKLÄRUNG II

II KURZFASSUNG III

III ABSTRACT IV

IV INHALTSVERZEICHNIS V

V ABBILDUNGSVERZEICHNIS IX

VI TABELLENVERZEICHNIS XI

VII ABKÜRZUNGSVERZEICHNIS XII

1 EINLEITUNG 1

1.1 KAPITELÜBERSICHT 1

1.2 HINTERGRUND UND MOTIVATION 2

1.3 PROBLEMSTELLUNG 4

1.4 ZIELSETZUNG 5

1.5 METHODIK 7

1.5.1 Methodik zur Marktrecherche 7

1.5.2 Methodik zur Auswahl der Hardware-Plattform 8

1.5.3 Methodik zur Programmierung der Datenaufnahme 8

1.5.4 Methodik zur Implementierung der Datenspeicherung 9

2 THEMATISCHE GRUNDLAGEN 10

2.1 ENERGIEEFFIZIENZ 10

2.1.1 Begriffsdefinition 10

2.1.2 Unterscheidung der Energieeffizienz 11

2.1.3 Energiemanagement und -monitoring 11

2.1.4 Energiedatenanalyse und prädiktive Regelung 12

2.2 ENERGIEEFFIZIENZ IN UNTERNEHMEN 13

2.2.1 Energieverbrauch deutscher Unternehmen 13

2.2.2 Energieflüsse eines Unternehmens 14

2.2.3 Motivation für Energiemanagement in Unternehmen 15

2.2.4 Energiemanagement nach ISO 50001 16

2.2.5 Verwendung von Gebäudeautomation für Energiemanagement 18

2.2.6 Einsparpotenziale in Unternehmen 19

3 GRUNDLAGEN ZUR HARD- UND SOFTWAREPROGRAMMIERUNG 24

3.1 BEGRIFFSDEFINITIONEN: HARDWARE, SOFTWARE UND PLATTFORM 24

3.2 GRUNDLAGEN ZUR SOFTWAREPROGRAMMIERUNG 25

3.2.1 Phasen der Softwareentwicklung 25

3.2.2 Programmablaufpläne als Visualisierungshilfe 27

3.2.3 Verwendung von Bibliotheken 28

3.3 CODESYS 29

3.3.1 Vorstellung von CODESYS 29

3.3.2 Verwendung der Programmiersprachen in CODESYS 29
3.3.3 Meldungen in CODESYS 31
3.3.4 Visualisierungen in CODESYS 31
3.4 MODBUS-TCP-PROTOKOLL 32
4 ABGRENZUNG ZU AKTUELLEN UND BISHERIGEN REFERENZPROJEKTEN 34
4.1 OPTICONTROL-II 34
4.2 BIG 35
4.3 SIOSTA 37
4.4 SMART-METER-ROLLOUT 38
5 MARKTRECHERCHE 41
5.1 ANFORDERUNGEN AN DIE HARD- UND SOFTWAREPLATTFORM 41
5.1.1 Beurteilungskriterien für reine Datenlogger 41
5.1.2 Beurteilungskriterien für Komplettsysteme zur Datenerfassung und -verarbeitung 43
5.2 VORSTELLUNG MARKTÜBLICHER DATENLOGGER 43
5.2.1 7KN POWERCENTER 3000 V1 (Siemens AG) 44
5.2.2 BDL 50 (Berg GmbH) 45
5.2.3 Controller PFC200 (WAGO GmbH & Co. KG) 45
5.2.4 EasyNet (4next Technology Systems) 46
5.2.5 Expert Logger (Delphin Technology AG) 46
5.2.6 gridBox (gridX GmbH) 47
5.2.7 HOBO RX3000 (Onset Computer Corporation) 47
5.2.8 MEMORY HiLOGGER LR8450 (HIOKI E.E. Corporation) 47
5.2.9 Tabellarischer Abgleich marktüblicher Datenlogger mit den Bewertungskriterien 48
5.3 VORSTELLUNG MARKTÜBLICHER SYSTEME FÜR DIE ENERGIEDATENERFASSUNG UND -ANALYSE 50
5.3.1 System der Beckhoff Automation GmbH & Co. KG 50
5.3.2 System der Berg GmbH 51
5.3.3 System der Delphin Technology AG 51
5.3.4 System der Eaton Corporation 51
5.3.5 System der OPTENDA GmbH 52
5.3.6 System der Robert Bosch GmbH 52
5.3.7 SENTRON-Reihe der Siemens AG 53
5.3.8 Zusammenfassung zur Recherche marktüblicher Systeme für die Energiedatenerfassung und -analyse 53
6 DEFINITION UND ERSTELLUNG EINER HARDWARE-PLATTFORM 54
6.1 ANFORDERUNGEN AN DIE HARDWARE 54
6.2 AUSWAHL GEEIGNETER HARDWARE 55
6.2.1 Bewertung der Leistungsfähigkeit 56

6.2.2 Entscheidung für ein Hardware-Modell ... 57
6.3 VORBEREITUNG DER HARDWARE ... 58
6.3.1 Installation des Linux-Betriebssystems ... 59
6.3.2 Einrichtung des Betriebssystems ... 59
6.4 ZWISCHENFAZIT: DEFINITION UND ERSTELLUNG EINER HARDWARE-PLATTFORM ... 60
7 SOFTWAREENTWICKLUNG FÜR DIE DATENAUFNAHME UND -SPEICHERUNG 61
7.1 KLÄRUNG DER VORGABEN ... 61
7.1.1 Ziele der Software-Lösung ... 61
7.1.2 Beschreibung der Anforderungen ... 62
7.2 AUSWAHL EINER IOT-PLATTFORM ... 63
7.2.1 Google Cloud IoT ... 63
7.2.2 Cisco IoT Cloud Connect ... 63
7.2.3 Salesforce IoT Cloud ... 64
7.2.4 Amazon AWS IoT Core ... 64
7.2.5 Microsoft Azure IoT Hub ... 64
7.3 AUFBAU UND PROGRAMMIERUNG DER SOFTWARE ... 65
7.3.1 Programmierung der Datenaufnahme ... 66
7.3.2 Programmierung der lokalen Datenspeicherung ... 68
7.3.3 Programmierung der Datenweiterleitung auf die IoT-Plattform ... 74
7.3.4 Umsetzungsvorschlag: Rückversand von Sollwerten ... 88
7.4 ZWISCHENFAZIT: SOFTWAREENTWICKLUNG FÜR DIE DATENAUFNAHME UND -SPEICHERUNG ... 89
8 VALIDIERUNG DER ARBEITSERGEBNISSE ... 91
8.1 KRITERIEN FÜR TESTS ... 91
8.2 AUFBAU UND VORBEREITUNG DER VALIDIERUNG ... 92
8.3 VERSUCHSDURCHFÜHRUNG ... 94
8.4 FAZIT DER VALIDIERUNG DER ARBEITSERGEBNISSE ... 97
9 VERGLEICH DER ARBEITSERGEBNISSE MIT MARKTÜBLICHEN LÖSUNGEN ... 98
9.1 ZUSAMMENFASSUNG DER ARBEITSERGEBNISSE ... 98
9.2 FESTLEGUNG DER KRITERIEN FÜR DEN VERGLEICH DER ARBEITSERGEBNISSE MIT MARKTÜBLICHEN LÖSUNGEN ... 99
9.3 VERGLEICHENDE DARSTELLUNG DER ARBEITSERGEBNISSE MIT MARKTÜBLICHEN LÖSUNGEN ... 100
9.4 FAZIT DES VERGLEICHS ... 101
10 HANDLUNGSEMPFEHLUNGEN ... 102
10.1 MÖGLICHE SCHWIERIGKEITEN BEI DER EINBINDUNG DER HSP IN DIE STRUKTUR EINES UNTERNEHMENS ... 102
10.2 ANBINDUNG DER HSP AN CLOUD-ANWENDUNG ... 103
10.3 ERWEITERUNG UND VERBESSERUNG DES FUNKTIONSUMFANGS ... 104
10.3.1 Ermittlung und Darstellung von Systemkennzahlen ... 104

10.3.2 Einführung einer Ausnahmebehandlung 104
10.3.3 Implementierung eines Betriebsstatus-Loggers 105
10.3.4 Einrichtung einer SAS-Authentifizierung 105
10.3.5 Berücksichtigung der Zeitumstellung 105
11 DISKUSSION 106
12 FAZIT 108
13 AUSBLICK 110
VIII LITERATURVERZEICHNIS XIV
IX ANHANG XXIV
ANHANG A : GEDÄCHTNISPROTOKOLL ZU PERSÖNLICHEM GESPRÄCH MIT THOMAS VOLLER BEZÜGLICH DER BIG-PROJEKTERGEBNISSE XXIV
ANHANG B : VOLLSTÄNDIGE LISTE MARKTÜBLICHER DATENLOGGER (AUFSTEIGEND NACH ALPHABET) XXVI
ANHANG C : VOLLSTÄNDIGE LISTE MARKTÜBLICHER SYSTEME FÜR DIE ENERGIEDATENERFASSUNG UND -ANALYSE XXIX
ANHANG D : VOLLSTÄNDIGE AUFLISTUNG DES BENCHMARKINGS DER VERSCHIEDENEN PROZESSOREN (AUFSTEIGEND SORTIERT NACH LEISTUNG) XXXII
ANHANG E : E-MAIL VON ALFRED TEKNIEPE ZUM THEMA „VORTEILE ALS MICROSOFT-SILBER-PARTNER“ XXXIV
ANHANG F : LÖSUNGSANSATZ 2 - C#-CODE ZUR VERARBEITUNG DER NACHRICHTEN AUS DEM MODBUS-MODUL FÜR DAS MESSAGE ROUTING XXXV
ANHANG G : LÖSUNGSANSATZ 3 - CODESYS-SKRIPT FÜR DAS MINÜTLICHE ABSPEICHERN DER MESSDATEN IN EINER .CSV-DATEI XXXVII
ANHANG H : LÖSUNGSANSATZ 3 - C#-CODE ZUR VERARBEITUNG VON .CSV-DATEIEN IN BLOCKBLOBS XLI
ANHANG I : LÖSUNGSANSATZ 4 - PYTHON-CODE ZUR VERARBEITUNG VON .CSV-DATEIEN IN BLOCKBLOBS XLIII

V ABBILDUNGSVERZEICHNIS

Abbildung 1 - Konzept des potenziellen Projektes 5

Abbildung 2 - Links: Schema "Stand Alone" Lösung; Rechts: Schema verknüpfter Anlagen durch IoT 7

Abbildung 3 - Endenergieverbrauch Deutschlands in Petajoule nach Sektor im Jahr 2020 14

Abbildung 4 - Beispielhafte Darstellung der Energieflüsse eines fiktiven Unternehmens 15

Abbildung 5 - Kontinuierlicher Verbesserungsprozess im *PDCA-Zyklus* nach *DIN ISO 50001* 17

Abbildung 6 - Phasen der Softwareentwicklung 25

Abbildung 7 - Beispiele für Programmablaufplan-Symbole nach *DIN 66001* 28

Abbildung 8 - Aufbau der Programmieroberfläche am Beispiel von "Strukturierter Text" 30

Abbildung 9 - Beispiel für die Verwendung eines komplexen Funktionsbausteins 30

Abbildung 10 - Darstellung von Programmcode mit mehreren Funktionsbausteinen 31

Abbildung 11 - Beispiel einer Veranschaulichung im Visualisierungseditor von *CODESYS* 32

Abbildung 12 - Schematische Darstellung des Datenflusses vom Messgerät über die HSP bis in die IoT-Plattform und Rückfluss über die potenzielle Cloud-Plattform 65

Abbildung 13 - Ausschnitt aus der Eingabemaske für die Offsets als Hexadezimalzahl und Anzahl der abzufragenden Register ("Länge") 67

Abbildung 14 - Beispiel für das Mapping von Variablen mit Registern 68

Abbildung 15 - Erster Programmteil der Auftrennung des DZS für die Umformatierung 71

Abbildung 16 - Verwendung des Funktionsbausteins *splitDTValue* und Neuformatierung des DZS über *CONCAT*-Operatoren 72

Abbildung 17 - Zusammensetzung der Messdaten und des Zeitstempels zu einem *SQL*-Befehl 73

Abbildung 18 - Blink-Funktionsbaustein für den Versand des *SQL*-Befehls 73

Abbildung 19 - Funktionsbaustein zum Versand des *SQL*-Befehls 74

Abbildung 20 - Einstellung des Message Routings im *Azure* Portal 76

Abbildung 21 - Einstellung des *Modbus*-Moduls im *Azure* Portal 76

Abbildung 22 - Konfiguration des Modul-Zwillings für das *Blockblob*-Modul im *Azure-Portal*..... 78

Abbildung 23 - Schematische Darstellung des Datenflusses vom Messgerät über die HSP bis in den *IoT Hub* für Lösungsansatz 2 79

Abbildung 24 - Schematische Darstellung des Datenflusses vom Messgerät über die HSP bis in den *IoT Hub* für Lösungsansatz 3 80

Abbildung 25 - Ausschnitt aus dem *CODESYS*-Programm zur Erstellung einer *.csv*-Datei - Hier: Befüllung des Arrays mit erstem Messwert (Spannung)..... 81

Abbildung 26 - Ausschnitt aus dem *CODESYS*-Programm zur Erstellung einer *.csv*-Datei – Hier: Erster Teil der *CASE*-Unterscheidung..... 82

Abbildung 27 - Programmablaufplan für das *C#*-Skript aus Lösungsansatz 3 84

Abbildung 28 - Ausschnitt aus dem *Python*-Programm für die Verarbeitung der *.csv*-Dateien zu *Blockblobs* – Hier: Import der Bibliotheken 86

Abbildung 29 - Programmablaufplan für das *Python*-Skript zur Verarbeitung der *.csv*-Dateien zu *Blockblobs* 87

Abbildung 30 - Zwischenstück zur Separierung der Phase des Heizlüfter-Kabels 92

Abbildung 31 - Schematische Darstellung des Versuchsaufbaus..... 93

Abbildung 32 - Realer Aufbau des Validierungsversuchs..... 94

Abbildung 33 - Ausschnitt der während des Tests aufgenommenen Daten abgefragt von dem lokalen *SQL*-Server 95

Abbildung 34 - Beispiel für während des Tests hochgeladene *Blockblobs* im *Azure IoT Hub* 96

Abbildung 35 - Beispiel für im Versuch in einem *Blockblob* gespeicherte Werte..... 96

VI TABELLENVERZEICHNIS

Tabelle 1 - Mögliche Probleme typischer Komponenten von strombasierten Basistechnologien in Unternehmen 20

Tabelle 2 - Mögliche Probleme typischer Komponenten von brennstoffbasierten Basistechnologien in Unternehmen 21

Tabelle 3 - Mögliche Probleme typischer Komponenten von strombasierten Querschnittsprozessen in Unternehmen 22

Tabelle 4 - Mögliche Probleme typischer Komponenten von brennstoffbasierten Querschnittsprozessen in Unternehmen 22

Tabelle 5 - Mögliche Probleme typischer Komponenten von gekoppelten und sonstigen Querschnittsprozessen in Unternehmen 23

Tabelle 6 - Auflistung bisher zertifizierter Unternehmen und Produkte 39

Tabelle 7 - Größe des internen Speichers und Anzahl der speicherbaren Messwerte marktüblicher Datenlogger 44

Tabelle 8 - Übersicht der vorgestellten Datenlogger und Vergleich mit Bewertungskriterien 49

Tabelle 9 - „Muss“-Anforderungen an die Hardware-Plattform 55

Tabelle 10 - „Kann“-Anforderungen an die Hardware-Plattform 55

Tabelle 11 - Auflistung der Top-3-Prozessoren (Top 10 %) des Benchmarkings (aufsteigend sortiert nach Leistung) 57

Tabelle 12 - Passende Hardware-Modelle in der Endauswahl 58

Tabelle 13 - Auflistung der Ziele und Anforderungen der zu erstellenden Software-Lösung 62

Tabelle 14 - Auflistung von Softwarekomponenten, die für die Implementierung des *SQL*-Servers vorausgesetzt werden 69

Tabelle 15 - Unterscheidung des DZS-Formats in *CODESYS* und für *SQL*-Server 71

Tabelle 17 - Vergleich der Arbeitsergebnisse mit marktüblichen Lösungen (sortiert aufsteigend nach Punktzahl) 100

VII ABKÜRZUNGSVERZEICHNIS

AIoTEL	*Azure IoT Edge*-Laufzeitumgebung
BHKW	Blockheizkraftwerk
BMWi	Bundesministerium für Wirtschaft und Energie
BSI	Bundesamt für Sicherheit in der Informationstechnik
CPU	Central Processing Unit (Englisch für “Zentrale Prozessoreinheit“)
DZS	Datums- und Zeitstempel
EMS	Energiemanagementsysteme
GB	Gigabyte
GHz	Gigahertz
HSP	Hard- und Software-Plattform
HVAC	Heating, Ventilation and Air Conditioning
IoT	Internet of Things
IP	Internetprotokoll
PV	Photovoltaik
SMGW	Smart-Meter-Gateway
SQL	Structured Query Language

1 EINLEITUNG

Das folgende Kapitel dient der Einführung in die Thematik dieser Masterarbeit mit dem Titel „Prototypische Entwicklung einer Hard- und Softwareplattform zur prädiktiven Regelung und Anbindung an ein open-source Energiemanagementsystem“. Zu diesem Zweck erfolgt übersichtshalber zunächst eine kurze Darstellung der einzelnen Kapitelinhalte gefolgt von der Beschreibung des Hintergrundes und der Motivation der Abschlussarbeit. Im Anschluss daran wird erst auf die Problemstellung und dann auf die Zielsetzung eingegangen. Den Abschluss des Kapitels bildet die Beschreibung der Vorgehensweise bei der Bearbeitung der einzelnen Arbeitsteile.

1.1 Kapitelübersicht

Nach der Einleitung in den Themenbereich und die Hintergründe dieser Arbeit folgt das Kapitel der *Thematischen Grundlagen*, in dem die Begriffe „Energieeffizienz“, „Energiemanagement und -monitoring“ und „prädiktive Regelung“ definiert werden. Im Anschluss werden die Energieeffizienz und das Energiemanagement in Unternehmen näher betrachtet und mögliche Motivationsfaktoren und typische Einsparpotenziale vorgestellt.

Im dritten Kapitel werden die Begriffe „Hardware“ und „Software“ voneinander abgegrenzt und die Grundlagen der Softwareprogrammierung werden betrachtet. Hier wird unter anderem auf die verschiedenen Phasen der Softwareprogrammierung, die Visualisierungsmöglichkeit von Programmcode und die Entwicklungsumgebung *CODESYS* eingegangen. Den Abschluss der Grundlagen bilden Hintergründe und Informationen zu dem verwendeten *Modbus-TCP*-Protokoll.

Nachfolgend findet die Abgrenzung zu aktuellen und bisherigen Referenzprojekten statt. Hier werden nicht nur drei Forschungsprojekte, sondern auch der sogenannte „Smart-Meter-Rollout“ vorgestellt.

Mittelpunkt des fünften Kapitels ist eine Marktrecherche, für die zunächst die Beurteilungskriterien definiert werden. Im Anschluss daran werden verschiedene marktübliche Produkte betrachtet, die eine vergleichbare Aufgabe wie die zu entwickelnde Lösung erfüllen.

Der nächste Abschnitt befasst sich mit der Definition und Erstellung einer Hardware-Plattform, für die in einem erstem Schritt Anforderungen definiert werden. Die Kriterien zur Auswahl einer geeigneten Hardware werden ebenso vorgestellt wie die letztendliche Entscheidung und die Vorbereitung des Systems für die Umsetzung dieser Arbeit.

Kapitel 7 ist den Softwarekomponenten gewidmet, die im Rahmen der Abschlussarbeit genutzt oder entwickelt wurden. Zu diesem Zweck werden zunächst die Vorgaben geklärt und aktuelle Lösungen für „Internet of Things“-Plattformen vorgestellt. Der Großteil des Kapitels befasst sich dann mit dem Aufbau und der Programmierung der Softwarekomponenten.

Aufbauend auf den zwei vorangegangenen Teilen steht in Kapitel 8 die Validierung der Arbeitsergebnisse im Vordergrund, die im Rahmen eines Versuchs erfolgt. Neben der Definition von Kriterien zur Bewertung des Versuchs werden dessen Aufbau, Vorbereitung, Durchführung und Ergebnisse vorgestellt.

Gegenstand des nachfolgenden Abschnitts ist der Vergleich der Arbeitsergebnisse mit marktüblichen Lösungen. Als Basis hierfür dienen eine Zusammenfassung der entwickelten Lösungen und eine Beschreibung der Vergleichskriterien. Es folgt ein tabellarischer Abgleich inklusive Bewertung, der zusammen mit einem Fazit des Vergleichs das Kapitel abschließt.

Auf Basis der Entwicklungen werden in Kapitel 10 nicht nur mögliche Schwierigkeiten beim Einsatz in einem realen Produktivsystem aufgedeckt, sondern auch Handlungsempfehlungen ausgesprochen. Diese betreffen vor allem die Erweiterung und Verbesserung des Funktionsumfanges und sollen zum Ausschluss möglicher Probleme während des Betriebs dienen.

Nachdem die Vorgehensweise bei der Umsetzung dieser Arbeit im darauffolgenden Abschnitt diskutiert worden ist, folgt ein Kapitel mit einem Fazit, das die wesentlichen Punkte und Schlussfolgerungen noch einmal zusammenfasst. Der letzte Teil gibt abschließend einen Ausblick auf künftige Entwicklungen in dem hier thematisierten Bereich.

1.2 Hintergrund und Motivation

Die Energiewende, die in Deutschland vor allem 2011 durch die Reaktorkatastrophe in Fukushima angestoßen wurde, enthält neben dem zügigen und umfassenden Ausstieg aus der Kernenergie [1] drei weitere wesentliche Änderungen:

- Ausbau der Erneuerbaren Energien
- Ausbau der Übertragungsnetze
- Regelungen zur Energieeffizienz. [2]

Besonders im Bereich der Energieeffizienz finden aktuell sowohl im privaten als auch gewerblichen Bereich stetige Entwicklungen statt. So wird zum Beispiel erfolgreich mit intelligenten Technologien wie Smart-Home-Systemen, Smart-Metering und Smart Grids der Verbrauch analysiert und durch Anpassung und Regelung der Gesamtenergieverbrauch verringert.

Die Analyse von großen Datenmengen (Stichwort: Big Data) mithilfe *künstlicher Intelligenz* ist bereits heute auf der ganzen Welt ein wichtiges Thema. Durch die rasante Digitalisierung der letzten Jahre und die daraus resultierende Entwicklung neuer digitaler Systeme, Produkte und Dienstleistungen sammeln sich mit jeder Sekunde neue Informationen an. Diese müssen aufgenommen, verarbeitet und analysiert werden, um im Anschluss Anpassungen vorzunehmen und Vorhersagen z.B. über Nutzerverhalten, Bedarfe und Verbräuche treffen zu können. Dies betrifft vor allem alltägliche Bereiche, wie den stationären oder elektronischen Handel (z.B. Einkaufsverhalten, Marketing), die Bewertung der Kreditwürdigkeit, die Beeinflussung politischer Wahlen und die Energiebranche.

Da konventionelle Methoden bei der Verarbeitung und Analyse dieser Datenmassen ihr Limit erreichen, kommen hier vor allem selbstlernende Algorithmen zum Einsatz. [3] Erst dadurch ergibt sich die Möglichkeit die riesigen Datenmengen bewältigen zu können, die z.B. bei Messungen von Stromverbräuchen in sekündlicher Auflösung entstehen.

Auf der Seite des Unternehmens *manageE* ist der Hintergrund dieser Masterarbeit ein potenzielles Projekt zur prädiktiven Regelung, welches gegebenenfalls zusammen mit Projektpartnern umgesetzt werden soll.

Dieses Projekt soll auf den Markt „Nichtwohngebäude und Industrieunternehmen“ abzielen. Als Nichtwohngebäude werden hier insbesondere Büro-, Verwaltungs-, Fabrik- und Werkstattgebäude aufgefasst, in denen diverse Systeme zur Energieerzeugung (z.B. Photovoltaik, Biomasse-Verbrennung und Wärmepumpen), Energienutzung und Energiespeicherung (z.B. Batterie-, Eis-, H2-Speicher und virtuelle Stromnetze) eingesetzt werden. Diese Technologien sind meist autark installiert. Durch intelligente Vernetzung mithilfe des Internet of Things (IoT) und der Möglichkeit einer prädiktiven Regelung sollen die einzelnen Systeme dann aufeinander abgestimmt werden, um so ein zusätzliches Potential an Energieeinsparung zu gewinnen und damit den Bedarf an fossilen Energieträgern und CO2-Emissionen zusätzlich zu reduzieren. Für diese intelligente Vernetzung ist eine übergeordnete Hardware erforderlich.

Die prädiktive Regelung, die auf Echtzeitoptimierungen basiert, soll alle verfügbaren Sensordaten über den aktuellen Systemzustand berücksichtigen und

Prognosen über zu erwartende Verbräuche in die Energieerzeugung und -verteilung mit einbeziehen.

Zu den möglichen Aufgaben von *manageE* könnten dabei die Entwicklung einer prototypischen Hardwareplattform zur Anbindung der prädiktiven Regelung in Nichtwohngebäuden und die Installation und technische Betreuung der Hardware für die Evaluation des Prototyps in den Demonstrationsgebäuden gehören. Weitere Aufgaben, die u.a. von den möglichen Projektpartnern übernommen werden könnten, lauten wie folgt:

- Entwicklung einer Cloud-Anwendung
- Erstellung modularer Bilanzmodelle der Systemkomponenten
- Entwicklung eines Regelungskonzeptes
- Erarbeitung von Methoden zur praktischen Nutzbarmachung des Konzepts
- Betreuung, Durchführung und Evaluation des Feldtestes.

Dabei ergibt sich die Möglichkeit den Erkenntnisrückfluss aus den Demonstrationsgebäuden zu nutzen, um Forschungsschwerpunkte zur weiteren Steigerung der Energieeffizienz im industriellen Umfeld zu identifizieren. Diese Erkenntnisse werden dabei helfen, weiterführende Forschungsprojekte zum Thema der modellbasierten Regelung in Nichtwohngebäuden und in industriellen Anlagen zu akquirieren.

Das Projekt könnte für *manageE* von Nutzen sein, da das Potenzial besteht, die entwickelte Lösung bei den vorhandenen Kunden des Unternehmens zu vermarkten. Dadurch wäre es trotz der im Anschluss an das Projekt notwendigen Weiterentwicklungen zur Marktreife möglich, zeitnah Erfolge verzeichnen zu können. Die existierenden Vertriebsstrukturen sollten über den gemeinsamen Vertrieb unter anderem mit den Projektpartnern, durch Nutzung der Kontakte zu allen bestehenden Kunden sowie ein Online-Marketing zur Gewinnung von Neukunden erweitert werden.

1.3 Problemstellung

Bei Betrachtung des oben beschriebenen Hintergrundes tritt unter anderem die vernetzte Datenaufnahme in den Vordergrund. Es gibt zwar theoretisch eine große Menge an Energiedaten, die aufgenommen werden könnten, aber bisher nur begrenzte Möglichkeiten und vereinzelte Ansätze, dies umzusetzen. Häufig werden entweder nur elektrische oder thermische Energiedaten aufgenommen und auch die Berücksichtigung und Aufnahme von Umgebungs- und Prozessdaten erfolgen meist nicht. Hier besteht demnach noch viel Entwicklungsbedarf und -potenzial.

Jede Anlage verfügt über eine spezifische Sensorik, die für den Betrieb relevante Informationen aufnimmt. Diese Messdaten werden in verschiedenen Formaten abgerufen und abgespeichert, sodass eine Zusammenführung der Messdaten (z.B. Energieerzeugung und -verbrauch, Zustands-, Temperatur- und Prozessdaten) eines gesamten Unternehmens eine große Herausforderung darstellt.

1.4 Zielsetzung

Wie in

Abbildung 1 zu sehen, wird für die Erfassung der Gesamtenergiedaten eine Hardware benötigt, die mit den vielfältigen Steuerungen von möglichen Energiequellen und -senken kommunizieren kann. Die Entwicklung der diesem Zweck dienlichen prototypischen Hard- und Software-Plattform (HSP) ist deshalb erklärtes Hauptziel dieser Masterarbeit.

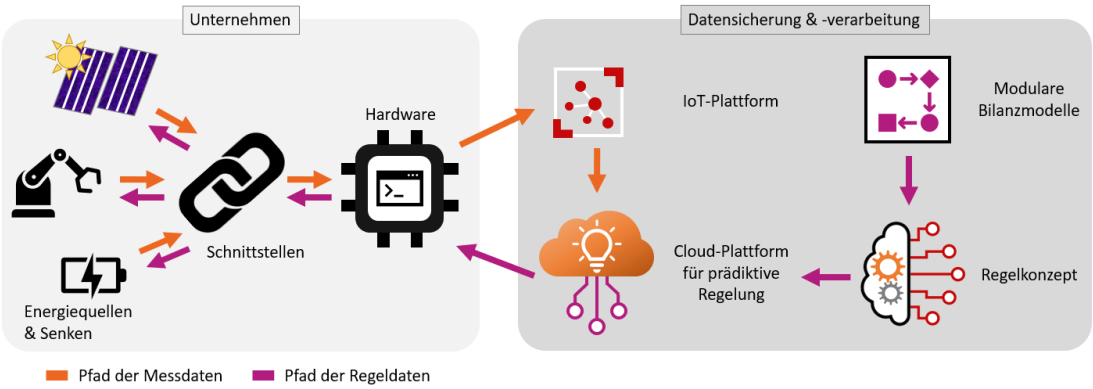


Abbildung 1 - Konzept des potenziellen Projektes

Die Hardware-Plattform soll mithilfe gängiger Kommunikationsschnittstellen von Automatisierungssystemen in der Lage sein, direkt mit der Sensorik, mit energieumwandelnden Systemen oder mit einem übergeordneten Gebäude-Automatisierungs-System zu kommunizieren. Als Basis soll ein industrietauglicher Mini-PC dienen, der aufgrund seiner kompakten Bauweise problemlos eingebunden und remote, d.h. ohne größere Peripherie wie Bildschirme und Bedienelemente, betrieben und gesteuert werden kann. Als Schnittstelle zwischen der Hardware und den Energiequellen und -senken des Unternehmens kommen intelligente Ein- und Ausgabe-Einheiten zum Einsatz, die seitens der Mini-PC Hardware via *Ethernet* angesprochen werden und sowohl Ein- als auch Ausgaben verarbeiten können. Diese Ein- und Ausgabeeinheiten werden von *manageE*-Mitarbeitern ausgewählt, programmiert und im Unternehmen installiert.

Neben der Auswahl einer für die Industrie geeigneten Hardware-Plattform muss auch die Erstellung und Programmierung einer Software erfolgen. Dies soll unter anderem mithilfe der Entwicklungsumgebung *CODESYS* geschehen. Mit dieser Umgebung kann das zu erstellende Programm flexibel und in jeder physikalischen beziehungsweise elektrischen Form die energetischen Messdaten aufnehmen und an eine zu definierende, externe Datenbank senden. Vor allem diese hohe Flexibilität ist für die Einbindung in Bestandsgebäuden unbedingt erforderlich. Da die Daten auch lokal auf der Hardware gespeichert werden sollten, besteht die Möglichkeit einer lokalen Visualisierung, die jedoch nicht Bestandteil der Masterarbeit ist.

Sobald die prototypische HSP entwickelt wurde, kann die Anbindung des Systems an eine noch zu erstellende Cloud-Plattform erfolgen, die neben der Aufbereitung und Analyse der Messdaten rechentechnisch komplexe Optimierungen durchführt. Zu diesem Zweck wird ein Regelkonzept benötigt, das auf modularen Bilanzmodellen beruht und in die Erstellung und Verbesserung von Lastprognosen integriert wird. Im Anschluss ist es dann möglich, dass die im Gebäude installierte Hardwarelösung die Übermittlung optimierter Soll- oder Stellwerte an die Komponentensteuerungen übernimmt. Allerdings schließt die hier vorliegende Abschlussarbeit die beiden Prozesse der Implementierung einer Cloud-Plattform zur Berechnung von Prognosen und die Erstellung des dahinterliegenden Regelungskonzeptes nicht mehr mit ein. Dafür wird erarbeitet, wie mithilfe von IoT-Technologie sichergestellt werden kann, dass alle Daten einer Vor-Ort Aufnahme immer vollständig in einer Cloud-basierten Datenbank ankommen. Vollständige Daten sind dringend notwendig, damit optimale Voraussagen für die Regelung getroffen werden können.

Die bereits erwähnten Soll- und Stellwerte sollen an die Komponentensteuerung zurückgegeben und von dort an vorhandene Systeme als externe Sollwerte vorgegeben werden. Die physikalische Anbindung erfolgt immer über Zweidraht- beziehungsweise Vierdrahtleitungen oder *Ethernet*-Anschlüsse, welche mit den Möglichkeiten der *CODESYS*-Software angebunden werden können.

Die zu erwartende Energieeinsparung erfolgt z.B. über die indirekte Anpassung des Betriebspunktes bei Heizungs- und Lüftungsanlagen oder auch Erzeugern wie Blockheizkraftwerken (BHKWs) aufgrund von prognostizierten Wetterdaten. Am höchsten wird jedoch der energetische Vorteil sein, weil mit einem solchen System Lastverschiebungen zwischen den Verbrauchern und Erzeugern optimal gesteuert werden können. Dies ist in sogenannten „Stand Alone“-Lösungen nicht der Fall, bei denen jeder Verbraucher beziehungsweise Erzeuger separat gesteuert werden muss. Dies geschieht ohne automatische

und direkte Abstimmung mit den anderen Verbrauchern und Erzeugern, was schematisch in Abbildung 2 (S.7) dargestellt ist.

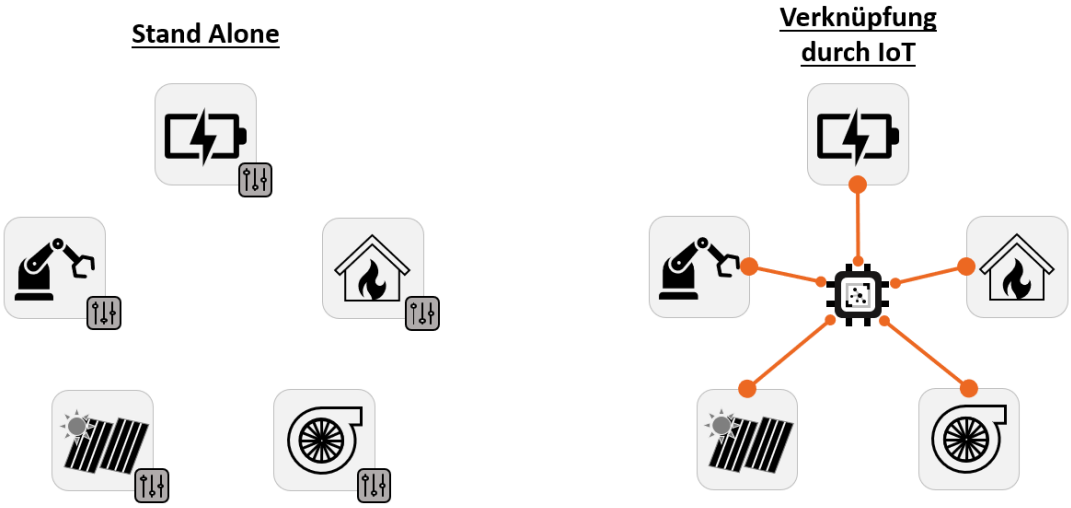


Abbildung 2 - Links: Schema "Stand Alone" Lösung; Rechts: Schema verknüpfter Anlagen durch IoT

Außerdem ist es wahrscheinlich, dass das System auf Basis der Daten von Produktionsanlagen bei Betrachtung ihrer Energieverbräuche und der erzeugten Stückzahlen Rückschlüsse auf deren Effizienz erlaubt. Auf dieser Basis könnte der Belegungsplan des Maschinenparks angepasst und Maschinen zeitweise komplett ausgeschaltet werden.

1.5 Methodik

Das folgende Unterkapitel widmet sich der Beschreibung der Vorgehensweisen, die bei der Erstellung dieser Abschlussarbeit eingesetzt wurden. Da je nach Teilergebnis eine unterschiedliche Methodik verwendet wird, erfolgt eine Einteilung in folgende Bereiche:

- Marktrecherche,
- Auswahl der Hardware-Plattform,
- Programmierung der Datenaufnahme und
- Implementierung der Datenspeicherung.

1.5.1 Methodik zur Marktrecherche

Bevor die Marktrecherche beginnen kann, werden in einem ersten Schritt die Anforderungen an die HSP definiert und in Beurteilungskriterien für marktübliche Lösungen gewandelt. Nachfolgend werden systematisch zunächst Hersteller und Anbieter adäquater Lösungen und im Anschluss deren Produktport-

folios recherchiert. Auf Basis dieser Recherche wird eine Auflistung von mindestens fünf Komplettsystemen und mindestens 15 einzelnen Datenloggern erstellt. Gibt es Unklarheiten oder fehlende, aber relevante Informationen, werden die Anbieter kontaktiert, um eine möglichst gute Grundlage für die spätere Beurteilung der Systeme und den Vergleich mit den angestrebten und tatsächlich umgesetzten Lösungen zu schaffen.

1.5.2 Methodik zur Auswahl der Hardware-Plattform

Zur Auswahl einer geeigneten Hardware-Plattform wird zunächst eine Anforderungsliste benötigt, um gezielt nach passenden Geräten suchen zu können. Diese Aufstellung wird im vorherigen Schritt für die Marktrecherche bereits abgeschlossen. Im Anschluss wird durch strukturierte Recherche eine Liste mit mindestens 15 verschiedenen Herstellern von Industrie-PCs ausgearbeitet. Auf Basis dieser Auflistung werden sowohl die Internetauftritte der Hersteller als auch Online-Vertriebskanäle einschlägiger Händler möglicher Hardware geprüft. Da es eine Vielzahl möglicher Produkte gibt, werden grobe Filtermöglichkeiten (z.B. Anzahl der *Ethernet*-Ports) verwendet, um die Liste der vorgeschlagenen Modelle vorab zu kürzen. Die einzelnen Produkte werden anschließend mit der zuvor erstellten Anforderungsliste abgeglichen und bei fehlenden Informationen der Händler beziehungsweise der Hersteller werden diese kontaktiert. Die technischen Eckdaten der potenziell einsetzbaren Mini-PCs werden innerhalb einer Tabelle gesammelt und auf Basis der Rückmeldungen der Unternehmen ergänzt. Abschließend wird ein erneuter Abgleich mit der Anforderungsliste durchgeführt und nach einer Bewertung die Entscheidung für ein Modell getroffen.

1.5.3 Methodik zur Programmierung der Datenaufnahme

Für die Erstellung der benötigten Skripte zur Datenaufnahme wurde die Software *CODESYS* als Entwicklungsumgebung ausgewählt. Zu Beginn wird versucht, die Kernprobleme durch die Suche nach ähnlichen Anwendungsfällen im Web mithilfe von Suchmaschinen zu lösen. Dabei werden, basierend auf bisherigen Erfahrungen, Webseiten wie *stackoverflow.com*, auf der sich Programmierer jeden Niveaus über ihre Skripte und die damit verbundenen Probleme austauschen können, als Hauptquellen dienen.

Sobald eine geeignete *CODESYS*-Bibliothek gefunden wird, die Methoden und Funktionen zur Lösung der Hauptaufgabe enthält, wird sie installiert und getestet. Enthält sie die geforderten Funktionen, wird ein erstes Programm erstellt. Danach werden die Skripte stetig debuggt, beschleunigt, kommentiert, verbessert und gegen vom Benutzer oder vom System produzierte Fehler geschützt. Während es teilweise notwendig sein wird, eine kleine Testumgebung

zu Debugging-Zwecken zu erstellen, wird versucht, die Skripte größtenteils mit realen Messdaten zu testen.

1.5.4 Methodik zur Implementierung der Datenspeicherung

Wie in Kapitel 1.4 bereits erwähnt, soll für die Speicherung der Daten ein IoT-Datenbank-System verwendet werden. Die Implementierung einer solchen Struktur erfordert zunächst die Auswahl einer IoT-Plattform, die die Speicherung und Verwaltung der Messdaten erlaubt. Nachdem eine geeignete Plattform ausgewählt wurde, wird ähnlich wie bei der Programmierung der Datenaufnahme nach vergleichbaren Anwendungsfällen im Internet recherchiert. Auch hier sind die Verwendung verschiedener Programmbibliotheken und das Erstellen von kleinen Testumgebungen notwendig, wobei nach Möglichkeit mit realen Daten gearbeitet wird.

2 THEMATISCHE GRUNDLAGEN

Um den Einstieg in das Thema der Energiedaten und der energetischen Effizienz zu schaffen, befasst sich das folgende Kapitel zunächst mit der Begriffsdefinition der Energieeffizienz und der Differenzierung der verschiedenen Bestandteile einer Gesamteffizienz. Im Anschluss wird auf die generelle Rolle von Energiemanagement und -monitoring, die Energiedatenanalyse und die Methode der prädiktiven Regelung eingegangen.

Die zweite Hälfte fokussiert sich auf Energieeffizienz in Unternehmen, wobei zunächst die prinzipiellen Energieflüsse eines Betriebs beleuchtet werden. Nachdem mögliche Motivationen für ein betriebliches Energiemanagement thematisiert wurden, werden nähere Informationen zur *ISO 50001* gegeben, die für das Thema von zentraler Wichtigkeit ist. Bevor das Kapitel mit der Klärung typischer Einsparpotenziale in Unternehmen abschließt, wird noch auf die Rolle der Gebäudeautomation für ein effizientes Energiemanagement eingegangen.

2.1 Energieeffizienz

Mittelpunkt dieses Abschnitts sind nicht nur die Definition des Begriffes „Energieeffizienz“, sondern auch die Unterscheidung unterschiedlicher Effizienzen. Zusätzlich wird generell das Energiemanagement betrachtet, von dem das Monitoring von Energiedaten ein wichtiger Bestandteil ist. Der letzte Teil des Kapitels geht näher auf die Energiedatenanalyse ein und erklärt den Begriff der „prädiktiven Regelung“.

2.1.1 Begriffsdefinition

Generell wird der Begriff der „**Effizienz**“ über die Relation von betriebenem Aufwand zu dem daraus resultierenden Nutzen definiert. So wird zum Beispiel eine Arbeit als effizient bewertet, wenn das Resultat mit geringem Aufwand an Ressourcen (Zeit, Kosten etc.) erreicht wird. Im Bereich der Energie kann der Begriff gleichwertig verwendet werden und bezieht sich hier auf eine energetische Arbeitsleistung, die mit einem verhältnismäßig kleinen Einsatz von Energie erreicht werden kann. Somit kann das Verhältnis dieser beiden Werte als Größe zur Messung der energetischen Effizienz verwendet werden. [4, S. 45] Ein einfaches Beispiel ist der Vergleich einer konventionellen Glühlampe mit einem LED-Leuchtmittel: Letzteres benötigt aufgrund des höheren Wirkungsgrades für die gleiche Lichtausbeute weniger Energie und besitzt somit eine höhere Energieeffizienz.

2.1.2 Unterscheidung der Energieeffizienz

Für die Bereitstellung von Energie kommen in einem System verschiedene Prozesse zum Einsatz, bei denen vor allem die Schritte der Gewinnung, Umwandlung und Übertragung von Energie mit Blick auf die Effizienz eine große Rolle spielen. Dabei wird Primärenergie (z.B. Gas) für die Gewinnung der Endenergie (z.B. Strom) verwendet, die dem Endnutzer zur Verfügung gestellt wird. Dieser wendet die Endenergie im Anschluss für eine Energiedienstleitung (z.B. das Erhitzen von Wasser) auf. Hier wird von Nutzenergie gesprochen.

Während dieser Schritte finden Wandlungsprozesse statt, bei denen ein Teil der Energie für den weiteren Prozess verloren geht. Das kann beispielsweise bei der Verbrennung von Gas entstehende Abwärme sein, die nicht vollständig genutzt werden kann. Die Relation der ursprünglich verwendeten Primärenergie zu der nach der Umformung nutzbaren Endenergie wird als **Wandlungseffizienz** bezeichnet.

Der zweite wichtige Begriff ist die sogenannte **Nutzungseffizienz**, die betrachtet, wie das Verhältnis von aufgewendeter Endenergie zu der tatsächlich für die Energiedienstleistung genutzten Energie ist. Diese wird ebenfalls vor allem durch Wandlungsprozesse reduziert und kann durch Verwendung von Energie aus der Umwelt erhöht werden. [4, S. 46-47]

Die **Gesamteffizienz** setzt sich aus den beiden oben beschriebenen Einzeleffizienzen zusammen und wird demnach maßgeblich durch diese beiden Arten beeinflusst. Deshalb kann die Gesamteffizienz durch Veränderung der beiden Größen direkt erhöht oder gesenkt werden. Dabei kann der zur Erfüllung einer Arbeitsleistung eingesetzte Betrag der Primärenergie zur Messung der Gesamteffizienz verwendet werden. [4, S. 48-49]

2.1.3 Energiemanagement und -monitoring

Die Betrachtung, Regelung und Verbesserung von Energieflüssen werden als „**Energiemanagement**“ bezeichnet. Durch Einführung eines Energiemanagements können Stellschrauben und Optionen ermittelt werden, die die Einsparung von Energie ermöglichen.

Zentraler Bestandteil ist dabei das sogenannte „**Energiemonitoring**“. Es beinhaltet die Aufnahme aller Energieflüsse eines Unternehmens unter Berücksichtigung der verwendeten Energieträger (z.B. Strom und Gas). Erst dadurch werden die Erzeugung und der Verbrauch der Energie in einem System transparent. Somit trägt ein gesamtheitliches Energiemonitoring maßgeblich zur Steigerung der Energietransparenz bei und ermöglicht das Erkennen und Ana-

lysieren der energetischen Effizienz. Auf diese Weise werden zentrale Einflussgrößen sichtbar und eine gezielte Regulierung und Steigerung der Energieeffizienz ist möglich. [5]

Grundsätzlich sind vor allem Energiezähler und Sensorik, die die Energieflüsse aufnehmen und in Form von Messdaten an ein zentrales Energiemanagement-System weiterleiten, essenzieller Bestandteil eines flächendeckenden Energiemonitorings.

Eine weitere Steigerung der Energietransparenz stellt die Visualisierung der Energieflüsse dar. Dies kann z.B. durch Übertragung der energetischen Messdaten auf Monitore dienen, sodass Verbraucher einen direkten Zusammenhang zwischen ihrem Handeln und dem Steigern oder Senken der Energieeffizienz sehen können. Hier kommen unter anderem grafische Darstellungen wie Diagramme zum Einsatz, die beispielsweise sowohl aktuelle als auch historische Verbräuche zeigen.

2.1.4 Energiedatenanalyse und prädiktive Regelung

Neben dem Energiemonitoring ist vor allem die Energiedatenanalyse ein weiterer wichtiger Bestandteil eines funktionierenden Energiemanagements. Erst dadurch kann ein Nutzen aus den energetischen Messdaten gezogen werden.

Mit steigender Komplexität eines Energiesystems nimmt auch die Menge der aufzunehmenden Messdaten zu. Gleichzeitig steigt aber auch der Erkenntnisgewinn, der auf Basis tiefgreifender Analysen generiert werden kann.

Je nach Bedarf wird die Vielschichtigkeit der Energiedatenanalyse gewählt. So kann eine einfache Auswertung von Energiemessdaten bereits in Form von Meldungen erfolgen, die den Anwender beim Erreichen festgelegter Sollwerte alarmieren. Ein nächster Schritt auf dem Weg zu einer mehrschichtigen Analyse ist beispielsweise die Deutung von Vergangenheitswerten. Dadurch können unter anderem ungünstige Betriebspunkte erkannt und Anpassungen und Maßnahmen für das zukünftige Energiemanagement entwickelt werden.

Werden zusätzliche Informationen wie die Saison, Produktionsdaten oder Wettervorhersagen in die Auswertungen mit einbezogen, können Prognosen über den zukünftigen Energiebedarf erstellt werden. Sie helfen dabei, das betrachtete System noch effizienter zu steuern und z.B. potenzielle Zeiten für hohe Lastspitzen zu erkennen oder ungeplante Ausfälle frühzeitig zu vermeiden. Diese Art der Analyse und Verarbeitung wird inzwischen bereits als **prädiktive Regelung** oder prädiktive Steuerung verstanden. [6]

Noch einen Schritt weiter gehen Systeme, die mit der sogenannten „modell-prädiktiven Regelung“ arbeiten. Sie entwickeln unter Zuhilfenahme von dynamischen Modellen, die auf Differentialgleichungen basieren, kontinuierlich das Systemverhalten und Prognosen passend zu den zur jeweiligen Zeit vorherrschenden Bedingungen für einen in der Zukunft liegenden Zeitraum. Es werden also alle Regelungen, die eine Relevanz besitzen, vollkommen automatisch beachtet. Aufgrund dieser Eigenschaft werden Regeln, die auf Basis unterschiedlicher Fälle abgeleitet werden, redundant. Dieser entscheidende Unterschied sorgt dafür, dass die Übertragung von einem Gebäudetyp auf Bauten anderer Art möglich ist. Regeln, die auf Erfahrungen basieren, ermöglichen zwar die Realisierung einer prädiktiven Arbeitsweise, können aber nicht auf alle Gebäudetypen angewandt werden. So gibt es beispielsweise viele Gebäude wie Bürokomplexe, in denen überhaupt keine Produktionsmaschinen vorkommen. Wird also eine Regel programmiert, die besagt, dass eine Anlage heruntergefahren werden soll, da sie in den nächsten Stunden nicht benötigt wird, kann sie nur auf eine bestimmte Art von System angewandt werden. [7, S. 38-41]

2.2 Energieeffizienz in Unternehmen

Das folgende Kapitel gibt zunächst einen Überblick über den Anteil des Energieverbrauchs von Unternehmen am Endenergiebedarf Deutschlands und stellt nähere Informationen zu den Energieströmen eines Unternehmens zur Verfügung. Im Anschluss erfolgt die Vorstellung möglicher Motivationen und von gesetzlichen Regelungen beim Einsatz von Energiemanagement. Zusätzlich werden dessen Einsatz im Betrieb unter Berücksichtigung der *ISO-Norm 50001* und bei Verwendung von Gebäudeautomation thematisiert, bevor die Erläuterung typischer Einsparpotenziale den Abschnitt abschließt.

2.2.1 Energieverbrauch deutscher Unternehmen

Wie in Abbildung 3 (S.14) zu sehen ist, tragen die Unternehmen in Deutschland mit ihrem Energiebedarf maßgeblich zum Gesamtenergieverbrauch der Bundesrepublik bei. Ihr Endenergiebedarf in Höhe von 3.637 Petajoule ist mit 44 % fast genauso groß wie der der privaten Haushalte und des Verkehrs zusammen. Unternehmen haben somit einen bedeutenden Einfluss auf die Erreichung der deutschen Klimaziele. Aufgrund des hohen Endenergieverbrauchs spielen vor allem der Umgang mit den vorhandenen Energieressourcen und die daraus resultierende Effizienz eine große Rolle.

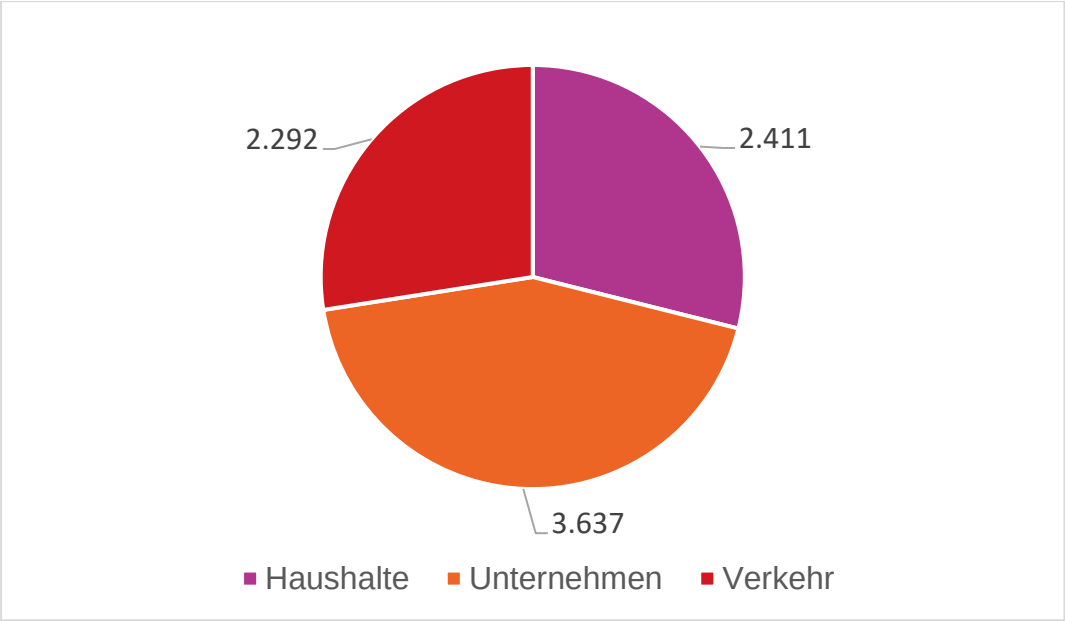


Abbildung 3 - Endenergieverbrauch Deutschlands in Petajoule nach Sektor im Jahr 2020 [nach 8]

2.2.2 Energieflüsse eines Unternehmens

Für die Bewertung der Energieeffizienz eines Unternehmens hilft es, den Bilanzrahmen stetig zu erweitern. So kann zu Beginn ein erster Schritt die Betrachtung der Wirkungsgrade, die per Definition über das Verhältnis von Leistungsabgabe zu Leistungsaufnahme im Beobachtungsmoment bestimmt werden, der einzelnen Geräte sein. Wird der Bilanzraum von einem einzelnen Betrachtungszeitpunkt auf eine Zeitspanne erweitert, wird dies als „Untersuchung des Nutzungsgrades“ bezeichnet. Das Einbeziehen der energetischen Bedarfe von untergeordneten Verbrauchern vergrößert ebenso das bilanzierte System und ermöglicht eine noch detailliertere Analyse.

Werden die energetischen Ströme eines Produktionsablaufes betrachtet, wird die nächste Detailebene erreicht. Die Energieflüsse eines Unternehmens sind von verschiedenen Faktoren wie unter anderem dem Standort und der Unternehmensgröße abhängig. So ist es beispielsweise bei großen Betrieben inzwischen üblich, dass ein Teil der benötigten Energie mithilfe von Photovoltaik (PV) oder BHKWs selbst produziert wird. Um diese Abläufe in Form von Energieflüssen zu visualisieren, kommen sogenannte Sankey-Diagramme zum Einsatz. [9, S. 11]

Eine beispielhafte Darstellung der Energieflüsse eines fiktiven Unternehmens mit einem Sankey-Diagramm ist in Abbildung 4 (S.15) zu sehen.

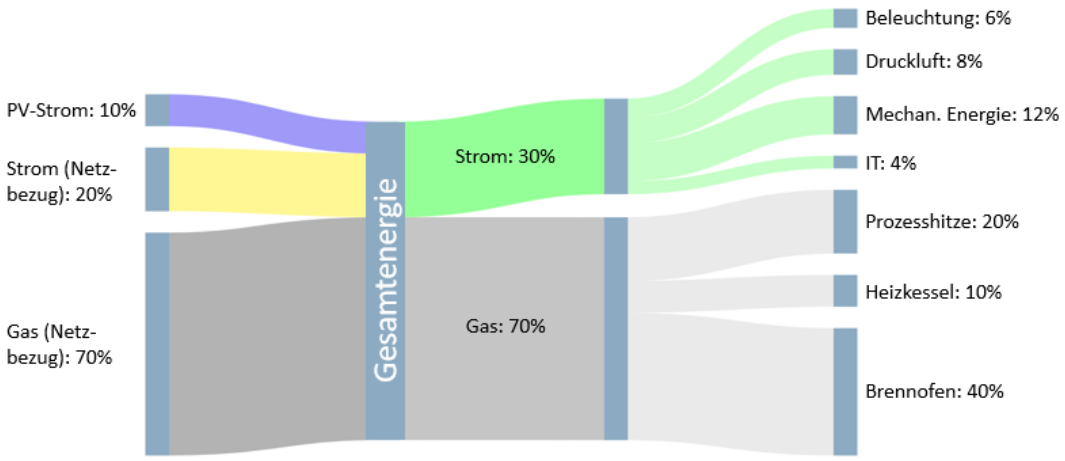


Abbildung 4 - Beispielhafte Darstellung der Energieflüsse eines fiktiven Unternehmens

Für die Erstellung eines Sankey-Diagramms werden sämtliche im Unternehmen vorhandene Energiequellen (z.B. PV- und Solarthermieanlagen) und -senken (z.B. Produktions- und Heizungsanlagen) einbezogen. Zu diesem Zweck müssen zuvor aufwändige Datenaufnahmen erfolgen, die das Gesamtsystem erfassen und detailgetreu darstellen.

Weiter an Komplexität nimmt die energetische Betrachtung zu, wenn die Unternehmensgrenzen überschritten werden und der Betrieb mit branchengleichen oder sogar -fremden Firmen verglichen wird. Dadurch können weitere hilfreiche Anregungen gewonnen werden, die über die bisherigen Maßnahmen und das gesammelte Wissen hinausgehen. [9, S. 11]

2.2.3 Motivation für Energiemanagement in Unternehmen

Die Motivation von Unternehmen sich mit dem Thema Energieeffizienz auseinanderzusetzen, kann divers sein. So kann der Wunsch nach einem bewussteren und nachhaltigeren Umgang mit den vorhandenen Ressourcen ebenso ein Grund sein wie die Aufbesserung des öffentlichen Images oder wirtschaftliche Vorteile in Form von Kostenreduktionen.

Je nach Größe des Unternehmens gibt es jedoch auch klare gesetzliche Regelungen, in denen Vorgaben zu Energieaudits und einem zertifizierten Energiemanagementsystem (EMS) festgelegt werden.

Befreit sind Unternehmen, die alle drei folgenden Punkte erfüllen:

1. Das Unternehmen beschäftigt maximal 249 Vollzeit-Mitarbeiter.
2. Es werden jährlich weniger als 50 Millionen Euro Umsatz erzielt und die Summe der Jahresbilanz liegt unter 43 Millionen Euro.

3. Andere Unternehmen beteiligen sich monetär mit maximal 25 % als Partner oder mit maximal 50 % als Verbund. [10]

Während eines Energieaudits analysiert ein sogenannter Auditor, der entweder dem Betrieb zugehörig oder extern angestellt ist, die energetischen Verbräuche und Ausgaben. Nachfolgend zeigt er Möglichkeiten für Einsparungen auf und beurteilt diese auf Basis von ökonomischen Gesichtspunkten. Stattfinden müssen diese Audits seit Ende 2015 in einem vierjährigen Rhythmus.

Damit die Optimierungen auf lange Sicht Teil der Betriebsorganisation werden, ist ein effektives Energiemanagement notwendig. Dieses erleichtert die Entwicklung, Realisierung und regelmäßige Überprüfung von Initiativen. Hat ein Betrieb bereits ein System für das Energiemanagement nach *DIN EN ISO 50001* oder eines für das Umweltmanagement nach der *Verordnung (EG) 1221/2009* realisiert, muss er kein zusätzliches Energieaudit durchführen.

Aber auch für kleinere Unternehmen, die zu diesen Maßnahmen nicht verpflichtet sind, kann ein Energieaudit auf freiwilliger Basis auch mit Blick auf Steuervergünstigungen sinnvoll sein. So erlangen sie etwa ein Recht auf legitime Kompensationsbestimmungen im Zuge des *Erneuerbare-Energien-Gesetzes* (§§ 63 ff EEG 2014) oder beim Spitzenausgleich (§ 55 Energie- beziehungsweise § 10 Stromsteuergesetz). [11]

2.2.4 Energiemanagement nach ISO 50001

Mithilfe der *ISO-Norm 50001* können EMS, wie im Kapitel *Motivation für Energiemanagement in Unternehmen* bereits erwähnt, für Unternehmen zertifiziert werden. Veröffentlicht wurde die für Managementsysteme geltende Norm, die eine weltweite Richtlinie für EMS ist, 2011. Sie beschränkt sich nicht auf einzelne Sektoren und kann von Firmen jeder Größe und auch Behörden verwendet werden. Die *ISO 50001* legt lediglich den Handlungsspielraum fest, in dem der eigene Einsatz des entsprechenden Betriebs notwendig ist. Dieser hat die Aufgabe die in der Norm festgehaltenen Ansprüche adäquat auf das individuelle Bestreben anzupassen und zu realisieren.

Sieben Jahre nach Veröffentlichung der ursprünglichen Version wurde 2018 eine Novelle publiziert, deren Hauptaugenmerk auf der Annäherung an den uniformen Grundaufbau von Richtlinien für *ISO*-Managementsysteme lag. Seitdem leitet sich die Struktur der *ISO 50001* von folgenden Standards ab:

1. der *ISO 9001*-Norm für Qualitätsmanagement,
2. der *ISO 14001*-Norm für Umweltmanagement und
3. der Arbeitssicherheitsmanagementnorm *ISO 45001.*

Zusätzlich dazu gab es eine Vereinheitlichung von grundlegenden Definitionen und übergreifend gültigen Formulierungen, sodass eine Verbindung unterschiedlicher Managementsysteme nun leichter fällt. [12]

Zentraler Teil der *DIN ISO 50001* ist der *PDCA-Zyklus*, der innerhalb eines fortlaufenden Optimierungsablaufs regelmäßig umgesetzt werden sollte. Dabei folgt die Methode der in Abbildung 5 dargestellten Struktur, die aus den vier Hauptphasen „plan“, „do“, „check“ und „act“ besteht. [9, S. 36-37]

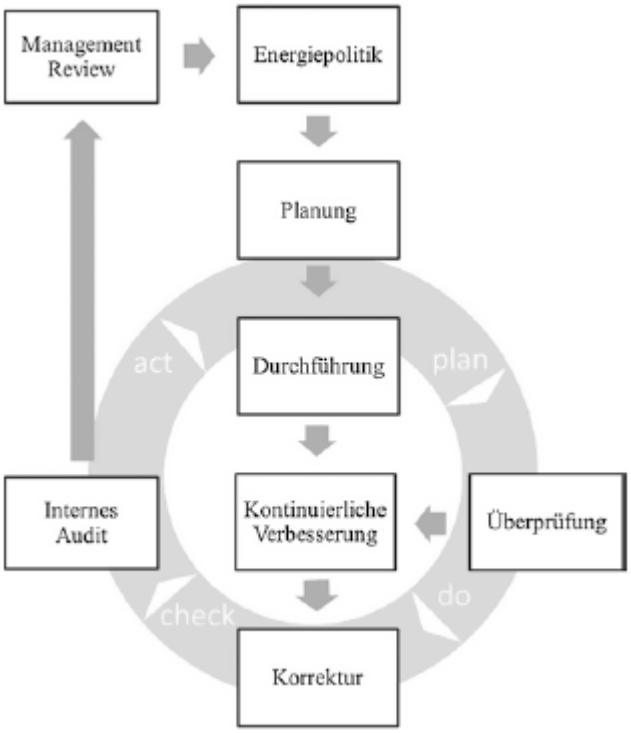


Abbildung 5 - Kontinuierlicher Verbesserungsprozess im *PDCA-Zyklus* nach *DIN ISO 50001* [9, S. 37]

Der erste Schritt des *PDCA-Zyklus* („plan“) widmet sich der Bündelung bestehender Bestandteile eines EMS, der Erstellung eines EMS-Entwurfes und der Dokumentation in einem Leitfaden. Basis dafür ist die von der Geschäftsführung beschlossene Strategie in Bezug auf das Thema „Energie“.

Während der nächsten Phase („do“) wird das EMS mit Unterstützung von Direktiven eingeführt, wobei das Messen energetischer Daten zuvor durchgeführte Auswertungen abrundet.

Der „Check“-Schritt beinhaltet innerbetriebliche Revisionen, mit deren Hilfe die beschlossenen Kriterien kontrolliert und Diskrepanzen in Form eines Berichtes festgehalten werden.

Innerhalb der letzten Phase („act“) stehen der Beschluss und die Umsetzung eines Handlungsentwurfs und ein Rückblick der Leitung im Vordergrund. Während letzterem werden sowohl der vollständige Ablauf als auch die Resultate betrachtet, damit fortlaufende Optimierungen realisiert werden können. [9, S. 36-37]

Damit ein Unternehmen entsprechend der Managementsystemnorm zertifiziert werden kann, müssen verschiedene Voraussetzungen eingehalten werden, die nachfolgend aufgelistet sind:

1. Festlegung des EMS-Geltungsbereiches
2. Spezifizierung einer Strategie bezüglich des Themas „Energie“ und Kommunikation an die Beschäftigten
3. Benennung eines Verantwortlichen für den Energiebereich
4. Registrierung und Evaluierung energetischer Kriterien
5. Bildung energetischer Indizes
6. Registrierung und Überwachung energetischer Verbraucher
7. Evaluierung des momentanen Energiekonsums
8. Definition, Verbesserung und Dokumentation der Vorgänge, die für den Energiebereich wichtig sind
9. Umsetzung von Energieaudits und Leitungsrückblicken
10. Identifikation energetischer Einsparmöglichkeiten
11. Gewährleistung fortlaufender Optimierung der Energieperformanz [9, S. 36-37]

Die Prozesse, die zu den Entschlüssen führen, und die Datengrundlage sollten bei der Aufnahme, der Zuweisung, der Analyse und der Administration von Energiedaten transparent aufgebaut werden. Auf solch einer internen Energie-Revision, die ein wichtiges Energiemanagement-Element ist, basieren der Fortschritt und die Realisierung von Ansätzen ökonomischer Energieeffizienz. In Abhängigkeit von der Komplexität geschieht die Auswertung inkrementell bedingt durch die Höhe der Detailebene. Dabei können einzelne Phasen auch Energieberatern externer Dienstleister übertragen werden. Gerade bei aufwändigen Ansätzen wird im Normalfall die Hilfe externer Experten in Anspruch genommen. [9, S. 36-37]

2.2.5 Verwendung von Gebäudeautomation für Energiemanagement

Wie bereits in den vorangegangenen Kapiteln erwähnt, sind das Messen, Monitoren und die Verarbeitung von Energiedaten essenziell für ein funktionie-

rendes Energiemanagement. Zu diesem Zweck werden für gewöhnlich Komponenten inklusive zweckdienlicher Schnittstellen aus dem Bereich der **Gebäudeautomation** verwendet.

Wird auf die Unabhängigkeit von Herstellern und die Übereinstimmung mit gesetzlich geregelten Normen geachtet, ist eine größtmögliche Freiheit in Bezug auf Planung und Ausschreibung gegeben. Die Eigenschaft der einzelnen Komponenten, diese komplett in das bestehende System einbinden zu können, ermöglicht eine problemlose Kommunikation der Produkte untereinander. Dadurch können kleine Anpassungen potenziell häufiger mithilfe von Programmänderungen realisiert werden und müssen nicht zwingend durch eine neue Verkabelung umgesetzt werden.

Zu den üblichen Einsatzmöglichkeiten von Komponenten der Gebäudeautomation gehören unter anderem die Steuerung der Beleuchtung, von Abschattungsmöglichkeiten wie Rollladen und Jalousien und die Regelung von Heizungs- oder Lüftungsanlagen. Mithilfe einer zentralen Steuereinheit können somit bereits Verhaltensregelungen implementiert und durch Abstimmung der verschiedenen Gebäudeautomations-Gewerke miteinander Effizienzsteigerungen erreicht werden. [13, 484-485]

2.2.6 Einsparpotenziale in Unternehmen

Die Einsparpotenziale eines Unternehmens hängen stark von der Struktur und den vorhandenen Energiequellen und -senken ab. Außerdem variieren die Einsparpotenziale stark nach Branche und hängen unter anderem von dem Alter der eingesetzten Technologien ab. Generell lassen sich aber folgende fünf Hauptkategorien zur Unterscheidung der Technologien und den jeweils charakteristischen Einsparpotenzialen bilden:

1. Strombasierte Basistechnologien
2. Brennstoffbasierte Basistechnologien
3. Strombasierte Querschnittsprozesse
4. Brennstoffbasierte Querschnittsprozesse
5. Gekoppelte und sonstige Querschnittsprozesse

Zu der ersten Kategorie „Strombasierte Basistechnologien“ gehören unter anderem Transformatoren und unterbrechungsfreie Stromversorgungen, die den Großteil der energetischen Einbußen verursachen. Bei beiden Komponenten ist eine Überdimensionierung typisch, die für ineffiziente Betriebsweisen sorgt und unnötig hohe Energieverbräuche verursacht. Weitere Technologien dieser Kategorie sind beispielweise die elektrische Beleuchtung und Antriebe, Venti-

latoren und Pumpen. Auch hier hat in der Vergangenheit häufig eine Überdimensionierung stattgefunden. Weitere Einsparpotenziale ergeben sich durch veraltete Produkte, deren Wirkungsgrad weit unter dem heutigen Durchschnitt liegt, fehlende Steuerung und Regelung und schlechte Wartung. Letztere äußert sich z.B. bei Pumpen in einem höheren Spiel, das wiederum zu einem ineffizienten Betrieb führt. Die nachfolgende Tabelle 1 gibt einen Überblick strombasierter Basistechnologien eines Betriebs und, listet zusammenfassend typische Probleme auf. [9, S. 53-117]

Tabelle 1 - Mögliche Probleme typischer Komponenten von strombasierten Basistechnologien in Unternehmen

[nach 9]

Kategorie	Typische Komponenten	Mögliche Probleme
Strombasierte Basistechnologien	- Transformatoren - Unterbrechungsfreie Stromversorgungen - Elektrische Beleuchtung - Elektrische Antriebe - Ventilatoren - Pumpen	- Falsche Auslegung (meist Überdimensionierung) - Veraltete Technologien mit geringen Wirkungsgraden - Keine ausreichende Wartung der Komponenten

Die nächste Gruppe enthält technische Basisverfahren, die zur Gewinnung und Verteilung von Wärme auf Basis von Brennstoffen genutzt werden. Auch hier sind Anlagen häufig überdimensioniert, weshalb diese beispielsweise nicht im optimalen Betriebspunkt laufen und somit nicht effizient sind. Zudem führen eine fehlende oder schlechte Steuerung sowie unregelmäßige Wartung zu einem erhöhtem Ressourcenverbrauch. [9, S. 117-127]

Zusätzlich zu der vorgeschalteten Wärmegenerierung sollten stets auch die Übertragung der Wärme bei deren Recycling und der Transport betrachtet werden. Häufig sind deutschlandweit in beiden Bereichen die Potenziale nicht ausgeschöpft. Dies bestätigen Berechnungen, die sagen, dass allein der Anteil von Flächen ohne Dämmung bei ungefähr 10 % liegt. Hinzu kommen Bereiche, die nur unzureichend oder mangelhaft gedämmt sind. Nicht durchgeführte Sichtkontrollen, die Maßnahmen zur Instandhaltung ins Leben rufen würden, führen ebenso zu energetischen Verlusten. Die nachfolgende Tabelle 2 (S.21) fasst die typischen Probleme von Komponenten der brennstoffbasierten Basistechnologien noch einmal zusammen. [9, S. 127-144]

Tabelle 2 - Mögliche Probleme typischer Komponenten von brennstoffbasierten Basistechnologien in Unternehmen [nach 9]

Kategorie	Typische Komponenten	Mögliche Probleme
Brennstoffbasierte Basistechnologien	- Brenner - Wärmeübertrager - Dämmung	- Falsche Auslegung (Unter- oder Überdimensionierung) - Kein Ausschöpfen der Potenziale - Keine regelmäßige Wartung & Instandhaltung

Auf den vorangegangenen Seiten wurden zunächst die „einfachen“ technischen Grundverfahren betrachtet. Die nächste Kategorie „strombasierte Querschnittsprozesse“ befasst sich nun mit den aufwändigeren Verfahren, die auf Strom basieren. Eine vielfach verwendete Technologie ist die Drucklufttechnik, die mit ungefähr 16,5 Terrawattstunden grob 8 % des Strombedarfs der deutschen Industrie ausmacht. Problematisch sind hier vor allem eine unregelmäßige Wartung und Instandhaltung, die zu unerkannten Leckagen und Druckverlusten führen. Außerdem werden die Anlagen teilweise mangelhaft oder gar nicht in Hinblick auf die Betriebsauslastung gesteuert. [9, S. 147-159]

Ebenfalls zu den „strombasierten Querschnittsprozessen“ werden Rechenzentren gezählt, die aufgrund von durch Nutzer abgespeicherte veraltete oder nicht benötigte Daten zum „Datengrab“ werden können. Dieser Umstand, eine ineffiziente Kühlung der Hardware und veraltete oder schlecht gesteuerte USV-Technik führen zu hohen Stromverbräuchen, die vermieden werden können. [9, 159–170]

Die dritte Technologie aus demselben Bereich ist die Galvanik, bei der eine Vielzahl an Problemen auftreten kann, die die Effizienz des Verfahrens stark beeinträchtigen. So sind beispielsweise hohe Spannungsverluste durch unregelmäßige Wartung, eine schlecht geregelte Luftführung und das Fehlen einer Abwärmerückgewinnung wichtige Faktoren. Zusätzlich dazu führen eine mangelhafte Wärmedämmung der Galvanik-Bäder und der Einsatz ineffizienter Komponenten zu hohen Energieverbräuchen und geringen Wirkungsgraden. Ein Überblick zu strombasierten Querschnittsprozessen eines Betriebs inklusive der Auflistung möglicher Probleme ist in Tabelle 3 (S.22) festgehalten. [9, 170-182]

Tabelle 3 - Mögliche Probleme typischer Komponenten von strombasierten Querschnittsprozessen in Unternehmen [nach 9]

Kategorie	Typische Komponenten	Mögliche Probleme
Strombasierte Querschnittsprozesse	- Druckluft - Rechenzentren - Galvanik	- Unregelmäßige Wartung und Instandhaltung - Mangelhafte Steuerung - Ineffiziente Kühlung und Luftführung - Mangelhafte Wärmedämmung

Zu den komplexeren Prozessen, die auf der Verwendung von Brennstoffen basieren, gehören unter anderem Kessel zur Dampferzeugung, bei denen vor allem eine Überdimensionierung, eine geringe Ausnutzung des Wärmerückgewinnungspotenzials und eine mangelhafte Dämmung problematisch sind. Letzteres ist bei Technologien dieser Kategorie eine generelle Problematik, die deshalb auch bei Industrieöfen, Trocknungs- und Lackieranlagen auftritt. Zusätzlich dazu mindern bei Industrieöfen der Einsatz veralteter Brenner und eine geringe Nutzung der Abwärme die Effizienz. Vor allem bei Trocknungs- und Lackieranlagen kann eine mangelhafte Prozesssteuerung zu einem ressourcenaufwändigen Betrieb führen. Eine Zusammenfassung kann Tabelle 4 entnommen werden. [9, 185-236]

Tabelle 4 - Mögliche Probleme typischer Komponenten von brennstoffbasierten Querschnittsprozessen in Unternehmen [nach 9]

Kategorie	Typische Komponenten	Mögliche Probleme
Brennstoffbasierte Querschnittsprozesse	- Kessel zur Dampferzeugung - Industrieöfen - Trocknungsanlagen - Lackieranlagen	- Falsche Auslegung (Überdimensionierung) - Mangelhafte Wärmedämmung - Nutzung veralteter Technologien - Mangelhafte Prozesssteuerung - Geringe Abwärmenutzung

Neben den vornehmlich strom- oder brennstoffbasierten Querschnittsprozessen gibt es auch gekoppelte Verfahren, die sich beider Ressourcen bedienen. Im Bereich der gekoppelten Produktion und Verwendung von Strom und

Wärme zählen beispielsweise BHKWs, Dampfturbinen, Wärmepumpen und Kältemaschinen zu typischen Technologien. Bei Anlagen dieser Art sind vor allem eine nicht bedarfsgerechte Dimensionierung, mangelhafte Regelung und unregelmäßige Wartungsarbeiten problematisch. Ebenso beeinflussen eine ungenügende Wärmedämmung und Fehler in der Mess-, Steuerungs- und Regelungstechnik die Effizienz negativ. [9, S. 239-251]

Bei der Betrachtung raumlufttechnischer oder kältetechnischer Anlagen stehen unter anderem die einzelnen Komponenten im Fokus, die womöglich ineffizient arbeiten und nicht regelmäßig gewartet werden. Zudem spielen gerade bei Anlagen zum Transport von Luft Undichtigkeiten eine große Rolle. Eine mangelhafte Einstellung der Temperaturen und ungenügende Wärmedämmung senken die Wirkungsgrade in der Kälteerzeugung weiter. [9, S. 252-273]

Weitere Verfahren, deren energetische Einsparpotenziale offenbart, sind die Förder-, Handhabungs- und Vakuumtechnik. Hier führt vor allem die Verwendung ineffizienter Antriebe und eine Überdimensionierung zu unnötigen energetischen Verlusten, wie Tabelle 5 entnommen werden kann. Auch die Verwendung von Industriegasen, Schweißprozesse und die Abwasserbehandlung sollten eingehend betrachtet werden, da hier auf verschiedene Weise Effizienzsteigerungen möglich sind. So können beispielsweise höhere Prozessgeschwindigkeiten beim Schweißen zu verbesserten Wirkungsgraden führen. [9, S. 273-282]

Tabelle 5 - Mögliche Probleme typischer Komponenten von gekoppelten und sonstigen Querschnittsprozessen in Unternehmen [nach 9]

Kategorie	Typische Komponenten	Mögliche Probleme
Gekoppelte und sonstige Querschnitts-prozesse	- Gekoppelte Erzeugung und Nutzung von Strom und Wärme (BHKWs, Dampfturbinen, Wärmepumpen und Kältemaschinen) - Raumlufttechnische Anlagen für Industriegebäude - Kälteerzeugung - Sonstige Querschnittsprozesse (Förder-, Handhabungs-, Vakuumtechnik, Industriegase, Schweißen, Abwasserbehandlung)	- Falsche Auslegung (Überdimensionierung) - Unregelmäßige Wartung und Instandhaltung - Mangelhafte Regelung - Verwendung ineffizienter Komponenten - Ungenügende Dämmung

3 GRUNDLAGEN ZUR HARD- UND SOFTWAREPROGRAMMIERUNG

Die Grenzen und Unterschiede zwischen den Begrifflichkeiten „**Hardware**“ und „**Software**“ sind nicht immer ganz klar, weshalb dieser Abschnitt zunächst die Begriffe definiert. Zusätzlich dazu wird auf die Grundlagen der Softwareprogrammierung eingegangen, um ein allgemeines Verständnis für die zugrundeliegenden Methoden zu schaffen. Dadurch sollen möglicherweise aufkommende Fragen in späteren Kapiteln, in denen die Kernthemen dieser Arbeit behandelt werden, bereits vorab beantwortet werden.

Neben der Betrachtung der verschiedenen Phasen bei der Softwareprogrammierung wird auch kurz die Visualisierungsmöglichkeit von Programmcode durch Verwendung von Programmablaufplänen erklärt. Eine weitere Hilfe bei der Erstellung von IT-Lösungen ist die Verwendung von Programmbibliotheken, die in Kapitel 3.2.3 näher betrachtet wird, bevor der Abschnitt mit der Beschreibung der verwendeten Entwicklungsumgebung *CODESYS* und einer Erläuterung des *Modbus-TCP*-Protokolls endet.

3.1 Begriffsdefinitionen: Hardware, Software und Plattform

Für die Definition des Begriffs „Hardware-Plattform“ sollte zunächst geklärt werden, was unter „Hard-“ beziehungsweise „Software“ verstanden wird. Als „Hardware“ werden technische Module bezeichnet, die Bestandteile eines Computers sind. Dies sind nicht nur alle Komponenten, die in einem solchen Gerät eingesetzt sind und auf Anregungen elektronischer oder mechanischer Art eine Reaktion zeigen, sondern auch externe Geräte, die Schnittstellen zu einem Computer besitzen, wie zum Beispiel Drucker. Zu den standardmäßig in einem Computer verbauten Hardware-Elementen gehören im Normalfall neben einem Prozessor (auch CPU[1] genannt), eine Grafikkarte, Arbeitsspeicher und Laufwerke beziehungsweise Festplatten zur Datenspeicherung. Die verschiedenen Bauteile werden mithilfe einer Hauptplatine physisch miteinander verbunden.

Die digitale Ergänzung der Hardware wird „Software“ genannt und umfasst die Computerbestandteile, die nicht physischer Natur sind, wie beispielsweise das Betriebssystem und Anwendungen. Diese werden für die Verwendung des

[1] CPU = central processing unit (Englisch für “zentrale Prozessoreinheit“)

Systems und die Verarbeitung von Daten benötigt, welche ebenfalls unter den Begriff der Software fallen. Anders als bei Hardwareanpassungen können Änderungen an Umfang und Art der Software leicht durch Installation und Deinstallation der Komponenten auf dem Computer vorgenommen werden. [14]

Als Plattform wird in der Informationstechnologie eine uniforme Grundlage gesehen, die mit verschiedenen Elementen verbunden wird und der Installation und Verwendung von Anwendungen dient. Typischerweise wird eine computertechnische Plattform über folgende Punkte spezifiziert:

1. Rechnerarchitektur
2. Betriebssystem
3. Programmiersprachen
4. Schnittstellen (benutzerspezifisch und graphisch)
5. Laufzeitumgebungen
6. Bibliotheken

Eine für Plattformen besonders wichtige Eigenschaft ist ihre Kombinierbarkeit mit anderen Systemen, sodass eine bestenfalls störungsfreie Kommunikation möglich ist. [15]

3.2 Grundlagen zur Softwareprogrammierung

3.2.1 Phasen der Softwareentwicklung

Wie in anderen Bereichen des Projektmanagements gibt es auch in der Softwareentwicklung verschiedene Modelle, die das Vorgehen bei gleichartigen Projekten festlegen. Grundlegend setzen sich die unterschiedlichen Modelle jedoch meist aus den in Abbildung 6 gezeigten Phasen zusammen. [16, S. 3-6]

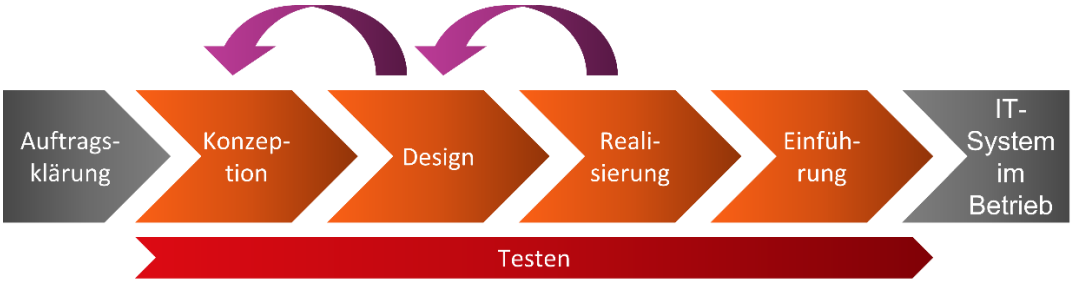


Abbildung 6 - Phasen der Softwareentwicklung [nach 16, S. 6]

Im ersten Schritt wird im Rahmen der Auftragsklärung eine grobe Idee nach ökonomischen Kriterien untersucht, mit denen ermittelt wird, ob eine Umsetzung aus Unternehmenssicht sinnvoll ist. Als Prüfungskriterien dienen dabei neben Aspekten der Wirtschaftlichkeit und Machbarkeit auch die Interessen der internen Unternehmenspolitik. Hält die Idee der Untersuchung stand, werden unter anderem nach einer Ist-Analyse die Projektziele und resultierenden Ablaufänderungen definiert. [16, S. 6-11]

Während des nächsten Abschnitts, der „Konzeptions-“ oder auch „Kundenphase“ genannt wird, werden Lösungsalternativen beispielsweise mithilfe einer Nutzwertanalyse verglichen. Es gilt die technische Plattform der informationstechnischen Lösung zu definieren und zum Beispiel die Art des Computers und das Betriebssystem festzulegen. Die Konzeptionsphase dient außerdem dazu zu klären, welche funktionalen Anforderungen existieren, wie die Datenhaltung gestaltet werden soll und welche essenziellen Kennzahlen und zusätzlichen Bedingungen berücksichtigt werden müssen. Zu diesem Zweck werden in der Regel zwei Dokumente erstellt: das Lastenheft, das eine exakte Beschreibung des Endproduktes von der Seite des (externen/internen) Kunden ist, und das Pflichtenheft, welches von dem Serviceunternehmen angefertigt wird und alle Vorgaben für das informationstechnische Endprodukt enthält. [16, S. 12-16]

Nachdem geklärt ist, welchen Umfang die IT-Lösung haben soll, wird innerhalb der Designphase die Struktur des Systems geplant, wobei folgende Punkte näher spezifiziert werden:

1. Hardwarearchitektur
2. Softwarearchitektur
3. Überführung und Eingliederung in die bestehende IT-Struktur

Mit der Definition der Hardwarearchitektur wird festgelegt, aus welchen Hardware-Komponenten die geplante IT-Struktur bestehen und wie die Verbindung der Bauteile untereinander gestaltet sein soll. Für die Festlegung der softwaretechnischen Komponenten, die in der Architektur zum Einsatz kommen, müssen die Schnittstellen und die Verhaltensweise bei spezifiziertem Input definiert werden. Damit eine problemlose Überführung und Eingliederung in die bestehende IT-Struktur erfolgen kann, muss geklärt werden, mit welchen Komponenten sich das zu entwickelnde Endprodukt austauschen muss und wie der Austausch ablaufen soll.

Im darauffolgenden Abschnitt, der Realisierungsphase, steht die Erstellung der IT-Lösung im Vordergrund. Besonders wichtig ist hier die Einhaltung der

in den vorangegangenen Schritten definierten Anforderungen. Die letztendliche Umsetzung erfolgt durch Programmierung in einer Entwicklungsumgebung. Wichtiger Teil dieser Phase ist das regelmäßige und ausgiebige Testen der Lösung, um mögliche Schwachstellen, Probleme und Bedienfehler durch den Anwender frühzeitig zu erkennen und zu beseitigen. [16, S. 16-18]

Die Einführungsphase dient dazu, das geplante und fertig entwickelte Endprodukt in das Zielsystem beim Kunden einzugliedern und es dort in Gang zu bringen. Da unter realen Bedingungen häufig Schwierigkeiten aufkommen, die in der Entwicklungsumgebung nicht aufgefallen sind, sind möglicherweise Nacharbeiten und Optimierungen notwendig. Erst wenn das System einwandfrei funktioniert und allen Anforderungen entspricht, erfolgt die Abnahme durch den Kunden. Die Art und Weise der Transformation in die bestehende informationstechnische Landschaft wird in zwei Vorgehensweisen unterschieden: Die Einführung aller Teile mit einem Mal („Big Bang“) und die schrittweise Einführung („Step-by-Step“). Eine weitere Aufgabe während dieser Phase ist das Training der späteren Benutzer. [16, S. 19-21]

Wie in Abbildung 6 zu sehen, ist die Testphase in der Regel kein ausschließlich am Ende stattfindender Schritt, sondern ein kontinuierlicher Prozess, der während der bereits genannten Phasen erfolgt. Ziel des Testens ist es, ein fehlerhaftes Verhalten der IT-Lösung frühzeitig zu entdecken und zu beseitigen. Dieser Prozess ist enorm wichtig, da die Wahrscheinlichkeit einer arbeits- und kostenintensiven Beseitigung steigt, je später ein Fehler entdeckt wird. Durch das Testen werden stetig Änderungsschleifen angestoßen, die zu Anpassungen in den jeweiligen Phasen und beispielsweise der Anforderungen führen können. [16, S. 22-25]

Der letzte Schritt ist der Betrieb des IT-Systems, das in der Zielumgebung implementiert wurde und nach der Abnahme durch den Kunden von den Anwendern im Produktivsystem genutzt werden sollte. Treten nun Fehler und Probleme auf, fällt die Beseitigung meist nicht mehr unter die Projektaufgaben des IT-Dienstleisters. Zu diesem Zweck werden Service- und Wartungsverträge geschlossen, die eine Betreuung des implementierten Systems und die Lösung aufkommender Probleme sichern.

3.2.2 Programmablaufpläne als Visualisierungshilfe

Mit der Komplexität eines Programms sinkt die Übersichtlichkeit des Programmcodes, weshalb in solchen Fällen häufig auf Visualisierungshilfen zurückgegriffen wird. Neben Struktogrammen sind Programmablaufpläne eine

häufig verwendete Methode zur Darstellung von softwaretechnischen Strukturen. Um eine Uniformität bei dieser Art der Visualisierung zu gewährleisten, ist in der *DIN 66001* festgelegt, welche Symbole für welche Programmabläufe verwendet werden sollen. Neben einer verbesserten Übersichtlichkeit hat dies den Vorteil, dass es auch fachfremden Personen mit wenig bis keinen Programmierkenntnissen leichter fällt, einen Programmablauf zu verstehen. Die Symbolik ist für alle Programmiersprachen gleich, sodass auch Programmierer selbst von einer solchen Visualisierung profitieren, da sie nicht zwingend mit der Syntax und Funktionen einer Programmiersprache vertraut sein müssen, um den Code zu verstehen.

Abbildung 7 zeigt eine Auswahl einiger Elemente, die es nach *DIN 66001* erlauben Algorithmen grafisch zu beschreiben. Die einzelnen Symbole werden mithilfe von Ablauflinien verbunden, die den Steuerfluss darstellen, sodass der Programmablauf über eine baumartige Struktur verbildlicht wird. [17]

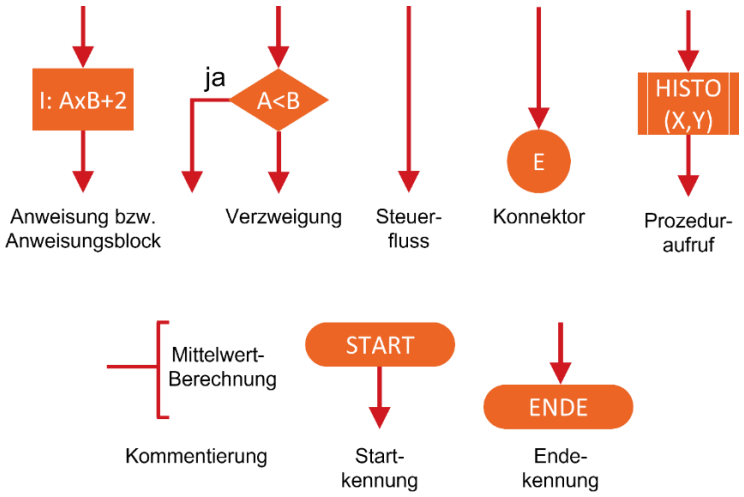


Abbildung 7 - Beispiele für Programmablaufplan-Symbole nach *DIN 66001* [nach 17]

3.2.3 Verwendung von Bibliotheken

Programmiersprachen bieten in der Regel die Verwendung sogenannter Programmbibliotheken an, die bereits programmierte Lösungen für verschiedene Funktionalitäten bereithalten. Programmbibliotheken unterscheiden sich insofern von Programmen, als sie selbst keine ausführbaren Komponenten, sondern Bausteine sind, die für Anwendungen bereitgestellt werden, um diese zu unterstützen. [18]

Solche Sammlungen von Programmfunktionen sind eine große Hilfe für Entwickler, da sie häufig thematisch zusammenhängende Klassen und Funktionen bereitstellen, die sonst selbst erdacht und händisch programmiert werden müssten. Als Beispiele seien hier die *pandas*-Bibliothek, die schnelle und

mächtige Werkzeuge für die Datenanalyse und -bearbeitung bereitstellt, und die *openpyxl*-Bibliothek genannt, die Methoden und Funktionen anbietet, um Verbindungen zu *SQL*-Servern herzustellen und *SQL*-Befehle auszuführen. Häufig werden Programmbibliotheken beispielweise für Programmiersprachen wie *Python* kostenfrei in Form eines Downloads zur Verfügung gestellt.

3.3 CODESYS

Die folgenden Unterkapitel beschreiben die *CODESYS*-Software, die für die Umsetzung dieser Arbeit genutzt wurde und der Erstellung von Automatisierungslösungen dient.

3.3.1 Vorstellung von CODESYS

CODESYS ist eine Automatisierungssoftware, die erstmalig im Jahr 1994 veröffentlicht, seitdem kontinuierlich weiterentwickelt wurde und von der *CODESYS GmbH* vertrieben wird. Die Software ist herstellerunabhängig, weshalb sie sich großer Beliebtheit bei internationalen Industrieunternehmen erfreut, und dient als Entwicklungsumgebung für benutzerdefinierte Steuerungen und Komponenten im Bereich der Automatisierung. Darunter fallen neben industriell verwendeten speicherprogrammierbaren Steuerungen unter anderem auch Einplatinen- und Industrie-Computer. [19]

Neben verschiedenen Editoren, die unterschiedliche Arten der Programmierung erlauben, bietet *CODESYS* zahlreiche Programmbibliotheken an. Diese erleichtern Anwendern bei komplexen Projekten die Arbeit, indem sie es ihnen beispielsweise ermöglichen, alle gängigen Industrie-Schnittstellen und -Protokolle anzusteuern beziehungsweise zu verwenden. Die Laufzeitumgebung, die die Programmierung der Automatisierungs-Komponenten erlaubt, kann sowohl auf *Windows*- als auch auf *Linux*-Betriebssystemen installiert werden, wodurch die Flexibilität weiter gesteigert wird. [20]

3.3.2 Verwendung der Programmiersprachen in CODESYS

Wie bereits in Kapitel 3.3.1 erwähnt, bietet *CODESYS* Entwicklern verschiedene Editoren für die Programmierung an. Diese ermöglichen unter anderem die Verwendung der Implementierungssprachen *Funktionsbausteinsprache* und *Strukturierter Text*. Unabhängig davon, welche der Sprachen ausgewählt wird, ist der Aufbau stets gleich gestaltet und entspricht der Darstellung in Abbildung 8 (S.30): Im oberen Bereich werden die Variablen deklariert und in der unteren Hälfte kann der eigentliche Programmcode erstellt werden.

```
fcInsertQuery
1  PROGRAM fcInsertQuery
2  VAR
3      sQuery: STRING(256);
4      sTabelle: STRING;
5      TestStr: STRING;
6      TestInt: INT;
7  END_VAR
8
```

```
1  sTabelle:= 'TestTabelle';
2
3  sQuery:= 'INSERT INTO ';
4  sQuery:= CONCAT(STR1:=sQuery, STR2:=sTabelle);
5  sQuery:= CONCAT(STR1:=sQuery, STR2:=' (IrgendeinText, IrgendeinInt) Values (N$'');
6  sQuery:= CONCAT(STR1:=sQuery, STR2:=TestStr);
7  sQuery:= CONCAT(STR1:='$',N$'');
8  sQuery:= CONCAT(STR1:=sQuery, STR2:=INT_TO_STRING(TestInt));
9  sQuery:= CONCAT(STR1:=sQuery, STR2:=')');
10 //fcInsertQuery:= sQuery;
```

Abbildung 8 - Aufbau der Programmieroberfläche am Beispiel von "Strukturierter Text"

Während bei der Programmierung in *Strukturierter Text* der Programmcode vollständig selbst geschrieben werden muss, bietet die Verwendung der *Funktionsbausteinsprache* fertige Bausteine für Funktionen, Operatoren und Ähnliches an. So kann beispielsweise die Verbindung zu einem *SQL*-Server inklusive Datenversand mithilfe eines einzigen Funktionsbausteins hergestellt werden, wie in Abbildung 9 zu sehen ist.

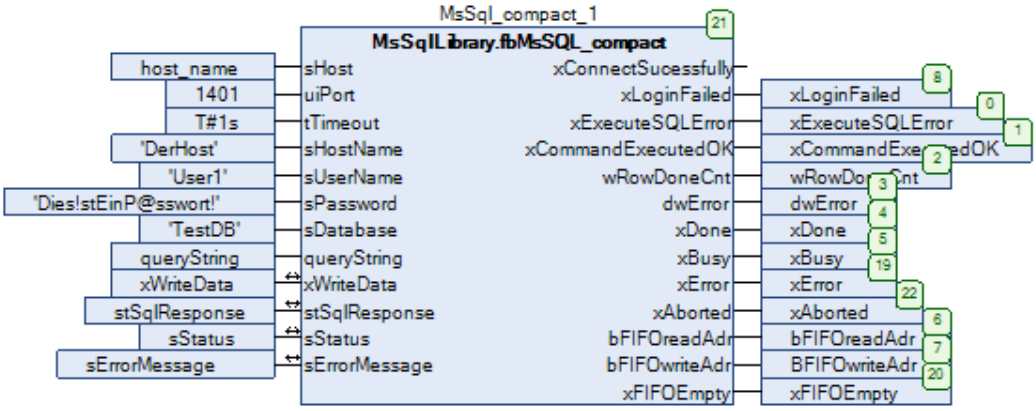


Abbildung 9 - Beispiel für die Verwendung eines komplexen Funktionsbausteins

Dies kann nicht nur eine Vereinfachung bei der Erstellung des Programmcodes sein, sondern dient auch bis zu einem gewissen Grad der Übersichtlichkeit. Diese verringert sich bei der Verwendung der *Funktionsbausteinsprache* mit steigender Anzahl der verwendeten Bausteine, da die Darstellungsmöglichkeiten begrenzt sind. Zwar ist es möglich, aus der Programmierebene heraus zu zoomen, um einen besseren Überblick zu erlangen, aber dadurch verringert sich gleichzeitig die Größe der Beschriftung. Dies führt zwangsläufig ab einer gewissen Zoomgröße dazu, dass der grobe Aufbau des Programmcodes

zwar sichtbar ist, Details aufgrund der sinkenden Lesbarkeit aber nicht mehr erkennbar sind (siehe Abbildung 10).

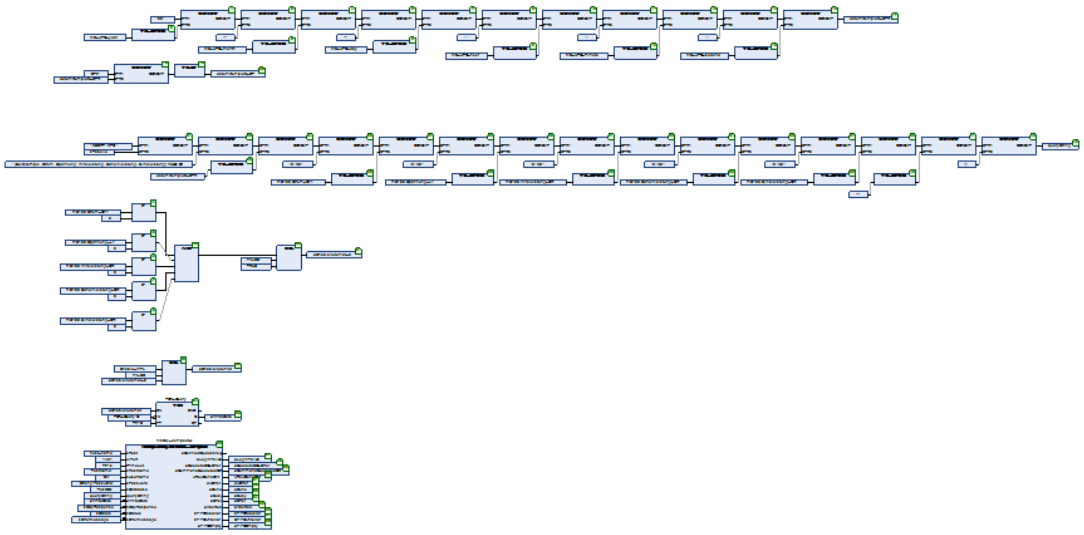


Abbildung 10 - Darstellung von Programmcode mit mehreren Funktionsbausteinen

Aus diesem Grund sollte stets im Einzelfall entschieden werden, welche der möglichen Programmiersprachen sich am besten für die angestrebte Lösung eignet.

3.3.3 Meldungen in CODESYS

Während im oberen Bereich der Softwareoberfläche standardmäßig die Programmierung des Codes stattfindet, ist gleich darunter ein Fenster zur Ausgabe von Meldungen positioniert. Wird ein Programmcode übersetzt und es treten Fehler, Warnungen oder sonstige Meldungen auf, werden diese dort dargestellt. Dadurch können Probleme bereits beseitigt werden, bevor der Code auf die zu programmierende Automatisierungskomponente übertragen wird. Durch einen Doppelklick mit dem Mauszeiger auf die entsprechende Meldung springt die Software direkt in den Programmcode und die problematische oder fehlerbehaftete Position wird angezeigt.

3.3.4 Visualisierungen in CODESYS

Eine weitere praktische Funktion in *CODESYS* ist die Möglichkeit, mithilfe eines Baukastens eine Visualisierung zu erstellen, die auf der programmierbaren Steuerung läuft und beispielsweise über einen Browser durch Eingabe der Internetprotokoll-Adresse (IP-Adresse) des Systems eingesehen werden kann. Dies ermöglicht es Anwendern Funktionen der Steuerung zu verwalten, Parameter zu setzen oder aktuelle Werte und Fehlermeldungen einzusehen.

Neben geometrischen Grundformen wie Rechtecken und Kreisen werden unter anderem auch folgende Visualisierungselemente zur Verfügung gestellt: Textfelder, Regler, Schaltflächen, Fortschrittsbalken, Knöpfe, Schalter, Taster, Messinstrumente, Lampen sowie Datums- und Zeitsteuerelemente. Durch Ziehen der Komponenten aus dem Baukasten und Loslassen auf der Visualisierungsebene werden die einzelnen Elemente eingefügt. Diese können nach dem Einfügen unter anderem in Größe und Form angepasst und Variablen aus dem Programmcode zugeordnet werden. Somit kann das Interface einfach und beliebig komplex gestaltet werden, wie beispielhaft in Abbildung 11 dargestellt ist.

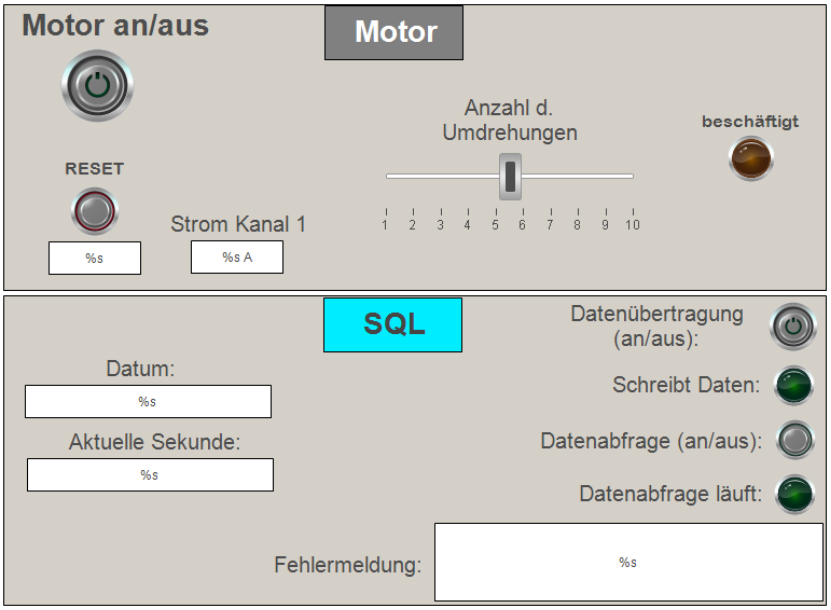


Abbildung 11 - Beispiel einer Veranschaulichung im Visualisierungseditor von *CODESYS*

3.4 Modbus-TCP-Protokoll

In der Industrie werden sogenannte Bussysteme (z.B. *Profibus*, *Ethernet*) zur sicheren Datenübertragung zwischen einzelnen Geräten verwendet, damit diese reibungslos miteinander kommunizieren können. Dabei kommen Bus-Schnittstellen wie beispielsweise *Ethernet*-Ports zum Einsatz, um über Verkabelung eine zuverlässige und direkte Verbindung der Kommunikationsteilnehmer untereinander zu realisieren. Elementare Komponenten dieser Systeme sind Protokolle, die festlegen, wie Informationen ausgetauscht werden. Zu diesem Zweck hat jedes Protokoll eigene Vorgaben für die Syntax (Einsatzmöglichkeiten von Zeichen- und Wortfolgen), die Semantik (Definition der Korrektheit der genutzten Folgen) und die Übermittlungszeit (welcher Teilnehmer bekommt wann welche Information). [21]

Eines der bekanntesten Bus-Protokolle ist *Modbus-TCP*, das von *Schneider Electric* entwickelt wurde und bereits seit 1979 auf dem Markt ist. Nachdem es über Jahrzehnte hinweg weiterentwickelt wurde, wird es inzwischen standardmäßig in der Industrieautomation verwendet wird. Dies ist auch einer der Gründe, weshalb dieses Protokoll für die Umsetzung der prototypischen HSP verwendet werden soll. *Modbus-TCP* wird in der Verbindung mit dem *Ethernet*-Bussystem verwendet und basiert auf einer Kommunikation zwischen einem sogenannten „Client“ und einem „Server“. In diesem Zusammenhang gibt es drei Rollen: Den *Modbus Client*, der anwendungsgesteuert Nachrichten erzeugt, den *Modbus Server*, der diese Nachrichten aufnimmt, verarbeitet und eine Antwort sendet, und das *Modbus User Application (Backend) Interface*, das den Server mit der Benutzeranwendung verbindet. Aufgrund dieses Aufbaus ermöglicht das Protokoll eine Kommunikation mit sogenannten „Remote devices“, also Geräten, die abseits verbaut sind, aber mithilfe von *Modbus-TCP* angesteuert werden können. [22]

4 ABGRENZUNG ZU AKTUELLEN UND BISHERIGEN REFERENZPROJEKTEN

Bereits seit mehreren Jahren wird die Verwendung von modellprädiktiven Steuerungen zum Management von sogenannten „Heating, Ventilation and Air Conditioning“-Systemen (HVAC-Systemen) in Nichtwohngebäuden innerhalb von wissenschaftlichen Vorhaben betrachtet. [23] In der Regel stehen die Optimierung der Beheizung und Kühlung von Räumen im Fokus. Dabei können deutliche Einsparungen auf Energie- und Kostenseite durch Verwendung dynamischer Modelle, die auf physikalischen Differentialgleichungen basieren, realisiert werden. Durch die Nachbildung der zu erwartenden Verhaltensweise des Systems ist das Einbinden von Prognosen bezüglich des Wetters oder der betrieblichen Auslastung möglich.

Der Großteil der modellprädiktiven Regelungen kommt bei Steuerungen der Temperatur zum Einsatz. [24], [25] Teilweise werden jedoch, wie beispielsweise im *BIG*-Projekt, auf das im folgenden Kapitel noch genauer eingegangen wird, große elektrische Konsumenten betrachtet. Das potenzielle Projekt der Firma *manageE* soll sich jedoch nicht allein auf die Bedarfe von Heizung, Kühlung und generell der Energie beziehen. Darüber hinaus soll eine Vielzahl an elektrischen und thermischen Konsumenten, Quellen und Speichern berücksichtigt werden. Vor allem die Integration verschiedener Energiespeicher (z.B. Strom-, Gas- und thermische Speicher) und die bestmögliche Distribution von eigens produzierter Energie aus diversen Quellen (z.B. BHKW, PV) übersteigen die Betrachtungen gleichartiger Projekte.

Um das potenzielle Projekt der Firma *manageE* von wichtigen wissenschaftlichen Projekten, die noch in der Bearbeitung oder bereits beendet sind, abzugrenzen, werden diese inklusive Betrachtungshorizont auf den folgenden Seiten kurz vorgestellt.

4.1 OptiControl-II

Als eines der bereits beendeten Projekte soll hier zunächst das *OptiControl-II*-Projekt genannt werden, das von der *Eidgenössischen Technischen Hochschule Zürich* zusammen mit verschiedenen Industrieunternehmen durchgeführt wurde. Die Betrachtungen zielten darauf ab zu analysieren, welche Gebäudearten generell einen Vorteil bei der Verwendung einer modellprädiktiven Regelung haben. Dabei wurde gezeigt, dass eine modellprädiktive Regelung der Temperatur an einem Bürogebäude realisiert werden konnte. Ein wichtiger Unterschied zu dem von *manageE* angestrebten Projekt ist, dass lediglich die

Regelung der Temperatur zum Einsatz kam und die Verwendung von Energiespeichern nicht betrachtet wurde. [26, S. 1-4]

In der Laufzeit des *OptiControl-II*-Projektes wurden alle geplanten Ziele erreicht, zu denen unter anderem die Weiterentwicklung, Implementierung und Bestätigung der Machbarkeit von modellprädiktiven Regelungsstrategien für die praktische Anwendung unter realen Bedingungen gehörten. Über die Dauer von eineinhalb Jahren wurden die entwickelten Strategien getestet und neue Methoden und Werkzeuge für die erweiterte Gebäudekontrolle entwickelt. Dabei wurde herausgefunden, dass die Anwendbarkeit der Regelstrategien davon abhängt, ob es möglich ist, ihre Komplexität hinter einfach zu verstehenden Konzepten, automatischen Prozeduren und benutzerfreundlichen Oberflächen zu verstecken.

Als Beweis für die Flexibilität und die Eignung der Kontrollalgorithmen für die alltägliche Anwendung in der Gebäudeautomatisierung wurde die einfache Übertragung ohne Anpassung der Konzepte von einem einzelnen Raum auf ein ganzes Demonstrationsgebäude gesehen. Es zeigte sich, dass die entwickelten Regelstrategien eine verbesserte Steuerleistung im Vergleich zu anderen Lösungen besaßen, die dem Stand der Technik entsprachen. Dies war unter anderem auch in der Einsparung von 10-17 % der aufgewendeten Primärenergie erkennbar. Eine wichtige Erkenntnis des Projektes war außerdem, dass der tatsächlich durch Einsatz modellprädiktiver Regelungen gewonnene Mehrwert stark von Gebäudetyp, Standort und den technischen Systemen abhängig ist.

Eines der Ergebnisse war jedoch auch, dass der Einsatz modellprädiktiver Regelungen zu höheren Kosten vor allem bei der Planung und dem Bau neuer Gebäude führt. Aus diesem Grund wurde geschlussfolgert, dass vor allem die Preisstrategie im Falle einer Adaption von hoher Bedeutung ist und, ob der jeweilige Auftraggeber bereit ist in eine qualitativ hochwertigere Lösung zu investieren. Zum Zeitpunkt des Projektabschlusses waren die Verringerung der energetischen Aufwände und die damit einhergehenden Kostenersparnisse zu gering, um den Einsatz eines solchen Systems zur modellprädiktiven Regelung zu rechtfertigen. Vor dem Hintergrund der steigenden Energiepreise wäre dies jedoch neu zu kalkulieren. [26, S. 153-156]

4.2 BIG

Ein weiteres bereits abgeschlossenes Projekt ist das *BIG*-Projekt, das im Rahmen der Fördermaßnahme *KMU-innovativ* vom *Fraunhofer-Institut für Produktionstechnologie IPT* in Aachen zusammen mit drei Industrie-Unternehmen

durchgeführt wurde. Fokus des Vorhabens war die Optimierung der Raumklima-Steuerung eines Betriebs auf Basis von meteorologischen Daten und Produktionsplänen. Zwar wurde in Simulationen und zur Regelabstrahierung eine modellprädiktive Regelung verwendet, im Anschluss jedoch eine prädiktive Strategie, die auf Regeln basierte, entwickelt und zusammen mit einem Hardware-Prototypen eingesetzt. Ähnlich wie bei dem zuvor beschriebenen *OptiControl*-Projekt fand lediglich die Untersuchung der thermischen Energieversorgung statt. Außerdem flossen die Kopplung verschiedene Energiespeicher und die bestmögliche Distribution von eigens produzierter Energie aus diversen Quellen (z.B. BHKW, PV) nicht in die Betrachtung mit ein. [27]

Im Laufe des *BIG*-Projektes wurden zunächst verschiedene Algorithmen verglichen, um zu prüfen, wie eine Regelung der Heiz- und Kühlprozesse mithilfe der Gebäudeautomation durchgeführt werden konnte. Basierend auf dem Vergleich wurde ein Modell zur Temperaturentwicklung abgeleitet, das auf reale Daten zugreifen konnte und eine ausreichende Genauigkeit besaß. Auf Basis des Modells wurden drei Strategien abgeleitet:

1. eine regelbasierte Strategie,
2. eine *PID*-Strategie der klassischen Regelungstechnik und
3. eine modellprädiktive Regelung.

Im Feldtest bei den Projektpartnern konnten neben der Raumtemperatur auch - wie geplant - andere Daten (z.B. Belegungspläne) aufgenommen werden. Hier stellte sich heraus, dass die regelbasierte Strategie eine höhere Praxistauglichkeit besaß und deshalb final für einen Prototypen umgesetzt wurde.

Ein weiteres Ergebnis war, dass die Energiekosten im Falle der Anwendung bei der regelbasierten Strategie am geringsten waren. Die Kosten der modellprädiktiven Steuerung lagen 20 % darüber und die der klassischen Regelungstechnik betrugen etwa das Vierfache der Kosten der regelbasierten Strategie. Im *BIG*-Projekt zeigte sich ebenso, dass die bei der regelbasierten Strategie erzielte Energieersparnis einfacher skaliert werden konnte.

Das Verhalten der unterschiedlichen Strategien bei der Raumtemperatursteuerung war wie folgt: Ohne Verwendung eines der Modelle lag die Raumtemperatur außerhalb des tolerierbaren Bereiches. Die regelbasierte Strategie und die modellprädiktive Regelung sorgten dafür, dass die Temperatur im tolerierbaren Rahmen lag. Im Fall der klassischen Regelungstechnik befand sich die Temperatur unterhalb der Toleranzen. Damit erfüllte dieses Modell als einziges nicht die Vorgaben, die an die klimatischen Voraussetzungen im Gebäude gestellt wurden. Dabei fand ein Abgleich der Temperaturvorhersagen mit den realen Werten statt, bei dem sich zeigte, dass die vorhergesagten Werte eine

hohe Genauigkeit aufwiesen. Die einzelnen Regelstrategien besaßen demnach dieselbe Datengrundlage, setzten die eingespielten Werte jedoch auf individuelle Art und Weise um. [28, siehe Anhang A]

4.3 SIOSTA

Das *SIOSTA*-Projekt gehört zu den noch laufenden Projekten und wird im Rahmen der Fördermaßnahme *KMU-innovativ* von der *Hochschule für Angewandte Wissenschaften Hamburg* zusammen mit der *Envidatec GmbH* bearbeitet. Der Fokus des Vorhabens liegt auf der Verwendung von Verfahren, die auf Maschinellem Lernen basieren, für HVAC-Anlagen-Steuerung in Produktionssystemen. Ziel ist es, durch die Nutzung von Algorithmen, die auf der Tensor-Mathematik basieren, vorhandene Leittechnik in der Industrie um vormals nicht registrierte Energieströme zu erweitern. Dabei soll ähnlich wie in dem von *manageE* angestrebten Projekt die vorhandene Leittechnik nicht ausgetauscht, sondern durch geschickte Ergänzung optimiert werden. [29]

Methodisches Kerninstrument des Vorhabens sind die bereits erwähnten Tensor-Algorithmen, die auf Daten basieren, multilinearer Struktur und besonders leistungsfähig sind. Wie zuvor bereits beschrieben, soll in dem potenziellen Projekt der Firma *manageE* allerdings mit physikalisch basierten Modellen gearbeitet werden, die auch nichtlineare Prozesse abbilden können. Auch wenn Methoden, die Maschinelles Lernen nutzen, über eine besondere Leistungsfähigkeit verfügen, sind Modelle, die ausschließlich auf Daten basieren, bei gleicher Datenlage stets nicht so genau wie bei zusätzlicher Betrachtung physikalischer Wechselbeziehungen und Ermittlung unbekannter Parameter aus Daten. [30]

Zusätzlich zu der Verwendung unterschiedlicher Modelle weicht das *SIOSTA*-Projekt von dem geplanten Projekt ab, da es sich auf Bauten konzentriert, bei denen ein sogenanntes „Supervisory control and data aquisition“-System (kurz: SCADA) verwendet wird. Dies ist vor allem bei kleinen und mittelständischen Unternehmen jedoch eher untypisch.

Ähnlich wie bei den zuvor erwähnten Projekten findet im *SIOSTA*-Projekt lediglich die Untersuchung der thermischen Energieversorgung statt. Außerdem wurden die Kopplung verschiedener Energiespeicher und die bestmögliche Distribution von eigens produzierter Energie aus diversen Quellen (z.B. BHKW, PV) nicht betrachtet. [29]

4.4 Smart-Meter-Rollout

Mit dem Ausbau der Erneuerbaren Energien steigen gleichzeitig die Zahl von dezentralen Energiequellen und der Anteil volatiler Energieerzeugung aufgrund wechselnder Wetterbedingungen. Dieser Umstand führt zu schwankenden Lasten, deren Abschätzung kompliziert ist, und stellt deshalb eine Gefahr für die Stromnetzstabilität dar. Aufgrund der begrenzten Speichermöglichkeiten für elektrischen Strom soll in Zukunft der Energieverbrauch so gesteuert werden, dass er stattfindet, wenn ausreichend Strom zur Verfügung steht. Voraussetzung dafür ist ein „intelligentes Netz“, das eine effiziente Verknüpfung und Balance der Produktion von Strom mit dessen Verbrauch realisiert. Ein solches Netz basiert auf intelligenten Komponenten, die zur Messung und Verarbeitung der Energiedaten verwendet werden und als „Smart Metering Systems“ bezeichnet werden. Dabei sollen aktuelle Verbräuche transparent und der Versand der gemessenen Daten sicher gestaltet werden. Zusätzlich soll die Steuerung von Konsumenten und Energiequellen zur Optimierung der Verwaltung von Lasten und Einspeisungen im Verteilnetz beitragen. [31]

Bereits im Jahr 2010 beauftragte das *Bundesministerium für Wirtschaft und Energie* (BMWi) das *Bundesamt für Sicherheit in der Informationstechnik* (BSI) mit der Entwicklung von technischen Vorgaben und Profilen zum Schutz für Komponenten, die zur Kommunikation in solchen „Smart Metering Systems“ dienen sollen. Diese Geräte werden als „Smart-Meter-Gateways“ (SMGW) bezeichnet. [32, S. 8-10] Mit Inkrafttreten des vom Bundestag verabschiedeten Gesetzes zur Digitalisierung der Energiewende im September 2016 wurde die Thematik weiter in den Fokus gerückt. [31]

Im Lauf der letzten Jahre befassten sich vereinzelt Unternehmen damit, Produkte als SMGWs zertifizieren zu lassen, die den Anforderungen nach dem Messstellenbetriebsgesetz § 22 Abs. 1 Nr. 1–3 gerecht werden. [33, S. 24] Inzwischen haben wie in Tabelle 6 (S.39) dargestellt vier Antragsteller Zertifikate für ihre SMGWs erhalten und bei fünf weiteren Unternehmen ist eine Zertifizierung noch in Bearbeitung. [34]

Tabelle 6 - Auflistung bisher zertifizierter Unternehmen und Produkte [34]

Antragsteller	**Produktname**
EMH metering GmbH & Co.KG	*CASA 1.0*
Power Plus Communications AG	*SMGW Version 1.2*
Sagemcom Dr. Neuhaus GmbH	*SMARTY IQ-GPRS / LTE, Version 1.1*
Theben AG	*CONEXA 3.0 Version 1.2*

Für die schrittweise Digitalisierung wurde der Migrationspfad der SMGW in Form eines Stufenmodells beschrieben, das zum aktuellen Stand in drei Schritte aufgeteilt ist. Im ersten und bereits abgeschlossenen Schritt sollten zunächst solche Fälle umgesetzt werden, die für die Aufnahme elektrischer, für die Abrechnung benötigter Daten am Netzanschlusspunkt eine Rolle spielen. Der zweite Schritt soll sich auf die Datenaufnahme von Energiequellen in Bezug auf die tatsächliche Einspeisung, die Abfrage von Informationen zum Netzzustand und die Bereitstellung gemessener Werte für Mehrwertdienste fokussieren. Die für diese Stufe benötigte Rezertifizierung erhielt bisher ein Hersteller, bei drei anderen läuft das Verfahren aktuell noch. Der dritte Schritt war für das Jahr 2021 geplant und enthielt unter anderem folgende zu prüfende Szenarien: Die Regelung von Niederspannungsverbrauchern, die Verwaltung steuerbarer Einrichtungen für die Energieerzeugung, -speicherung und -verbrauch und das Aufladen mithilfe öffentlicher und privater Ladeinfrastruktur. [33, S. 37-38]

Die geringe Zahl der bisherigen Zertifizierungen zeigt, dass der Aufwand, ein Produkt für den Rollout des SMGWs zu entwickeln groß ist und dementsprechend Ressourcen benötigt. Berücksichtigt werden müssen bei der Entwicklung ebenso die sicherheitstechnischen Anforderungen des BSI wie verschiedene Normen und Gesetze. Dies ist nach Aussagen von *manageE* für ein kleines Unternehmen nur umsetzbar, wenn 100 % der Arbeitskapazität aufgewendet werden. Neben der Bearbeitung eines solch großen Projektes muss ein Unternehmen wie *manageE* zur eigenen Finanzierung allerdings auch kleinere Projekte abwickeln, was unter diesen Umständen nicht mehr möglich wäre. Werden jedoch nur Messungen innerhalb eines Unternehmens betrachtet ohne, dass es eine Grenze zum Energieversorgungsunternehmen gibt, werden zurzeit keine Zertifizierungen benötigt. Aus diesen Gründen hat sich *manageE* gegen eine Teilnahme am SMGW-Rollout entschieden.

Da der Rollout für immer mehr Bereiche der Energiedatenerfassung und des Energiemanagements Pflicht werden soll, wird sich in der Zukunft zeigen, ob

das geplante Projekt von *manageE* dem Markt auf lange Sicht ein relevantes Produkt bieten kann. Womöglich sind in ein paar Jahren auch für die angedachte HSP Zertifizierungen notwendig. Ob diese im Laufe der Zeit auch mit geringeren Anforderungen und Aufwänden erreicht werden können, ist zum jetzigen Zeitpunkt nicht klar.

5 MARKTRECHERCHE

Die Marktrecherche dient dazu, einen Überblick über aktuell verfügbare Produkte zu schaffen, sodass ein Vergleich zu dem in dieser Arbeit entwickelten Prototypen gezogen werden kann. Dies soll vor allem die Relevanz der Erarbeitung noch einmal betonen.

Zu diesem Zweck werden zunächst die Anforderungen an die HSP erörtert, um im Anschluss marktübliche Produkte von der zu entwickelnden Lösung abzugrenzen.

5.1 Anforderungen an die Hard- und Softwareplattform

Für eine detaillierte Marktrecherche müssen in einem ersten Schritt zunächst die Anforderungen näher definiert werden, die an die HSP gestellt werden. Hierzu sollten zwei Betrachtungsrahmen für die Unterscheidung gewählt werden, auch wenn sich deren Vorgaben teilweise überschneiden:

1. Der Vergleich mit reinen Datenloggern
2. Die Gegenüberstellung mit Systemen, die eine vergleichbare Lösung wie die des angestrebten Projektes ermöglichen

Der erste Betrachtungsrahmen, der sich mit reinen Datenloggern befasst, basiert auf den Anforderungen der prototypischen HSP und ist Hauptaugenmerk dieses Kapitels. Zusätzlich werden für den Fall, dass das Projekt der Firma *manageE* umgesetzt wird, im zweiten Teil der Marktrecherche Systeme vorgestellt, die vergleichbare Lösungen anbieten. Die Kriterien für die beiden unterschiedlichen Betrachtungsrahmen werden in den folgenden Unterkapiteln näher erörtert.

5.1.1 Beurteilungskriterien für reine Datenlogger

Für den Vergleich mit Produkten, die allein zur Erfassung von Daten geeignet sind, wird sich hauptsächlich auf die zentrale Hardwarekomponente inklusive Betriebssystem fokussiert. Erstes Kriterium ist hier die Wahl des Betriebssystems, das bereits großen Einfluss auf die Eigenschaften der Plattform besitzt. Gängige Systeme für Netzwerkanwendungen wie beispielsweise die Verwendung als Server sind *Windows* und *Linux*. Auch wenn heutzutage (Stand September 2021) *Windows* mit einem Marktanteil von 75,4 % den Großteil der verwendeten Betriebssysteme ausmacht, hat der Einsatz *Linux*-basierter Systeme, deren Anteil bei etwa 2,33 % liegt, seine Berechtigung und Vorteile. [35]

Die Vorteile der vielfältigen *Linux*-Versionen beginnen bereits bei den Kosten, da der Großteil im Vergleich zu den *Windows*-Systemen kostenfrei verfügbar ist. Hinzu kommen der in der Regel offene Quellcode, der das Auffinden von Sicherheitslücken verbessert, die generell hohe Sicherheit vor Cyber-Attacken und die hohe Zuverlässigkeit. Letztere zeigt sich unter anderem darin, dass Hardware, auf der ein *Linux*-basiertes System läuft, beispielsweise nach einem Stromausfall problemlos neustarten kann, ohne dass es zu Laufzeitfehlern und ähnlichen kommt. Diese Eigenschaft ist bei *Windows* nicht gegeben, aber für den Betrieb eines Loggers essenziell, da er möglichst schnell nach dem Stromausfall wieder Daten erfassen soll. Dies hilft vor allem dabei, große Lücken in Datenbanken zu vermeiden. Ein weiterer Vorteil von *Linux* ist neben der konstant hohen Geschwindigkeit, die bei *Windows*-Systemen im Laufe der Zeit abnimmt, die hohe individuelle Anpassbarkeit des Systems. [36] Aus diesen Gründen wird die Fähigkeit, *Linux* als Betriebssystem nutzen zu können als Anforderung für die Hardware gesetzt.

Als nächste Vorgabe wird die Tauglichkeit des Systems für Industrieanwendungen festgelegt, da diese das geplante Einsatzgebiet des Datenloggers sind. Zur Beurteilung dieser Eigenschaft werden der mögliche Bereich der Betriebstemperatur, der auf mindestens -20 bis +60°C festgelegt wird, und die Möglichkeit betrachtet, die Hardware an einer genormten Tragschiene, nachfolgend „Hutschiene“ genannt, befestigen zu können. Der letzte Punkt zielt auf den Einbau in einen industriellen Schaltschrank ab, was neben der Hutschienenmontage auch eine geringe Baugröße voraussetzt, die auf maximal 200x150x100mm (Länge x Breite x Höhe) beschränkt wird.

Zusätzlich dazu sollte das Gerät dazu in der Lage sein, sowohl mit dem Internet als auch mit einem vom Internet getrennten internen Netzwerk kommunizieren zu können. Aus diesem Grund werden zwei *Ethernet*-Schnittstellen benötigt, die separat voneinander konfigurierbar sind. Dies bedeutet, dass jeder eine eigene IP-Adresse zugeordnet werden kann. Ebenfalls wird von der HSP gefordert, dass diese die Energiedaten für mindestens sechs Monate lokal speichert und lückenlos an eine Cloud-Speicherlösung überträgt. Für diesen Messzeitraum wird eine Speicherkapazität von mindestens 500 Gigabyte Größe vorausgesetzt. „Lückenlos“ bedeutet in diesem Fall, dass überprüft wird, ob die versendeten Daten angekommen sind und zur Not noch einmal versandt werden, bis die Informationen in der Cloud abgespeichert wurden. Bezüglich der Daten selbst ist auch die Auflösung der Aufzeichnungsrate zu beurteilen, die beispielsweise sekündlich, minütlich oder 15-minütig sein kann und in diesem Fall sekündlich gefordert ist.

Um die Wahrscheinlichkeit zu erhöhen, dass die Hardware eine Verbindung zum Internet herstellen kann, wäre es außerdem sinnvoll, wenn der Logger einen Steckplatz für eine *SIM*-Karte besitzt. Dies kann in externen Bereichen, die nicht an das Firmennetzwerk angeschlossen sind, hilfreich sein, wird hier aber nicht als „Muss“-Kriterium angesehen.

5.1.2 Beurteilungskriterien für Komplettsysteme zur Datenerfassung und -verarbeitung

Neben der reinen Funktion Messdaten über eine zentrale Hardware aufnehmen zu können, sollen in dem von *manageE* angestrebten Projekt wie in Kapitel 1.4 beschrieben, intelligente Ein- und Ausgabeeinheiten eingesetzt werden, die die digitalen und analogen Daten der Anlagen abgreifen können. Außerdem soll eine nachgeschaltete Plattform die aufgenommenen Daten analysieren, visualisieren und auf ihrer Basis Algorithmen der modellprädiktiven Regelung nutzen.

Die Beurteilungskriterien, die auf Basis der an die HSP gestellten Anforderungen im vorherigen Kapitel formuliert wurden, gelten theoretisch auch für die Komplettsysteme. Zusätzlich dazu ist im Kontext dieser Arbeit die Betrachtung der Auswertungsmöglichkeiten vor allem im Hinblick auf die Verwendung prädiktiver Analysen sinnvoll. Da insgesamt nur wenige Informationen zu Komplettsystemen frei verfügbar sind, kann eine Beurteilung dieser Systeme nur oberflächlich stattfinden. Details hierzu sind in Kapitel 5.3 zu finden.

5.2 Vorstellung marktüblicher Datenlogger

Auf dem Datenlogger-Markt sind verschiedene Unternehmen tätig, die sich in der Regel auf ein paar Bereiche spezialisiert haben. Aus diesem Grund werden hier von den zwanzig Datenloggern, die während der Recherche gefunden wurden, lediglich acht relevante vorgestellt. Die vollständige Liste inklusive der Produkte anderer Anwendungsbereiche ist in Anhang B zu finden.

Nicht für alle Datenlogger konnte in Erfahrung gebracht werden, wie viele Messdaten diese zwischenspeichern können. Aus diesem Grund wird auf Basis der Speichergröße und des datenspezifischen Speicherbedarfs anderer marktüblicher Geräte die Anzahl theoretisch speicherbarer Messwerte berechnet. Wie Tabelle 7 (S.44) entnommen werden kann, unterscheidet sich die Anzahl der Messwerte, die pro GB auf den Geräten gespeichert werden können, stark (bis zu Faktor 8,75). Aus diesem Grund wird das arithmetische Mittel aller Werte gebildet, das bei 53.646.400 Messwerten pro GB liegt.

Tabelle 7 - Größe des internen Speichers und Anzahl der speicherbaren Messwerte marktüblicher Datenlogger

Hersteller	Gerätename	Messwerte	Speicher [GB]	Messwerte / GB
Ammonit Measurement GMBH	*Meteo 40 Plus*	32.000.000	2	16.000.000
Delphin Technology AG	*Expert Logger*	420.000.000	14	30.000.000
ELPRO-BUCHS AG	*ECOLOG-PRO Base*	2.371.680.000	120	19.764.000
manageE GmbH	*mE180*	279.936.000	2	139.968.000
Onset Computer Corporation	*HOBO RX3000*	2.000.000	0,032	62.500.000

Zwar sorgt die Anzahl theoretisch speicherbarer Messwerte für eine bessere Vergleichbarkeit, jedoch sollten die Werte zur Darstellung der Speichermenge mit Blick auf die am Anfang des Kapitels aufgestellten Kriterien noch weiter bewertet werden. Eines dieser Kriterien besagt, dass die gemessenen Informationen für mindestens sechs Monate abgespeichert werden sollen. Deshalb wird berechnet, für wie viele Tage theoretisch Messdaten auf den Geräten gespeichert werden könnten. Zu diesem Zweck wird angenommen, dass der Datenlogger die Daten von 30 verschiedenen Geräten aufnimmt, die jeweils fünf Messwerte liefern. Die mögliche Speicherdauer in Tagen (mSD) wird dann wie folgt berechnet:

$$mSD = \frac{Anzahl\ [Messwerte]}{30\ [Geräte] * 5\ \left[\frac{Messwerte}{Gerät * Sekunde}\right] * 60\left[\frac{Sekunden}{Minute}\right] * 60\left[\frac{Minuten}{Stunde}\right] * 24\left[\frac{Stunden}{Tag}\right]}$$

Bei einem internen Speicher von 16 GB lassen sich auf Basis dieser Annahmen und Berechnungen also circa 860 Millionen sekündliche Messwerte abspeichern, was einer Speicherkapazität von fast 70 Tagen Messdaten entspricht.

5.2.1 7KN POWERCENTER 3000 V1 (Siemens AG)

Das *7KN POWERCENTER 3000 V1* der *Siemens AG* entspricht den Vorgaben der *ISO 50001* und ist hauptsächlich für die Datenaufnahme in Verbindung mit Geräten der *SENTRON*-Reihe des Herstellers gedacht. Mithilfe der zwei *Ethernet*-Schnittstellen, die über das *Modbus-TCP*-Protokoll kommunizieren können, ist es auch möglich, andere Hardwarekomponenten anzusteuern. Als Betriebssystem kommt eine *Linux*-Distribution zum Einsatz. Das *7KN*

POWERCENTER 3000 V1 kann in einem Temperaturbereich zwischen 0 und 55 °C verwendet werden und besitzt 32 GB internen Speicher, der nicht erweiterbar ist. Er ermöglicht die Speicherung von Messdaten in zehn-sekündlicher Auflösung für einen Tag und von 15-minütigen Werten für bis zu 14 Monate. Aufgrund der Abmessungen von 85x85x41 mm und der Befestigungsmöglichkeit an einer Hutschiene eignet sich das Gerät für den Einbau in einem industriellen Schaltschrank. Ob bei einem Netzwerkausfall versendete Daten, die nicht auf dem Server angekommen sind, erneut versendet werden, konnte nicht in Erfahrung gebracht werden. [37]

5.2.2 BDL 50 (Berg GmbH)

Die *Berg GmbH* besitzt mit dem *BDL 50* einen Energie-Datenlogger im Portfolio, der mithilfe verschiedener Schnittstellen, von denen zwei *Ethernet*-Anschlüsse sind, in der Lage ist, Messdaten von maximal 50 Geräten aufzunehmen. Dabei wird unter anderem die Kommunikation via *M-Bus* und *Modbus-RTU* ermöglicht. Betrieben wird der *BDL 50* mit einem *Linux*-System (*OSADL*) bei möglichen Temperaturen zwischen 0 und 60 °C. Der Einbau in einen Schaltschrank ist aufgrund der geringen Abmessungen von 106,3x62x76,5 mm und einem Hutschienenadapter problemlos möglich. Der interne Speicher beträgt 16 GB und ermöglicht auf Basis der im Kapitel 5.2 dargestellten Berechnung des durchschnittlichen spezifischen Speicherbedarfs von Messdaten marktüblicher Geräte die Abspeicherung von fast 860 Millionen Messwerten. Die Abtastrate kann zwischen einminütig und 15-minütig variiert werden, was zwar bedeutet, dass Daten von hochgerechnet fast 130 Monaten auf dem Gerät gespeichert werden können, was aber auch relativ ungenau ist. Bricht die Verbindung des *BDL 50* zum Netzwerk ab, werden die Daten nachversendet, sobald das Gerät wieder online ist. [38]

5.2.3 Controller PFC200 (WAGO GmbH & Co. KG)

Ein Datenlogger der *WAGO GmbH & Co. KG* ist der *Controller PFC200*, der vielfältige Kommunikationsmöglichkeiten mithilfe verschiedener modular hinzufügbarer Schnittstellen und diverser Protokolle bietet. Auch die Möglichkeit zwei *Ethernet*-Anschlüsse zu nutzen ist gegeben. Mit einer Größe von 100x78,6x71,9 mm und der Eigenschaft an einer Hutschiene montiert werden zu können ist die Eignung für Schaltschränke gegeben. Der *PFC200* läuft mit *Linux* und kann in einem Temperaturbereich von 0 bis 55 °C betrieben werden. Mit einem internen Speicher von 36 GB kann er hochgerechnet mehr als 1,9 Milliarden sekündliche Messwerte abspeichern, die in einem Zeitraum von nicht ganz fünf Monaten entstehen. Wird der Speicher um die maximal mögli-

chen 32 GB erweitert, können Daten von über neun Monaten gespeichert werden, was dann auch den hier zugrunde gelegten Anforderungen entspricht. Sowohl der Nachversand der Messdaten als auch die Abtastrate, die ab etwa 20 Millisekunden frei einstellbar ist, müssen selbst durch den Nutzer programmiert werden. [39]

5.2.4 EasyNet (4next Technology Systems)

Ein Datenlogger des eher unbekannten Unternehmens *4next Technology Systems* ist der *EasyNet*, der eines der wenigen Geräte ist, die über ein integriertes *SIM*-Karten-Modul verfügen. Zusammen mit einer Schnittstelle für *Modbus-RTU* und einem *Modbus-TCP*-fähigen *Ethernet*-Anschluss kann er vielfältig eine Verbindung zu einem Server für die Datenspeicherung aufbauen. Der *EasyNet* basiert auf einer Mikrocontroller-Architektur und nutzt ein *FreeRTOS*-Betriebssystem. Der Einbau in einen Schaltschrank ist durch die Größe von 90x70x65 mm und eine Hutschienenbefestigung möglich. Ein weiter Punkt, der die Industrietauglichkeit des Gerätes bestätigt, sind die möglichen Umgebungstemperaturen zwischen -20 bis +60 °C. Der interne Speicher in der Höhe von 64 GB ermöglicht hochgerechnet die Messwertspeicherung von etwa 3,4 Milliarden Messwerten, die in nicht ganz neun Monaten abgefragt werden. Damit ist der *EasyNet* das Gerät mit der höchsten Speicherkapazität von Messwerten. Die Abtastrate lässt sich frei zwischen einer Sekunde und einer Stunde einstellen. Was bei fehlender Netzwerkverbindung mit nicht versendeten Daten passiert, ist nicht klar. [40]

5.2.5 Expert Logger (Delphin Technology AG)

Der *Expert Logger* der *Delphin Technology AG* besitzt ebenso wie das zuvor beschriebene Gerät von *4Next* sowohl zwei *Ethernet*-Ports als auch ein *SIM*-Karten-Modul. Zusätzlich ermöglichen eine Wireless-LAN-Antenne, analoge sowie digitale Eingänge und verschiedene Protokolle die vielfältige Aufnahme von Messdaten. Betrieben wird der *Expert Logger* mit einer *Linux-Debian*-Distribution. Mit Abmessungen von 217x88x128 mm ist er größer als der Durchschnitt der betrachteten Geräte, ermöglicht aber trotzdem aufgrund einer passenden Vorrichtung die Hutschienenmontage in einem Schaltschrank. Dabei liegt der mögliche Temperaturbereich für den Betrieb bei -20 bis +50 °C. Der interne Speicher von 14 GB erlaubt das Ablegen von bis zu 420 Millionen Messwerten, die im Fall einer unterbrochenen Netzwerkverbindung nach deren Wiederherstellung nachgesendet werden. Diese Datenmenge entspricht Messungen von etwas mehr als 32 Tagen und ist damit weit unter der angestrebten Größe.[41]

5.2.6 gridBox (gridX GmbH)

Die Firma *gridX* vertreibt mit der *gridBox* einen kompakten Datenlogger (110x110x30 mm), der zwar nicht über die Möglichkeit der Hutschienenmontage verfügt, dafür aber 33 unterschiedliche Hersteller unterstützt. Die Kommunikation läuft standardmäßig über einen *Ethernet*-Anschluss und kann optional um weitere Schnittstellen wie *RS232* und *RS485* erweitert werden. Basis des Gerätes ist ein eigens entwickeltes Betriebssystem namens *gridOS*, dem *Linux-Yocto* zugrunde liegt. Die *gridBox* kann bei Temperaturen zwischen 0 und 40 °C betrieben werden und auf dem internen Speicher von 8 GB Daten von bis zu 10 Tagen speichern. Kommt es zu einem Netzausfall, wird die Übertragung nicht versendeter Daten automatisch neu angestoßen. Gespeichert werden können die Messwerte im Millisekunden-Bereich. [42]

5.2.7 HOBO RX3000 (Onset Computer Corporation)

Ein weiteres Gerät, das zwar keine Hutschienenmontage ermöglicht, aber über ein *SIM*-Karten-Modul verfügt, ist der *HOBO RX3000* der *Onset Computer Corporation*, die hier ein eigenes Betriebssystem verwendet. Weitere Möglichkeiten für eine Netzwerkkonnektivität bieten eine Wireless-LAN-Antenne und ein *Ethernet*-Anschluss. Maximal 15 Datenkanäle kann das Gerät, dessen Abmessungen mit 186x181x118 mm groß sind, verarbeiten, und lediglich zwei Millionen Messwerte abspeichern. Ein Vorteil ist die Möglichkeit, eine interne Batterie zu verwenden, die das Gerät bei einem Stromausfall unterbrechungsfrei mit Energie versorgt. Bei Umgebungstemperaturen zwischen -40 und +60 °C kann der *HOBO RX3000* Messdaten in sekündlicher, minütlicher oder stündlicher Auflösung aufnehmen. Wird die Netzwerkverbindung unterbrochen, werden die nicht versendeten Messdaten automatisch nachgesendet. [43]

5.2.8 MEMORY HiLOGGER LR8450 (HIOKI E.E. Corporation)

Der *MEMORY HiLOGGER LR8450* der *HIOKI E.E. Corporation* ist mit 272x145x43 mm nicht nur das größte der hier vorgestellten Geräte, sondern verfügt auch als einziges über ein Display zur direkten Anzeige der Messdaten. Bis zu 120 Kanäle stehen zur Verfügung, die Messungen über einen *Ethernet*-Port, Wireless-Lan sowie digitale und analoge Eingänge ermöglichen. Das Gerät sieht keine Hutschienenmontage vor, kann zwischen -10 und +50 °C betrieben werden und ermöglicht hochgerechnet die Speicherung von circa 1,3 Milliarden Messwerten, die Messdaten von etwas über drei Monaten

entsprechen. Eingestellt werden kann die Aufzeichnungsrate mit 22 Möglichkeiten zwischen zehn Millisekunden und einer Stunde. Ähnlich wie der *HOBO RX3000* besitzt auch der *MEMORY HiLOGGER LR8450* eine interne Batterie, die das Gerät für bis zu vier Stunden mit Strom versorgen kann. Informationen bezüglich des Betriebssystems und des Datenversands bei Netzwerkausfall konnten nicht in Erfahrung gebracht werden. [44]

5.2.9 Tabellarischer Abgleich marktüblicher Datenlogger mit den Bewertungskriterien

Nachdem die einzelnen Datenlogger kurz vorgestellt wurden, ist zur besseren Übersicht in Tabelle 8 (S.49) ein Abgleich der einzelnen Geräteeigenschaften mit den Bewertungskriterien auf Basis der gegebenen Informationen zu finden.

Wie der Tabelle entnommen werden kann, erfüllen einige Geräte den Großteil der Anforderungen und bieten teilweise sogar zusätzliche Vorteile wie zum Beispiel ein Display oder eine verbaute Batterie für eine unterbrechungsfreie Spannungsversorgung. Es ist jedoch keiner der gefundenen Datenlogger in der Lage, alle festgelegten Kriterien zu erfüllen. Die Geräte mit den höchsten Übereinstimmungen sind der *CONTROLLER PFC200* von *WAGO* und der *EasyNet*-Datenlogger der Firma *4Next Technology Systems*.

Tabelle 8 - Übersicht der vorgestellten Datenlogger und Vergleich mit Bewertungskriterien

Name des Produktes	**Unternehmen**	**Abmessungen kleiner als: 200x150x100mm**	**Hutschie-nenmon-tage**	**Zwei Ether-net-Ports**	**SIM-Kar-ten-Adapter**	**Linux als Betriebs-system möglich**	**Betriebstem-peratur erfüllt Industriestan-dard (-20 bis +60 °C)**	**Speicherung von Daten für min. 6 Monate**	**Nachversand der Daten bei Netzwerkausfall**
7KN POWER-CENTER 3000 V1	*SIEMENS*	Ja	Ja	Ja	Nein	Ja	Nein	Nein	*unklar*
BDL 50	*Berg GmbH*	Ja	Ja	Ja	Nein	Ja	Nein	Nein	Ja
CONTROL-LER PFC200	*WAGO*	Ja	Ja	Ja	Nein	Ja	Nein	Ja	kann programmiert werden
EasyNet	*4Next Technology Systems*	Ja	Ja	Nein	Ja	Nein	Ja	Ja	*unklar*
Expert Logger	*Delphin Technology AG*	Nein	Ja	Ja	Nein	Ja	Nein	Nein	Ja
gridBox	*gridX*	Ja	Nein	Nein	Nein	Ja	Nein	Nein	Ja
HOBO RX3000	*ONSET*	Nein	Nein	Nein	Ja	Nein	Ja	Nein	Ja
MEMORY Hi-LOGGER LR8450	*HIOKI*	Nein	Nein	Nein	Nein	*unklar*	Nein	Nein	*unklar*

5.3 Vorstellung marktüblicher Systeme für die Energiedatenerfassung und -analyse

Für den Fall, dass das Projekt der Firma *manageE* umgesetzt werden sollte, werden nun auch marktübliche Systeme vorgestellt, die die Energiedatenerfassung ermöglichen. Da Informationen zu solchen Systemen auf den Internetauftritten der Hersteller meist nur in geringem Maße vorhanden sind, wird vermutet, dass genauere Details zu der Systemstruktur nur an potenzielle Kunden weitergegeben werden. Anfragen bei unterschiedlichen Unternehmen bestätigen diese Vermutung teilweise. Zudem wird bei Kontaktaufnahme in manchen Fällen darauf verwiesen, dass kein weiterer Informationsaustausch stattfindet, da *manageE* ein ähnliches Lösungsspektrum beziehungsweise Wettbewerbsprodukte zu den Systemen der jeweiligen Unternehmen anbietet.

Grundlage von Systemen für das Energiemonitoring und die Messdatenverarbeitung ist - wie in vorherigen Kapiteln bereits erwähnt - eine Hardwarestruktur. Diese ist in der Regel ähnlich aufgebaut und besteht aus einzelnen Energiezählern, die direkt an die Anlagen angeschlossen sind, und einer zentralen Hardware, die vor allem für die Übertragung der Messdaten in eine Online-Datenbank zuständig ist. Zu diesem Zweck greifen vor allem kleinere Unternehmen häufig auf Hardware-Lösungen anderer Hersteller zurück, zu denen beispielsweise *Siemens* gehört.

In den nachfolgenden Kapiteln werden Systeme für die Energiedatenerfassung und -analyse von sieben verschiedenen Unternehmen vorgestellt. Eine tabellarische Zusammenfassung, die Systeme weiterer Hersteller enthält, ist in Anhang C zu finden.

5.3.1 System der Beckhoff Automation GmbH & Co. KG

Ein Hersteller, der Kunden eine Komplettlösung aus dem eigenen Hause anbietet, ist *Beckhoff*. Als Grundlage der Datenaufnahme dienen verbaute Zähler für beispielsweise Strom oder Gas, bei denen mithilfe von Stromwandlern oder Energiemessklemmen Messdaten abgegriffen werden. Diese Daten können dann unter anderem mit dem sogenannten IoT-Koppler *EK9160* über verschiedene digitale und analoge Standard-*EtherCat*-Klemmen gesammelt und an Datenbanken und Server weitergeschickt werden. Softwaretechnisch bietet *Beckhoff* hier die Verwendung von *TwinCat* an, das punktuelle oder kontinuierliche Datenanalysen, die automatische Erzeugung von Quellcode und die Übertragung an IoT-Plattformen wie *Microsoft Azure* ermöglicht. Kommt es zu einem Ausfall des Netzwerkes, können bestehende Algorithmen genutzt werden, um Daten zwischenzuspeichern und diese zu einem späteren Zeitpunkt

zu versenden. Auch wenn *TwinCat* die Weiterleitung von Datenströmen an Plattformen ermöglicht, auf denen weitere Analysen stattfinden können, bietet es von Haus aus keine Mechanismen und Methoden für eine prädiktive Regelung an. [45]

5.3.2 System der Berg GmbH

Ebenso wie *Beckhoff* bietet auch die *Berg GmbH*, deren *BDL 50* bereits in Kapitel 5.2.2 vorgestellt wurde, ein Komplett-Portfolio für die Aufnahme energetischer Messdaten und deren Analyse an. So können für die Datenaufnahme an Anlagen beispielsweise die *Skalar.pro Universal* Datenlogger genutzt werden, deren Daten mit dem *BDL 50* gesammelt werden. Das hauseigene Software-Tool zur Auswertung der Messdaten heißt *Efficio* und stellt dem Anwender verschiedene Funktionen zur Verfügung. So ermöglicht es zum Beispiel die Erstellung verschiedener Diagramme und Berichte, Zustandsanalysen und die Festlegung von Kennzahlen. Auch hier sind jedoch prädiktive Analysen und Regelungen nicht Teil des Produktumfangs. [46]

5.3.3 System der Delphin Technology AG

Ergänzend zur Nutzung des in Kapitel 5.2.5 beschriebenen *Expert Loggers* als zentrale Hardware für die Aggregierung der gemessenen Werte können Geräte der *Loggito*-Serie der *Delphin Technology AG* genutzt werden. Diese erlauben dem Kunden, Messdaten von dezentralen Anlagen aufzunehmen, zu denen keine direkte Verbindung mit dem *Expert Logger* hergestellt werden kann. Neben der Weiterleitung der Daten ermöglichen die *Loggito*-Produkte auch das Zwischenspeichern von 240 Millionen Messwerten. [47] Die Auswertung der gesammelten Informationen erfolgt schlussendlich über die *ProfiSignal 20*-Software der *Delphin Technology AG*. Das Programm besitzt allerdings nur eine kleine Auswahl an Auswertungsmöglichkeiten, zu denen die Darstellung von historischen und Live-Messdaten und die Visualisierung von Anlagen und Abläufen gehören. Die Möglichkeit, Prognosen unter Berücksichtigung von beispielsweise Klimadaten zu erstellen, ist ebenso wenig gegeben wie eine Regelung der Anlagen auf Basis solcher Prognosen. [48]

5.3.4 System der Eaton Corporation

Die weltweit tätige *Eaton Corporation* gibt auf ihren Webseiten an, dass auch sie im Bereich der Erfassung und Verarbeitung energetischer Messdaten tätig ist und kundenindividuelle Lösungen anbietet. Wie genau diese Lösungen aussehen und welche Hardware-Produkte zum Einsatz kommen, geht jedoch weder aus den Informationen der Webseiten noch aus verfügbaren Dokumenten

hervor. Es wird allerdings beworben, dass verschiedene Verbraucher (z.B. von Strom, Gas, Druckluft) eingegliedert und analysiert werden können. Auf Anfragen zu detaillierteren Informationen gab es keine Reaktion. [49] Ausführlicher wird die Energiemanagementsoftware von *Eaton* vorgestellt: Das Tool mit dem Namen *Eaton EPMS* bietet neben Energiemonitoring-Funktionen, die Analyse historischer Daten zur Entwicklung von Trends unter anderem auch Prognosen zum Geräteverhalten und Kapazitätsplanungen. Damit bietet *Eaton* als einziges der hier beschriebenen Unternehmen prädiktive Analysen an. Ob auf Basis dieser Betrachtungen die gemessenen Anlagen und Geräte geregelt werden können, ist jedoch unklar. [50]

5.3.5 System der OPTENDA GmbH

Ein Tochterunternehmen der *STEAG Solar Energy Solutions GmbH* ist die *OPTENDA GmbH*, die sich vor allem mit Energiemanagement auseinandersetzt. Messkonzepte zur Aufnahme energetischer Daten werden kundenindividuell erstellt und beinhalten eine sogenannte *Energy Box* als zentrale Einheit. Diese kann mithilfe verschiedener Schnittstellen und Protokolle alle benötigten Energie- und Betriebsdaten aufnehmen und an Server weiterleiten. Wie die gemessenen Daten bei den Anlagen abgegriffen werden, ist unklar und wurde auf Nachfrage nicht näher spezifiziert. *OPTENDA*s Software *Energy Monitor*, die der Verarbeitung und Analyse der aufgenommenen Informationen dient, erlaubt unter anderem die Analyse von Lastgängen und Verbräuchen, die Erstellung diverser Diagramme und die Aufnahme aktueller Wetterdaten. Diese werden jedoch nicht für prädiktive Analysen, sondern lediglich zur rückwirkenden Betrachtung der Auswirkungen des Wetters auf die Verbräuche genutzt. [51]

5.3.6 System der Robert Bosch GmbH

Teil des Energie-Monitoringsystems der *Robert Bosch GmbH* ist die sogenannte *Energy Platform*, die unter anderem die Messung der Maschinenleistung, eine umfangreiche Berichtserstellung und automatisierte Analysen erlaubt. Welcher Art diese Analysen sind und wie die zu Grunde liegende Hardwarestruktur zur Aufnahme der energetischen Messdaten aufgebaut ist, ist jedoch unklar. [52] Die Herausgabe weiterer Informationen wurde auf Nachfrage abgelehnt, da *manageE* ein ähnliches Lösungsspektrum beziehungsweise Wettbewerbsprodukte zur *Energy Platform* und den Energieeffizienzlösungen von *Bosch* anbietet. Dass die entwickelte Softwarelösung zusammen mit einem individuellen Messkonzept und unter Anwendung *künstlicher Intelligenz* von großem Erfolg geprägt ist, zeigt ein Bericht des Unternehmens. Dieser

beschreibt die Einführung der Software zur Verbesserung des Energiemanagements eines Werkes in China. [53]

5.3.7 SENTRON-Reihe der Siemens AG

Das Produktportfolio der *Siemens AG* besitzt mit den Lösungen der *SENTRON*-Reihe eine eigene Sparte für die Energiedatenerfassung. Hierzu gehört auch das in Kapitel 5.2.1 vorgestellte *7KN POWERCENTER 3000 V1*, das die Messdaten anderer *SENTRON*-Produkte empfängt, verarbeitet und an Datenbanken weiterleitet. Abgestimmt auf die eigenen Hardware-Komponenten bietet *Siemens* die ebenso unter dem Namen *SENTRON* vertriebenen Softwareprodukte *powermanager* und *powermind* an. Neben dem Versand der gemessenen Daten an Cloud-Lösungen (z.B. *Microsoft Azure*) bieten diese Programme diverse Funktionen für die Verwaltung des Messsystems und die Visualisierung und Analyse der energetischen Messdaten. So dient der *SENTRON powermanager* beispielsweise dazu, das System einzurichten, Geräte anzulegen, die Gerätedaten grafisch darzustellen und die Datenverarbeitung in Berichtsform. Mit *SENTRON powermind* bietet das Unternehmen hingegen eine Möglichkeit, um Energieverbräuche zu vergleichen, Referenzperioden festzulegen und „Condition-Monitoring“ zu betreiben. Auch wenn das Gesamtsystem in der Lage ist, verschiedene Messdaten aufzunehmen und zu verarbeiten, werden keine Prognosen über zukünftige Bedarfe und Verbräuche erstellt. [54] Hier würde sich das von *manageE* angedachte Projekt durch Verwendung der prädiktiven Regelung abheben.

5.3.8 Zusammenfassung zur Recherche marktüblicher Systeme für die Energiedatenerfassung und -analyse

Wie aus der Vorstellung marktüblicher Systeme für die Energiedatenerfassung und -analyse hervorgeht, gibt es auf Basis der zur Verfügung gestellten Informationen aktuell kein Produkt der betrachteten Hersteller, das eine prädiktive Regelung ermöglicht. Allein das System von *Eaton* ermöglicht die Erstellung von Prognosen, die jedoch nach den vorhandenen Informationen nicht für eine automatische Regelung genutzt werden. Ob Produkte wie die *Energy Platform* von *Bosch* in diesem Punkt bereits ausgereifter sind, ist aufgrund der wenigen Informationen, die recherchiert werden konnten, nicht ersichtlich.

6 DEFINITION UND ERSTELLUNG EINER HARDWARE-PLATTFORM

Bevor der nächste Abschnitt des Hauptteils die tatsächliche Programmierung der angestrebten Lösung behandelt, steht im folgenden Kapitel zunächst die Definition der Hardware im Vordergrund. Dies beinhaltet neben einer Beschreibung der Anforderungen auch - basierend darauf - die Auswahl einer geeigneten Hardware. Dabei werden für diesen Schritt nicht nur Auswahlkriterien, sondern auch die Bewertung der Leistungsfähigkeit unterschiedlicher Geräte definiert. Im Anschluss daran erfolgt die Entscheidung für ein Computer-Modell, das ohne Betriebssystem ausgeliefert werden muss. Deshalb schließt das Kapitel mit der Vorbereitung des Gerätes, die neben der Installation eines Betriebssystems auch dessen Einrichtung beinhaltet.

6.1 Anforderungen an die Hardware

In Kapitel 5.1 erfolgte bereits ein erster Blick auf die Anforderungen, die an die HSP gestellt werden. Diese entsprechen bereits zum Großteil den Voraussetzungen, die allein an die Hardware gestellt werden. Bereits erwähnt wurden folgende Punkte: das Betriebssystem, die Industrietauglichkeit, Anschlussmöglichkeiten für zwei Netzwerke, die Speicherung von Daten für sechs Monate und ein *SIM*-Karten-Adapter. Zusätzlich zu diesen Kriterien sollte die gewählte Hardware-Plattform ausreichend leistungsfähig sein, um aufwändige Berechnungen und Simulationen durchführen zu können. Zwar ist der Plan, dass die tatsächlichen Berechnungen auf einer Cloud-Plattform stattfinden, die Erfahrung von *manageE* ist jedoch, dass deutsche Unternehmen zurückhaltend sind, ihre Daten mit einer solchen Lösung abzuspeichern. Dies wird auch durch eine Umfrage von *GROHMANN BUSINESS CONSULTING* bestätigt, die besagt, dass 43 % der Unternehmen sich aus Sorge um ihre Daten gegen Cloud Computing entscheiden. [55]

Aus diesem Grund sollte es auch die Möglichkeit geben, dass die Daten lediglich lokal gespeichert werden und die Berechnungen und Simulationen vor Ort durchgeführt werden. Außerdem sollte zur einfacheren Behebung von Fehlern der Anschluss eines externen Bildschirms möglich sein. Zur besseren Übersicht sind in Tabelle 9 (S.55) noch einmal alle „Muss“-Anforderungen inklusive der Spezifikation des jeweiligen Auswahlkriteriums aufgeführt.

Tabelle 9 - „Muss“-Anforderungen an die Hardware-Plattform

Anforderung		Auswahlkriterium
Anschlussmöglichkeit an zwei Netzwerke		Zwei separat konfigurierbare *Ethernet*-Anschlüsse
Kostengünstiges und stabiles Betriebssystem		*Linux*-Distribution
Speicherung von Daten für mindestens sechs Monate		Mindestens 500 GB
Industrietauglichkeit	Montage in Schaltschrank	Hutschienenmontage
	Äußere Abmessungen	Maximal 200x150x100 mm
Leistungsfähige Komponenten	Arbeitsspeicher	Mindestens 8 GB
	Prozessor	Top 10 % marktüblicher Lösungen
	Schnelle Festplatte	Solid State Disk

Der Vollständigkeit halber sind die „Kann“-Anforderungen inklusive der Spezifikation des jeweiligen Auswahlkriteriums in Tabelle 10 aufgelistet.

Tabelle 10 - „Kann“-Anforderungen an die Hardware-Plattform

Anforderung		Auswahlkriterium
Anschlussmöglichkeit für externen Bildschirm		HDMI, VGA
Fähigkeit, an Standorten ohne direkten Zugang mit dem Internet zu kommunizieren		SIM-Karten-Adapter
Industrietauglichkeit	Betriebstemperatur	Mindestens -20 bis +60 °C
Leistungsfähige Komponenten	Grafikkarte	Dedizierte Grafikkarte

6.2 Auswahl geeigneter Hardware

Für die Auswahl einer geeigneten Hardware wurden zunächst unter anderem durch Sichtung der Online-Auftritte von Händlern einzelne Hersteller von industrietauglichen Computern recherchiert. Nachdem das Produktportfolio der Unternehmen grob anhand der angebotenen Möglichkeiten gefiltert wurde,

blieb am Ende eine Liste von 41 potenziell passenden Modellen, inklusive Varianten mit unterschiedlichen Prozessorleistungen, übrig, die von 15 verschiedenen Produzenten angeboten werden.

Im Anschluss daran erfolgte anhand der im vorherigen Abschnitt gelisteten Anforderungen eine genauere Untersuchung der Spezifikationen. Da die auf den Internetauftritten verfügbaren Informationen häufig unvollständig waren, wurden die Hersteller zur Klärung kontaktiert.

6.2.1 Bewertung der Leistungsfähigkeit

Für die Bewertung der Leistungsfähigkeit eines Computers wird in der Regel vor allem die zentrale Prozessoreinheit in Verbindung mit der Größe des Arbeitsspeichers und den Merkmalen der Grafikeinheit betrachtet. Da die Größe des Arbeitsspeichers bereits auf mindestens 8 GB festgelegt wurde und keines der recherchierten Geräte eine dedizierte Grafikkarte besitzt, wird nachfolgend lediglich das Potenzial des Prozessors betrachtet.

Nach der Zusammenführung sämtlicher Informationen wurde eine Bewertung der insgesamt 26 verschiedenen Prozessoren, die sich hauptsächlich durch ihre Taktfrequenz und die Anzahl der Prozessorkerne unterscheiden, vorgenommen. Zu diesem Zweck wurden die Datenbanken verschiedener Benchmarking-Webseiten, die unter anderem die Leistungsfähigkeit von Computer-Komponenten bewerten, genutzt. Für eine bessere Verifikation der Bewertungen wurden die Informationen der folgenden drei Webseiten verwendet und verglichen:

1. *cpubenchmark.net* (CPUB)
2. *amd-intel-cpu-benchmark.com* (AICB)
3. *browser.geekbench.com*

Da die Datenbanken von *browser.geekbench.com* nur zu knapp 58 % aller betrachteten Prozessoren Informationen zur Verfügung stellen konnten, wurden vor allem die Bewertungen der anderen beiden Quellen verwendet.

Wie der vollständigen Auflistung aller Prozessoren in Anhang D entnommen werden kann, unterscheidet sich die Performanz der diversen Prozessor-Modelle sehr stark. So ist beispielsweise die stärkste CPU etwa 60-mal so leistungsfähig wie die schwächste der betrachteten CPUs. Zu den besten 10 % der 26 verschiedenen Prozessoren gehören die folgenden Varianten, die aufsteigend nach Leistung sortiert in Tabelle 11 (S.57) dargestellt sind:

Tabelle 11 - Auflistung der Top-4-Prozessoren (Top 10 %) des Benchmarkings (aufsteigend sortiert nach Leistung)

Bezeichnung des Prozessors inklusive Anzahl der Kerne und Taktfrequenz	**CPUB [56]**	**AICB [57]**
Intel Core i3-8145UE 2x 2,2 – 3,9 GHz	4.133	-
Intel Core i5-8365UE 4x 1,6 - 4,1 GHz	5.555	6.215
Intel Core i5-8265U 4x 1,6 – 3,9 GHz	6.132	6.424
Intel Core i7-8665UE 4x 1,7 - 4,4 GHz	6.587	6.774

6.2.2 Entscheidung für ein Hardware-Modell

Nachdem sämtliche Informationen zusammengeführt und mit den Vorgaben abgeglichen wurden, konnten die folgenden drei Computer in die Endauswahl genommen werden:

1. *UP Xtreme Edge* (*Intel Core i3-8145UE*)
2. *UP Xtreme Lite* (*Intel Core i5-8365UE*)
3. *UP Xtreme Lite* (*Intel Core i7-8665UE*)

Dabei wird der Computer, in dem der *Intel Core i5-8265U (*4x 1,6 – 3,9 GHz) verbaut wurde, nicht in der engeren Auswahl betrachtet, da er keine Hutschienenmontage vorsieht. Jedes der drei Modelle in der Endauswahl stammt aus dem Haus des Herstellers *AAEON Technology Inc.* und ist in der Lage, sowohl *Windows* als auch *Linux* und teilweise sogar *Android* als Betriebssystem zu nutzen. Inklusive eines passenden Chassis liegen die äußeren Abmessungen bei 190x129x78,6 mm. Der interne Speicher variiert zwischen 8 und 16 GB und ist mithilfe einer *Solid State Disk*-Festplatte erweiterbar, sodass sogar die Abspeicherung von mehreren 1.000 GB möglich ist. Die Geräte verfügen alle über die Möglichkeit, eine *SIM*-Karte zur Herstellung einer mobilen Internetverbindung zu verwenden und besitzen jeweils zwei separat konfigurierbare *Ethernet*-Anschlüsse und einen *HDMI*-Anschluss. Auch die Hutschienenmontage und der Einbau einer zusätzlichen Grafikeinheit, die noch einmal zu einer Leistungssteigerung führt, ist bei allen drei Computern möglich. [58, 59] Die zwei wichtigsten Unterschiede liegen in der Betriebstemperatur und dem verwendeten Prozessor, wie Tabelle 12 (S.58) entnommen werden kann.

Tabelle 12 - Passende Hardware-Modelle in der Endauswahl

Name	Prozessor [58, 59]	CPU-Benchmark [56]	Betriebstemperatur [58, 59]
UP Xtreme Edge	*Intel Core i3-8145UE 2x 2.2 - 3.9GHz*	4.133	-20 bis +70 °C
UP Xtreme Lite	*Intel Core i5-8365UE 4x 1,6 - 4,1 GHz*	5.555	0 bis +60 °C
UP Xtreme Lite	*Intel Core i7-8665UE 4x 1,7 - 4,4 GHz*	6.587	0 bis +60 °C

Bei den Geräten der *UP Xtreme Lite*-Serie handelt es sich um Einplatinencomputer, die standardmäßig vom Hersteller ohne Gehäuse geliefert werden. Es besteht allerdings die Möglichkeit ein zusätzliches Chassis zu erwerben, das je nach Ausführung auch eine aktive Kühlung mithilfe von Lüftern erlaubt, weshalb die Computer auch bei höheren Temperaturen betrieben werden können. Dadurch und aufgrund einer gewissen thermischen Isolierung, die das Gehäuse bietet, wird angenommen, dass die Betriebstemperatur beim Einbau in einem Schaltschrank auch bei *UP Xtreme Lite*-Modellen ausreichend ist. Da es sich bei einer Betriebstemperatur im Bereich von -20 bis 60 °C lediglich um ein „Kann“-Kriterium handelt, ist letztendlich die Rechenleistung der Geräte ausschlaggebend. Hier wird durch Vergleich des CPU-Benchmarks deutlich, dass der *UP Xtreme Lite*, in dem der *Intel Core i7-8665UE* verbaut ist, mit Abstand das leistungsfähigste Modell der betrachteten Computer ist. Da er somit bis auf ein „Kann“-Kriterium alle Anforderungen erfüllt, fiel die Wahl der Hardware auf dieses Gerät, das im Anschluss an diese Entscheidung bestellt wurde.

Aufgrund des während der Umsetzung dieser Arbeit vorherrschenden Chipmangels, der auch zu Lieferengpässen des Herstellers *AAEON* führte, wurde die Hardware allerdings nicht rechtzeitig zur Verfügung gestellt. [60] Deshalb wurde eine ältere Version des Modells *UP squared*, das aufgrund früherer Arbeiten bereits bei *manageE* vorhanden war, für die Entwicklung des Prototyps verwendet.

6.3 Vorbereitung der Hardware

Die ausgewählte Hardware wird ohne jegliche Softwarekomponenten ausgeliefert, weshalb zunächst ein Betriebssystem ausgewählt, installiert und eingerichtet werden muss. Erst im Anschluss daran können Programme aufgespielt werden, die für die Nutzung des Gerätes benötigt werden.

6.3.1 Installation des Linux-Betriebssystems

Für die Programmierung eines *Linux*-Computers mit *CODESYS* muss zunächst ein Laufzeitsystem, das *CODESYS Control for Linux SL* heißt, auf dem zu programmierenden Gerät installiert werden. Dieses System unterstützt lediglich *Debian*-basierte *Linux*-Versionen, weshalb ein solches Betriebssystem ausgewählt und installiert werden muss. [61]

Debian ist wie die meisten anderen *Linux*-Distributionen ein kostenfreies Betriebssystem, das von der *Debian*-Organisation zur Verfügung gestellt wird. Installiert werden kann das System mithilfe verschiedener Datenträger wie beispielsweise *Universal Serial Bus*-Massenspeicher. [62] Da die Installation von *Debian* gerade für bisherige *Windows*-Nutzer ungewohnt sein kann, empfiehlt sich die Verwendung einer Anleitung in deutscher Sprache, die beispielsweise von Frank Rosenberger zur Verfügung gestellt wird. [63] Diese wurde auch für die Umsetzung dieser Arbeit verwendet.

6.3.2 Einrichtung des Betriebssystems

Nach der erfolgreichen Installation des *Debian*-Betriebssystems sollte dieses eingerichtet und das Gerät vor fremden Zugriffen geschützt werden. Auch wenn *Linux*-Geräte prinzipiell als besonders sicher gelten, ist das Vornehmen von ein paar Anpassungen sinnvoll. So sollte zum Beispiel möglichst nach dem Abschluss der Installation ein sogenanntes „Root-Passwort“ vergeben werden, das den administrativen Zugriff schützt, denn in den Werkseinstellungen ist hier keine Eingabe definiert. [64]

Eine praktische Hilfe bei der initialen Einrichtung und auch für spätere Änderungen und Anpassungen während des Betriebs ist die Installation des sogenannten *sudo*-Paketes. Das Wort „sudo“ steht für „SuperUser DO“ und erlaubt Anwendern die Nutzung zugriffsbeschränkter Anwendungen und Dateien. Während andere *Linux*-Distributionen dieses Paket standardmäßig installiert haben, muss es auf *Debian* manuell hinzugefügt werden. [65]

Ebenfalls sinnvoll ist die Einrichtung einer sogenannten *Firewall*, die basierend auf zuvor definierten Vorgaben prüft, wenn das Gerät auf ein Netzwerk zugreift und andersherum. Dadurch wird der Computer vor Bedrohungen wie beispielsweise Virusattacken geschützt, die Datenänderungen, deren Diebstahl oder ähnliche schadhafte Handlungen beabsichtigen. [66] Diese Funktion wird bei *Linux* durch die Anwendung *IP-Tables* erfüllt, die mithilfe des *UFW*-Paketes auf einfache Art und Weise verwaltet werden kann. Die *UFW-Firewall*

ermöglicht es dabei nicht nur IP-Adressen zu blockieren, sondern auch sogenannte „Ports“, die das Zugangstor zu dem anzusteuernden Gerät darstellen, zu erlauben oder zu blockieren. [67]

Damit die Hardware später per Fernzugriff sicher angesteuert werden kann, empfiehlt es sich, auf dem Gerät die Möglichkeit einer sogenannten *Secure Shell*-Verbindung zu aktivieren. Dieses Sicherheitsverfahren ermöglicht mithilfe eines Protokolls eine geschützte Kommunikation, die zu diesem Zweck verschlüsselt wird, zwischen zwei Rechnern aufzubauen. [68] Nachdem das zur Nutzung dieser Funktion benötigte *net-tools*-Paket installiert wurde, sollte der in den Werkseinstellungen festgelegte Wert für den Zugangs-Port, der zu Beginn mit der Zahl 22 definiert ist, geändert werden. Durch diese Maßnahme wird die Wahrscheinlichkeit eines fremdartigen Zugriffs stark verringert. [64]

Zu den weiteren Anpassungen für die Verbesserung der Sicherheit des Gerätes gehören neben der Einrichtung einer E-Mail-Benachrichtigung, die bei unberechtigten Zugriffen erfolgt, beispielweise auch das Sperren von Ports, die nicht verwendet werden. Berücksichtigt werden sollte außerdem, dass nur die Softwarekomponenten installiert werden, die tatsächlich für den späteren Betrieb benötigt werden, da jedes Programm die Wahrscheinlichkeit von Sicherheitslücken erhöht. Dies gilt vor allem dann, wenn keine automatischen oder regelmäßigen Updates erfolgen. [64]

6.4 Zwischenfazit: Definition und Erstellung einer Hardware-Plattform

Aufgrund der vielfältigen Anforderungen kommt nur ein kleiner Teil der marktüblichen Hardware-Computer als Grundlage für die Erstellung der HSP in Frage. Sollte im Lauf des potenziellen Projektes von *manageE* eine Erstellung der Prognosen lokal auf der Hardware benötigt werden, sollte dies jedoch wegen der Entscheidung für den leistungsfähigen *UP Xtreme Lite* umsetzbar sein. Im Zweifelsfall kann die Leistung des Gerätes durch den Einbau einer dedizierten Grafikeinheit noch weiter gesteigert werden.

Da die Automatisierungssoftware *CODESYS* lediglich die Programmierung *Debian*-basierter *Linux*-Systeme erlaubt, wird ein solches als Grundlage für die HSP verwendet. Obwohl die Verwendung von *Linux* generell eine hohe Sicherheit gewährleistet, werden nach der Installation des Betriebssystems noch verschiedene Einrichtungsschritte wie beispielweise die Sicherung der *Secure Shell*-Verbindung vorgenommen. Dadurch kann das Risiko von schädlichen Angriffen vor allem auf Basis von Viren weiter verringert werden.

7 SOFTWAREENTWICKLUNG FÜR DIE DATENAUFNAHME UND -SPEICHERUNG

Nachdem im vorangegangenen Kapitel die Definition und Erstellung der Hardware-Plattform im Vordergrund stand, beleuchtet dieser Abschnitt die softwaretechnischen Komponenten, die den Großteil dieser Arbeit ausmachen. Dabei werden vorab die Ziele und Anforderungen definiert, die als Grundlage für die Erstellung der Softwarelösung dienen. Aufbauend auf diesen werden marktführende Produkte und Anbieter für IoT-Plattformen und die Entscheidung für eine der Lösungen vorgestellt.

Es folgen die Unterkapitel, die sich mit dem Aufbau und der Erstellung der unterschiedlichen Programme und deren Implementierung auf der HSP befassen. Für die Umsetzung der geplanten Programmierung und Funktionen werden zusätzlich zu den bereits installierten Programmen weitere Softwarekomponenten sowohl auf dem sogenannten „Entwicklungsrechner“ als auch auf der HSP benötigt. Als „Entwicklungsrechner“ wird dabei der Computer bezeichnet, über den der Zugriff und die Programmierung der HSP erfolgen. Welche Komponenten für welchen Teil der aufgebauten Softwarestruktur vorausgesetzt werden, wird in dem jeweiligen Unterkapitel geklärt.

7.1 Klärung der Vorgaben

Vor der Programmierung einer Software-Lösung werden in aller Regel als erstes die Projektziele und in einem nächsten Schritt die detaillierten Anforderungen geklärt. Die genaue Vorgehensweise hierzu wurde bereits ausführlich in Kapitel 3.2.1 beschrieben, weshalb hier nun der Fokus auf die Beschreibung der Anforderungen und Ziele gelegt wird. Da es sich bei dieser Arbeit lediglich um die Vorbereitung für ein potenzielles Projekt und nicht bereits um ein solches handelt, wurden im Rahmen dieser Ausarbeitung weder ein Lasten- noch ein Pflichtenheft erstellt. Die Klärung der Ziele und Anforderungen erfolgte in Form von mündlichen Absprachen, die nachfolgend zusammengefasst sind.

7.1.1 Ziele der Software-Lösung

Neben einer leichten Nachrüstbarkeit, die die Anbindung an und Verwendung von bestehenden Systemen voraussetzt, soll die softwaretechnische Lösung auch kostengünstig sein. Ebenso soll die Anpassbarkeit der Software an spezielle Komponenten wie konkret verfügbare Quellen, Senken und Speicher gegeben sein. Das Hauptziel ist dabei die lückenlose Erfassung aller lokal verfügbaren Messgrößen und Daten, die für energetische Effizienzsteigerungen relevant sind. Zusätzlich zu der lokalen Speicherung auf dem Gerät soll die

Möglichkeit realisiert werden, die Daten über eine Internetverbindung auch außerhalb des unternehmensinternen Netzwerkes abzurufen, damit wie bereits beschrieben Prognosen über eine leistungsfähige Cloud-Plattform erstellt werden können. Auf Basis dieser Prognosen soll die Steuerung der Komponenten erfolgen, deren Daten aufgenommen wurden, sodass am Ende Effizienzsteigerungen durch eine optimierte Betriebsweise erfolgen.

7.1.2 Beschreibung der Anforderungen

Nachdem im vorherigen Kapitel die Ziele beschrieben wurden, die den übergeordneten Rahmen festlegen, wird nun näher auf die Anforderungen eingegangen, die die Details der Umsetzung bestimmen. Der Übersichtlichkeit halber sind die Ziele nachfolgend noch einmal aufgeschlüsselt und in tabellarischer Form mit den detaillierten Anforderungen verknüpft (siehe Tabelle 13).

Tabelle 13 - Auflistung der Ziele und Anforderungen der zu erstellenden Software-Lösung

Nr.	Beschreibung des Ziels	Resultierende Anforderungen
1	Einfache Nachrüstbarkeit	- Fähigkeit, mit einer Vielzahl anderer Systeme kommunizieren zu können
2	Anpassbarkeit an spezielle Komponenten	- Kommunikation über das *Modbus-TCP*-Protokoll - Verarbeitung gängiger Datentypen
3	Kostengünstig	- Verwendung von *Linux* - Nutzung lizenzfreier oder kostengünstiger Softwarelizenzen
4	Lückenlose Datenerfassung	- Prüfung, ob Daten angekommen sind / Synchronisation - Wiederaufnahme des Loggens beim Neustart
5	Lokale Speicherung der Daten	- Einrichtung einer lokal verfügbaren Datenbank
6	Zugänglichkeit der Daten über das Internet ermöglichen	- Einrichtung einer Online-Datenbank - Kommunikation mit einer IoT-Datenbank in der Cloud
7	Regelung der Komponentensteuerungen	- Fähigkeit, Sollwerte zu empfangen und an Steuerungen zu übergeben

7.2 Auswahl einer IoT-Plattform

Wie bereits zuvor beschrieben wurde und aus den Anforderungen hervorgeht, soll eine IoT-Plattform genutzt werden, um die Messdaten abzuspeichern und für die Berechnung der Prognosen verfügbar zu machen. Da die Sparte der IoT-Technologie in immer mehr alltäglichen Bereichen Anwendung findet, hat sich inzwischen eine Vielzahl von Unternehmen mit der Entwicklung von IoT-Lösungen befasst. Darunter zählen auch diverse IoT-Plattformen, die im Grunde als Schnittstelle zwischen Hardware-Komponenten wie Datenloggern und der Analyse und Verwertung der Messdaten stehen. Auch wenn manche dieser Plattformen, die in der Regel als Webseite aufrufbar sind, auch Analyse-Möglichkeiten bieten, liegt ihr Hauptaugenmerk darauf, die Messdaten vieler IoT-Geräte zu speichern und zu verwalten. [69] Für einen besseren Überblick über die Unterschiede marktüblicher Lösungen werden im Folgenden fünf IoT-Plattformen unterschiedlicher Hersteller vorgestellt.

7.2.1 Google Cloud IoT

Der IoT-Dienst von *Google* basiert auf der *Google Cloud Plattform* und gehört zu den führenden IoT-Plattformen des Marktes. Dabei wurden verschiedene andere *Google*-Produkte in die *Google Cloud IoT* integriert, um eine breite Palette an Lösungen anzubieten. Die Funktionen der Plattform reichen neben den Grundbausteinen, die die Registrierung von IoT-Geräten und die Speicherung und Visualisierung ihrer Daten ermöglichen, von Routing-Mechanismen bis zu verschiedenen Analysemöglichkeiten. Diese sind vor allem auf große Datenmengen ausgerichtet und bedienen sich bei den Berechnungen unter anderem Verfahren *künstlicher Intelligenz*. Zusätzlich können durch die Integration der sogenannten *Google Maps Platform* die Standorte der angebundenen Geräte dargestellt werden. [69]

7.2.2 Cisco IoT Cloud Connect

Als Anbieter, der unter anderem im Bereich der Netzwerktechnik tätig ist, bietet *Cisco* mit *IoT Cloud Connect* ein Produkt an, das in den Anfängen vor allem für Mobilfunkbetreiber gedacht war. Inzwischen wurde die Plattform zu einer weit verbreiteten Lösung für generelle IoT-Anwendungen weiterentwickelt, für die *Cisco* auch passende Hardware-Komponenten anbietet. Während mit dem sogenannten *IoT Control Center* die Mobilfunkkonnektivität verwaltet und IoT-Hardware eingebunden werden kann, bietet die Funktion *Edge Intelligence* einfache Möglichkeiten für die Verarbeitung der Messdaten. Ebenso können mit weiteren Teilprodukten direkt Sensorik-Elemente integriert und Daten durch ein eigenes Sicherheits-Feature geschützt werden. [69]

7.2.3 Salesforce IoT Cloud

Da sich der Hersteller *Salesforce* auf die Verwaltung von Kundenbeziehungen spezialisiert hat, dient die *IoT Cloud* der Firma vor allem dazu die Verbindung von Unternehmen zu ihren Kunden zu stärken. Deshalb gibt es eine enge Verzahnung mit *Salesforce CRM*, das die Möglichkeit bietet, die gemessenen IoT-Daten und Alarme oder Warnungen darzustellen und Gegenmaßnahmen ergreifen zu können. Dabei sind durch Verwendung einfacher Bedienoberflächen keine Programmierkenntnisse zum Beispiel für die Erstellung von Regelmechanismen notwendig. [69] Damit ist die *Salesforce IoT Cloud* ein in sich geschlossenes Produkt, das sich in das Portfolio des Herstellers eingliedert. Die Verwendung zur Bereitstellung und Weiterleitung von Messdaten für andere Plattformen ist nicht Teil des Produktes, weshalb es für die Umsetzung dieser Arbeit nicht von Interesse ist.

7.2.4 Amazon AWS IoT Core

Ebenfalls führend im Bereich der IoT-Anwendungen ist *Amazon* mit *AWS IoT Core*, das die zuverlässige und sichere Verwaltung einer Vielzahl an IoT-Geräten ermöglicht. Dabei bietet das Produkt nicht nur die Verwendung verschiedener Verbindungsprotokolle, sondern ähnlich wie *Google* auch die Nutzung weiterer *Amazon*-Dienste an, die für IoT-Anwendungen nützlich sein können. Die Funktionen der integrierten Dienste beinhalten neben diversen Möglichkeiten für die Datenspeicherung, -verwaltung und -verarbeitung auch die automatisierte Ausführung von Code zum Beispiel für Verfahren des Maschinellen Lernens. [69]

7.2.5 Microsoft Azure IoT Hub

Das letzte der hier vorgestellten Produkte stammt vom Hersteller *Microsoft*, der mit dem *Azure IoT Hub* eine quelloffene IoT-Plattform anbietet, die eine Vielfalt von Lösungen anbietet. Viele der Funktionen können ohne aufwändige Einstellungen genutzt und flexibel auf die individuellen Bedürfnisse eines Unternehmens angepasst werden. Aufgrund der hohen Sicherheit, der breiten Auswahl an Funktionen für den Umgang mit großen Datenmengen und der Möglichkeit, Methoden der *künstlichen Intelligenz* für die Analyse zu nutzen, gehört der *Azure IoT Hub* ebenfalls zu den häufig verwendeten Produkten. Zudem zeichnet sich *Microsofts* IoT-Plattform durch eine hohe Skalierbarkeit aus. [69]

Der Funktionsumfang des *Azure IoT Hub* erfüllt alle Anforderungen zur Verwirklichung des von *manageE* angedachten Projektes. Außerdem ist das Unternehmen in sechs Bereichen als *Microsoft Silver Partner* zertifiziert, sodass *manageE* viele der *Microsoft*-Produkte kostenfrei oder vergünstigt nutzen kann. Zusätzlich werden an *Microsoft Silver Partner* Freikontingente für verschiedene Produkte wie beispielsweise für die Cloud-Nutzung vergeben und es existieren Vorteile beim technischen Support. Aus diesen Gründen wurde der *Azure IoT Hub* für die Umsetzung dieser Arbeit gewählt. [70, siehe Anhang E]

7.3 Aufbau und Programmierung der Software

Ebenfalls auf der Seite der Software befindet sich die Programmierung, mit der die Funktionalitäten der HSP realisiert werden sollen. Die Aufgaben, die entsprechend den Anforderungen übernommen werden sollen, können in die folgenden vier Bereiche unterteilt werden:

1. Datenaufnahme
2. Lokale Datenspeicherung
3. Datenweiterleitung an die IoT-Plattform
4. Rückversand von Sollwerten

Der theoretische Datenfluss der vier Kernaufgaben ist in Abbildung 12 schematisch dargestellt. Dabei ist zu berücksichtigen, dass die Schnittstellen sowohl für den Datenfluss zur Cloud-Plattform für prädiktive Regelung als auch für den Rückversand der Sollwerte noch unklar sind.

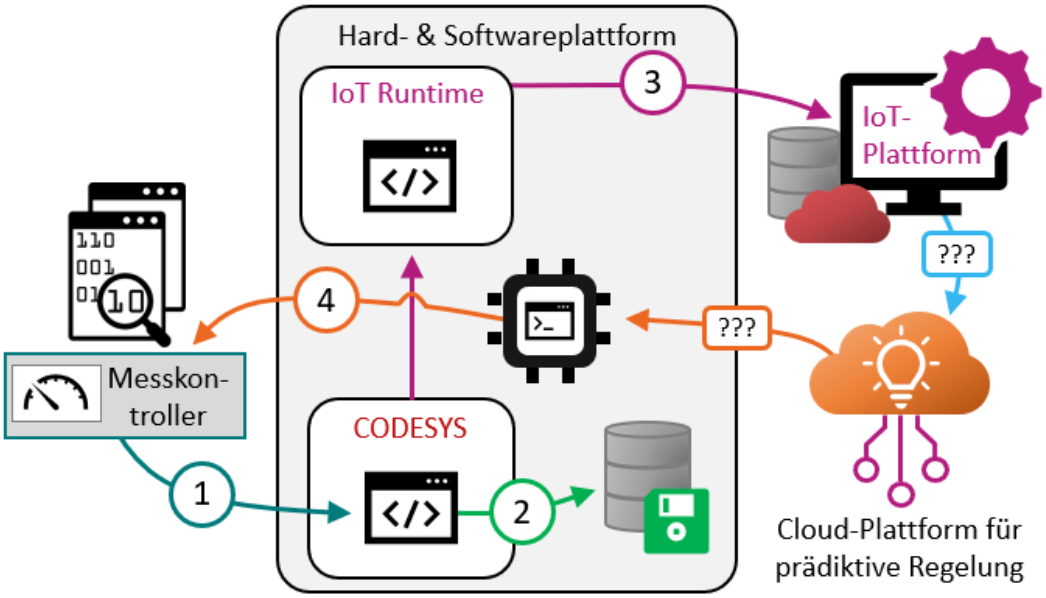


Abbildung 12 - Schematische Darstellung des Datenflusses vom Messgerät über die HSP bis in die IoT-Plattform und Rückfluss über die potenzielle Cloud-Plattform

Auf Basis dieser Unterscheidung werden nachfolgend die einzelnen Teile der Programmierung inklusive der verwendeten Softwarekomponenten vorgestellt.

7.3.1 Programmierung der Datenaufnahme

Essenziell für die Verarbeitung und Analyse von Messdaten ist zunächst ihre möglichst lückenlose Erfassung. Deshalb muss in einem ersten Schritt eine Verbindung zu den Anlagen hergestellt werden. Wie bereits in vorangegangenen Kapiteln erwähnt, gibt es verschiedene Möglichkeiten, dies zu realisieren. In den meisten Fällen existiert eine Maschinensteuerung, die mit einem Netzwerk verknüpft ist oder verknüpft werden kann. Ist dies nicht der Fall, besteht die Möglichkeit, mit intelligenten Ein- und Ausgabeeinheiten zu arbeiten, die über verschiedene Schnittstellen direkt an die Anlagen angeschlossen werden und die aggregierten Messdaten beispielsweise via *Ethernet* versenden können. Auf diesem Weg können prinzipiell alle Maschinen mit einem Netzwerk verbunden werden. Da - wie in Kapitel 3.4 beschrieben - das *Modbus-TCP*-Protokoll standardmäßig in der Industrie für automatisierte Kommunikation verwendet wird, wird dies zum Zweck der Datenaufnahme für die Umsetzung dieser Arbeit verwendet.

Bevor mit der Programmierung begonnen werden kann, müssen zum einen die *CODESYS* Software selbst und das Erweiterungspaket für die Steuerung von *Linux*-Geräten (*CODESYS Control for Linux SL*) auf dem Entwicklungsrechner installiert werden. Erst im Anschluss daran ist es möglich das *CODESYS* Laufzeitsystem auf die Hardware-Plattform zu überspielen, dort zu starten und Skripte und Regelkonzepte auszuführen.

Die Rolle der Anlagensteuerung beziehungsweise der intelligenten Ein- und Ausgabeeinheiten übernimmt für diese Arbeit ein Modell des *mE180 Messcontrollers* der Firma *manageE*. Dieses Gerät ist dazu in der Lage, die Messdaten von 18 unterschiedlichen Verbrauchern aufzunehmen und via *Ethernet* unter Verwendung diverser Protokolle zur Verfügung zu stellen. Da der Kontroller unter anderem auch als *Modbus-TCP-Slave* verwendet werden kann, eignet er sich für die Umsetzung dieser Arbeit. [71] Ein solches Messgerät ist an die Stromunterverteilung der *manageE*-Büroräume angeschlossen und nimmt elektrische Größen wie unter anderem die Spannung, den Strom und den Phasenwinkel auf.

In einem ersten Schritt wird in einem neuen *CODESYS*-Projekt, bei dessen Erzeugung das Laufzeitsystem *CODESYS Control for Linux* ausgewählt werden muss, ein neuer *Ethernet-Modbus-TCP-Master* angelegt. Dieser ermöglicht die Abfrage der Daten eines *Modbus-TCP-Slaves*, bei dem es sich um

den *mE180* Messcontroller handelt und der nun ebenfalls in *CODESYS* erzeugt und mit dem *Master* verknüpft wird. Um nun wiederum diesen *Slave*-Baustein mit dem *mE180* Messcontroller verknüpfen zu können, wird dessen IP-Adresse benötigt, die vor der Erstkonfiguration auf den Wert „192.168.1.7“ eingestellt ist und beliebig angepasst werden kann. [72, S. 26-29] Diese IP-Adresse wird nun in dem *Slave*-Baustein eingetragen. Standardmäßig ist hier für die Verbindung der Port „502“ eingetragen, der angepasst werden muss, sofern das Gerät nur über einen anderen Port erreicht werden kann.

Modbus-Schnittstellen sind so aufgebaut, dass sie die unterschiedlichen Daten in Registern abspeichern, die jeweils entweder nur gelesen, geschrieben oder gelesen und geschrieben werden können. [73] Da die Anzahl der Register bei einigen 10.000 Stück liegen kann, informiert in der Regel der Hersteller eines *TCP*-Gerätes in der Betriebsanleitung darüber, auf welchem Register welche Daten gelesen und/oder gespeichert werden können. Auf Basis dieser Angaben muss in *CODESYS* festgelegt werden, ab welcher Registernummer die Werte abgerufen werden sollen und für wie viele Folgeregister dies geschehen soll (siehe Abbildung 13). Dies wird als sogenannter „READ“- beziehungsweise "WRITE“-Offset bezeichnet.

Trigger	READ-Offset	Länge	Fehlerbehandlung	WRITE Offset	Länge
Zyklisch, t#1000ms	16#07D0	121	Letzen Wert beibehalten		
Zyklisch, t#1000ms				16#07D1	1
Zyklisch, t#100ms	16#0BB8	97	Letzen Wert beibehalten		

Abbildung 13 - Ausschnitt aus der Eingabemaske für die Offsets als Hexadezimalzahl und Anzahl der abzufragenden Register ("Länge")

Hierbei ist es gegebenenfalls notwendig verschiedene Offsets zu testen, da unterschiedliche Systeme und Softwarekomponenten mit unterschiedlichen Indizes (in der Regel 0 oder 1) für die Register starten, sodass es leicht zu einer Verschiebung kommen kann. Ebenfalls festgelegt werden können die sogenannten „Trigger“, die bestimmen, auf welche Art und wie häufig die Daten abgefragt werden sollen.

Sobald die Offsets eingetragen sind, können nun die einzelnen Register mit Variablen verknüpft werden (auch „Mapping“ genannt) sodass die Werte in Skripten verwendet werden können. Auch dies geschieht in tabellarischer Form wie in Abbildung 14 (S.68) zu erkennen ist.

Variable	Mapping	Kanal	Adres...	Typ	Beschreibung
		Data[21]	%IW21	WORD	0x07E5
MB_Phase1_V_rms		Data[22]	%IW22	WORD	0x07E6
MB_Phase1_Resolution_U		Data[23]	%IW23	WORD	0x07E7
MB_Phase1_Frequency		Data[24]	%IW24	WORD	0x07E8
MB_ch1_active_power_high		Data[25]	%IW25	WORD	0x07E9
MB_ch1_active_power_low		Data[26]	%IW26	WORD	0x07EA
MB_ch1_power_factor		Data[27]	%IW27	WORD	0x07EB
MB_ch1_resolution_i		Data[28]	%IW28	WORD	0x07EC
MB_ch1_error		Data[29]	%IW29	WORD	0x07ED
MB_ch2_active_power_high		Data[30]	%IW30	WORD	0x07EE

Abbildung 14 - Beispiel für das Mapping von Variablen mit Registern

Die Werte können nun in Programmen beispielsweise für Berechnungen neuer Größen wie der Scheinleistung verwendet, weitergeleitet oder abgespeichert werden. Um allerdings die fertige Programmierung auf die HSP zu übertragen, wird sich über *CODESYS* auf dem Gerät eingeloggt und das Programm übertragen. Sobald dies geschehen ist, können die erstellten Skripte gestartet werden. Diese laufen nun bis zu ihrer Beendigung, dem Auftreten eines fatalen Fehlers oder dem Herunterfahren des Gerätes auf diesem.

7.3.2 Programmierung der lokalen Datenspeicherung

Nachdem im vorangegangenen Abschnitt die Datenaufnahme erörtert wurde, befasst sich dieser Teil mit der lokalen Datenspeicherung. *CODESYS* bietet verschiedene Arten der Speicherung an, die in Form von beispielswese *.txt*- oder *.csv*-Dateien erfolgen kann. Dies entspricht jedoch nicht den Vorgaben einer sicheren und strukturierten Datenbank. Da die Automatisierungssoftware jedoch auch Bibliotheken für die Nutzung von *Structured Query Language*-Datenbanken (*SQL*) zur Verfügung stellt, bietet sich die Nutzung einer solchen Struktur an.

SQL-Datenbanken zählen zu den relationalen Datenbanken und setzen sich aus verschiedenen Tabellen zusammen, in denen Informationen auf Basis einer festgelegten Struktur gespeichert werden können. Sie benötigen lediglich geringe Speichermengen, vermeiden durch ihren Aufbau Dopplungen und ermöglichen es Nutzern, mithilfe von Befehlen die gespeicherten Informationen zu verwalten und beispielsweise abzurufen. [74] Dadurch eignet sich diese Datenbank-Art auch für die Umsetzung dieser Arbeit. Da *SQL*-Datenbanken in der Regel lediglich über das Internet verfügbar sind, aber eine lokale Offline-Speicherung der Daten auf der Hardware gefordert ist, muss ein eigener *SQL*-Server auf der HSP implementiert werden.

Der *Azure IoT Hub* bietet die Möglichkeit, fertige oder selbst programmierte Software-Module, zu denen auch eines für den Betrieb eines *SQL*-Servers verfügbar ist, auf IoT-Geräten zu installieren und zu verwalten. Für die Implementierung dieses Servers müssen zunächst sowohl auf dem Entwicklungsrechner als auch auf der HSP die in Tabelle 14 gelisteten Softwarekomponenten installiert werden. [75]

Tabelle 14 - Auflistung von Softwarekomponenten, die für die Implementierung des *SQL*-Servers vorausgesetzt werden

Plattform	**Softwarekomponente**
Entwicklungsrechner	*Docker CE* (mit *Linux*-Containern)
	Visual Studio Code
HSP	*Azure IoT Edge*
	Docker

Docker ist eine Anwendung, die es ermöglicht, Lösungen zu virtualisieren, indem sogenannte *Docker-Container* aus einer Art virtuellen Vorlage, einem „Image“, erzeugt werden. Diese *Container*, deren Kommunikation über das Netzwerk läuft, bieten dann die Möglichkeit, die entwickelte Anwendung von ihrem Umfeld zu isolieren und diese zur Verwendung bereitzustellen. [76] Sie muss sowohl auf dem Entwicklungsrechner als auch auf der HSP installiert werden, damit die erstellte Applikation übertragen werden kann.

Visual Studio Code ist ein Produkt von *Microsoft*, das als Code-Editor verschiedener Programmiersprachen dient und es ermöglicht, Skripte und Anwendungen in *Azure* bereitzustellen. [77] In diesem Fall wird das Programm genutzt, um den *SQL*-Server für die Verwendung auf der HSP vorzubereiten und das Modul auf dem Gerät zu installieren.

Die *Microsoft*-Software *Azure IoT Edge* wird auf die HSP gespielt und bietet die Möglichkeit, Module auf dem Gerät zu installieren, die auf die *Azure*-Umgebung zugreifen und mit den bereitgestellten Strukturen arbeiten können. Auf diesem Weg können Schnittstellen zu dem *Azure IoT Hub* geschaffen und Daten übertragen werden. [78]

Nach der Installation der Programme kann das *SQL*-Server-Modul mithilfe von *Visual Studio Code* auf Basis der offiziellen Anleitung installiert und konfiguriert

werden. Beachtet werden sollte hierbei vor allem die Änderung des standardmäßig festgelegten Passwortes auf eine sichere Kombination aus Buchstaben, Zahlen und Sonderzeichen. Da der *SQL*-Server auch über das Internet verfügbar ist und über entsprechende Programme angesprochen werden kann, besteht ansonsten ein erhöhtes Risiko für Fremdzugriff.

Wird das Modul korrekt ausgeführt, werden unter Verwendung der entsprechenden Befehle eine neue Datenbank und die benötigten Tabellen angelegt, sodass der Server nun bereit ist, Daten abzuspeichern. Die Betrachtung und Verwaltung der Daten kann beispielsweise über *SQL*-Befehle in der Eingabeaufforderung der HSP oder über Softwareprogramme wie *Microsoft SQL Server Management Studio* erfolgen. Dieses Tool bietet eine Bedienoberfläche, die eine gute Übersicht über die Struktur des Servers und die Inhalte der Datenbank schafft.

Für die Weiterleitung der Daten werden nun die folgenden *CODESYS*-Bibliotheken benötigt: *CAA DTUtil* und *MsSqlLibrary*. Während die *CAA DTUtil*-Bibliothek Funktionen zum Auslesen der Systemzeit zur Verfügung stellt, um zum Beispiel einen Datums- und Zeitstempel (DZS) zu generieren, erlaubt die *MsSqlLibrary*-Bibliothek den Zugriff auf *SQL*-Server, dessen Daten dadurch verwaltet und abgefragt werden können.

Zwar liefert auch der *mE180* alle Daten, die für die Erstellung eines DZS benötigt werden würden, allerdings ist bei dessen Verwendung beispielsweise nicht ersichtlich, wenn keine Daten des Messkontrollers abgefragt werden konnten. Wird ein von der Systemzeit abhängiger Zeitstempel generiert, werden weiter Daten auf dem *SQL*-Server gespeichert. Dadurch, dass bei einer unterbrochenen Verbindung zwischen dem *mE180* und der HSP keine neuen Werte geliefert werden, ändert sich zwar der Wert für den DZS, die gemessenen Werte jedoch nicht mehr. So ist auch nachträglich auf Basis der aufgenommenen Daten ersichtlich, dass eine Fehlfunktion aufgetreten ist.

Aus diesem Grund wird zunächst über ein kleines Programm ein neuer Zeitstempel auf Basis der Systemzeit der HSP erzeugt. Hier muss jedoch beachtet werden, dass der DZS in *CODESYS* sich von dem standardmäßigen Format, das bei DZS von *SQL*-Servern verwendet wird, unterscheidet (siehe Tabelle 15, S.71).

Tabelle 15 - Unterscheidung des DZS-Formats in *CODESYS* und für *SQL*-Server

Name	**Format**
CODESYS	DT#JJJJ-MM-TT-hh:mm:ss Beispiel: DT#2021-12-31-13:54:23
SQL-Server	JJJJ-MM-TT hh:mm:ss Beispiel: 2021-12-31 13:54:23

Aus diesem Grund muss das Format des DZS angepasst werden, was mit einem weiteren Programm namens *splitDTValue* erfolgt, dessen Funktionsweise am Beispiel des DZS „DT#2021-12-31-13:54:23“ erklärt wird. Dabei wird das Programm als Funktionsbaustein deklariert, um im späteren Hauptprogramm übersichtlich die Inputs und Outputs des Skriptes nutzen zu können. Vor seiner Zerlegung muss der vom System ausgelesene Wert zunächst von dem Typ *DATE_AND_TIME*, der für DZS verwendet wird, in den Datentyp *STRING*, also eine Buchstabenfolge, umgewandelt werden. Im Anschluss wird der Zeitstempel in seine einzelnen Bestandteile zerlegt. Dies geschieht, indem automatisch die Position des „#“-Zeichens mithilfe des *FIND*-Operators gesucht und die Buchstabenfolge über den *LEFT*- und den *DELETE*-Operator in zwei Teile (Teil 1: „DT#“; Teil 2: „2021-12-31-13:54:23“) getrennt wird. Wie in Abbildung 15 zu erkennen ist, muss die ermittelte Position um den Wert „1“ reduziert werden, damit der *LEFT*-Operator die korrekte Länge (*SIZE*) der Zeichenfolge verwendet und sie entsprechend auftrennt.

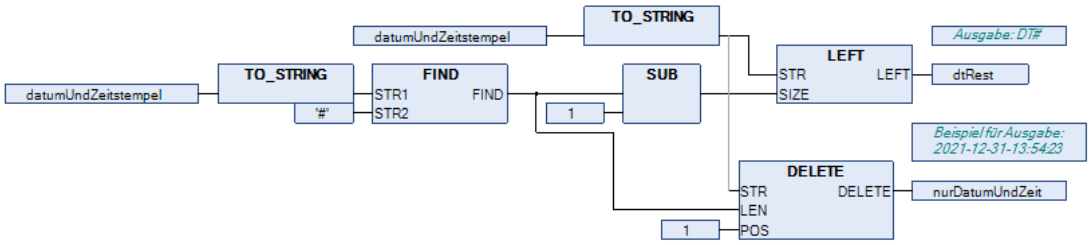


Abbildung 15 - Erster Programmteil der Auftrennung des DZS für die Umformatierung

Nachfolgend wird auf Basis des Ausgangs des *DELETE*-Operators, der den zweiten beziehungsweise rechten Teil des *STRING*-Wertes („2021-12-31-13:54:23“) ausgibt, die weitere Separierung vorgenommen. Dabei bleibt das Schema gleich dem gerade beschriebenen, außer, dass das Zeichen, nach dem für die Auftrennung gesucht wird, nun von „#“ auf „-“ geändert, und der links davon gefundene Teil der Zeichenfolge in Variablen für die spätere Ausgabe des Funktionsbausteins gespeichert wird. Auf diese Art und Weise werden das Jahr, der Monat, der Tag und der vollständige Zeitstempel separiert

abgespeichert und können in beliebiger Reihenfolge mithilfe von *CONCAT*-Operatoren wieder zusammengesetzt werden (siehe Abbildung 16).

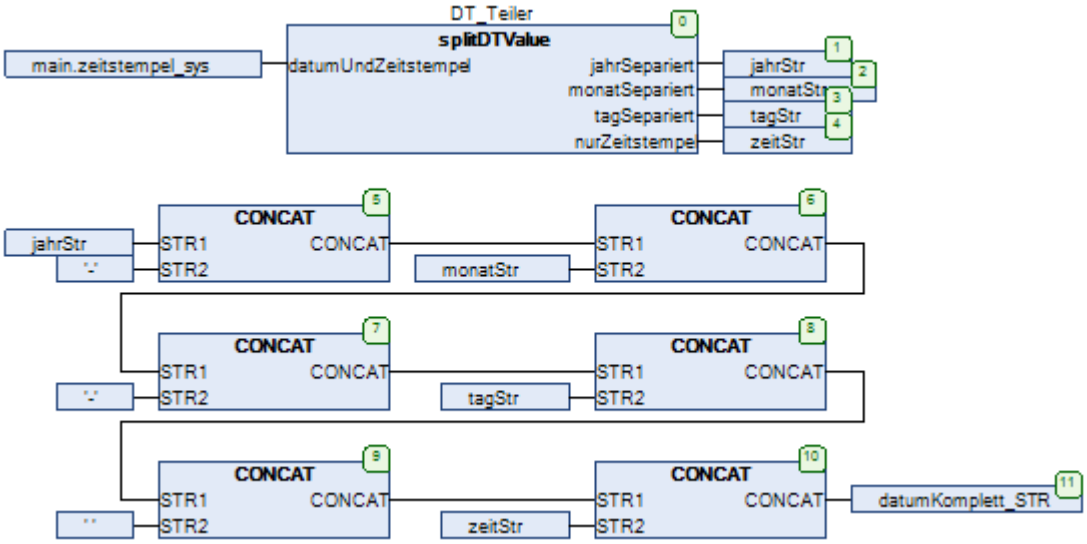


Abbildung 16 - Verwendung des Funktionsbausteins *splitDTValue* und Neuformatierung des DZS über *CONCAT*-Operatoren

Der nächste Teil des Programmes setzt die gemessenen Werte und den Zeitstempel zu einem *SQL*-Befehl, der auch „Query“ genannt wird, zusammen, damit die Daten versendet werden können. Dies erfolgt erneut über *CONCAT*-Befehle, wobei die Messdaten, die dem Datentyp *REAL* (Gleitkommazahl) entsprechen, zunächst in Strings umgewandelt werden müssen (siehe Abbildung 17, S.73).

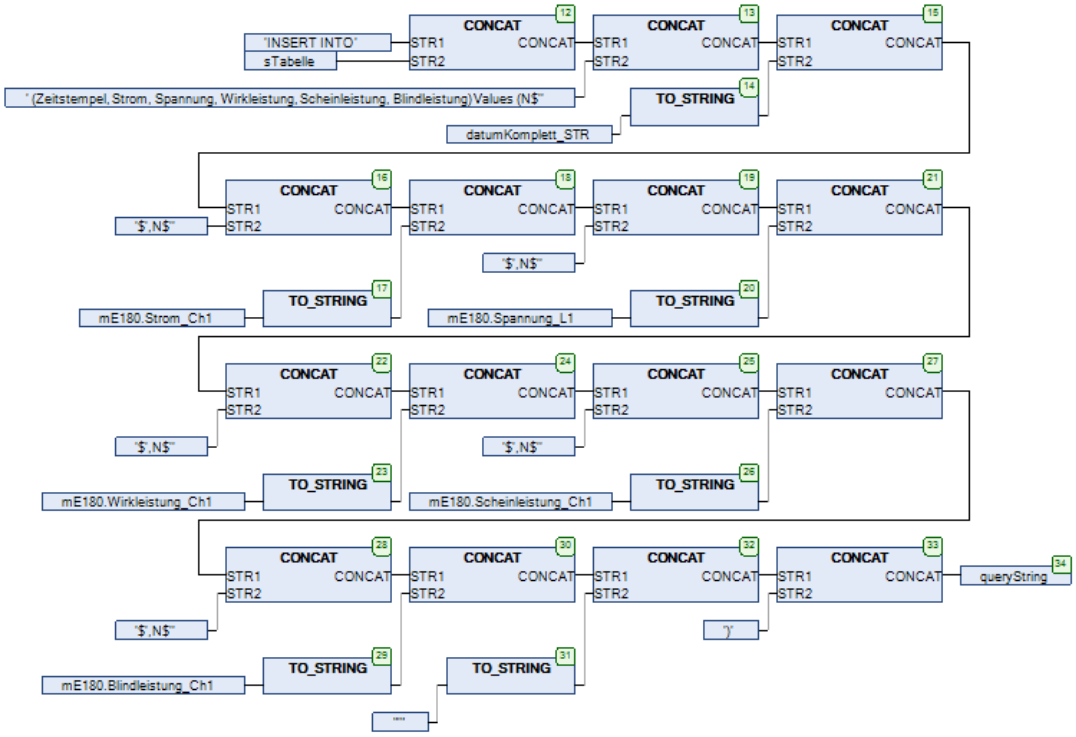


Abbildung 17 - Zusammensetzung der Messdaten und des Zeitstempels zu einem *SQL*-Befehl

Würde der Befehl auf Basis des bisherigen Programms verwendet werden, würden die Daten im Millisekunden-Takt versandt werden. Da lediglich eine sekündliche Abfrage der Daten erfolgt, wäre die Folge, dass doppelte Werte in die Datenbank geschrieben werden. Um dies und damit Dopplungen zu vermeiden, wird durch den Einsatz eines *BLINK*-Bausteins, der für eine festgelegte Zeitdauer (*TIMELOW*) den Wert „0“ oder „FALSE“ ausgibt und für eine festgelegte Zeitdauer (*TIMEHIGH*) den Wert „1“ oder „TRUE“ ausgibt. Empirische Versuche haben ergeben, dass die Zeitdauer, in der der *SQL*-Befehl versandt werden soll, mit 10 Millisekunden ausreichend bemessen ist. Da der Takt des Datenversands auf 1 Sekunde festgelegt werden soll, muss das Zeitintervall, in dem die Ausgabe des Funktionsbausteins den Wert 0 oder „FALSE“ hat, dementsprechend 990 Millisekunden andauern. Immer nur wenn der Ausgang den Wert „TRUE“ beziehungsweise 1 besitzt, wird der Versand des *SQL*-Befehls mithilfe der Variablen *xWriteData* freigegeben (siehe Abbildung 18).

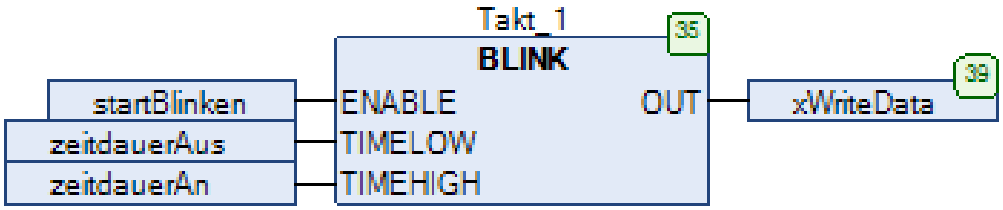


Abbildung 18 - Blink-Funktionsbaustein für den Versand des *SQL*-Befehls

Nachdem nun die zu versendenden Informationen entsprechend aufbereitet wurden, kann in einem letzten Schritt der Versand der Daten erfolgen. Zu diesem Zweck wird der *MsSqlLibrary.fbMsSQL_compact*-Baustein verwendet, der neben der Eingabe der Anmeldedaten für den *SQL*-Server, den Namen der Tabelle, die zusammengesetzte *SQL*-Query und den Startbefehl in Form der *xWriteData*-Variable benötigt. Sollten bei der Ausführung des Funktionsbausteins Fehler auftreten, können diese - wie in Abbildung 19 zu erkennen ist - ausgelesen werden.

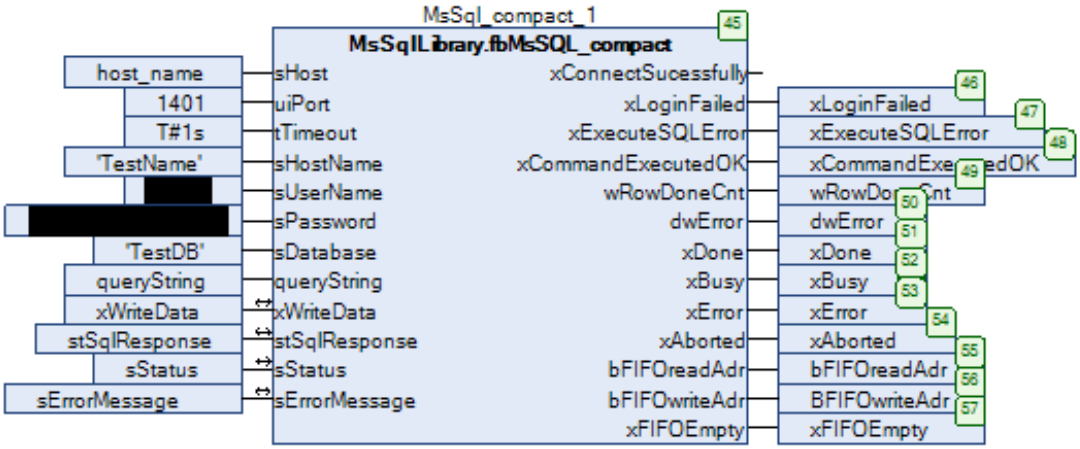


Abbildung 19 - Funktionsbaustein zum Versand des *SQL*-Befehls

7.3.3 Programmierung der Datenweiterleitung auf die IoT-Plattform

Sobald die Messdaten abgerufen und lokal auf dem Gerät gespeichert wurden, findet die lückenlose Datenweiterleitung auf die IoT-Plattform statt. Dieser Prozess kann auf vielfältige Art und Weise gestaltet werden, je nachdem welche Softwarekomponenten genutzt werden sollen. Nachfolgend sind deshalb verschiedene Lösungsansätze beschrieben, die alle bis auf eine für die Umsetzung dieser Arbeit getestet wurden. Vorab sollte darauf hingewiesen werden, dass lediglich einer der Lösungsansätze vollständig umgesetzt werden konnte. Ebenso ist zu beachten, dass lediglich Lösungen getestet wurden, die eine lückenlose Datenübertragung garantieren, sodass die Vollständigkeit der letztendlichen Datenbank auf der IoT-Plattform gewährleistet werden kann. Eine einfache Datenweiterleitung ohne diese Gewährleistung wäre ohne viel Aufwand beispielsweise über *SQL*-Befehle oder einen direkten Upload der *Modbus-TCP*-Daten in den *Azure IoT Hub* möglich.

Lösungsansatz 1 – SQL-Datensynchronisierung

Eine vorgefertigte Lösung zur lückenlosen Übertragung der Messdaten bietet *Microsoft* mit dem Produkt *SQL Azure Data Sync Agent* an. Dieses ermöglicht

die direkte Synchronisation von zwei *SQL*-Datenbanken von *Azure* oder *Microsoft SQL*-Servern. Dabei kann ausgewählt werden, welche der beiden Datenbanken die führende Datenquelle ist, sodass die jeweils andere lediglich aktualisiert und um die fehlenden Informationen ergänzt wird. Zu diesem Zweck wird im *Azure*-Portal eine sogenannte „Synchronisierungsgruppe“ erstellt, mit der die beiden Datenbanken verknüpft werden können. Zur Erhöhung der Sicherheit wird ein Schlüssel generiert, der für die Kommunikation der beiden Synchronisationsteilnehmer benötigt wird. Nachdem der *SQL Azure Data Sync Agent* auf dem Gerät installiert wurde, auf dem der *SQL*-Server betrieben wird, und verschiedene Einstellungen vorgenommen wurden, kann die Synchronisation beginnen.

Dieser Lösungsweg scheint simpel und schnell umsetzbar, kommt allerdings für die Umsetzung dieser Arbeit aus folgendem Grund nicht in Frage: Der *SQL Azure Data Sync Agent* muss auf dem Gerät installiert werden, auf dem der *SQL*-Server betrieben wird, ist zurzeit jedoch nur für *Windows*-Betriebssysteme verfügbar. Aus diesem Grund ist ein Test dieses Lösungsansatzes nicht erfolgt. Sollte in der Zukunft die Überlegung bestehen, die HSP mit *Windows* zu betreiben, wäre dieser Ansatz jedoch eine weitere Option, die lückenlose Datenübertragung zu gewährleisten. [79]

Lösungsansatz 2 – Message Routing

Der *Azure IoT Hub* bietet verschiedene Möglichkeiten, um Daten von einem *Edge*-Gerät, das heißt einer Hardware, auf der die *Azure IoT Edge*-Laufzeitumgebung (AIoTEL) betrieben wird, in eine Datenbank auf dem Hub selbst zu übertragen. Eine Option hierfür ist beispielsweise die Verwendung des sogenannten *Message Routings*, das in *IoT Edge* integriert ist und das Weiterleiten von Nachrichten erlaubt. Dies ermöglicht es, Informationen sowohl innerhalb der AIoTEL auf dem *Edge*-Gerät zwischen einzelnen Modulen als auch zwischen Modulen und dem *IoT Hub* auszutauschen. Dabei werden die Nachrichten eines Moduls, die versandt werden sollen, in dessen vordefinierten Ausgang geleitet und von dort entweder in die Cloud oder den Nachrichteneingang eines anderen Moduls versandt. [80]

Die Festlegung des *Message Routings*, bei der definiert wird, wohin die Daten eines Moduls geleitet werden sollen, kann über das *Azure* Portal erfolgen. Hier können nicht nur bestehende Weiterleitungen umbenannt, umgeleitet oder priorisiert, sondern auch neue Datenverbindungen angelegt oder aber auch wieder entfernt werden (siehe Abbildung 20, S.76).

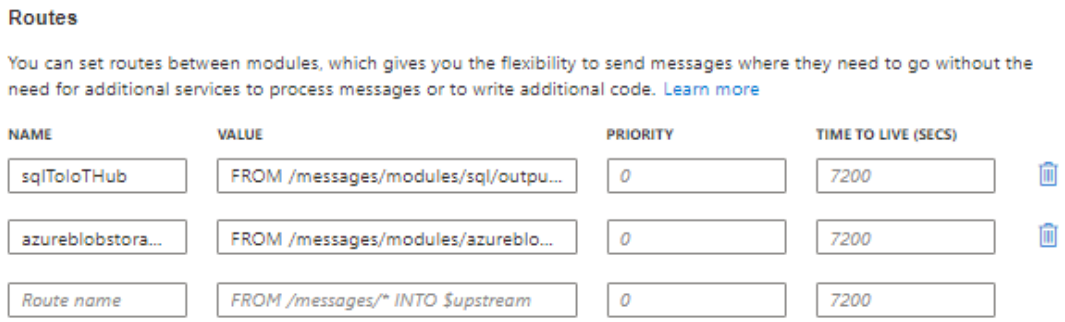


Abbildung 20 - Einstellung des Message Routings im *Azure* Portal

Damit die Daten mithilfe des *Message Routings* in den *IoT Hub* weitergeleitet werden können, müssen diese zunächst der AIoTEL bereitgestellt werden. Dies kann über ein von *Microsoft* kostenfrei zur Verfügung gestelltes *Modbus*-Modul erfolgen, das unter anderem die Daten eines *Modbus-TCP-Slaves* abfragen kann. Mithilfe des *Azure* Portals kann das Modul zum Beispiel über den Marktplatz auf dem *Edge*-Gerät, also der HSP, installiert und - wie in Abbildung 21 zu sehen ist - konfiguriert werden. [81]

Abbildung 21 - Einstellung des *Modbus*-Moduls im *Azure* Portal

Dabei sind neben der Definition des Veröffentlichungs- und Abfrageintervalls vor allem die IP-Adresse des *Modbus-TCP-Slaves*, die Start-Registeradresse und die Anzahl der abzufragenden Register festzulegen. Die in der Abbildung

gezeigten, vorgenommenen Einstellungen dienen lediglich Testzwecken, weshalb die Abfrage der *Modbus*-Register auf zwei Stück begrenzt wurde. Die abgefragten Daten werden nach der Einrichtung bei laufender Übertragung standardmäßig direkt in den *IoT Hub* gesendet und können dort zum Beispiel unter der *Troubleshoot*-Funktion live eingesehen werden.

Für die sichere und verlustfreie Übertragung und Speicherung von unstrukturierten Daten bietet *Microsoft* den sogenannten *Azure Blob Storage* an. Mithilfe der dazugehörigen zur Verfügung gestellten Services können Informationen in den *IoT Hub* übertragen werden. Ein Vorteil hierbei ist, dass die verschlüsselte Kommunikation zwischen Sender und Empfänger überprüft und der Sender informiert wird, falls ein Datenpaket nicht sein Ziel erreicht hat. Da die zu versendenden Daten oder Dateien so lange lokal gespeichert werden, bis der Versand erfolgreich war und die Datenbank ihre Ankunft bestätigt, wird eine lückenlose Übertragung ermöglicht. [82]

Für die Nutzung des *Azure Blob Storage* auf einem *Edge*-Gerät stellt der Hersteller ein sogenanntes *Blockblob*-Modul zur Verfügung, das aus Daten oder Dateien *Blockblobs* erzeugt, die zunächst lokal gespeichert werden. Ein solcher *Blockblob* ähnelt einer Box, die mit Informationen gefüllt wird. Durch die Verknüpfung mit einem *Azure-Portal*-Speicherkonto, in dem zunächst ein neuer Container für den Upload der Daten erstellt werden muss, können die Informationen zielgerichtet hochgeladen werden.

Ebenso wie das *Modbus*-Modul kann auch das *Blockblob*-Modul über das *Azure-Portal* eingerichtet und auf dem *Edge*-Gerät installiert werden. Dem offiziellen *Microsoft*-Tutorial für die Einrichtung des *Azure Blob Storage* auf einem *Edge*-Gerät folgend werden zunächst die sogenannten „Container Create Options“ in JSON-Form eingetragen. Diese legen die Zugangsdaten des lokalen Speicheraccounts, über den später auf dem Gerät auf die *Blockblob*-Funktion zugegriffen wird, den Port-Anschluss und den Speicherpfad für die *Blockblobs* fest. Während der weiteren Konfiguration des sogenannten „Modul-Zwillings“ werden die Vorgehensweise beim Löschen versandter Datenpakete, der lokale Speicher-Container und die Zugangsdaten für das *Azure*-Speicherkonto und der Name des zuvor erstellten Containers in der Cloud angegeben (siehe Abbildung 22, S.78). [83]

Module Settings | Environment Variables | Container Create Options | Module Twin Settings

Module twin desired properties will be updated to reflect entered JSON.
Learn more about deployments

```
2     "deviceAutoDeleteProperties": {
3         "deleteOn": true,
4         "deleteAfterMinutes": 14400,
5         "retainWhileUploading": true
6     },
7     "deviceToCloudUploadProperties": {
8         "uploadOn": true,
9         "uploadOrder": "OldestFirst",
10        "cloudStorageConnectionString": "DefaultEndpointsProtocol=https;AccountName=
11        "storageContainersForUpload": {
12            "localblobcontainer": {
13                "target": "uploaded"
14            }
15        },
16        "deleteAfterUpload": false
17    }
```

Abbildung 22 - Konfiguration des Modul-Zwillings für das *Blockblob*-Modul im *Azure-Portal*

Wird das neu eingerichtete Modul nun gestartet, kommt es zunächst zu einigen Fehlermeldungen, da der lokale Speicher-Container nicht gefunden werden konnte. Aus diesem Grund muss in einem nächsten Schritt mithilfe der Software *Storage Explorer* von *Microsoft* ein solcher Speicher-Container angelegt und mit einer eindeutigen Bezeichnung, die bei der Konfiguration des Modul-Zwillings angegeben wird, benannt werden.

Wie alle *Azure IoT Edge*-Module verfügt auch das *Blockblob*-Modul über *Message Routing*-Kanäle. Nun gilt es, die von dem *Modbus*-Modul abgefragten Daten intern an das *Blockblob*-Modul zu versenden. Der Übersicht halber ist der vollständige Datenfluss von der Messstelle bis zum Cloud-Speicher in Abbildung 23 (S.79) schematisch dargestellt.

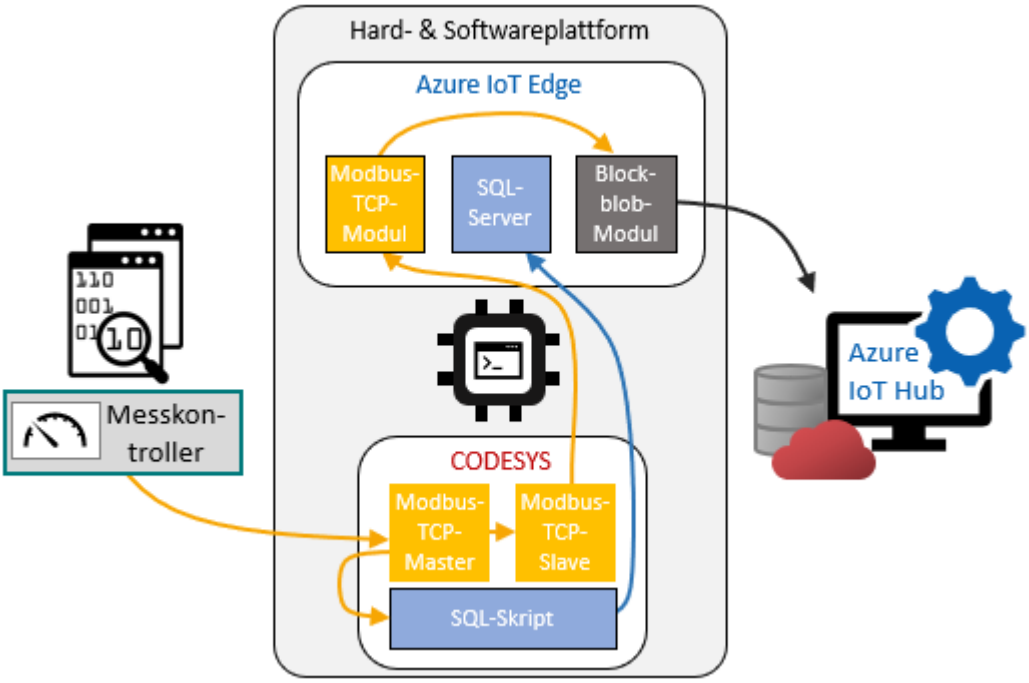


Abbildung 23 - Schematische Darstellung des Datenflusses vom Messgerät über die HSP bis in den *IoT Hub* für Lösungsansatz 2

Die Daten, die über das *Modbus*-Modul versendet werden, werden im JSON-Format versendet, bei dem Informationen nach bestimmten Verschachtelungsstrukturen in Objekten, Arrays und Werten gespeichert werden. Diese müssen dann aus den Nachrichten extrahiert werden, damit eine Nutzung oder Weiterverarbeitung möglich ist. [84] Aus diesem Grund können die Daten nicht direkt von dem Ausgang der *Modbus*-Schnittstelle in den *Blockblob*-Eingang gesendet werden, sondern bedürfen einer vorgeschalteten Filterung. Zu diesem Zweck wurde mithilfe von *Visual Studio Code* auf dem Entwicklungsrechner ein Skript in der Programmiersprache *C#* erstellt, das diese Aufgabe übernehmen sollte. Während der Extrahierung und Weiterverarbeitung der empfangenen Daten kam es jedoch zu einer Vielzahl unterschiedlicher Fehler. Diese konnten auch durch das Einbeziehen von verschiedenen Foren und *Microsoft*-Entwicklern nicht gelöst werden, sodass letztendlich kein funktionsfähiges Programm entwickelt werden konnte. Aus diesem Grund und aufgrund anderer Lösungsansätze, die eine einfachere Umsetzung versprachen, wurde die Verwendung des *Message Routings* nicht weiterverfolgt. Der anonymisierte *C#*-Code, der für den Lösungsansatz erstellt wurde, ist in Anhang F zu finden.

Lösungsansatz 3 – Ausführbares C#-Skript

Aufbauend auf dem vorherigen Lösungsansatz mithilfe von *Message Routing* wurde eine neue Herangehensweise entwickelt, die während der Versuche zu

Lösungsansatz 2 entstand. Während die Verarbeitung der gestreamten Daten innerhalb der AIoTEL nicht funktionierte, waren Tests, bei denen einzelne *.csv*-Dateien verarbeitet wurden, zur Funktionsüberprüfung des *Blockblob*-Moduls problemlos abgelaufen. Aus diesem Grund wurde der bereits programmierte *C#*-Code so angepasst, dass einzelne *.csv*-Dateien, die von *CODESYS* erzeugt werden sollen, aus einem definierten Ordner in das *Blockblob*-Modul geladen werden, um nach erfolgreichem Upload wieder gelöscht zu werden (siehe Abbildung 24).

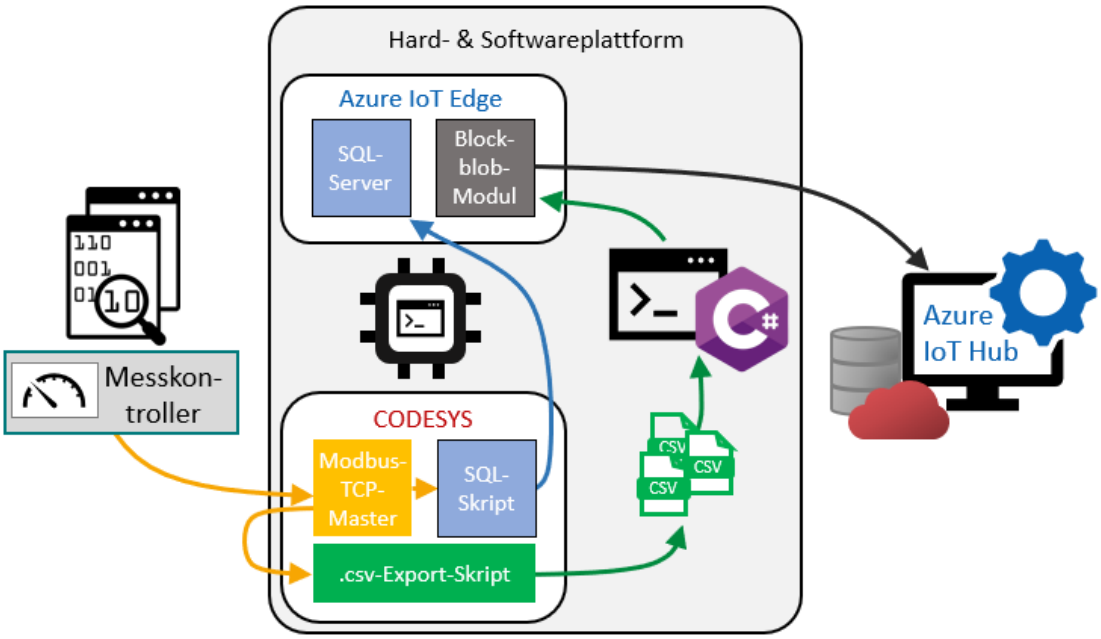


Abbildung 24 - Schematische Darstellung des Datenflusses vom Messgerät über die HSP bis in den *IoT Hub* für Lösungsansatz 3

Für die Anpassungen des bereits laufenden *CODESYS*-Programms wird zunächst eine Bibliothek benötigt, die das Erstellen von *.csv*-Dateien erlaubt, was beispielsweise mit der *File Access*-Bibliothek möglich ist. Da der Vorgang der Dateierzeugung und -speicherung mehrere Sekunden in Anspruch nimmt, werden zunächst die Messdaten von 60 Sekunden in einer Variable des Datentyps *ARRAY* gesammelt. Ein Array ähnelt einer Tabelle, in der jede Zelle einzeln ausgelesen werden kann. Dabei wird in *CODESYS* der Dateityp der abgespeicherten Informationen, die in diesem Fall Strings sind, bei der Deklaration der Array-Variablen festgelegt. Die gemessenen Daten werden dann in Zeichenfolgen umgewandelt und mithilfe des *CONCAT*-Befehls zusammengefügt. Da es sich bei den Messdaten um Gleitkommazahlen handelt, die in *CODESYS* mit einem Punkt und nicht wie im deutschen Sprachraum mit einem Kommazeichen getrennt werden, wird die Position des Punktes in der Zeichenfolge gesucht und mit einem Komma ersetzt. Dies ist vor allem dann wichtig wenn die *.csv*-Datei beispielsweise in einem deutschsprachigen Excel-

Programm geöffnet wird, da dies mit einem Punkt getrennte Gleitkommazahlen meist in Datumswerte umwandelt. In einer *.csv*-Datei wird stets ein Trennzeichen (meist ein Semikolon) verwendet, um die eingetragenen Werte voneinander zu separieren. Aus diesem Grund wird an das Ende des eingetragenen Wertes noch ein Semikolon angefügt. Wie in Abbildung 25 zu sehen ist, wurde das Programm nicht mit Funktionsbausteinen, sondern der Programmiersprache *Strukturierter Text* umgesetzt.

```
13        // Fügt den Zeitstempel in das Array ein
14        measuredData[logIndex]:=CONCAT(Zeitstempel,';');
15
16        // Kopiert die Spannung von Kanal 1 in das Array
17        // Wandelt den gemessenen Wert in einen String um
18        K1_Spannung := TO_STRING(inputSpannung);
19        // Fügt den umgewandelten Wert in das Array ein
20        measuredData[logIndex]:=CONCAT(measuredData[logIndex],K1_Spannung);
21        // Ermittelt die Position des REAL-Wert-Punktes im String
22        Pos := Find(measuredData[logIndex],'.');
23        // Ersetzt den Punkt durch ein Komma
24        measuredData[logIndex]:= Replace(measuredData[logIndex], ',' , 1, Pos);
25        // Fügt ein Semikolon als Trennzeichen hinzu
26        measuredData[logIndex]:=CONCAT(measuredData[logIndex],';');
```

Abbildung 25 - Ausschnitt aus dem *CODESYS*-Programm zur Erstellung einer *.csv*-Datei - Hier: Befüllung des Arrays mit erstem Messwert (Spannung)

Ausgeführt wird das Skript aufgrund eines Triggers, der - wie der für das *SQL*-Programm verwendete - auf Basis eines *BLINK*-Bausteins in sekündlichem Abstand erzeugt wird. Nachdem die Messdaten einer Sekunde alle in dem Array aufgenommen wurden, wird eine Zählvariable um 1 hochgesetzt und sobald diese nach 1 Minute gleich 60 ist, wird die Erzeugung der *.csv*-Datei gestartet.

Um zu vermeiden, dass die Befehle für die Erzeugung der *.csv*-Datei gleichzeitig ablaufen und es dadurch zu Fehlern kommt, wird der Prozess in fünf Abschnitte separiert. Die einzelnen Programmteile werden dann auf Basis einer *CASE*-Unterscheidung ausgeführt, die hierfür den Wert einer festgelegten Variable untersucht und je nachdem, welchen Wert sie besitzt, einen bestimmten Programmcode ausführt (siehe Abbildung 26, S.82). Nachdem einer dieser Programmcodes ausgeführt wurde, wird die *CASE*-Variable (*uiStpDataLogger*) auf den Wert gesetzt, der den nächsten Teil des Programms startet. Die einzelnen Programmteile sind durch die vorangestellten Zahlen zu erkennen („0:“, „10:“, „20:“ etc.).

```
76
77   CASE uiStpDataLogger OF
78       (*Wartet auf den Trigger*)
79       0:  IF bWriteData THEN
80               bWriteData:=FALSE;
81               uiStpDataLogger:=10;
82           END_IF
83
84       10: (*Erstellt die neue .csv-Datei*)
85           fopen.sFileName:=FileName;
86           fopen.eFileMode:=File.MODE.MRDWR;
87           fopen.xExclusive:=TRUE;
88           fopen( xExecute:=TRUE);
89           IF fopen.xDone THEN
90               hFile:=fopen.hFile;
91               sStringToWrite:=CONCAT('Zeit;Spannung_L1;Cosphi_Ch1;Strom_Ch1
92               writeIndex:=0;
93               uiStpDataLogger:=20;
94           END_IF
95
96       20: (*Kopiert die Daten aus dem Array*)
97           sStringToWrite:=CONCAT(sStringToWrite,measuredData[writeIndex]);
98           IF writeIndex < messIntervall THEN
99                   sStringToWrite:=CONCAT(sStringToWrite,CRLF);
100          END_IF
101          uiStpDataLogger:=30;
102
103      30: (*Schreibt die Messdaten in die Datei*)
104          fwrite.hFile:=hFile;
```

Abbildung 26 - Ausschnitt aus dem *CODESYS*-Programm zur Erstellung einer *.csv*-Datei – Hier: Erster Teil der *CASE*-Unterscheidung

Während im ersten Teil zunächst lediglich überprüft wird, ob der Befehl für das Schreiben der Messdaten gegeben wurde, wird im zweiten Abschnitt die neue *.csv*-Datei erzeugt und die Spaltenbezeichnungen werden in die Zeichenfolge kopiert. Die nächsten zwei Teile des Programmcodes sind iterativ miteinander verknüpft, wobei in Abschnitt drei eine *STRING*-Variable mit einer Zeile aus dem Array gefüllt und der Befehl für das Erstellen einer neuen Zeile angefügt wird, sofern es sich nicht um die letzte Zeile des Arrays handelt. Nun springt das Programm in den nächsten Schritt, bei dem die gerade in den String kopierte Zeile in die *.csv*-Datei geschrieben wird. Sobald dies geschehen ist, wird der Zähler zum Auslesen der einzelnen Zeilen aus dem Array um 1 erhöht und das Programm springt wieder in den vorherigen Abschnitt, um die nächste Zeile auszulesen. Diese Iteration erfolgt so lange, bis die letzte Zeile des Arrays extrahiert wurde. Zum Abschluss wird die Datei im fünften und letzten Abschnitt geschlossen und gespeichert. Der vollständige Code für dieses Programm kann in Anhang G eingesehen werden.

Nach dem Export der Messdaten in Form von *.csv*-Dateien wird nun der Programmcode für die Verarbeitung dieser Dateien zu *Blockblobs* beschrieben.

Um die Umsetzung dieser Aufgabe zu ermöglichen, wird zunächst die Entwicklungsumgebung *Visual Studio 2022* von *Microsoft* installiert. Für die Implementierung der geplanten Funktion wird neben ein paar standardmäßig installierten System-Bibliotheken die *Azure.Storage.Blobs*-Bibliothek benötigt, die mithilfe des *NuGet-Paket-Managers* in der Projektmappe installiert wird.

Der Programmcode startet mit der Implementierung eines Prozesses in Form einer *Main*-Methode, da dies für das Programm als Einstiegspunkt benötigt wird. Diese Methode wird als *async* initialisiert, sodass mithilfe des *await*-Operators auf die Rückmeldung von einzelnen Programmteilen gewartet werden kann, bevor die nächsten Befehle ausgeführt werden, und startet die Kernfunktion *UploadBlob*. Nun findet zunächst die Deklaration der benötigten Zugangsdaten statt, die sich aus der Verbindungszeichenfolge für das lokale *Blockblob*-Modul und dem Namen des lokalen *Blockblob*-Containers zusammensetzen. Zudem wird der Pfad, der zu dem Ordner führt, in dem die *.csv*-Dateien liegen, in einer Variable gespeichert. Unter Verwendung dieser Information werden dann mithilfe der *GetFiles*-Methode die kompletten Pfade inklusive der Namen der Dateien, die sich in dem vorgesehenen Ordner befinden, ausgelesen und in einem Array gespeichert.

Wie auf dem Programmablaufplan in Abbildung 27 auf Seite 84 zu sehen ist, iteriert im nächsten Schritt eine *foreach*-Schleife durch die einzelnen Elemente des Arrays und trennt zunächst den Dateipfad auf, sodass der Dateiname separiert werden kann. Danach wird der sogenannte *BlobClient* initialisiert, der hierfür die Verbindungszeichenfolge, den Container-Namen und den Namen des zu erzeugenden *Blockblobs* benötigt, um eine Verbindung zum *Azure Blob Storage* herstellen zu können. Nun kann die Datei mithilfe der *UploadAsync*-Methode hochgeladen werden, wobei durch Verwendung des *await*-Operators das Feedback des Aufrufs abgewartet wird. Zum Schluss des Programms wird die Unterfunktion *DeleteCsv* aufgerufen, die nach erfolgreichem Upload die *.csv*-Datei auf Basis des Dateipfades nach dessen Prüfung löscht.

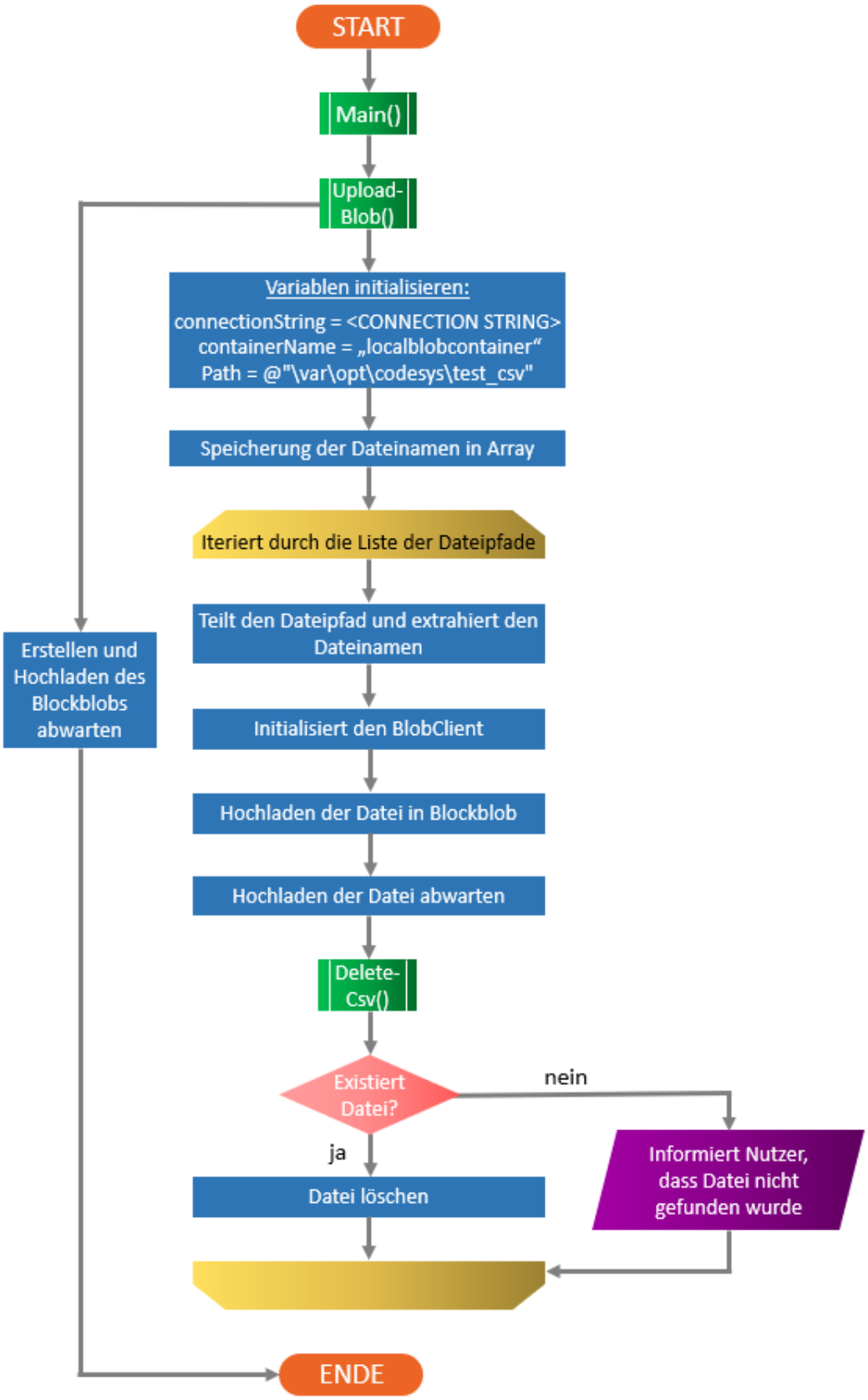


Abbildung 27 - Programmablaufplan für das *C#*-Skript aus Lösungsansatz 3

Damit dieses *C#*-Skript auf einem *Linux*-Computer ausgeführt werden kann, muss auf Basis des Programms eine ausführbare *.exe*-Datei erzeugt werden. Diese muss dann kurz nach dem Neustart des Gerätes automatisch gestartet werden, um eine möglichst lückenlose und automatisierte Dateiübertragung zu ermöglichen. Außerdem muss zu diesem Zweck auf der HSP die Software-Plattform „Mono“ installiert werden, die die Ausführung von .exe-Dateien auf *Linux* ermöglicht.

Theoretisch bestehen verschiedene Möglichkeiten, solch eine benötigte .exe-Datei auf Basis eines *C#*-Skriptes zu erzeugen. Wird beispielsweise in *Visual Studio* die Projektmappe erstellt, werden automatisch ausführbare Dateien (z.B. *.exe*, *.dll*) erzeugt, die in den festgelegten Speicherordnern zu finden sind. Diese Dateien enthalten jedoch nicht die zuvor installierten und für die Ausführung des Skriptes benötigte Bibliothek *Azure.Storage.Blobs*. Eine Möglichkeit, die vollständige Projektmappe inklusive dieser Bibliothek zu exportieren, konnte im Rahmen der Recherchen nicht gefunden werden. Deshalb wurde geprüft, ob es möglich ist, die benötigte Bibliothek ebenfalls auf einem *Linux*-Betriebssystem zu installieren. Ergebnis dieser weiteren Recherche, während der ebenfalls keine Lösung für das *C#*-Problem gefunden werden konnte, war allerdings die Möglichkeit, die benötigte Funktionalität mithilfe der Programmiersprache *Python* umzusetzen. Diese bietet mit der sowohl auf *Windows* als auch auf *Linux* frei verfügbaren *azure.storage.blob*-Bibliothek vergleichbare Methoden wie die zuvor verwendete *C#*-Bibliothek. Da auch die Ausführung von *Python*-Dateien auf *Linux*-Geräten umgesetzt werden kann, wurde nun diese Herangehensweise fokussiert. Der für den dritten Lösungsansatz programmierte *C#*-Code ohne Anmeldeinformationen kann ebenfalls im Anhang betrachtet werden (Anhang H).

Lösungsansatz 4 – Programmierung mit Python

Da *Python* standardmäßig mit der Installation von *Linux* aufgespielt wird, muss es lediglich auf dem Entwicklungsrechner zusätzlich installiert werden. Wie bereits beschrieben, wird die *azure.storage.blob*-Bibliothek für die Umsetzung der benötigten Funktionalitäten benötigt. Diese muss deshalb zunächst ebenso wie die Bibliotheken *schedule* und *time* mithilfe des *pip*-Befehls über eine Eingabeaufforderung heruntergeladen und installiert werden.

Das mit *Python* programmierte Skript ist dem in Lösungsansatz 4 beschriebenen Code sehr ähnlich und unterscheidet sich hauptsächlich dadurch, dass es zusätzlich überprüft, ob es sich um *.csv*-Dateien handelt und dass bereits die Ausführung in einem Zeitintervall von einer Minute implementiert ist. Anders als beispielsweise bei *CODESYS*, in dem die benötigten Bibliotheken in jedem

Projekt installiert werden müssen, findet in *Python* lediglich ein Import der Bibliotheken statt. Der Übersichtlichkeit halber sollten sämtliche Importvorgänge in der Regel am Anfang des Skriptes stattfinden (siehe Abbildung 28).

```
1   from azure.storage.blob import ContainerClient
2   # für os.remove()
3   import os
4   # lisdir muss einzeln importiert werden
5   from os import listdir
6   from os.path import isfile, join
7   import schedule
8   import time
9
10
11  CONNECT_STR = "<CONNECTION STRING>"
12
13  CONTAINER_NAME = "localblobcontainer"
14
15  dateiPfad = "/var/opt/codesys/test_csv"
16
17  def getFilesSendAndDelete():
18      # Speichert die Namen aller Dateien im aktuellen Pfad
19      filenames = [f for f in listdir(dateiPfad) if isfile(join(dateiPfad, f))]
```

Abbildung 28 - Ausschnitt aus dem *Python*-Programm für die Verarbeitung der *.csv*-Dateien zu *Blockblobs* – Hier: Import der Bibliotheken

Wie ebenfalls aus der Abbildung hervorgeht, werden im Anschluss die Anmeldedaten für den Zugriff auf den *Azure Blob Storage* und der Speicherort der *.csv*-Dateien festgelegt. Danach beginnt die Definition der Funktion *getFilesSendAndDelete*, die die Verarbeitung der Dateien vornimmt und mit der Namensspeicherung der sich in dem Ordner befindenden Dateien beginnt. Nun wird mithilfe der Anmeldedaten erst der *ContainerClient* initialisiert und dann eine *for*-Schleife gestartet, die durch das Array mit den Dateinamen iteriert und prüft, ob es sich bei den gefundenen Dokumenten um *.csv*-Dateien handelt. Ist dies der Fall, werden der zu Beginn deklarierte Pfad des Ablageordners und der Dateiname zu einem Dateispeicherpfad zusammengefügt. Im Anschluss wird der Titel des *Blockblobs* gleich dem des Dateinamens gesetzt und der Upload wird gestartet. Sobald der *Blockblob* hochgeladen wurde, wird die ursprüngliche *.csv*-Datei gelöscht und die Schleife beginnt von vorne.

Damit die definierte Funktion in regelmäßigen Abständen ausgeführt wird, folgt der Definition die Verwendung der *every().minutes.do*-Funktion, die der *schedule*-Bibliothek entstammt. Sie initialisiert die Ausführung der *getFilesSendAndDelete*-Funktion und sorgt zusammen mit einer fest auf „TRUE“ gesetzten *while*-Schleife und einem „Sleep-Timer“ von 1 Minute für eine minütliche Ausführung des Skriptes. Somit läuft das Programm in einer absichtlich erzeugten Endlosschleife, die erst durch dessen Beendigung oder durch das Auftreten eines Fehlers unterbrochen wird. Während zur besseren Übersicht

der Programmablaufplan in Abbildung 29 dargestellt ist, kann der funktionierende *Python*-Programmcode in Anhang I eingesehen werden.

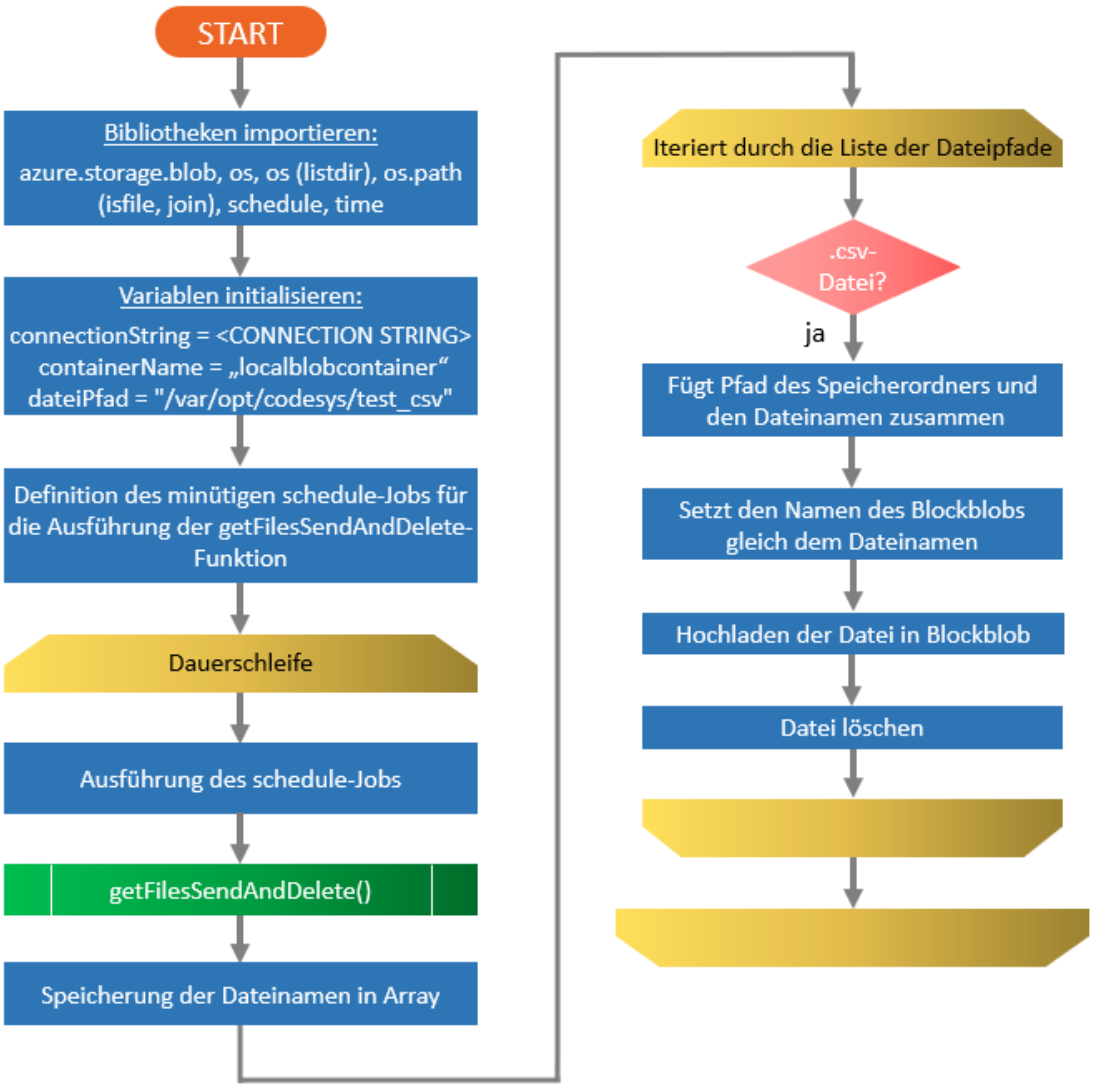


Abbildung 29 - Programmablaufplan für das *Python*-Skript zur Verarbeitung der *.csv*-Dateien zu *Blockblobs*

Wie bereits beschrieben, ermöglicht *Linux* das direkte Ausführen von *Python*-Dateien, was beispielsweise durch die Verwendung der Eingabeaufforderung erfolgen kann, indem der Befehl „python3“ in Verbindung mit dem Dateinamen (z.B. „python3 test.py“) eingegeben und ausgeführt wird. Liegt das Skript nicht im Start-Verzeichnis, ist zusätzlich die Angabe des Speicherortes notwendig (z.B. „python3 /pfad/zum/skript/test.py“).

Nun soll das Skript beim Start der HSP automatisch ausgeführt werden, damit der Logging-Prozess schnellstmöglich starten kann. Für die Umsetzung dieser Funktionalität bietet *Linux* verschiedene Möglichkeiten, von denen eine die Verwendung der *rc.local*-Skriptdatei ist, die in früheren *Linux*-Distributionen vorinstalliert war und heutzutage manuell angelegt werden muss. Dabei wird

gleichzeitig der Befehl zur Ausführung der *Python*-Datei eingefügt, der auf Basis des folgenden Prinzips aufgebaut ist:

> *sleep <ZEIT IN SEKUNDEN> sudo -H -u <BENUTZERNAME> /usr/bin/python3 /pfad/zum/skript/<SKRIPTNAME>.py*

Der *sleep*-Befehl sorgt dafür, dass das Skript erst nach einer festgelegten Zeit (z.B. 60 Sekunden) nach einem Reboot gestartet wird. Auf diesem Weg kann vermieden werden, dass das Programm mit dem Start anderer Softwarekomponenten kollidiert und Fehler verursacht. [85]

Nachdem die Datei angelegt und der Befehl zum Start des Skriptes eingefügt wurde, müssen die entsprechenden Berechtigungen für *rc.local* und die Ordner, auf die die Skripte zugreifen, vergeben werden. Dies erfolgt unter Verwendung der Befehle *chmod* und *chown*, der Ordner und Dateien auf den im Befehl angegebenen Benutzer überträgt. [85] Im Anschluss daran wird das Skript automatisch beim Hochfahren der HSP gestartet und aufgrund der implementierten Dauerschleife in einem minütlichen Takt ausgeführt.

7.3.4 Umsetzungsvorschlag: Rückversand von Sollwerten

Der Rückversand der Sollwerte ist von den Prognosen abhängig, die von der noch zu entwickelnden Cloud-Plattform auf Basis von Modellen des Maschinellen Lernens an die HSP gesendet werden. Da diese Plattform zurzeit noch nicht existiert, besteht zum aktuellen Zeitpunkt noch Unklarheit bezüglich folgender Punkte:

1. Verwendbare Schnittstellen
2. Struktur der Daten
3. Typen der Daten (z.B. *STRING*, *INTEGER*, *REAL*)
4. Auflösung der Daten (z.B. sekündlich, minütlich, stündlich)

Aufgrund dieser Unklarheiten werden nachfolgend lediglich zwei Umsetzungsvorschläge vorgestellt.

Rückversand der Sollwerte über SQL-Server

Eine simple Möglichkeit den Rückversand der Sollwerte zu realisieren, wäre es, den bereits vorhandene *SQL*-Server zu nutzen und dort eine neue Tabelle zu erstellen, die durch *SQL*-Befehle, die von der Cloud-Plattform gesendet werden, befüllt wird. Diese Tabelle sollte zur Speicherung der Sollwerte für sämtliche Anlagen und Maschinen dienen, um den Einrichtungsaufwand zu verringern, da ansonsten für jede Anlage eine eigene Tabelle eingerichtet und die spätere Abfrage der Daten aufwändig im Programm angepasst werden

muss. Dabei sollte die Tabelle über die Spalten „Zeitstempel“ und „Anlagenname“ verfügen. Zu klären ist, wie viele Sollwerte maximal pro Anlage vorgegeben werden können, sodass dementsprechend viele Spalten in der *SQL*-Tabelle berücksichtigt werden können. Hier ist es sinnvoll, eher zu viele Spalten anzulegen und zu reservieren, um eine höhere Flexibilität gewährleisten zu können. Jeder dieser Spalten kann nur ein Datentyp zugewiesen werden, weshalb dies zur besseren Übersicht bei ihrer Benennung berücksichtigt werden sollte.

Für die Weiterleitung der gespeicherten Sollwerte könnte beispielsweise ein *CODESYS*-Programm auf Basis des in dieser Arbeit erstellen Skriptes entwickelt werden. Dieses fragt dann in einem vorgegebenen Takt die auf dem *SQL*-Server gespeicherten Sollwerte ab und separiert den Anlagennamen aus der Antwort des Servers. Im Anschluss wird/werden der/die Sollwert/e mithilfe des bereits eingerichteten *Modbus-TCP-Slave*-Moduls an die entsprechende Anlage versandt.

Rückversand der Sollwerte über Blockblob-Modul

Obwohl der Versand der Sollwerte über den *SQL*-Server den Ansprüchen des Projektes laut Aussage von *manageE* genügen sollte, da nicht gewährleistet werden muss, dass 100 % der Sollwerte weitergeleitet werden, wird nun eine sichere Möglichkeit erörtert. Diese bedient sich des *Blockblob*-Moduls, das über die AIoTEL betrieben wird. Für die Umsetzung des Vorschlags muss es der Cloud-Plattform möglich sein, beispielsweise mithilfe von *C#* oder *Python Blockblobs* auf Basis der berechneten Sollwerte zu erzeugen und an den *IoT Hub* zu versenden. Diese können dann durch Erweiterung des in dieser Arbeit erstellten *Python*-Programms in vorgesehenen Abständen heruntergeladen und in *.csv*-Dateien, für die ähnliche Vorgaben wie für die oben beschriebene *SQL*-Tabelle gelten, umgewandelt werden. Durch Anpassung des erstellten *CODESYS*-Programms können die Sollwerte aus den *.csv*-Dateien extrahiert und an die Anlagensteuerungen versandt werden. Auf diese Art und Weise könnte aufgrund der *Blockblob*-Funktionalität gewährleistet werden, dass sämtliche Sollwerte zumindest an die HSP übergeben wurden.

7.4 Zwischenfazit: Softwareentwicklung für die Datenaufnahme und -speicherung

Zur besseren Übersicht werden nachfolgend noch einmal die Ergebnisse des kompletten Kapitels zur Softwareentwicklung zusammengefasst, in dem zunächst die Vorgaben definiert wurden. Hier werden die einfache Nachrüstbar-

keit und Integration des Systems in bestehende Strukturen, eine hohe Flexibilität bei Anpassung an spezielle Anlagen, die Möglichkeit, Sollwerte an Anlagensteuerungen zu senden, und die Verwendung von *Linux* gefordert. Außerdem sind eine lückenlose Datenerfassung und sowohl eine lokale Speicherung als auch die Speicherung in einer IoT-Datenbank zentral.

Im Lauf der letzten Jahre haben vermehrt Anbieter eigene IoT-Plattformen entwickelt, sodass der Markt inzwischen eine breite Auswahl von Lösungen bereithält, die meist neben den Grundfunktionen der Datenaufnahme und Speicherung auch weitere IoT-Dienste anbieten. Da *manageE* zertifizierter *Microsoft Silver Partner* ist, woraus sich einige Vorteile für das Unternehmen ergeben, und aufgrund des umfassenden Funktionsumfangs, wurde der *Azure IoT Hub* von *Microsoft* für die Umsetzung dieser Arbeit ausgewählt.

Die entwickelte Software-Lösung besteht streng genommen aus vier Teilen, die wie folgt separiert werden können:

1. *CODESYS*-Programm für die Datenaufnahme, Speicherung der Daten auf dem *SQL*-Server, Erzeugung der *.csv*-Dateien und die Verarbeitung und den Rückversand der Sollwerte

2. *Azure IoT Edge Runtime* mit *Microsoft SQL-Server* und *Blockblob*-Modul

3. *Python*-Skript zur Verarbeitung der *.csv*-Dateien zu *Blockblobs*

4. *Azure IoT Hub* für die Speicherung der *Blockblobs* und als Schnittstelle zwischen der HSP und der noch zu entwickelnden Cloud-Plattform für die Erstellung der Prognosen

8 VALIDIERUNG DER ARBEITSERGEBNISSE

Die Validierung der Arbeitsergebnisse dient dazu zu überprüfen, ob die entwickelten Lösungen auch in einem konkreten Anwendungsfall problemlos funktionieren. Zu diesem Zweck werden zunächst die Bewertungskriterien für die Validierung festgelegt, deren Grundlage die Anforderungslisten an die Hard- und Software sind. Basierend auf diesen wird ein Versuchsaufbau entwickelt, der sämtliche von der HSP geforderten Funktionen überprüfen soll. Nach der Beschreibung der Versuchsdurchführung endet das Kapitel mit dem Fazit des Validierungsversuchs.

8.1 Kriterien für Tests

Um einen Maßstab für die Validierung der entwickelten Lösungen zu haben, müssen zunächst entsprechende Kriterien festgelegt werden, die auf dem Anforderungsprofil basieren, das an die HSP gestellt wird. Nachfolgend sind zur besseren Übersicht noch einmal kurz die Anforderungen aus Kapitel 7.1 dargestellt:

1. Fähigkeit, mit einer Vielzahl anderer Systeme kommunizieren zu können
2. Kommunikation über das *Modbus-TCP*-Protokoll
3. Verarbeitung gängiger Datentypen
4. Verwendung von *Linux*
5. Nutzung lizenzfreier oder kostengünstiger Softwarelizenzen
6. Prüfung, ob Daten angekommen sind / Synchronisation
7. Wiederaufnahme des Loggens beim Neustart
8. Lokale Speicherung der Messdaten
9. Sichere Übertragung der Daten in IoT-Datenbank
10. Fähigkeit, Sollwerte zu empfangen und an Steuerungen zu übergeben

Bei der Betrachtung der Anforderungen fällt auf, dass ein Teil der Vorgaben bereits durch die Auswahl der Hard- und Softwarekomponenten erfüllt wurden. Darunter fallen beispielsweise die oben gelisteten Anforderungen 1 und 2, da das System über *Modbus-TCP* kommuniziert, was dem Industriestandard entspricht. Sollte eine Anlage nicht in der Lage sein, das *Modbus-TCP*-Protokoll zu nutzen, wird sie indirekt mithilfe der intelligenten Ein- und Ausgabeeinheiten dazu befähigt. Zudem bietet *CODESYS* notfalls neben *Modbus-TCP* auch weitere Protokolle für die Kommunikation über *Ethernet* an. Nicht zum Umfang dieser Arbeit gehört die Integration der Ein- und Ausgabeeinheiten. Da sie jedoch *Modbus-TCP*-fähig sind, wird davon ausgegangen, dass die Anbindung an die HSP ohne Probleme funktionieren wird. Anforderung 4 und 5 werden

ebenfalls als erfüllt betrachtet, da ein *Debian*-basiertes *Linux*-Betriebssystem und Software verwendet wird, die dem Unternehmen entweder kostenfrei oder kostengünstig zur Verfügung steht. Die letzte Anforderung bezüglich der Übertragung der Sollwerte wird aufgrund der in Kapitel 7.3.4 beschriebenen Unklarheiten und der fehlenden Programmierung nur teilweise validiert. Für die Validierung relevant sind demnach noch folgende Anforderungen, die es zu überprüfen gilt:

3. Verarbeitung gängiger Datentypen
6. Prüfung, ob Daten angekommen sind / Synchronisation
7. Wiederaufnahme des Loggens beim Neustart
8. Lokale Speicherung der Messdaten
9. Sichere Übertragung der Daten in IoT-Datenbank
10. Fähigkeit, Sollwerte an Steuerung zu übergeben

8.2 Aufbau und Vorbereitung der Validierung

Aufgrund der aktuell wieder steigenden Zahlen von *Covid-19*-Erkrankten wurde davon abgesehen, einen Systemtest in einem realen Betrieb zu starten. Deshalb wird in diesem Abschnitt der Aufbau eines Versuches beschrieben, der eine Anlage inklusive Sensorik und Aktorik nachempfindet.

Eines der Hauptelemente des Versuchsaufbaus, das die Energiesenke des Unternehmens darstellt, ist ein Heizlüfter, der über zwei Regler eingestellt werden kann. Während der erste Regler das Zu- beziehungsweise Abschalten des Heizelements in 3 Stufen steuert, wird mit dem zweiten Regler das eingebaute Thermostat stufenlos eingestellt. Dieses schaltet den Heizlüfter ab, sobald die konfigurierte Raumtemperatur erreicht ist.

Stellvertretend für die Sensorik der Anlage kommt ein Stromwandler zum Einsatz, der die Stromstärke des Heizlüfters misst. Damit dies möglich ist, wird das Netzkabel des Heizlüfters mit einem Zwischenstück verbunden, das die Phase von Schutz- und Neutralleiter separiert (siehe Abbildung 30).

Abbildung 30 - Zwischenstück zur Separierung der Phase des Heizlüfter-Kabels

Damit die von der Sensorik gemessenen Daten aufgenommen und verwendet werden können, ist der Stromwandler an einen *mE180 Messcontroller* angeschlossen. Dieser ist wiederum mit dem Netzwerk verbunden, in dem sich auch die HSP befindet, sodass die Daten entsprechend abgerufen werden können. Zusätzlich zu der Messung des Stromflusses greift das Gerät auch die Eingangsspannung ab.

Die Aktorik der Anlagensteuerung wird im Versuchsaufbau durch einen *WAGO*-Controller, der ebenfalls über *Ethernet* mit dem Netzwerk verbunden ist, und einen mit dem Controller verbundenen Schrittmotor dargestellt. Dieser symbolisiert ein Regelventil, das je nach Energiebezug des Heizlüfters entsprechend eingestellt werden soll.

Neben dem bereits erwähnten Netzwerk, zu dem der Zugang im Test durch Verwendung eines Switches ermöglicht wird, gehören, wie in Abbildung 31 schematisch dargestellt, die bereits erwähnte HSP und der *Azure IoT Hub* zu den restlichen Bestandteilen der Validierung.

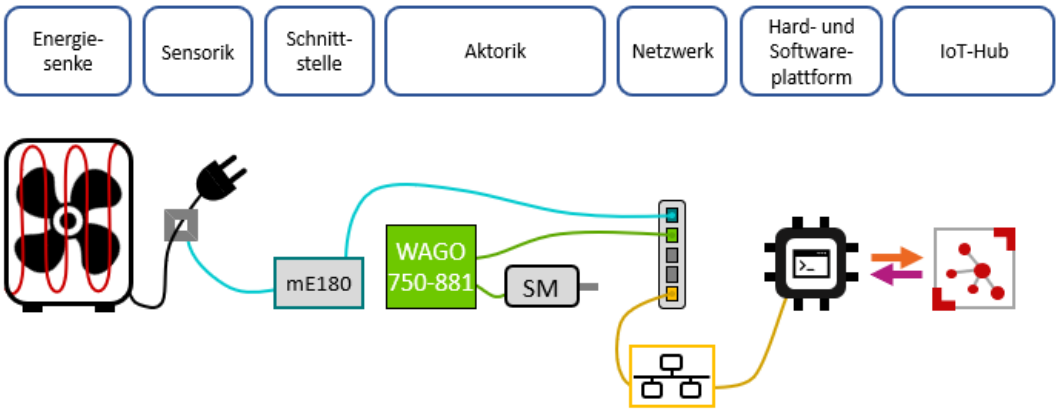


Abbildung 31 - Schematische Darstellung des Versuchsaufbaus

Zusätzlich zu den hardwaretechnischen Elementen des Versuchsaufbaus werden zwei Softwarekomponenten benötigt, die die folgenden Rollen einnehmen:

1. Softwarekomponente 1: Ansteuerung des WAGO-Controllers
2. Softwarekomponente 2: Ansteuerung des Schrittmotors

Für die Ansteuerung des *WAGO*-Controllers wird das bereits implementierte *CODESYS*-Skript um einen weiteren *Modbus-TCP-Slave* erweitert, der unter der Berücksichtigung der Registeradressen, des Offsets und der IP-Adresse des Gerätes eingerichtet wird. Zusätzlich dazu wird das Programm um ein Unterprogramm erweitert, das bei Überschreiten eines Grenzwertes, der sich am Stromfluss orientiert und auf 7 Ampere festgelegt wird, einen Regelbefehl an den *WAGO*-Controller sendet.

Das zweite Skript wird ebenfalls mit *CODESYS* geschrieben und auf dem *WAGO*-Controller implementiert. Es legt fest, auf welche Art und Weise der Schrittmotor angesteuert werden kann, wie die Standardparameter (z.B. die Drehgeschwindigkeit) lauten und auf welcher Registeradresse welcher der Werte geändert werden kann.

8.3 Versuchsdurchführung

Nachdem die einzelnen Komponenten, die für den Test benötigt werden, wie im vorangegangenen Kapitel beschrieben und in Abbildung 32 dargestellt, miteinander verbunden wurden, kann die Validierung der Arbeitsergebnisse beginnen.

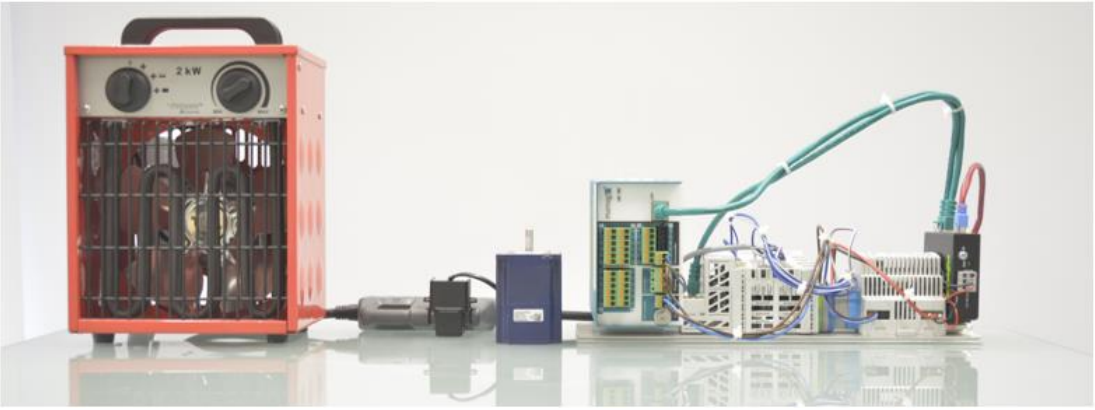


Abbildung 32 - Realer Aufbau des Validierungsversuchs

Zunächst werden die HSP, der *WAGO*-Controller und der *mE180 Messcontroller* mit dem Stromnetz verbunden und gestartet. Sobald die einzelnen Geräte hochgefahren sind, kann die HSP eine Verbindung zu den beiden Controllern über deren IP-Adresse herstellen. Steht diese Verbindung, werden aufgrund der implementierten Programme automatisch Daten von dem *mE180* abgefragt, die in sekündlicher Auflösung an den *SQL*-Server geschickt werden (siehe Abbildung 33, S.95).

Results | Messages

	Zeitstempel	Strom	Spannung	Wirkleistung	Scheinleistung	Blindleistung
1	2022-01-31 13:17:12.000	0	0	0	0	0
2	2022-01-31 13:17:13.000	0	226,59	0	0	0
3	2022-01-31 13:17:14.000	0	226,65	0	0	0
4	2022-01-31 13:17:15.000	0	226,63	0	0	0
5	2022-01-31 13:17:16.000	0,1403183	226,6	13,1	31,79612	28,97211
6	2022-01-31 13:17:17.000	2,673272	225,16	421,4	601,914	429,7936
7	2022-01-31 13:17:18.000	2,673272	225,16	421,4	601,914	429,7936
8	2022-01-31 13:17:19.000	3,966791	223,43	886,3	886,3	0
9	2022-01-31 13:17:20.000	3,963895	223,24	884,9	884,9	0
10	2022-01-31 13:17:21.000	3,962264	223,13	884,1	884,1	0
11	2022-01-31 13:17:22.000	3,963439	223,19	884,6	884,6	0
12	2022-01-31 13:17:23.000	3,962898	223,17	884,4	884,4	0
13	2022-01-31 13:17:24.000	3,962171	223,11	884	884	0
14	2022-01-31 13:17:25.000	3,962171	223,11	884	884	0
15	2022-01-31 13:17:26.000	3,96371	223,2	884,7	884,7	0
16	2022-01-31 13:17:27.000	4,675491	222,78	998,9	1041,606	295,1976
17	2022-01-31 13:17:28.000	7,720645	220,15	1699,7	1699,7	0
18	2022-01-31 13:17:29.000	7,713571	219,95	1696,6	1696,6	0
19	2022-01-31 13:17:30.000	7,718557	220,08	1698,7	1698,7	0
20	2022-01-31 13:17:31.000	7,718557	220,08	1698,7	1698,7	0

Abbildung 33 - Ausschnitt der während des Tests aufgenommenen Daten abgefragt von dem lokalen *SQL*-Server

Zu erkennen ist in der Darstellung, dass zu Beginn der Messung noch keine Daten geliefert wurden, was an dem Umstand liegt, dass der Heizlüfter erst beim Start der Messung angeschaltet wurde. Werden die einzelnen Sekundenwerte genauer betrachtet und mit den vorher und danach aufgenommenen Daten verglichen, fällt auf, dass die Messwerte mit zunehmender Zeit steigen. Dies wird durch die Betätigung des Drehreglers verursacht, der das Zuschalten des Heizelementes steuert. Für etwa 1 Sekunde wird das Gerät nur mit dem Ventilator, also ohne Heizelemente, betrieben (Höhe des Stromflusses ca. 0,14 Ampere). Danach wird in zwei Stufen das Heizelement zugeschaltet, das zuerst auf halber Leistung läuft (Höhe des Stromflusses ca. 3,9 Ampere) und dann auf voller Leistung betrieben wird (Höhe des Stromflusses ca. 7,7 Ampere).

Sobald der Stromfluss über 7 Ampere liegt, sendet die HSP ein Signal an den *WAGO*-Controller, der die Steuerung der Heizungsanlage darstellt und nun den Schrittmotor ansteuert. Dieser symbolisiert ein Regelventil, das nun entsprechend der steigenden Heizleistung aufgedreht wird, um den Durchfluss zu erhöhen. Diese Regelung entspricht zwar nicht zu 100 % dem Prozess, Sollwerte, die über die Cloud-Plattform erstellt wurden, zurückzuspielen, zeigt aber die Regelfähigkeit der HSP, die während des Versuches bewiesen wurde, da die Regelungen wie programmiert durchgeführt wurden.

Gleichzeitig sammelt eines der entwickelten Skripte die sekündlichen Messdaten und fasst diese in *.csv*-Dateien von jeweils einer Minute zusammen. Das ebenfalls nach dem Hochfahren der HSP gestartete *Python*-Skript prüft minütlich, ob eine neue *.csv*-Datei erstellt wurde. Ist dies der Fall, wird die Datei in *Blockblobs* umgewandelt und auf den *IoT Hub* hochgeladen, was - wie in Abbildung 34 zu sehen ist - während des Versuchs ohne Probleme funktioniert hat.

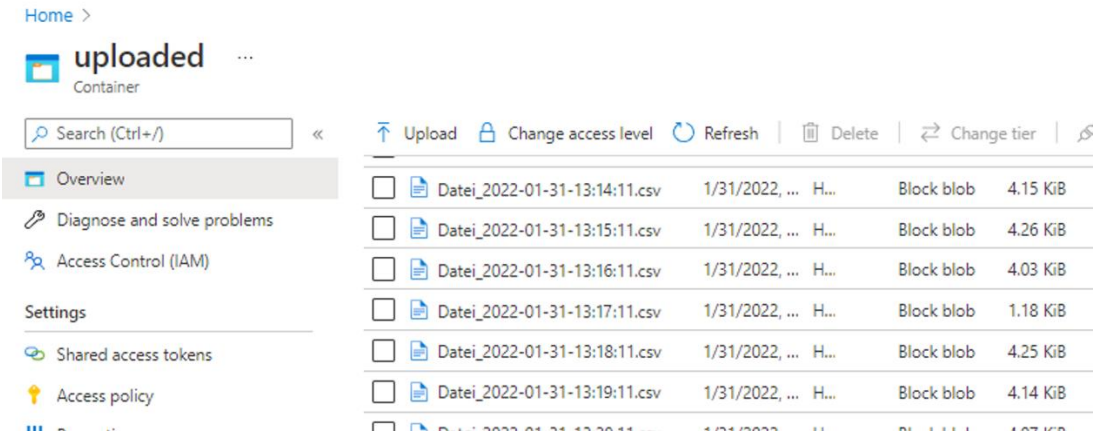


Abbildung 34 - Beispiel für während des Tests hochgeladene *Blockblobs* im *Azure IoT Hub*

Zur Kontrolle, ob alle Werte korrekt übertragen wurden, werden die *Blockblobs* im *IoT Hub* geöffnet und ihr Inhalt eingesehen. Auch hier zeigt sich, dass die Programmierung wie gewünscht funktioniert, da wie in Abbildung 35 zu sehen ist, sämtliche Werte, die auf dem *SQL*-Server zu finden sind, auch hier in dem entsprechenden *Blockblob* vorhanden sind.

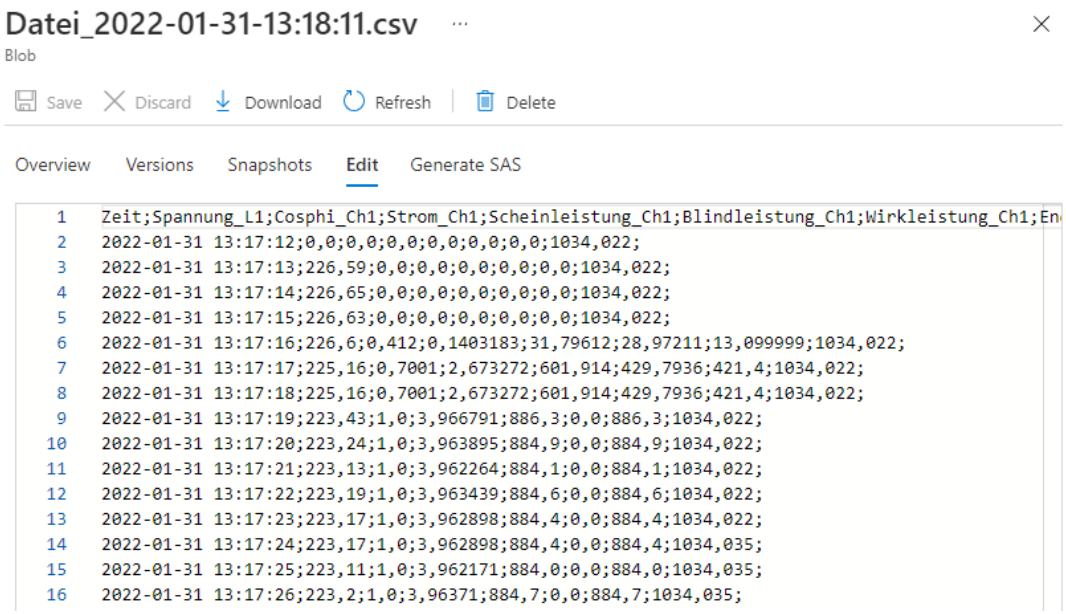


Abbildung 35 - Beispiel für im Versuch in einem *Blockblob* gespeicherte Werte

8.4 Fazit der Validierung der Arbeitsergebnisse

Wie bereits bei der Beschreibung der Versuchsdurchführung erwähnt, funktionierten die einzelnen Softwarekomponenten wie geplant, sodass zuerst Messdaten aufgenommen und diese nach der Speicherung in *.csv*-Form in *Blockblobs* umgewandelt und in den *IoT Hub* hochgeladen wurden. Auch das Senden von Sollwerten an die symbolische Anlagensteuerung konnte durchgeführt werden, sodass alle an die HSP gestellten Funktionen und somit die Arbeitsergebnisse validiert werden konnten.

9 VERGLEICH DER ARBEITSERGEBNISSE MIT MARKTÜBLICHEN LÖSUNGEN

Nachdem die im Rahmen dieser Arbeit entwickelten Lösungskomponenten vorgestellt und innerhalb eines Versuchsaufbaus validiert wurden, werden sie nun noch einmal mit den in Kapitel 5 vorgestellten Lösungen verglichen, die während der Marktrecherche gefunden wurden. Zu diesem Zweck werden die Arbeitsergebnisse zunächst noch einmal zusammengefasst. Im Anschluss daran erfolgt die Festlegung der Kriterien, die für den zu ziehenden Vergleich benötigt werden, um eine Beurteilung zu ermöglichen. Nach einem tabellarischen Vergleich der Arbeitsergebnisse mit den marktüblichen Lösungen schließt das Kapitel den Vergleich mit einem Fazit ab.

9.1 Zusammenfassung der Arbeitsergebnisse

Zentrales Ergebnis dieser Arbeit ist die Entwicklung einer prototypischen HSP, die für die Aufnahme, Speicherung und Weiterleitung von Energie- und Betriebsdaten eingesetzt werden soll. Als Basis dient ein *UP Xtreme Lite* Computer, der nicht nur besonders leistungsstark, sondern auch kompakt ist und industriellen Anforderungen für den Einbau und Betrieb in einem Schaltschrank genügt. Aufgrund der Leistungsfähigkeit des Gerätes und der Möglichkeit, diese durch Verwendung dedizierter Grafikeinheiten noch zu erhöhen, können zumindest theoretisch auch lokal aufwändige Berechnungen durchgeführt werden.

Durch die Verwendung eines *Debian*-basierten Betriebssystems wird sichergestellt, dass das System flüssig und zuverlässig läuft und auch ein Stromausfall keine Probleme hinterlassen dürfte. Ist die Stromversorgung wieder hergestellt, startet sich die HSP automatisch neu und führt die implementierten Skripte aus.

Mithilfe der verwendeten Softwarekomponenten wird die lückenlose Datenaufnahme und der vollständige Versand der Messdaten realisiert. Dabei können die Daten via *Ethernet* unter Verwendung des *Modbus-TCP*-Protokolls von Anlagen und Maschinen abgegriffen und mithilfe des *Blockblob*-Moduls sicher an den *Azure IoT Hub* weitergeleitet werden.

Aufgrund der Speicherung der Daten auf einem lokalen *SQL*-Server stehen diese nicht nur für Analysen und Visualisierungen auf dem Gerät selbst, sondern auch in dem kompletten Netzwerk zur Verfügung. Hier ermöglicht es die verwendete Hardware sogar, die Konnektivität auf zwei Netzwerke und die Kommunikation über mobile Daten zu erweitern.

Der Rückversand von berechneten Sollwerten wurde betrachtet und kann ohne großen Mehraufwand auf Basis der entwickelten HSP umgesetzt werden. Hier ist eine Implementierung jedoch von den Rahmenbedingungen abhängig, die eine mögliche Cloud-Plattform zur Erstellung prädiktiver Analysen bieten kann.

9.2 Festlegung der Kriterien für den Vergleich der Arbeitsergebnisse mit marktüblichen Lösungen

Basierend auf den für die verwendete Hardware und die Softwarekomponenten festgelegten Anforderungen werden nun die Kriterien für den Vergleich festgelegt. Dabei werden diese in „Muss“- und „Kann“-Bedingungen unterteilt, die wie folgt aufgeschlüsselt sind:

Muss-Kriterien:

- Nutzung von *Linux* als Betriebssystem
- Möglichkeit, die Hardware notfalls mit einem vom Hersteller angebotenen Adapter an einer Hutschiene montieren zu können
- Maximale Abmessungen von 200x150x100mm
- Zwei getrennt voneinander konfigurierbare *Ethernet*-Anschlüsse
- Ausreichend Speicherplatz, um die Datenmenge zu bevorraten, die in sechs Monaten abgerufen wird (Berechnung in Kapitel 5.2)
- Verwendung eines leistungsfähigen Prozessors (Top 10 % marktüblicher Industrie-Computer) und mindestens 8 GB Arbeitsspeicher
- Verwendung des *Modbus-TCP*-Protokolls
- Zwischenspeicherung der Messdaten bei Netzwerkausfall und lückenloser Nachversand bei wiederhergestellter Konnektivität
- Eine mindestens sekündliche Aufnahme- beziehungsweise Logging-Rate
- Möglichkeit, Sollwerte an Anlagensteuerungen zu geben, um eine Regelung dieser zu realisieren

Kann-Kriterien:

- Möglichkeit eine *SIM*-Karte zu verwenden, sodass bei fehlender *Ethernet*-Verbindung zum Internet über mobile Daten kommuniziert werden kann
- Betrieb der Hardware bei Temperaturen zwischen -20 und +60 °C

9.3 Vergleichende Darstellung der Arbeitsergebnisse mit marktüblichen Lösungen

Die Zusammenfassung des Vergleichs unter Berücksichtigung der zuvor festgelegten Kriterien ist in Tabelle 16 dargestellt. Um eine Bewertung der unterschiedlichen Produkte zu ermöglichen, wurden Punkte wie folgt verteilt: Kriterium erfüllt = +1 Punkt, Kriterium nicht erfüllt = +0 Punkte, unklar / programmierbar = +0,5 Punkte. Im Anschluss wurde die Punktzahl der Muss-Kriterien mit Faktor 2 multipliziert und zu der Punktzahl der Kann-Kriterien addiert.

Tabelle 16 - Vergleich der Arbeitsergebnisse mit marktüblichen Lösungen (sortiert aufsteigend nach Punktzahl)

		Muss-Kriterien											Kann-Kriterien	
Produkt	**Punkte**	**Linux-fähig**	**Hutschienen-montage**	**Maximal 200x 150x100mm**	**2x Ethernet**	**Speicherung für 6 Monate**	**Leistungsfähig**	**Modbus-TCP**	**Nachversand**	**Logging-Rate (1/s)**	**Regelfähig**	**SIM-Karten-Adapter**	**Betrieb bei -20 bis +60 °C**
MEMORY Hi-LOGGER LR8450	**4**	un-klar	Nein	Nein	Nein	Nein	Nein	Nein	unklar	Ja	Nein	Nein	Nein
gridBox	**6**	Ja	Nein	Ja	Nein	Nein	Nein	Nein	Ja	Nein	Nein	Nein	Nein
HOBO RX3000	**6**	Nein	Nein	Nein	Nein	Nein	Nein	Nein	Ja	Ja	Nein	Ja	Ja
BDL 50	**10**	Ja	Ja	Ja	Ja	Nein	Nein	Nein	Ja	Nein	Nein	Nein	Nein
7KN POWER-CENTER 3000 V1	**11**	Ja	Ja	Ja	Ja	Nein	Nein	Ja	unklar	Nein	Nein	Nein	Nein
Expert Log-ger	**12**	Ja	Ja	Nein	Ja	Nein	Nein	Ja	Ja	Ja	Nein	Nein	Nein
EasyNet	**13**	Nein	Ja	Ja	Nein	Ja	Nein	Ja	unklar	Ja	Nein	Ja	Ja
CONTROL-LER PFC200	**17**	Ja	Ja	Ja	Ja	Ja	Nein	Ja	program-mierbar	Ja	Ja	Nein	Nein
HSP	**21**	Ja	Ja	Ja	Ja	Ja	Ja	Ja	Ja	Ja	Ja	Ja	Nein

9.4 Fazit des Vergleichs

Die an die entwickelte HSP gestellten Anforderungen sind vielseitig und die zuvor dargestellte Gegenüberstellung zeigt, dass es aktuell kein frei verfügbares Produkt auf dem Markt gibt, das diesen Vorgaben zu 100 % entspricht. Mit insgesamt 17 von 22 möglichen Punkten erreicht der *Controller PFC200* von *WAGO* die höchste Punktzahl von den acht Datenloggern anderer Anbieter. Er verfügt allerdings lediglich über einen einkernigen *ARM Cortex-A8* Prozessor, der mit 1 Ghz getaktet ist, und einen 512 Megabyte Arbeitsspeicher, der zusammen mit der CPU lediglich für eine geringe Leistungsfähigkeit sorgt. [39] Hinzu kommt, dass der Nachversand der Daten bei Netzwerkausfall nicht standardmäßig implementiert ist und vom Nutzer eigenständig programmiert werden muss. Inwiefern dies unter Verwendung des *e!COCKPIT* von *WAGO* möglich wäre, ist zu klären. Ein weiterer Nachteil ist das Fehlen eines *SIM*-Karten-Adapters, auch wenn dieser nicht unter die Muss-Kriterien fällt. Das nächstbeste Modell ist der *EasyNet* von *4Next Technology Systems*, der mit 13 Punkten unter 60 % der geforderten Merkmale und Möglichkeiten bietet.

Der Vergleich zeigt, dass die in dieser Arbeit entwickelte HSP allen hier betrachteten Datenloggern weit überlegen ist und ein großes Potenzial mitbringt, sich im Markt zu etablieren. Sollte das geplante Projekt zum Einsatz prädiktiver Regelmechanismen umgesetzt werden, bietet die im Rahmen dieser Abschlussarbeit konzipierte Lösung noch weitere Vorteile. Dazu gehören beispielsweise die Steuerung der Anlagen und Maschinen auf Basis von Prognosen und damit einhergehende Energieeinsparungen.

10 HANDLUNGSEMPFEHLUNGEN

Die im Rahmen dieser Untersuchung entwickelte HSP soll nach Weiterentwicklung des Prototyps und Erreichen der Serienreife für den Einsatz in Industrieunternehmen verwendet werden. Damit dieser Schritt vollzogen werden kann und das Verhalten der Hard- und Softwarelösung auch in realen Tests wie gewünscht abläuft, wurden verschiedene Handlungsempfehlungen formuliert. Diese geben Hinweise auf mögliche Schwierigkeiten und Verbesserungspotenzial, um die Wahrscheinlichkeit von Komplikationen im realen Betrieb zu reduzieren.

10.1 Mögliche Schwierigkeiten bei der Einbindung der HSP in die Struktur eines Unternehmens

In der Regel verwenden Unternehmen, die über eine informationstechnische Infrastruktur verfügen, eine soft- oder hardwaretechnische *Firewall*, die die eingebundenen Computer vor virtuellen Attacken schützt (für weitere Informationen zur Funktion einer *Firewall* siehe Kapitel 6.3.2). Bei der hier entwickelten HSP handelt es sich zunächst einmal um einen Fremdkörper in dem „Ökosystem“ dieser Infrastruktur, weshalb es möglich ist, dass die Zugriffe auf angesteuerte Ports und Geräte als schadhaft erkannt und blockiert werden. Dies führt dazu, dass keine Verbindung zu den gewünschten Anlagen und Systemen hergestellt werden kann. Aus diesem Grund sollte, wenn möglich, das für die informationstechnische Struktur zuständige Personal des Unternehmens mit in die Integration einbezogen werden.

Wie bereits in Kapitel 6.1 erwähnt, entscheiden sich viele deutsche Unternehmen aus Sorge um ihre Daten gegen Cloud Computing. Deshalb ist es nicht unwahrscheinlich, dass sich ein Unternehmen zwar für die Verwendung einer HSP als Datenlogger, jedoch gegen die Verarbeitung der Messdaten in einer Cloud-Anwendung ausspricht. Folglich müssen die Berechnungen zur Realisierung der prädiktiven Regelung lokal auf dem Gerät stattfinden. Zu diesem Zweck ist es notwendig, dass die Softwarekomponenten, die für die Datenanalysen und die Erstellung der Prognosen in der Cloud zum Einsatz kommen, auch auf der HSP verwendet werden können. Dies wird eine Anpassung der Software und potenzielle Schwierigkeiten nach sich ziehen. Zudem ist trotz der großzügigen Auslegung der Hardware-Komponenten unklar, ob die Rechenleistung tatsächlich für die angestrebten Berechnungen ausreichend ist. Hier müssen womöglich Einbußen in Bezug auf die Funktionalität akzeptiert werden.

Eine weitere potenzielle Problemquelle ist die Verbindung der HSP zu den Anlagen und Maschinen, deren Daten abgerufen und verarbeitet werden sollen. Neben den bereits erwähnten Schwierigkeiten bezüglich der *Firewall* muss für jedes anzusteuernde Gerät klar sein, auf welchen *Modbus*-Registeradressen welche der benötigten Informationen liegen. Hier kann es aufgrund veralteter oder fehlender technischer Dokumentation und Informationen zu Problemen kommen. Sobald der in Kapitel 7.3.1 beschriebene Offset, der die Nummer des Startregisters für die Datenabfrage festlegt, um den Wert 1 geringer oder höher ist, werden andere als die gewünschten Daten gelesen oder geschrieben.

In Abhängigkeit von der Beschaffenheit des Standortes zur Datenaufnahme besteht zwar die Möglichkeit, ein Offline-*Ethernet*-Netzwerk für die Verbindung der HSP zu den Anlagen aufzubauen, aber womöglich kann keine Verbindung zum Internet geschaffen werden. Dies ist vor allem in abgelegenen Regionen oder großen Gebäudekomplexen, die über eine unzureichende informationstechnische Infrastruktur verfügen, wahrscheinlich. Zwar bietet die ausgewählte Hardware die Verwendung einer *SIM*-Karte für die Nutzung von mobilen Daten an, aber auch dies kann gerade in entlegenen Gegenden oder Kellerräumen nicht umsetzbar sein. In solch einem Fall wäre zu überlegen, ob die Daten regelmäßig manuell exportiert werden sollten, oder ob sich der Einsatz der Technologie auf eine offline Speicherung und Nutzung der Daten beschränken sollte.

10.2 Anbindung der HSP an Cloud-Anwendung

Das von *manageE* angedachte Projekt sieht die Anbindung der HSP an eine Cloud-Anwendung vor, die neben der Erstellung der Prognosen auch Sollwerte generiert, die an die Anlagensteuerungen zurückgesendet werden. Damit dies ermöglicht werden kann, müssen zunächst geeignete Schnittstellen ermittelt werden, die eine zuverlässige Verbindung von der Cloud-Anwendung zu der HSP erlauben. Dies betrifft nicht nur den Abruf der im *Azure IoT Hub* gespeicherten *Blockblobs*, sondern auch den Rückversand der Sollwerte. Geeignete Schnittstellen können hier wie in Kapitel 7.3.4 bereits erwähnt *SQL*-Befehle, Befehle aus den *Blockblob*-Bibliotheken oder andere Tools sein.

Zusätzlich zur Festlegung der Schnittstellen muss vor der Übertragung der Messdaten geklärt werden, welche Informationen genau abgefragt und versandt werden sollen. Womöglich sind neben der Anbindung von Informationsquellen für meteorologische Daten auch Schnittstellen zu einer sogenannten „Enterprise-Resource-Planning“-Software (enthält beispielsweise Produktionsdaten) sinnvoll, die das Abfragen betrieblicher Daten erlauben. Hier muss

neben der Definition der zu sammelnden Daten auch festgelegt werden um welche Datentypen es sich im Einzelnen handelt, damit diese korrekt verarbeitet werden können.

10.3 Erweiterung und Verbesserung des Funktionsumfangs

Als Ergänzung der bisher genannten Handlungsempfehlungen werden nachfolgend fünf weitere Punkte genannt, mit deren Hilfe der Funktionsumfang der HSP erweitert oder verbessert werden kann.

10.3.1 Ermittlung und Darstellung von Systemkennzahlen

Da die gesammelten Informationen lokal auf der entwickelten Plattform gespeichert werden, können durch Erweiterung der Programmierung bereits erste Analysen der Messdaten erfolgen. Diese erlauben beispielsweise die Ermittlung und Darstellung von Systemkennzahlen, sodass direkte Aussagen über die Effizienzen der Systeme getroffen werden können (z.B. benötigte Energie pro produziertem Stück). Dies schließt auch Berechnungen zu erzielten Energie- und CO2-Einsparungen ein, die dann in übergeordneten Energiemanagementsystemen weiterverarbeitet und unter anderem für Unternehmensberichte verwendet werden können. Mithilfe der Visualisierungsumgebung von *CODESYS* können diese Werte jedoch auch lokal auf dem Gerät dargestellt und in Form einer Webseite über das Netzwerk eingesehen werden.

10.3.2 Einführung einer Ausnahmebehandlung

Während der Entwicklung der prototypischen HSP traten zwar bereits verschiedene Fehler auf, gegen die Maßnahmen ergriffen wurden, es kann allerdings jederzeit zu weiterem unvorhergesehenem Verhalten kommen. Damit solche Einzelfälle nicht zu einem ungewollten Stopp der Softwarekomponenten führen, ist es sinnvoll, eine Ausnahme- beziehungsweise Fehlerbehandlung in den Programmcode zu implementieren. Diese wird initialisiert, sobald festgelegte Werte oder Rückmeldungen in einem Skript nicht den Erwartungen entsprechen und verhindert dadurch einen Programmabsturz. Für die Entwicklung einer solchen Funktion sollte viel Zeit in das Testen des Systems investiert werden, um möglichst viele Fehler bereits vor dem Einsatz beim Kunden zu entdecken und Gegenmaßnahmen ergreifen zu können. Sobald das Produkt im Betrieb ist, sollte die Ausnahmebehandlung auf Basis der im Feld gemachten Erfahrungen stetig weiterentwickelt und optimiert werden.

10.3.3 Implementierung eines Betriebsstatus-Loggers

Ergänzend zu der im vorangegangenen Abschnitt beschrieben Einführung einer Ausnahmebehandlung sollte ein Großteil der Programmschritte im Betrieb dokumentiert werden. Dies kann durch die Implementierung eines Loggers erfolgen, der nach der Ausführung verschiedener Operationen den Betriebsstatus aufnimmt. Bei diesen Informationen zum Betriebsstatus kann es sich zum Beispiel um gefundene oder versandte Dateien, empfangene Werte oder aufgetretene Fehler handeln. Die durch diesen Logger aufgenommenen Daten können dann - ergänzt durch den jeweiligen Zeitstempel - beispielsweise in einer Textdatei festgehalten werden, sodass im Nachhinein das gezeigte Verhalten der Hard- und Software nachvollzogen werden kann.

10.3.4 Einrichtung einer SAS-Authentifizierung

Falls das elektronische Datenverarbeitungssystem des Unternehmens eine Sicherheitslücke aufweist, steigt die Möglichkeit, dass Cyber-Attacken die Firewall des Unternehmens überwinden und die Messdaten der HSP gefährdet sind. Deshalb sollten zusätzliche Maßnahmen ergriffen werden, um die HSP, die darauf gespeicherten Daten und die Datenverbindungen besser zu schützen. Eine solche Maßnahme ist beispielsweise die Einrichtung eines sogenannten *Shared Acess Signature*-Services für die verwendeten Container und das *Blockblob*-Modul. Dieser Service bietet durch Einführung einer weiteren Authentifizierungsabfrage die Möglichkeit, den Zugang zu den genannten Softwarestrukturen stärker einzuschränken. [86]

10.3.5 Berücksichtigung der Zeitumstellung

Die prototypische HSP speichert neben den aufgenommenen Messdaten auch den jeweiligen DZS ab, zu dem die Informationen abgefragt wurden. Der im DZS enthaltene Zeitstempel orientiert sich dabei an der Systemzeit, die nicht zwangsläufig der Ortszeit entspricht. Da die entwickelte Lösung vornehmlich in Deutschland eingesetzt werden soll, muss die halbjährliche Zeitumstellung zwischen Sommer und Winterzeit berücksichtigt werden, damit keine Differenzen bezüglich der zeitlichen Komponente der Messdaten entstehen. Dieser Umstand muss vor allem bei der Abfrage und Abspeicherung der Daten beachtet werden.

11 DISKUSSION

Der Prozess zur Entwicklung der in dieser Arbeit beschriebenen Lösungen war nicht ausschließlich von Erfolgen, sondern auch von zahlreichen Rückschlägen geprägt. Ein Großteil der Zeit wurde damit verbracht, tiefer in die unterschiedlichen Programmiersprachen und -umgebungen einzusteigen und nach ähnlichen Anwendungsfällen zu suchen, die nur selten gefunden werden konnten. Zuweilen traten Situationen auf, in denen ein Teil eines Skripts zu funktionieren begann, ein anderer zuvor laufender Code-Teil jedoch nicht mehr wie geplant arbeitete. Hier wurde viel Zeit mit der Fehler- und Lösungssuche verbracht.

Im Allgemeinen kann die Fehlersuche bei der Programmierung recht zeitaufwändig und ineffizient sein, da es diverse Fehler gibt, die in verschiedene Richtungen interpretiert werden können. Während manchmal schon die Eingabe der Fehlermeldung in eine Suchmaschine zur richtigen Lösung führt, sind in anderen Situationen Recherchen von mehreren Stunden nötig, um die Ursachen des Fehlers zu entschlüsseln.

Gerade in Bezug auf die Software *CODESYS* ist eine frei verfügbare Dokumentation spärlich vorhanden und meist veraltet, sodass sowohl die Einarbeitung in das Programm als auch die Arbeit mit diesem erschwert war. Auch wenn Informationen bezüglich des *Azure IoT Hubs* und den damit verbunden Diensten und Programmiermöglichkeiten vermehrt zur Verfügung stehen, fehlen teils essenzielle Erklärungen für Laien in diesem Bereich. Vor Beginn der Arbeit war die Annahme getroffen worden, dass ausreichend Dokumentation vorhanden und nutzbar ist und im Notfall der Support der Hersteller in Anspruch genommen werden kann. Diese Annahme beruhte auf den bisherigen Erfahrungen im Bereich der Programmierung, konnte aber in diesem Fall nicht bestätigt werden, da Support-Anfragen meist nur unzureichend oder gar nicht beantwortet wurden. Sowohl die Nutzung bekannter Foren, die generell zum Austausch von Entwicklern dienen, als auch die Nutzung der offiziellen Austauschmöglichkeiten der Softwarehersteller waren den Großteil der Zeit nur von geringer Hilfe. Dies änderte sich erst als der Lösungsansatz mit der Programmiersprache *Python* begonnen wurde, da es hier eine sehr aktive Community gibt.

Ein weiterer Punkt, der zu Beginn unterschätzt wurde, war die Vorbereitung der Hardware zur Nutzung der benötigten Softwarekomponenten, da diese teilweise das vorherige Aufspielen anderer Programme voraussetzen. Die Installation dieser weiteren Software führte jedoch zur Erhöhung der Komplexität und dadurch vermehrt zu neuen Problemen. Hier fehlte häufig grundlegendes

Wissen über die Arbeitsweise und die Funktionalitäten der zusätzlich verwendeten Strukturen, sodass das Lösen von auftretenden Problemen sich ebenfalls als sehr zeitaufwändig zeigte.

Die Schlussfolgerung daraus ist, dass diese Punkte bei einem zukünftigen Projekt, in dem die Verwendung von Programmiersprachen und -umgebungen angedacht ist, zu denen keine oder kaum Vorerfahrung vorhanden ist, bereits im Vorfeld berücksichtigt werden sollten. Alternativ kann das Vorhandensein von versierten Ansprechpartnern, die viel Erfahrung mit den genutzten Programmen haben, im direkten Arbeitsumfeld von Vorteil sein. Diese können bei auftretenden Problemen unterstützen und Erklärungen oder sogar Lösungsansätze geben. Hätte vor Beginn der Programmierung eine strukturierte Einarbeitung von mehreren Monaten in die genutzten Softwarekomponenten stattgefunden, wären viele Probleme vermutlich nicht aufgetreten oder hätten mit geringerem Zeitaufwand gelöst werden können. Eine solche intensive Auseinandersetzung mit den genutzten Strukturen vorab hätte jedoch den Rahmen dieser Untersuchung gesprengt.

Nichtsdestotrotz wurden ungeachtet aller Hindernisse und unvorhergesehenen Probleme im Endeffekt die gewünschten Ergebnisse erreicht, die einer Validierung standhalten konnten. Gerade die Arbeit mit *Python*, deren Anwendung auf bisherigen Erfahrungen mit der Programmiersprache beruhte, zeigte sich als verhältnismäßig unkompliziert und flüssig. Dies ist jedoch unter anderem auf den bereits erwähnten leicht zugänglichen und verlässlichen Austausch mit anderen Entwicklern zurückzuführen.

Die im Rahmen dieser Arbeit entwickelten Lösungen heben sich von den in Kapitel 4 genannten Forschungsprojekten ab, da nicht nur sämtliche Betriebsdaten aufgenommen, sondern auch verschiedene Energiesenken und -quellen angebunden werden können. Gerade letzteres ist ein entscheidender Vorteil, da die vorgestellten Projekte beispielsweise Energiespeicher nicht berücksichtigten. Zudem hat die hier entworfene HSP den Vorteil, dass sie über eine Vielzahl möglicher Schnittstellen verfügt, die es ihr erlauben, mit diversen Anlagen, Steuerungen und Energiemanagementsystemen zu kommunizieren. Der dadurch generierte Mehrwert ist vor allem bei einem Einbau in bestehende Strukturen und Bestandsgebäude nicht zu unterschätzen.

12 FAZIT

In dieser Arbeit wurde die lückenlose Aufnahme und Speicherung energetisch relevanter Daten von Anlagen und Maschinen untersucht und in Form einer prototypischen Hard- und Softwareplattform umgesetzt. Da die von Unternehmen benötigte Energie den Großteil des deutschen Gesamtenergieverbrauchs ausmacht, ist dies von besonderer Bedeutung, um weitere Energieeinsparungen, eine Energieeffizienzsteigerung und die geplanten Klimaziele Deutschlands zu erreichen.

Essenziell für die Umsetzung des von *manageE* angedachten Projektes ist die Verwendung einer Hardware, die als Schnittstelle für die Kommunikation zwischen den Anlagen und Maschinen eines Unternehmens und der noch zu entwickelnden Cloud-Anwendung für modellprädiktive Regelung dient. Aus diesem Grund wurde im Rahmen dieser Masterarbeit zur Vorbereitung des möglichen Projektes die prototypische Hard- und Softwareplattform entwickelt.

Die für die Konzipierung dieser Plattform benötigten Anforderungen und Ziele wurden definiert, um darauf aufbauend geeignete Lösungen zu entwickeln. Dabei wurde sowohl auf Basis einer Hardware-Recherche ein leistungsfähiger Computer (*UP Xtreme Edge*), der alle geforderten Charakteristika erfüllt, ausgewählt und bestellt, als auch die gewünschten Funktionalitäten auf dem Gerät implementiert.

Der Fokus bei der Auswahl und Entwicklung der Softwarekomponenten lag darauf, die folgenden Aufgaben zu realisieren: Die lückenlose Messdatenaufnahme verschiedener Geräte, die lokale Speicherung der Daten, die sichere Datenübertragung an eine IoT Plattform und den Rückversand von Sollwerten zu ermöglichen. Da *manageE* zertifizierter *Microsoft Silver Partner* ist, woraus sich einige Vorteile für das Unternehmen ergeben, und aufgrund des umfassenden Funktionsumfangs, wurde der *Azure IoT Hub* von *Microsoft* für die Umsetzung dieser Arbeit ausgewählt. Die entwickelte Software-Lösung besteht aus den folgenden vier Teilen:

1. Einem *CODESYS*-Programm für die Datenaufnahme über *Modbus-TCP*, die lokale Speicherung der Daten auf dem *SQL*-Server, die Erzeugung von *.csv*-Dateien und den Rückversand der Sollwerte,

2. der *Azure IoT Edge Runtime*, mit deren Hilfe ein *Microsoft SQL*-Server und *Blockblob*-Modul betrieben werden,

3. einem *Python*-Skript zur Verarbeitung der *.csv*-Dateien zu *Blockblobs* und deren Versand und

4. dem eingerichteten *Azure IoT Hub* für die Speicherung der *Blockblobs* und als Schnittstelle zwischen der HSP und der noch zu entwickelnden Cloud-Anwendung für die Erstellung der Prognosen.

Die korrekte Funktion dieser Softwarekomponenten konnte mithilfe eines Versuchsaufbaus, mit dem die energetische Messung einer Energiesenke in einem Unternehmen und ihre Regelung simuliert wurden, validiert werden.

Durch den Vergleich mit marktüblichen Produkten, die innerhalb einer Recherche ermittelt wurden, zeigte sich, dass keiner der Datenlogger den hohen Anforderungen und dem Funktionsumfang der HSP entspricht. Die Gegenüberstellung mit Systemen für die Energiedatenerfassung und -analyse ergab ein ähnliches Bild, da lediglich ein Unternehmen mit der Verwendung prädiktiver Analysen warb. Sollte das potenzielle Projekt umgesetzt werden, ist deshalb die Wahrscheinlichkeit hoch, dass sich die HSP in Verbindung mit der modellprädiktiven Cloud-Anwendung schnell etablieren wird. Hier sind vor allem ein zuverlässiges Produkt und eine gute Marketingstrategie vonnöten.

13 AUSBLICK

Nachdem die prototypische Umsetzung der gewünschten Funktionalitäten erfolgreich war, sollte das Unternehmen *manageE* nun das geplante Projekt umsetzen und die HSP zu einem serienreifen Produkt weiterentwickeln. Zu diesem Zweck ist es sinnvoll, unter anderem die in Kapitel 10 beschriebenen Handlungsempfehlungen zu berücksichtigen. Außerdem gilt es nun, das Produkt im Feld, das heißt in einem realen Unternehmen, einzusetzen.

Nicht klar ist in diesem Zusammenhang, wie lange es *manageE* möglich sein wird, das spätere Serienprodukt zu vermarkten, da die Entwicklung des SMGW weiter fortschreitet und auch die gesetzlichen Rahmenbedingungen geändert werden. Sollte hier im Laufe der kommenden Jahre eine Zertifizierungspflicht für sämtliche Datenaufnahmeprozesse innerhalb deutscher Unternehmen beschlossen werden, könnten neue Herausforderungen auf *manageE* zukommen. Eine solche Pflicht könnte sogar das Aus des Produktes bedeuten, da wie in Kapitel 4.4 geschildert der Aufwand einer solchen Zertifizierung zumindest aktuell die Kapazitäten des Unternehmens überschreitet.

Aus Sicht der Forschung stellt vor allem das potenzielle Projekt zur Nutzung modellprädiktiver Regelung eine interessante Gelegenheit dar, um diese Technologie zusammen mit einem Industriepartner zu entwickeln und den tatsächlichen Nutzen zu verifizieren. Eine solche Entwicklung besitzt das Potenzial, einen neuen Meilenstein für das Energiemanagement und die effizientere Regelung von Anlagen und Maschinen zu setzen. Gerade aufgrund des hohen Anteils der Industrie am deutschen Gesamtenergieverbrauch könnten somit große Fortschritte in Richtung der politisch gesetzten Klimaziele gemacht werden.

VIII LITERATURVERZEICHNIS

[1] Bundesregierung, *Ausstieg aus der Kernkraft.* [Online]. Verfügbar unter: https://www.bundesregierung.de/breg-de/themen/energiewende/energie-erzeugen/ausstieg-aus-der-kernkraft-394280 (Zugriff am: 8. Dezember 2021).

[2] Bundesministerium für Wirtschaft und Klimaschutz, *Unsere Energiewende: sicher, sauber, bezahlbar.* [Online]. Verfügbar unter: https://www.bmwi.de/Redaktion/DE/Dossier/energiewende.html (Zugriff am: 9. Dezember 2021).

[3] M.-L. Stadler, „Big Data“, *mindsquare*, 7. Juli 2021, 2021. [Online]. Verfügbar unter: https://mindsquare.de/knowhow/big-data/. Zugriff am: 3. Februar 2022.

[4] M. Günther, Hg., *Energieeffizienz durch Erneuerbare Energien*. Wiesbaden: Springer Fachmedien Wiesbaden, 2015.

[5] ENGIE Deutschland, *Mehr Effizienz durch Energiemonitoring.* [Online]. Verfügbar unter: https://www.engie-deutschland.de/de/magazin/mehr-effizienz-durch-energiemonitoring (Zugriff am: 26. Dezember 2021).

[6] Fraunhofer-Allianz Energie, *Energiedatenanalyse.* [Online]. Verfügbar unter: https://www.energie.fraunhofer.de/de/geschaeftsbereiche/energie-digital/energiedatenanalyse.html (Zugriff am: 26. Dezember 2021).

[7] R. Dittmar und B.-M. Pfeiffer, *Modellbasierte prädiktive Regelung: Eine Einführung für Ingenieure*. München: Oldenbourg, 2004.

[8] Statista, *Energieverbrauch in Deutschland nach Sektoren 2020 | Statista.* [Online]. Verfügbar unter: https://de.statista.com/statistik/daten/studie/321069/umfrage/energieverbrauch-in-deutschland-nach-sektoren/ (Zugriff am: 26. Dezember 2021).

[9] M. Blesl und A. Kessler, *Energieeffizienz in der Industrie*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2017.

[10] consulting.de, *Energiemanagement im Unternehmen – Warum und wie?* [Online]. Verfügbar unter: https://www.consulting.de/hintergruende/fachartikel/einzelansicht/energiemanagement-im-unternehmen-warum-und-wie/ (Zugriff am: 26. Dezember 2021).

[11] Deutsche Energie-Agentur GmbH, *Energieaudit und Energiemanagement.* [Online]. Verfügbar unter: https://www.dena.de/themen-projekte/energieeffizienz/unternehmen/energieaudit-und-energiemanagement/ (Zugriff am: 27. Dezember 2021).

[12] Umweltbundesamt, *Energiemanagement mit System.* [Online]. Verfügbar unter: https://www.umweltbundesamt.de/themen/wirtschaft-konsum/wirtschaft-umwelt/umwelt-energiemanagement/energiemanagement-system#wie-funktioniert-ein-energiemanagement (Zugriff am: 26. Dezember 2021).

[13] F. Wosnitza und H. G. Hilgers, *Energieeffizienz und Energiemanagement*. Wiesbaden: Vieweg+Teubner Verlag, 2012.

[14] WinTotal.de, *Hardware & Software: Unterschied und Zusammenhang einfach erklärt.* [Online]. Verfügbar unter: https://www.wintotal.de/hardware-software/ (Zugriff am: 1. Januar 2022).

[15] ITWissen.info, *Plattform.* [Online]. Verfügbar unter: https://www.itwissen.info/Plattform-platform.html (Zugriff am: 1. Januar 2022).

[16] H. Brandt-Pook und R. Kollmeier, *Softwareentwicklung kompakt und verständlich: Wie Softwaresysteme entstehen,* 1. Aufl. Wiesbaden: Vieweg + Teubner, 2008.

[17] ITWissen.info, *Programmablaufplan (PAP).* [Online]. Verfügbar unter: https://www.itwissen.info/Programmablaufplan-PAP-program-flowchart.html (Zugriff am: 2. Januar 2022).

[18] Academic dictionaries and encyclopedias, *Programmbibliothek.* [Online]. Verfügbar unter: https://de-academic.com/dic.nsf/dewiki/1133576 (Zugriff am: 2. Januar 2022).

[19] CODESYS GROUP, *Unternehmen: Zahlen, Daten, Fakten.* [Online]. Verfügbar unter: https://de.codesys.com/unternehmen/zahlen-daten-fakten.html (Zugriff am: 3. Januar 2022).

[20] CODESYS GROUP, „CODESYS Development System V3 — Product Data Sheet V0.0.0.1“. [Online]. Verfügbar unter: https://store.codesys.com/media/n98_media_assets/files/1101000000/3/CODESYS%20Development%20System%20V3_de.pdf

[21] KUNBUS, *Grundlagen digitaler Bussysteme und wesentliche Grundbegriffe - KUNBUS GMBH.* [Online]. Verfügbar unter: https://www.kunbus.de/grundlagen-digitaler-bussysteme-und-wesentliche-grundbegriffe.html (Zugriff am: 9. Januar 2022).

[22] L. Stiegler, „Industrielle Bussysteme Teil 5 – Modbus-TCP“, 2015. [Online]. Verfügbar unter: http://wwwlehre.dhbw-stuttgart.de/~srupp/IBS/05_Modbus-TCP.pdf

[23] G. Serale, M. Fiorentini, A. Capozzoli, D. Bernardini und A. Bemporad, „Model Predictive Control (MPC) for Enhancing Building and HVAC System Energy Efficiency: Problem Formulation, Applications and Opportunities“, *Energies*, Jg. 11, Nr. 3, S. 631, 2018, doi: 10.3390/en11030631.

[24] A. Afram und F. Janabi-Sharifi, „Theory and applications of HVAC control systems – A review of model predictive control (MPC)“, *Building and Environment*, Jg. 72, S. 343–355, 2014, doi: 10.1016/j.buildenv.2013.11.016.

[25] H. Thieblemont, F. Haghighat, R. Ooka und A. Moreau, „Predictive control strategies based on weather forecast in buildings with energy storage system: A review of the state-of-the art“, *Energy and Buildings*, Jg. 153, S. 485–500, 2017, doi: 10.1016/j.enbuild.2017.08.010.

[26] M. Gwerder, D. Gyalistras, C. Sagerschnig, R. S. Smith, D. Sturzenegger, *Final Report: Use of Weather And Occupancy Forecasts For Optimal Building Climate Control: Part II: Demonstration (OptiControl-II).*

[27] Fraunhofer-Institut für Produktionstechnologie, *Das Projekt BIG.* [Online]. Verfügbar unter: https://www.big-gebaeudeautomation.de/de/projekt.html (Zugriff am: 5. Januar 2022).

[28] T. Vollmer, "Erklärung der Ergebnisse des BIG-Projektes", mündlich, Jan. 2022.

[29] Envidatec GmbH, *SIOSTA – Supervision von Industriestandorten mit Open-Source Tensor-Algorithmen.* [Online]. Verfügbar unter: http://siosta.de/ (Zugriff am: 5. Januar 2022).

[30] B. Recht, *Reflections on the Learning-to-Control Renaissance: Plenary Talk at 21st IFAC World Congress 2020 (Virtual).* [Online]. Verfügbar unter: https://www.youtube.com/watch?v=IEZFwh8sw8s (Zugriff am: 5. Januar 2022).

[31] Bundesanzeiger Verlag, *Budesgesetzblatt Jahrgang 2016 Teil I Nr. 43* (Zugriff am: 6. Januar 2022).

[32] Bundesamt für Sicherheit in der Informationstechnik, *Das Smart-Meter-Gateway: Sicherheit für intelligente Netze.* [Online]. Verfügbar unter: http://www.lebenswertes-hochplateau.de/images/pdf/dokumente/Smart-Meter-Gateway.pdf (Zugriff am: 6. Januar 2022).

[33] Ernst & Young GmbH, *Barometer – Digitalisierung der Energiewende: Digitalisierung 2020: Spürbare Fortschritte sowie neue Hindernisse bei Regulierung und Umsetzung.* [Online]. Verfügbar unter: https://www.bmwi.de/Redaktion/DE/Publikationen/Studien/barometer-digitalisierung-der-energiewende-berichtsjahr-2020.html (Zugriff am: 6. Januar 2022).

[34] Bundesamt für Sicherheit in der Informationstechnik, *Zertifizierte Produkte - Intelligente Messsysteme.* [Online]. Verfügbar unter: https://www.bsi.bund.de/DE/Themen/Unternehmen-und-Organisationen/Standards-und-Zertifizierung/Smart-metering/Smart-Meter-Gateway/Zertifikate24Msbg/zertifikate24MsbG_node.html (Zugriff am: 6. Januar 2022).

[35] Statista, *Marktanteile der führenden Betriebssysteme weltweit von Januar 2009 bis September 2021*. [Online]. Verfügbar unter: https://de.statista.com/statistik/daten/studie/157902/umfrage/marktanteil-der-genutzten-betriebssysteme-weltweit-seit-2009/ (Zugriff am: 7. Januar 2022).

[36] IONOS Digitalguide, *Linux vs. Windows Betriebssysteme im Vergleich.* [Online]. Verfügbar unter: https://www.ionos.de/digitalguide/server/knowhow/linux-vs-windows-betriebssystem/ (Zugriff am: 7. Januar 2022).

[37] Siemens AG, *7KN1310-0MC00-0AA8: 7KN POWERCENTER 3000 V1*. [Online]. Verfügbar unter: https://support.industry.siemens.com/cs/pd/1383610?pdti=td&dl=de&lc=de-DE (Zugriff am: 9. Januar 2022).

[38] Berg - Die Energieoptimierer, *Datenlogger - BDL 50.* [Online]. Verfügbar unter: https://www.berg-energie.de/produkte-loesungen/mess-und-kommunikationstechnik/datenlogger/ (Zugriff am: 9. Januar 2022).

[39] WAGO GmbH & Co. KG, *Controller PFC200 (750-8212).* [Online]. Verfügbar unter: https://www.wago.com/de/sps/controller-pfc200/p/750-8212 (Zugriff am: 9. Januar 2022).

[40] 4next S.r.l.s., *EasyNET.* [Online]. Verfügbar unter: https://www.4next.eu/wp-content/uploads/EasyNET_Brochure_EN_1.0.1.pdf (Zugriff am: 9. Januar 2022).

[41] Delphin Technology AG, *Expert Logger.* [Online]. Verfügbar unter: https://www.delphin.de/produkte/mess-und-pruefgeraete/expert-logger.html (Zugriff am: 9. Januar 2022).

[42] gridX, *gridBox - Das IoT-Gateway für Energie.* [Online]. Verfügbar unter: https://de.gridx.ai/edge-services#facts (Zugriff am: 9. Januar 2022).

[43] Onset Computer Corporation, *HOBO RX3000 Remote Monitoring Station Data Logger.* [Online]. Verfügbar unter: https://www.onsetcomp.com/products/data-loggers/rx3000-indoor (Zugriff am: 9. Januar 2022).

[44] HIOKI E.E. Corporation, *MEMORY HiLOGGER LR8450.* [Online]. Verfügbar unter: https://www.hioki.com/global/products/multichannel-data-loggers/multichannel/id_6767 (Zugriff am: 9. Januar 2022).

[45] Beckhoff Automation GmbH & Co. KG, *Die vollständige Energiemesskette: vom Sensor bis in die Cloud.* [Online]. Verfügbar unter: https://www.beckhoff.com/media/downloads/informationsmedien/flyer-energiemesstechnik.pdf (Zugriff am: 13. Januar 2022).

[46] Berg - Die Energieoptimierer, *Energiemanagement.* [Online]. Verfügbar unter: https://www.berg-energie.de/produkte-loesungen/energiemanagement/ (Zugriff am: 13. Januar 2022).

[47] Delphin Technology AG, *Loggito: Lückenlose Datenerfassung an dezentralen Messstellen.* [Online]. Verfügbar unter: https://www.delphin.de/produkte/mess-und-pruefgeraete/loggito.html (Zugriff am: 13. Januar 2022).

[48] Delphin Technology AG, *ProfiSignal 20: Plattformunabhängige Software für die Mess- und Prüftechnik.* [Online]. Verfügbar unter: https://www.delphin.de/produkte/software/profisignal-20.html (Zugriff am: 13. Januar 2022).

[49] Eaton, *Energie-Monitoring-Systeme.* [Online]. Verfügbar unter: http://www.eaton.de/EatonDE/ProdukteundLoesungen/Electrical/Produkte-Dienstleistungen/EnergieverteilungMonitoring/Energie-Mess-undMonitoring-Loesungen/Energie-Monitoring-Systeme/index.htm#tabs-1 (Zugriff am: 13. Januar 2022).

[50] Eaton, *EPMS - Eaton Electrical Power Monitoring System.* [Online]. Verfügbar unter: https://www.eaton.com/us/en-us/digital/brightlayer/brightlayer-data-centers-suite/eaton-epms.html (Zugriff am: 13. Januar 2022).

[51] OPTENDA, *Energiemonitoring: System & Software.* [Online]. Verfügbar unter: https://www.optenda.de/energiemonitoring/ (Zugriff am: 13. Januar 2022).

[52] Bosch Energy and Building Solutions Deutschland, *Energy Platform.* [Online]. Verfügbar unter: https://www.boschbuildingsolutions.com/de/de/news-und-stories/energy-platform/ (Zugriff am: 13. Januar 2022).

[53] Bosch Global, *Energiemanagement-Software: Energy Platform.* [Online]. Verfügbar unter: https://www.bosch.com/de/stories/energiemanagement-software/ (Zugriff am: 13. Januar 2022).

[54] Siemens AG, *SENTRON - Energiemonitoring einfach gemacht.* [Online]. Verfügbar unter: https://cache.industry.siemens.com/dl/files/328/109744328/att_1088673/v1/LV14_112021_DE_202111301728257342.pdf (Zugriff am: 12. Januar 2022).

[55] GROHMANN BUSINESS CONSULTING, *Cloud Computing Marktbarometer Deutschland: Ergebnisbericht 2021.* [Online]. Verfügbar unter: https://www.grohmann-business-consulting.de/cloud-computing-marktbarometer-deutschland/ (Zugriff am: 14. Januar 2021).

[56] PassMark Software, *List of Benchmarked CPUs.* [Online]. Verfügbar unter: https://www.cpubenchmark.net/cpu_list.php (Zugriff am: 14. Januar 2022).

[57] Cpu Specs, *Benchmarks reviews.* [Online]. Verfügbar unter: https://amd-intel-cpu-benchmark.com/ (Zugriff am: 14. Januar 2022).

[58] AAEON Technology Inc., *UP Xtreme Lite Series.* [Online]. Verfügbar unter: https://up-shop.org/up-xtreme-lite-series.html (Zugriff am: 15. Dezember 2021).

[59] AAEON Technology Inc., *UP Xtreme Edge Compute.* [Online]. Verfügbar unter: https://up-shop.org/upx-edge-series.html (Zugriff am: 15. Dezember 2021).

[60] Business Insider, „Gründe für den Chipmangel und Wege, die aus der Halbleiter-Krise führen“, *Business Insider*, 13. Dez. 2021, 2021. [Online]. Verfügbar unter: https://www.businessinsider.de/wirtschaft/diese-ereignisse-haben-zum-internationalen-chipmangel-gefuehrt-und-so-soll-wieder-fuer-genug-halbleiter-gesorgt-werden/. Zugriff am: 30. Januar 2022.

[61] CODESYS GROUP, *CODESYS Control for Linux SL.* [Online]. Verfügbar unter: https://store.codesys.com/de/codesys-control-for-linux-sl-bundle.html (Zugriff am: 17. Januar 2022).

[62] Debian-Organisation, *Debian -- Debian bekommen.* [Online]. Verfügbar unter: https://www.debian.org/distrib/ (Zugriff am: 17. Januar 2022).

[63] F. Rosenberger, *Debian – die Installation – Rosenberger.* [Online]. Verfügbar unter: https://ro-kom.de/debian-die-installation/ (Zugriff am: 17. Januar 2022).

[64] IONOS Digitalguide, *Die richtige Konfiguration, um den Server abzusichern* (Zugriff am: 17. Januar 2022).

[65] Unix Tutorial, *How To: Setup sudo in Debian.* [Online]. Verfügbar unter: https://www.unixtutorial.org/how-to-setup-sudo-in-debian/ (Zugriff am: 17. Januar 2022).

[66] IONOS Digitalguide, *Die Firewall – Ihr Türsteher zwischen Rechner und Netzwerk.* [Online]. Verfügbar unter: https://www.ionos.de/digitalguide/server/sicherheit/was-ist-eine-firewall/ (Zugriff am: 17. Januar 2022).

[67] M. Mierke, *Ubuntu: Firewall einrichten.* [Online]. Verfügbar unter: https://www.heise.de/tipps-tricks/Ubuntu-Firewall-einrichten-4633959.html (Zugriff am: 17. Januar 2022).

[68] IONOS Digitalguide, *Was ist eigentlich SSH? Die Secure Shell einfach erklärt.* [Online]. Verfügbar unter: https://www.ionos.de/digitalguide/server/tools/secure-shell-ssh/ (Zugriff am: 17. Januar 2022).

[69] N. Sakovich, „10 Best IoT Platforms for 2021-2022“, *SaM Solutions*, 21. Apr. 2021, 2021. [Online]. Verfügbar unter: https://www.sam-solutions.com/blog/top-iot-platforms/. Zugriff am: 19. Januar 2022.

[70] A. Tekniepe, "Vorteile als Microsoft-Silber-Partner", schriftlich, Jan. 2022.

[71] manageE, *mE180 Controller – Technische Daten » manageE.* [Online]. Verfügbar unter: https://www.managee.de/produkte/me180-controller/me180-controller-technische-daten/ (Zugriff am: 20. Januar 2022).

[72] manageE, *manageE mE 180: Benutzerhandbuch V1.2.10.* [Online]. Verfügbar unter: http://www.managee.de/wp-content/uploads/2016/05/manageE-mE-180-Benutzerhandbuch_V1.2.10.pdf (Zugriff am: 22. Januar 2022).

[73] Martens, *MODBUS – Schnittstelle: Schnittstellenbeschreibung und Datenübertragungsprotokoll.* [Online]. Verfügbar unter: https://www.ghm-group.de/fileadmin/GhmProduct/PDF/Diverses/GHM_DMS50_ModBus_D.pdf (Zugriff am: 22. Januar 2022).

[74] S. Niemeier, *SQL Datenbank.* [Online]. Verfügbar unter: https://www.tenmedia.de/de/glossar/sql-datenbank (Zugriff am: 22. Januar 2022).

[75] K. Gremban, *Tutorial: Speichern von Daten mit dem SQL-Modul in Azure IoT Edge.* [Online]. Verfügbar unter: https://docs.microsoft.com/de-de/azure/iot-edge/tutorial-store-data-sql-server?view=iotedge-2020-11 (Zugriff am: 22. Januar 2022).

[76] IONOS Digitalguide, *Docker-Container.* [Online]. Verfügbar unter: https://www.ionos.de/digitalguide/server/knowhow/docker-container/ (Zugriff am: 24. Januar 2022).

[77] Microsoft Corporation, *Visual Studio Code – Code Editor | Microsoft Azure.* [Online]. Verfügbar unter: https://azure.microsoft.com/de-de/products/visual-studio-code/ (Zugriff am: 24. Januar 2022).

[78] Microsoft Corporation, *IoT Edge | Cloud Intelligence | Microsoft Azure.* [Online]. Verfügbar unter: https://azure.microsoft.com/de-de/services/iot-edge/ (Zugriff am: 24. Januar 2022).

[79] M. Steiu, *Einrichten der SQL-Datensynchronisierung zwischen Datenbanken in Azure SQL-Datenbank und SQL Server.* [Online]. Verfügbar unter: https://docs.microsoft.com/de-de/azure/azure-sql/database/sql-data-sync-sql-server-configure (Zugriff am: 26. Januar 2022).

[80] N. Singh, *Use IoT Hub message routing to send device-to-cloud messages to different endpoints.* [Online]. Verfügbar unter: https://docs.microsoft.com/en-us/azure/iot-hub/iot-hub-devguide-messages-d2c (Zugriff am: 26. Januar 2022).

[81] GitHub, *GitHub - Azure/iot-edge-modbus: Modbus protocol module for use with the Azure IoT Edge.* [Online]. Verfügbar unter: https://github.com/Azure/iot-edge-modbus (Zugriff am: 26. Januar 2022).

[82] Microsoft Corporation, *Azure Blob Storage | Microsoft Azure.* [Online]. Verfügbar unter: https://azure.microsoft.com/de-de/services/storage/blobs/#overview (Zugriff am: 26. Januar 2022).

[83] K. Gremban, S. Adams, B. Hamilton, A. Jenks und A. Duppala, *Deploy the Azure Blob Storage on IoT Edge module to your device.* [Online]. Verfügbar unter: https://docs.microsoft.com/en-us/azure/iot-edge/how-to-deploy-blob?view=iotedge-2020-11 (Zugriff am: 26. Januar 2022).

[84] json.org, *Einführung in JSON.* [Online]. Verfügbar unter: https://www.json.org/json-de.html (Zugriff am: 26. Januar 2022).

[85] S. Wijman, *Enable rc.local in Debian Bullseye.* [Online]. Verfügbar unter: https://blog.wijman.net/enable-rc-local-in-debian-bullseye/ (Zugriff am: 28. Januar 2022).

[86] T. Myers, A. Buck und M. Hopkins, *Create a service SAS for a container or blob - Azure Storage* (Zugriff am: 30. Januar 2022).

[87] PRIMATE LABS, *Processor Benchmarks - Geekbench Browser.* [Online]. Verfügbar unter: https://browser.geekbench.com/processor-benchmarks (Zugriff am: 14. Januar 2022).

IX ANHANG

Anhang A : Gedächtnisprotokoll zu persönlichem Gespräch mit Thomas Voller bezüglich der BIG-Projektergebnisse

Datum: 06.01.2022

Medium: Telefon

Inhalt:

Vor dem Telefonat hatte ich Thomas Vollmer per E-Mail kontaktiert, da er Coautor des Artikels „Energieeffiziente Produktionsstätte durch bedarfsabhängige und innovative Gebäudeautomation. Reduzierung von Energiekosten durch Vernetzung von Gebäudeautomation, PPS-Systemen und Wetter-Forecasts“ ist, der unter anderem die Ergebnisse des BIG-Projektes zusammenfassen soll.

Zu Beginn des Gesprächs bittet er mich darum, noch einmal in wenigen Sätzen zusammenzufassen, welche Erkenntnisse ich mir von dem Artikel erhoffe. Ich erkläre ihm kurz die Hintergründe meiner Anfrage und beschreibe meine Vorgehensweise beim Vergleich abgeschlossener und laufender Projekte, die Algorithmen zur modellprädiktiven Regelung verwenden/ verwendet haben.

Er sagt daraufhin, dass Prozesse, die eine gewisse Abwärme produzieren, nicht aufeinander abgestimmt wären und dies in der Gebäudeautomation bei Wärme und Kühlung nicht berücksichtigt werde. Um zu prüfen, wie eine solche Regelung durchgeführt werden kann, wurden zunächst verschiedene Algorithmen verglichen. Im Anschluss wurde ein Modell zur Temperaturentwicklung abgeleitet, das in Matlab Simulink realisiert wurde, auf reale Daten zugriff und eine ausreichende Genauigkeit besaß. Dabei wurden drei Strategien abgeleitet: Eine regelbasierte Strategie (RBC), eine PID-Strategie (klassische Regelungstechnik) und eine modellprädiktive Regelung (MPC). Neben den Temperaturen im Raum konnten während der Überprüfung bei Projektpartnern auch andere Daten wie Maschinenbelegungspläne und Personalanwesenheiten aufgenommen werden. Es zeigt sich, dass die RBC-Strategie eine höhere Praxistauglichkeit besaß und deshalb final für einen Prototypen umgesetzt wurde.

Für die Regelungen wurden unter anderem Abweichungen zwischen dem Sollwert und prognostizierter Raumtemperatur und die Wärmeverteilung in der

Produktionsstätte auf Basis der Wärmekoeffizienten der Gebäudehülle berechnet.

In den letztendlichen Auswertungen habe sich herausgestellt, dass die prognostizierten Energiekosten bei Umsetzung der Strategien in etwa wie folgt wären:

- RBC-Strategie: Etwas unter 100 Euro/Tag
- PID-Strategie: Faktor 4 zur RBC-Strategie
- MPC: Etwa 20 % über RBC

Außerdem zeigte sich laut Vollmer, dass die RBC im Vergleich eine besser skalierbare Energieersparnis besaß.

Vollmer sagte ebenfalls, dass bei der Regelung der Raumtemperatur die unterschiedlichen Modelle wie folgt abgeschnitten hätten:

- Ohne Regler: Temperatur läuft aus den Toleranzen raus
- RBC-Regler: Innerhalb der Toleranzen
- MPC-Regler: Innerhalb der Toleranzen
- PID-Regler: Läuft in Toleranzen unten aus Bereich raus → erfüllt nicht die Anforderungen an die klimatischen Bedingungen im Gebäude

Weiter führte der Interviewpartner aus, dass die Temperaturprognosen mit den tatsächlichen Temperaturen verglichen worden seien und sich dabei zeigt, dass die Prognosen entsprechend akkurat waren (unter 1° Abweichung). Das bedeutet, dass die Regelstrategien mit einem guten Dateninput agieren konnten, die Daten aber unterschiedlich umsetzten.

Anhang B : Vollständige Liste marktüblicher Datenlogger (aufsteigend nach Alphabet)

Produkt (Unternehmen)	Anmerkung	Abmessungen [mm]	Hutschienenmontage	Schnittstellen	Betriebssystem (BS)	Betriebstemperatur	Datenspeicher	Versand der Daten	Logging-Rate	USV	Quelle
7KN POWERCENTER 3000 V1 (SIEMENS)		85x85x41	Ja	2x Ethernet (Modbus TCP), Bluetooth, Funk	Linux	0 bis +55 °C	32 GB (nicht erweiterbar)	unklar	- 10 s -> 1 Tag - 15 min -> bis 14 Monate	nein	https://mall.industry.siemens.com/mall/de/de/Catalog/Product/7KN1310-0MC00-0AA8
BDL 50 (Berg GmbH)	max. 50 Geräte	106,3x62x76,5	Ja	2x Ethernet, COM1(RS485), COM2(M-Bus) S	OSADL Linux	0-60 °C	16 GB	Nachversand bei Netzausfall	1min, 15min	Nein	https://www.berg-energie.de/produkte-loesungen/mess-und-kommunikationstechnik/datenlogger/
CONTROLLER PFC200 (WAGO)		100x78,6x71,9	Ja	Modbus (TCP, UDP), ETHERNET, EtherNet/IPTM-Adapter, Modbus® RTU, RS-232, RS-485, MQTT, EtherCAT-Master, BACnet/IP	Linux	0 bis +55 °C	36 GB (bis zu 32GB mit SD-Karte erweiterbar)	kann programmiert werden	einstellbar (ab 20ms)	nein	https://www.wago.com/de/sps/controller-pfc200/p/750-8212
DL4000 Universal Data Logger (Vaisala)	Nur 4 Kanäle Datenaufnahme nur mit viewlinc Monitoring	85×59×26	Nein	RS-232 serial, Wifi module, Ethernet, Ethernet (vNet)	BS von Hersteller	-40 bis +85 °C	120.000 Messdaten (12-bit)	Nachversand bei Netzausfall	10 seconds to once a day	10 Jahre Batterie bei 1min Logging-Rate oder länger	https://www.vaisala.com/en
EasyNet (4Next Technology Systems)	Integriertes Modem	90x70x65	Ja	1x Ethernet (ModBus RTU and TCP), LTE	FreeRTOS	-20 °C +60 °C	64 GB	unklar	1s bis 1h (frei einstellbar)	nein	EasyNET_Brochure_EN_1.0.1.pdf (4next.eu) https://www.4next.eu/en/prodotto/gsm-data-logger/
ECOLOG-PRO Base (ELPRO)	Für Raumüberwachung	282×201×56	Nein	Ethernet, WiFi	Windows 10 LTSB	0-40 °C	120 GB (50 Messpunkte, 18 Monate)	bis 10.000 Werte / Modul nachladbar	unklar	integrierte USV	https://shop.elpro.com/EL20/daten/img/Documents/TechSpecs/ECOLOG-PRO/TS_ECOLOGPRO_Base_V3_DE_web.pdf

Produkt (Unternehmen)	Anmerkung	Abmess-ungen [mm]	Hut-schie-nen-mon-tage	Schnittstellen	Betriebs-system (BS)	Betriebs-tempera-tur	Datenspei-cher	Versand der Daten	Logging-Rate	USV	Quelle
Energy Manager L (SMA)	Nur für PV-Anlagen	150x120.2x48.8	Ja	2x Ethernet (Modbus TCP), 2 x RS-485, 4x digitale Eingänge	unklar	-40 bis +70 °C	128 GB	unklar	unklar	nein	https://files.sma.de/downloads/EDML-10-DS-en-13.pdf
Expert Logger (Delphin Technology AG)		217x88x128	Ja	- 2x Ethernet (Modbus-TCP), Wlan, LTE, RS485, RS232, ProfiBus, ModBus, CAN-Bus, OPC UA, serielle Schnittstellen	Linux Debian	-20-50 °C	14 GB (420 Millionen Messwerte)	Nachversand bei Netzausfall	1s	nein	https://www.delphin.de/produkte/mess-und-pruefgeraete/expert-logger.html
Fluke 1742, 1746 and 1748 (FLUKE)	Nur für Stromverteilung	230cmx180cmx54	Nein	Ethernet, Wlan, Bluetooth	unklar	-25 bis 50 °C	4 Wochen mit 1-min	unklar	1 / 5 / 10 / 30 s 1 / 5 / 10 / 15 / 30 min	4h Batterie	https://www.fluke.com/en-us/product/electrical-testing/power-quality/1742-1746-1748#
GP2 (Delta-T Devices)	Fürs Feld gedacht mit Feuchtigkeitssensoren etc.	225x185x75	Nein	Ethernet, SDI-12, RS232	unklar	-20 bis 60 °C	2.500.000 Datenpunkte	unklar	1s bis mehr als 24h	530 Tage Batterie	https://delta-t.co.uk/product/gp2/#specification
gridBox (gridX)	Info: 33 unterstütze Hersteller	110x110x30	Nein	Ethernet optional: RS232, RS485, RJ12, CAN-Bus, IO-Link	gridOS (Yocto-basiertes Linux)	0 °C bis 40 °C	8 GB (Bis zu 10 Tage Daten)	Nachversand bei Netzausfall	lokal: Millisekunden Cloud: 2s, 10m, 15m etc.	nein	https://de.gridx.ai/edge-services
HOBO RX3000 (ONSET)	maximal 15 Datenkanäle	186x181x118	Nein	LTE, Wlan, Ethernet, Analoger Input	BS von Hersteller	-40° bis 60 °C	0,032 GB (nicht erweiterbar)	Nachversand bei Netzausfall	1s, 1min - bis zu alle 18h	Ja	https://assets.omega.com/spec/OM-SQ2040.pdf
MEMORY HiLOGGER LR8450 (HIOKI)	bis zu 120 Kanäle	272x145x43	Nein	Ethernet, Wlan, analoge & digitale Eingänge	unklar	-10 bis 50 °C	Intern: 0,5 GB + 8GB SD-Karte + 16 GB USB-Stick	unklar	22 Optionen: 10 ms bis 1 h	4h Batterie	https://www.hioki.com/global/products/multichannel-data-loggers/multichannel/id_6767
Meteo 40 Plus (Ammonit Measurement)	Für Wetterstationen, Windfarmen, Solarparks etc. (max. 12 Sensoren verbindbar)	260x194x50	Nein	LTE, Ethernet (Modbus RTU and TCP)	Linux	-40 bis +65 °C	2 GB (ca. 32.000.000 Messwerte)	Nachversand bei Netzausfall	1/4s bis 1h (frei einstellbar)	nein	https://www.ammonit.com/en/products/data-loggers/meteo-40-plus/
OM-SQ2040-2F16 (OMEGA)	Leider keine Reaktion auf Anfragen erhalten	175x235x95	Nein	Ethernet, Wlan, RS-232	unklar	unklar	unklar	unklar	unklar	unklar	https://www.omega.com/en-us/data-acquisition/data-loggers/multichannel-programmable-and-universal-input-data-loggers/p/OM-SQ2040-Series

Produkt (Unternehmen)	Anmerkung	Abmessungen [mm]	Hutschienenmontage	Schnittstellen	Betriebssystem (BS)	Betriebstemperatur	Datenspeicher	Versand der Daten	Logging-Rate	USV	Quelle
PowerLogic PM5000 Serie (Schneider Electric)	Nur Strommessung mit 4 Eingängen	96x96x72	Ja	2x Ethernet (S0-Impuls, Modbus RTU, Modbus TCP, BACnet/IP, SNMP), 4 Stromeingänge	BS von Hersteller	-25 bis +70 °C	15 min-Lastprofilaufzeichung für z.B. 14 Messgrößen für 90 Tage	Nachversand bei Netzausfall	unklar	nein	https://www.se.com/de/de/product-range/61281-power-logic-pm5000-serie/?parent-subcategory-id=4115&filter=business-2-geb%C3%A4udeautomation#overview
PSK RTU 50 (Phoenix Contact)	Unterstützte Protokolle: IEC 60870-5-101, Modbus/RTU, IEC 62056-21	210x110x45	Ja	Ethernet, 2 analoge, 4 digitale Eingänge, serielle Schnittstelle, GSM/GPRS-Modem	BS von Hersteller	-20 bis 65 °C	1 MB	unklar (vermutlich programmierbar)	programmierbar ab 1s	nein	https://www.phoenixcontact.com/online/portal/de?urile=wcm%3apath%3a/dede/web/main/products/subcategory_pages/Complete_packages_for_data_logging_P-14-04-02/bcfc8319-5935-43e2-882f-098b66633ce5
RMS-LOG-L - DATENLOGGER (Rotronic)	- Pharma Monitoring - Lebensmittel Monitoring - Museum Monitoring)	105x113x38	Nein	Ethernet, Funk (Modbus TCP)	BS von Hersteller	-40 bis +70 °C	44.000 Messwertpaare	Nachversand bei Netzausfall	10-15 min	interne Batterie (1 min)	https://www.rotronic.com/pub/media/productattachments/files/59057D_RMS-Datenlogger.pdf
Touchscreen Data Logger (Dickson)	Nur für Raumüberwachung / Umgebungsdaten	210x45x180	Nein	Dickson-Sensoren	unklar	-10 bis 60 °C	1.000.000 Datenpunkte	unklar	60 / 90 s 2 / 5 / 10 / 30 / 60 min	70h Batterie	https://dicksondata.com/product-category/data-loggers/
Verschiedene (Dwyer Instruments)	Datenlogger für Prozesse (Füllstand, Druck, Durchfluss, Temperatur)	-	-	unklar	-	-	-	-	-	-	https://dwyer-products.de/
Verschiedene (SENSITECH)	Für Kühlketten, Fracht und Logistik	-	-	-	-	-	-	-	-	-	https://www.sensitech.com/en/products/

Anhang C : Vollständige Liste marktüblicher Systeme für die Energiedatenerfassung und -analyse

Anbieter	Anmerkung	Energiezähler etc.	Zentrale Hardware / Datenlogger	Schnittstellen	Auswertungs-tool	Funktionen für Auswertung	Quelle
Beckhoff Automation GmbH & Co. KG	Bei Netzwerkausfall stehen Buffering-Algorithmen zur Verfügung, um Datenverlust zu vermeiden	etablierte Zähler, die über Impulszähler (Stromwandler, Energiemess-klemmen)eingebunden werden	IoTKoppler EK9160	digitale und analoge Standard-EtherCAT-Klemmen	TwinCAT	- punktuelle oder kontinuierliche Datenanalyse - automatische Programmcode-Generierung - TC3 IoT Data Agent überträgt Daten in Cloud (z.B. Microsoft Azure)	https://www.beckhoff.com/media/downloads/informationsmedien/flyer-energiemesstechnik.pdf https://www.beckhoff.com/de-de/produkte/i-o/leistungsmessung/
Berg GmbH	Auswertungssoftware Efficio auch mobil als App nutzbar	Skalar.pro Universal Datenlogger	BDL 50	- 2x Ethernet - COM1(RS485) - USB Slot für USB-Stick - COM2(M-Bus) S	Efficio	- Sankey-Diagramme - Erstellung von Berichten - Zustandsanalysen - Auftragskennzahlen - Streudiagramme mit Regressionsanalysen	https://www.berg-energie.de/produkte-loesungen/energiemanagement/ https://www.berg-energie.de/produkte-loesungen/energiemanagement/efficio-energiemanagement/
Delphin Technology AG		Loggito Logger	Expert Logger	- 2x Ethernet (Modbus-TCP) - Wlan - LTE diverse andere (RS485, RS232, ProfiBus, ModBus, CAN-Bus. OPC UA, serielle Schnittstellen)	ProfiSignal 20	- Darstellung der Messdaten (live und historisch) - Anlagen- und Prozessvisualisierung	https://www.delphin.de/produkte/software/profisignal-20.html https://www.delphin.de/produkte/mess-und-pruefgeraete/expert-logger.html
Eaton Corporation	Wenige Informationen auf Webseite, keine Antwort auf Anfrage	unklar	unklar	unklar	Eaton Forseer EPMS	- Energiemonitoring - Analyse historischer Daten zur Entwicklung von Trends - Prognosen zu Geräteverhalten und Kapazitätsplanung - Alarmfunktionen	https://www.eaton.com/us/en-us/digital/brightlayer/brightlayer-data-centers-suite/eaton-epms.html http://www.eaton.de/EatonDE/ProdukteundLoesungen/Electrical/ProdukteDienstleistungen/EnergieverteilungMonitoring/Energie-Mess-undMonitoring-Loesungen/index.htm
Janitza	- Zwischenspeicherung von 10 Millionen Messwerten (pro "ProData 2"-Gerät) - Erfassung nicht elektrischer Werte	Integration bereits vorhandener Zähler	ProData 2	15 digitale Eingänge, Modbus RTU, Ethernet, Protokolle: TCP/IP, DHCP-Client (BootP), Modbus/TCP (Port 502), ICMP (Ping), NTP	GridVis®-Energy	- Charts - Bargraph - Energietabellen - Heatmaps - Mengenflussdiagramme - Leistungskennzahlen - Onlinewerte	https://www.janitza.de/gridvis-energy-demozugang.html

Anbieter	Anmerkung	Energiezähler etc.	Zentrale Hardware / Datenlogger	Schnittstellen	Auswertungstool	Funktionen für Auswertung	Quelle
Meistro	Nur für elektrischen Strom Wenige Informationen auf Webseite	unklar	CONTROLIT Expert	unklar	Mesitro Monitoring-System		https://www.meistro.de/unser-business/effizienz/energiemonitoring/
OP-TENDA GmbH	Umfassendes Monitoring von Energie- und Betriebsdaten	unklar	Energy Box	Modbus TCP/IP / RTU, OPC U/A, BACnet IP, Ethernet RS 232, W-LAN, SAMLAN, OCPP, WIFI, 3G/4G, SPS Steuerungen (alle gängigen Hersteller)	Energy Monitor	- Lastganganalyse - Verbrauchsdiagramme - Dauerkennlinien - Sankey-Diagramme - Definition von Kennzahlen - Regressionsanalyse - Aufnahme aktueller Wetterdaten	https://www.optenda.de/energiemonitoring/ https://www.optenda.de/energiemonitoring/messkonzepte/
Qualitas AMS	Wenige Informationen auf Webseite	verschiedene Sensorik	bit.B-Hardware	unklar	bit.B-Online-Monitor	angefragt	https://www.qualitas-ams.de/energiemonitoring/
Robert Bosch GmbH	Kein weiterer Informationsaustausch, da manageE ein ähnliches Lösungsspektrum bzw. Wettbewerbsprodukte zur Energy Platform und den Energieeffizienzlösungen von Bosch anbietet.	unklar	unklar	unklar	Energy Platform	- Messung der Maschinenleistung - Alarmierung in Echtzeit - Umfassendes Reporting - Automatisierte Analyse	https://www.boschbuildingsolutions.com/de/de/news-und-stories/energy-platform/
Siemens	Energie- und Anlagendaten	Verschiedene Messgeräte aus dem Sentron-Portfolio(z.B. 7KT PAC1600 und 7KM PAC2200)	7KN POWERCENTER 3000 V1	- 2x Ethernet (Modbus TCP) - Bluetooth - Funk	SENTRON powermanager SENTRON powermind	**Versand von Daten an Cloud-Lösungen (z.B. Microsoft Azure)** **powermanager:** - Einrichten des Systems - Anlegen von Geräten - grafischen Darstellung der Gerätedaten - Verarbeitung der Daten in Berichten **powermind:** - Vergleich von Energieverbräuchen - Festlegung von Referenzperioden - Informationen zu Gerätezuständen - Condition Monitoring	https://cache.industry.siemens.com/dl/files/328/109744328/att_1088673/v1/LV14_112021_DE_202111301728257342.pdf https://new.siemens.com/de/de/produkte/energie/niederspannung/komponenten/sentron-messgeraete-und-energiemonitoring.html#Energiez%C3%A4hler

Anbieter	Anmerkung	Energiezähler etc.	Zentrale Hardware / Datenlogger	Schnittstellen	Auswertungstool	Funktionen für Auswertung	Quelle
SMA	Nur für PV-Anlagen	I/O System von MOXA und WAGO	Energy Manager L	unklar	SUNNY PORTAL	- Live-Daten zum Anlagenstatus - Informationen zum aktuellen Energiefluss (Netzbezug, Batterieladung) - Überwachung der Kommunikation zum Portal - Überwachung der Wechselrichter-Performance - Wetterinformationen für den Standort	https://www.sma.de/produkte/monitoring-control/data-manager-m.html
Solarwatt / Kiwigrid	Nur für PV-Anlagen	Smart Plug	Solarwatt Manager flex	- 1x Ethernet - 2x SO / Digital In	SOLARWATT Manager Portal	- Fernwartung - Einsehen historischer Daten	https://solarwatt.canto.global/direct/document/6hg28ijq4d1p335i7pt5jooo2c/f0x2Xu6s107xLt-GXJCPsAglUuts/original?content-type=application%2Fpdf&name=extended-datasheet-sw-manager-de.pdf&source=solarwatt.de&medium=Website&referrer=solarwatt.de/loesungen/unsere-produkte/uebersicht/manager https://www.solarwatt.de/loesungen/unsere-produkte/uebersicht/manager

Anhang D : Vollständige Auflistung des Benchmarkings der verschiedenen Prozessoren (aufsteigend sortiert nach Leistung)

			Geekbench [87]	
Bezeichnung des Prozessors inklusive Anzahl der Kerne und Taktfrequenz	**CPUB [56]**	**AICB [57]**	**Einkernbetrieb**	**Mehrkernbetrieb**
Intel Atom E3815 1x 1,46 Gigahertz (GHz)	159	110	-	-
ARM Cortec-A53 4x 1,6 GHz	202	-	-	-
Intel Atom x5-E3930 2x 1,3 - 1,8 GHz	878	1.110	-	-
Intel Atom x5-E8000 4x 1,04 GHz	985	1.116	182	603
Intel Celeron N2930 4 x 1,83 - 2,16 GHz	994	1.001	190	644
Intel Atom E3845 4x 1,91 GHz	1.024	1.008	-	-
Intel Celeron N3160 4x 1,6 - 2,24 GHz	1.069	1.256	206	650
Intel Celeron J1900 4x 2,0 GHz	1.136	1.113	216	713
Intel Apollo Lake N3350 2x 1,1 - 2,4 GHz	1.147	1.137	287	522
Intel J3355 2x 2,0 GHz	1.235	1.329	311	578
Intel Celeron 4205U 2x 1,8 GHz	1.321	1.341	-	-
Celeron 3955U 2x 2 GHz	1.499	1.596	460	806
Intel N3450 4x 1,1 - 2,2 GHz	1.942	1.944	274	892
Intel Atom E3940 4x 1,60 - 1,8 GHz	1.960	1.868	-	-
Intel Pentium N4200 4x bis 2,5 GHz	2.142	2.230	309	942
Intel Atom x7-E3950 4x 1,6 - 2 GHz	2.159	-	260	929

Intel J3455 4x 1,5 - 2,3 GHz	2.237	2.263	290	985
Intel J4205 4x 1,5 GHz	2.384	2.326	307	1.004
Core i5 6200U 2x 2,3 - 2,8 GHz	3.024	3.039	558	1.333
Intel Celeron J4125 4x 2,0 - 2,7 GHz	3.042	3.035	433	1.405
Core i7 6600U 2x 2,6 - 3,4 GHz	3.519	3.643	-	-
Intel Core i3-8145U 2x 2,1 - 3,9 GHz	3.825	2.048	776	1.595
Intel Core i3-8145UE 2x 2,2 - 3,9 GHz	4.133	-	-	-
Intel Core i5-8365UE 4x 1,6 - 4,1 GHz	5.555	6.215	-	-
Intel Core i5-8265U 4x 1,6 – 3,9 GHz	6.132	6.424	-	-
Intel Core i7-8665UE 4x 1,7 - 4,4 GHz	6.587	6.774	-	-

Anhang E : E-Mail von Alfred Tekniepe zum Thema „Vorteile als Microsoft-Silber-Partner“

Hallo Fabian,

die *manageE* GmbH + Co. KG ist zurzeit in sechs Kompetenzbereichen als Microsoft Silver Partner zertifiziert.

Jedes Jahr muss eine Rezertifizierung erfolgen und je nach Kompetenzbereich müssen unterschiedliche Voraussetzungen erfüllt werden (Bezahlen von Kompetenzgebühren, Zertifizierung selbst entwickelter Software, Erreichen von Umsatzzielen, Erfüllung von Assessment- und Prüfungsanforderungen durch Mitarbeiter).

Das hat für uns u.a. folgende Vorteile:

- Zugriff auf Microsoft-Produkte (Download mit Lizenzen) für die meisten Server- und Desktop-Produkte
- Office 365 E3-Lizenzen für bis zu 25 Mitarbeiter
- Visual Studio Enterprise-Abonnements für bis zu 15 Mitarbeiter und hiermit Downloadmöglichkeit (mit Entwicklungs- und Testlizenzen) für alle Microsoftprodukte die von Microsoft veröffentlicht wurden, sowie Vorabversionen
- Monatliche Freikontingente für Azure- und Cloudprodukte
- Technische Supportvorteile
- Hilfestellung bei der Markteinführung und exklusive Ressourcen zur Förderung des Unternehmenswachstums

Viele Grüße

Alfred Tekniepe
Dipl.-Ing., M.Sc.
Teammanager IT und Grid Automation

Mobil +49 (0)151 44 879 489
Fax +49 (0)221 975803-89

Alfred.Tekniepe@manageE.de
www.manageE.de

manageE GmbH & Co. KG
Im Technikhof
Dillenburger Straße 93
51105 Köln

Sitz der Gesellschaft
Marbergweg 79, D-51107 Köln
Amtsgericht Köln HRA 29993
Geschäftsführer Michael Pauli

Anhang F : Lösungsansatz 2 - C#-Code zur Verarbeitung der Nachrichten aus dem Modbus-Modul für das Message Routing

```
using Microsoft.WindowsAzure.Storage;
using Azure.Storage.Blobs;
using System;
using System.IO;
using System.Text;
using System.Collections.Generic;
using System.Threading.Tasks;
using Microsoft.Azure.Devices.Client;
using Microsoft.Azure.WebJobs;
using Microsoft.Azure.WebJobs.Extensions.EdgeHub;
using Microsoft.Azure.WebJobs.Host;
using Microsoft.Extensions.Logging;
using Newtonsoft.Json;

namespace Functions.Samples
{
    public static class streamToBlob
    {
        [FunctionName("streamToBlob")]
        public static async Task getMessageAndStream(
            [EdgeHubTrigger("input1")] Message messageReceived,
            ILogger logger)
        {

            //Console.WriteLine("TEST");

            byte[] messageBytes = messageReceived.GetBytes();
           var messageString = System.Text.Encoding.UTF8.GetString(message-
Bytes);

            if (!string.IsNullOrEmpty(messageString))
            {
               logger.LogInformation("Info: Received one non-empty message:
" + messageString);
                using (var pipeMessage = new Message(messageBytes))
                {

                   foreach (KeyValuePair<string, string> prop in messageRe-
ceived.Properties)
                    {

                    }

                    //Console.WriteLine(pipeMessage);

                }
            }

             logger.LogInformation("Info: Nachricht erhalten" + messageRe-
ceived);
```

```
            var account = CloudStorageAccount.Parse("<CONNECTION STRING FÜR
MICROSOFT SPEICHER-ACCOUNT>");

            var client = account.CreateCloudBlobClient();

            // Erzeugt einen neuen CloudBlobContainer
              var container = client.GetContainerReference("localblobcon-
tainer");

            var now = DateTime.Now;

            var filename = "Datei_" + now.ToString("dd-MM-yyyy, HH:mm:ss");

            var blob = container.GetBlockBlobReference(filename);

            blob.UploadFromStreamAsync(pipeMessage);

            // existiert bereits ein Blob?
            var exists = blob.ExistsAsync().Result;

            if (!exists)
            {
                        await blob.UploadTextAsync($"Text geschrieben um
{DateTime.Now.ToLocalTime()}");
                logger.LogInformation("Info: Blob wurde hochgeladen");
            }
            else
            {
                await blob.UploadTextAsync($"Blob existiert bereits!");
                logger.LogInformation("Info: Blob existiert bereits!");
            }
        }
    }

    // Definiert das erwartete Body-Schema der hereinkommenden Nachrichten.
    class MessageBody
    {
        public DisplayName displayname {get; set;}
        public HwId hwid {get; set;}
        public Address adresse {get; set;}
        public Value wert {get; set;}
        public SourceTimestamp zeitstempel {get; set;}
    }
}
```

Anhang G : Lösungsansatz 3 - CODESYS-Skript für das minütliche Abspeichern der Messdaten in einer .csv-Datei

Deklaration der Variablen

```
FUNCTION_BLOCK csv_creator
VAR_INPUT
	Trigger:BOOL:=FALSE;
	Zeitstempel:STRING:='';
	Dateiname:STRING:='';
	inputSpannung:REAL:=0;
	inputCosphi:REAL:=0;
	inputStrom:REAL:=0;
	inputScheinleistung:REAL:=0;
	inputBlindleistung:REAL:=0;
	inputWirkleistung:REAL:=0;
	inputEnergy:REAL:=0;
END_VAR
VAR_OUTPUT
END_VAR
VAR
	bWriteData:BOOL:=FALSE;		// Auslöser um Messdaten zu schreiben
	uiStpDataLogger:UDINT:=0;	//sequencer

	sFileName:  CAA.FILENAME;
	hFile:	CAA.HANDLE;

	Dateipfad: STRING := '\var\opt\codesys\test_csv\';
	Filename: STRING :='';

	//example data
	measuredData: ARRAY [0..60] OF STRING[256];

	sStringToWrite:	STRING[1024]:='no data';

	writeIndex:UINT:=0;
	spaltenNamen:STRING[256]:=	'Zeit;Spanung_L1;Cosphi_Ch1;Strom_Ch1;Scheinleis-
tung_Ch1;Blindleistung_Ch1;Wirkleistung_Ch1;Energy_Ch1;';

	// Kontroll-Charakter
	CRLF:STRING(2):= '$R$N';
	LF:STRING(1):= '$N';

	// Funktionsblock für die Datei-Befehle
	fopen  : FILE.Open;
	fwrite : file.Write;
	fclose : file.Close;

	Pos:INT:=0;
	readdata: BOOL:= FALSE;	//neue Daten einlesen start
	firstline: STRING[256] := '';
	logIndex:UINT:=0;
	writeFile:BOOL:=FALSE;

	K1_Spannung:STRING:='';
```

```
        K1_Cosphi:STRING:='';
        K1_Strom:STRING:='';
        K1_Scheinleistung:STRING:='';
        K1_Blindleistung:STRING:='';
        K1_Wirkleistung:STRING:='';
        K1_Energy:STRING:='';

        // Legt den Messintervall und die Länge der .csv-Datei fest
        // für nur 10 Zeilen in 10 Sekunden
        //messIntervall: INT:= 10;
        // Für 60 Zeilen nach 1 Minute
        messIntervall: INT:= 60;

END_VAR
```

Haupt-Skript

```
//Setzt den Dateipfad zusammen
Filename :=CONCAT(Dateipfad,Dateiname);
Filename :=CONCAT(Filename,'.csv');

IF Trigger = 0  THEN
        readdata := TRUE;
END_IF

IF Trigger = 1 AND readdata = 1 THEN

   // Fügt den Zeitstempel in das Array ein
        measuredData[logIndex]:=CONCAT(Zeitstempel,';');

        // Kopiert die Spannung von Kanal 1 in das Array
        // Wandelt den gemessenen Wert in einen String um
        K1_Spannung := TO_STRING(inputSpannung);
        // Fügt den umgewandelten Wert in das Array ein
        measuredData[logIndex]:=CONCAT(measuredData[logIndex],K1_Spannung);
        // Ermittelt die Position des REAL-Wert-Punktes im String
        Pos := Find(measuredData[logIndex],'.');
        // Ersetzt den Punkt durch ein Komma
        measuredData[logIndex]:= Replace(measuredData[logIndex], ',' , 1, Pos);
        // Fügt ein Semikolon als Trennzeichen hinzu
        measuredData[logIndex]:=CONCAT(measuredData[logIndex],';');

        K1_Cosphi := TO_STRING(inputCosphi);
        measuredData[logIndex]:=CONCAT(measuredData[logIndex],K1_Cosphi);
        Pos := Find(measuredData[logIndex],'.');
        measuredData[logIndex]:= Replace(measuredData[logIndex], ',' , 1, Pos);
        measuredData[logIndex]:=CONCAT(measuredData[logIndex],';');

        K1_Strom := TO_STRING(inputStrom);
        measuredData[logIndex]:=CONCAT(measuredData[logIndex],K1_Strom);
        Pos := Find(measuredData[logIndex],'.');
        measuredData[logIndex]:= Replace(measuredData[logIndex], ',' , 1, Pos);
        measuredData[logIndex]:=CONCAT(measuredData[logIndex],';');

        K1_Scheinleistung := TO_STRING(inputScheinleistung);
        measuredData[logIndex]:=CONCAT(measuredData[logIndex],K1_Scheinleistung);
        Pos := Find(measuredData[logIndex],'.');
        measuredData[logIndex]:= Replace(measuredData[logIndex], ',' , 1, Pos);
        measuredData[logIndex]:=CONCAT(measuredData[logIndex],';');
```

```
        K1_Blindleistung := TO_STRING(inputBlindleistung);
        measuredData[logIndex]:=CONCAT(measuredData[logIndex],K1_Blindleistung);
        Pos := Find(measuredData[logIndex],'.');
        measuredData[logIndex]:= Replace(measuredData[logIndex], ',' , 1, Pos);
        measuredData[logIndex]:=CONCAT(measuredData[logIndex],';');

        K1_Wirkleistung := TO_STRING(inputWirkleistung);
        measuredData[logIndex]:=CONCAT(measuredData[logIndex],K1_Wirkleistung);
        Pos := Find(measuredData[logIndex],'.');
        measuredData[logIndex]:= Replace(measuredData[logIndex], ',' , 1, Pos);
        measuredData[logIndex]:=CONCAT(measuredData[logIndex],';');

        K1_Energy := TO_STRING(inputEnergy);
        measuredData[logIndex]:=CONCAT(measuredData[logIndex],K1_Energy);
        Pos := Find(measuredData[logIndex],'.');
        measuredData[logIndex]:= Replace(measuredData[logIndex], ',' , 1, Pos);
        measuredData[logIndex]:=CONCAT(measuredData[logIndex],';');

        readdata := 0;

        // Sekundenzähler wird um 1 erhöht
        logIndex := logIndex + 1;

        // Prüft, ob bereits 60 Werte in das Array geladen wurden
        IF logIndex >= messIntervall THEN
                bWriteData :=1;
                logIndex :=0;
                writeFile :=1;

        END_IF
END_IF


CASE uiStpDataLogger OF
        (*Wartet auf den Trigger*)
        0:      IF bWriteData THEN
                        bWriteData:=FALSE;
                        uiStpDataLogger:=10;
                END_IF

        10:     (*Erstellt die neue .csv-Datei*)
    fopen.sFileName:=FileName;
    fopen.eFileMode:=File.MODE.MRDWR;
    fopen.xExclusive:=TRUE;
    fopen( xExecute:=TRUE);
    IF fopen.xDone THEN
      hFile:=fopen.hFile;
                        sStringToWrite:=CONCAT('Zeit;Span-
nung_L1;Cosphi_Ch1;Strom_Ch1;Scheinleistung_Ch1;Blindleistung_Ch1;Wirkleis-
tung_Ch1;Energy_Ch1;',CRLF);
                        writeIndex:=0;
      uiStpDataLogger:=20;
    END_IF

        20: (*Kopiert die Daten aus dem Array*)
                sStringToWrite:=CONCAT(sStringToWrite,measuredData[writeIndex]);
                IF writeIndex < messIntervall THEN
                        sStringToWrite:=CONCAT(sStringToWrite,CRLF);
                END_IF
                uiStpDataLogger:=30;
```

```
        30:        (*Schreibt die Messdaten in die Datei*)
    fwrite.hFile:=hFile;
    fwrite.pBuffer:=ADR(sStringToWrite);
    fwrite.szSize:=LEN(sStringToWrite);
    fwrite.udiTimeOut:=100000;  //Timeout
    fwrite( xExecute:=TRUE);
    IF fwrite.xDone THEN
                        fwrite( xExecute:=FALSE);
                        writeIndex:=writeIndex+1;
                        IF writeIndex <= messIntervall THEN
                                    uiStpDataLogger:=20;
                                    sStringToWrite:=";
                              ELSE
                                    uiStpDataLogger:=40;
                        END_IF
    END_IF
    IF fwrite.xError THEN
                        // Fehler
      uiStpDataLogger:=16#DEADBEEF;
    END_IF

        40:  (* Schließt die .csv-Datei *)
    fclose.hFile:=hFile;
    fclose( xExecute:=TRUE);
    IF fclose.xDone THEN
                        fopen( xExecute:=FALSE);
                        fwrite( xExecute:=FALSE);
                        fclose( xExecute:=FALSE);
      uiStpDataLogger:=0;
    END_IF
    IF fclose.xError THEN
      // Fehler
      uiStpDataLogger:=16#DEADBEEF;
    END_IF

        16#DEADBEEF:
                  // ErrorHandling
                  ;
END_CASE
```

Anhang H : Lösungsansatz 3 - C#-Code zur Verarbeitung von .csv-Dateien in Blockblobs

```
using Azure.Storage.Blobs;
using System;
using System.IO;
using System.Threading.Tasks;

namespace BlobContainerConsoleApp
{
    class Program
    {
        static async Task Main(string[] args)
        {
            await UploadBlob();

            //Console.WriteLine("Press a key to exit");
            //Console.ReadKey();
        }

        private static async Task UploadBlob()
        {
                var connectionString = "<CONNECTION STRING FÜR MICROSOFT
SPEICHER-ACCOUNT>";

            string containerName = "localblobcontainer";

            //  Holt den aktuellen Zeitstempel
            var now = DateTime.Now;

            // Pfad (Windows) in dem die .csv-Dateien gespeichert sind
            //string path = @"C:\test_upload";

            // Pfad (Linux) in dem die .csv-Dateien gespeichert sind
            string path = @"\var\opt\codesys\test_csv";

            string[] files = Directory.GetFiles(path);

            foreach (string file in files)
            {
                //Console.WriteLine(file);

               // Teilt den Dateipfad in Bestandteile auf, um Namen abgreifen
zu können
                string[] subs = file.Split('\\');

                var filename = subs[2];

                //Console.WriteLine(filename);

             //string[] subs2 = filename.Split('.');

                // Wenn Dateinamen sich ändern
                //var filenameOhneEndung = subs2[0];
                // Wenn Dateinamen sich nicht ändern
               //var filenameOhneEndung = subs2[0] + "_" + now.ToString("dd-
MM-yyyy, HH:mm:ss");
```

```
                //Console.WriteLine(filenameOhneEndung);

                // Initialisiert BlobClient
                 Azure.Storage.Blobs.BlobClient blobClient = new Azure.Sto-
rage.Blobs.BlobClient(
                    connectionString: connectionString,
                    blobContainerName: containerName,
                    blobName: filename);
                    //blobName: filenameOhneEndung);

                await blobClient.UploadAsync(path);

                DeleteCsv(path);
            }
        }

        private static void DeleteCsv(string filepathTodelete)
        {
            try
            {
                // Prüft, ob die Datei existiert
                if (File.Exists(filepathTodelete))
                {
                    // Wenn Datei existiert, lösche sie
                    File.Delete(filepathTodelete);
                }
               else Console.WriteLine("Datei konnte nicht gefunden werden");
            }
            catch (IOException ioExp)
            {
                Console.WriteLine(ioExp.Message);
            }
        }
    }
}
```

Anhang I : Lösungsansatz 4 - Python-Code zur Verarbeitung von .csv-Dateien in Blockblobs

```
from azure.storage.blob import ContainerClient
# für os.remove()
import os
# lisdir muss einzeln importiert werden
from os import listdir
from os.path import isfile, join
import schedule
import time

CONNECT_STR = <CONNECTION STRING>

CONTAINER_NAME = "localblobcontainer"

dateiPfad = "/var/opt/codesys/test_csv"

def getFilesSendAndDelete():

    # Speichert die Namen aller Dateien im aktuellen Pfad
    filenames = [f for f in listdir(dateiPfad) if isfile(join(dateiPfad, f))]

    container_client = ContainerClient.from_connection_string(conn_str=CONNECT_STR, container_name=CONTAINER_NAME)

    # Iteriert durch die gefundenen Dateien
    for file in range(len(filenames)):

        # print("Datei gefunden: " +str(filenames[file]))

        # Prüft, ob es sich bei den gefundenen Dateien um .csv-Dateien handelt und überspringt
        # alle anderen Dateien (z.B. die .py-Datei und die Excel-Vorlage)
        if filenames[file].endswith(".csv"):

            input_file_path = dateiPfad + "/" +str(filenames[file])

            # print(input_file_path)

            output_blob_name = str(filenames[file])

            # Lädt die Datei hoch
            with open(input_file_path, "rb") as data:
                container_client.upload_blob(name=output_blob_name, data=data)

            os.remove(input_file_path)


# Führt das Programm jede Minute aus
schedule.every(1).minutes.do(getFilesSendAndDelete)

while True:
    schedule.run_pending()
    time.sleep(1)
```