Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

Bachelorarbeit

Thorvin Kruse

Verantwortungsrückkopplung für Energieproportionale CO2-Emissionen in der IKT

Fakultät Technik und Informatik
Studiendepartment Informations- und Elektrotechnik

Faculty of Engineering and Computer Science
Department of Information and Electrical Engineering

Thorvin Kruse

Verantwortungsrückkopplung für Energieproportionale CO2-Emissionen in der IKT

Bachelorarbeit eingereicht im Rahmen der Bachelorprüfung

im Studiengang Bachelor of Science Elektro- und Informationstechnik
am Department Informations- und Elektrotechnik
der Fakultät Technik und Informatik
der Hochschule für Angewandte Wissenschaften Hamburg

Betreuender Prüfer: Prof. Rainer Schoenen
Zweitgutachter: Prof. Kolja Sören Eger

Eingereicht am: 24. Mai 2024

Thorvin Kruse

Thema der Arbeit
Verantwortungsrückkopplung für Energieproportionale CO2-Emissionen in der IKT

Stichworte
User-in-the-Loop, IKT, CO2-Emissionen

Kurzzusammenfassung
Diese Arbeit untersucht die Ermittlung des Stromverbrauchs eines Computers anhand der Leistungsdaten der CPU. Durch den Einsatz regressionsanalytischer Verfahren wird der Stromverbrauch approximiert, um ihn anschließend spezifischen Prozessen des Rechners zuordnen zu können. Weiterhin wird der Energieverbrauch des Gesamtsystems und der einzelnen Prozesse im Kontext ihres Beitrags zur globalen Erwärmung analysiert. Die Ergebnisse werden grafisch in einer Benutzeroberfläche (GUI) dargestellt, basierend auf dem User-in-the-Loop-Prinzip, das den Ressourcenverbrauch dem Endnutzer visuell zurückmeldet. Diese Methode soll das Bewusstsein für den Energieverbrauch und dessen Umweltauswirkungen schärfen.
Thorvin Kruse

Title of the paper
Responsibility feedback for energy-proportional CO2 emissions in ICT.

Keywords
User-in-the-Loop, IKT, CO2-Emissionen

Abstract
This thesis explores the determination of a computer's power consumption based on CPU performance data. By using regression analysis techniques, power consumption is approximated and then attributed to specific processes within the computer. Furthermore, the energy consumption of the entire system and individual processes is analyzed in the context of their contribution to global warming. The results are graphically presented in a graphical user interface (GUI), based on the User-in-the-Loop principle, which visually feeds back resource consumption to the end-user. This method aims to increase awareness of energy consumption and its environmental impacts.

Inhaltsverzeichnis

Tabellenverzeichnis

Abbildungsverzeichnis

1 Einleitung

1.1 Hintergrund und Bedeutung des Themas

Der zunehmende Einsatz von Computertechnologie in fast allen Bereichen des täglichen Lebens hat zu einem drastischen Anstieg des globalen Energieverbrauchs geführt. Während die Digitalisierung der Gesellschaft unbestreitbare Vorteile mit sich bringt, sind sich viele Nutzer nicht der versteckten Kosten bewusst, die dieser Fortschritt mit sich bringt. Insbesondere in Bezug auf den CO2-Ausstoß und die damit verbundenen Auswirkungen auf den Klimawandel, welche durch den Konsum von Software entstehen.
Der Energieverbrauch, der durch die Nutzung von Computern entsteht, trägt signifikant zu den gesamten CO2-Emissionen bei, die unserer Umwelt schaden und den bereits bestehenden Klimawandel vorantreiben.
Dieses Thema gewinnt vor dem Hintergrund der globalen Klimakrise an Bedeutung, da der Klimawandel nicht nur eine Bedrohung für zukünftige Generationen darstellt, sondern bereits jetzt spürbare Auswirkungen auf unser aller Leben hat.
Die direkte Verbindung zwischen der Nutzung von Informationstechnologie und dem Klimawandel ist vielen Nutzern jedoch nicht bewusst. Viele betrachten ihren persönlichen oder beruflichen Gebrauch von Computern als notwendig und unvermeidlich, ohne die daraus resultierenden Umweltbelastungen zu bedenken. Dies führt zu einer Diskrepanz zwischen dem individuellen Handeln und den kollektiven Auswirkungen auf die Umwelt. Das Bewusstsein für den Energieverbrauch und die CO2-Emissionen, die durch die Computernutzung entstehen, ist daher ein entscheidender Schritt, um den ökologischen Fußabdruck der digitalen Ära zu verstehen und zu verringern.

In dieser Arbeit wird untersucht, wie der Stromverbrauch von Computern anhand der CPU-Auslastung ermittelt werden kann und welche Auswirkungen dies auf die Umwelt hat. Durch die Quantifizierung des Energieverbrauchs und dessen Rückführung auf einzelne Prozesse soll ein tieferes Verständnis dafür geschaffen werden, wie Nutzer innerhalb der IKT ihren Beitrag zum Klimawandel minimieren können. Dies ist von entscheidender Bedeutung, da der Kampf

gegen den Klimawandel eine gemeinsame Anstrengung erfordert, bei der jeder Einzelne eine Rolle spielt. Die Sensibilisierung für den CO2-Ausstoß, der durch alltägliche Aktivitäten wie die Computernutzung verursacht wird, ist ein wichtiger Schritt, um Verhaltensänderungen zu fördern und den Übergang zu nachhaltigeren Praktiken zu beschleunigen.

1.2 Zielsetzung der Arbeit

Das Hauptziel dieser Arbeit ist die Ermittlung des Stromverbrauchs eines Personal Computers (PC) durch die Analyse der CPU-Auslastung mit Hilfe regressionsanalytischer Verfahren. Die vorliegende Untersuchung konzentriert sich darauf, den Einfluss der auf einem Computer ausgeführten Aktivitäten und Prozesse auf seinen Gesamtenergieverbrauch zu verstehen. Indem die CPU-Auslastung als Indikator für den Energieverbrauch herangezogen wird, zielt diese Arbeit darauf ab, ein Modell zu entwickeln, das nicht nur den Gesamtstromverbrauch des PCs schätzt, sondern diesen auch spezifisch den einzelnen laufenden Prozessen zuordnen kann.
Diese Analyse soll hierbei für Vergleichszwecke auf unterschiedlichen Rechnern sowie Betriebssystemen durchgeführt werden.
Ein weiteres Ziel besteht in der Entwicklung einer grafischen Benutzeroberfläche (GUI), die es den Nutzern ermöglicht, den Energieverbrauch ihres Computers in Echtzeit zu überwachen. Diese Visualisierung soll nicht nur informativ sein, sondern nach dem Prinzip des User-in-the-Loop auch als Sensibilisierungsinstrument dienen, um die Nutzer zu einem bewussteren Umgang mit den Ressourcen ihres Computers zu motivieren.
Durch die Darstellung des direkten Zusammenhangs zwischen der Nutzung spezifischer Programme und dem Energieverbrauch beabsichtigt die Arbeit, einen Beitrag zur Sensibilisierung des Nutzers für seinen Beitrag zum Klimawandel zu leisten.

Zusammengefasst verfolgt diese Arbeit folgende Ziele:

1. Entwicklung eines Modells zur Ermittlung des Gesamtstromverbrauchs eines PCs basierend auf der CPU-Auslastung.

2. Detaillierte Zuordnung des Stromverbrauchs zu einzelnen Prozessen, um dessen Einfluss auf das Klima identifizieren zu können.

3. Erstellung einer benutzerfreundlichen GUI nach dem Prinzip des User-in-the-Loop, die die genannten Informationen visualisiert und das Bewusstsein sowie Verständnis der Nutzer für den Energieverbrauch ihres Computers steigert.

1.2.1 Forschungsfrage und Hypothesen

Die zentrale Frage dieser Arbeit lautet: „Wie kann der Stromverbrauch eines PCs auf Basis der CPU-Auslastung ermittelt und spezifischen Prozessen zugeordnet werden, um dem Nutzer seinen Beitrag zum Klimawandel aufzuzeigen und diesen in eine bessere Richtung zu lenken?“ Diese Frage adressiert die wachsende Notwendigkeit, den Energieverbrauch von IT-Geräten im Kontext des globalen Klimawandels zu verstehen und zu bewerten.
Um diese Frage zu untersuchen, werden folgende Hypothesen aufgestellt:

1. **Hypothese 1:** Die CPU-Auslastung ist ein zuverlässiger Indikator für den Energieverbrauch eines PCs. Diese Hypothese basiert auf der Annahme, dass höhere CPU-Auslastungen mit einem erhöhten Energieverbrauch korrelieren und somit eine präzise Schätzung des Stromverbrauchs ermöglichen.

2. **Hypothese 2:** Durch die Anwendung regressionsanalytischer Verfahren kann der Stromverbrauch eines PCs aufgrund seiner CPU-Auslastung mit einer gewissen Genauigkeit vorhergesagt werden.

3. **Hypothese 3:** Die Zuordnung des Stromverbrauchs zu einzelnen Prozessen sowie die grafische Darstellung der Auswirkungen auf das Klima sind geeignet, den Nutzer von seinem Verhalten abzubringen und ihm ein Bewusstsein für die Auswirkungen auf das Klima zu schaffen.

Die Überprüfung dieser Hypothesen soll nicht nur einen wissenschaftlichen Beitrag zur Verbindung zwischen CPU-Auslastung und Energieverbrauch leisten, sondern auch praktische Empfehlungen für Nutzer hervorbringen, um den ökologischen Fußabdruck digitaler Technologien zu reduzieren.

1.2.2 Aufbau der Arbeit

Die vorliegende Arbeit ist in mehrere Kapitel gegliedert, die systematisch den Weg von der Einleitung über die theoretischen Grundlagen und Methodik bis hin zu den Ergebnissen und ihrer Diskussion beschreibt. Ziel ist es, einen klaren und logischen Pfad durch die Arbeit zu bieten.

1. **Einleitung:** Dieses Kapitel setzt den Rahmen für die Arbeit, hierbei wird der Hintergrund der Arbeit beleuchtet, die zentrale Forschungsfrage vorgestellt und die Bedeutung der Arbeit im Kontext des Klimawandels vorgestellt. Des Weiteren werden die Zielsetzungen der Arbeit und die Hypothesen dargelegt.

2. **Theoretische Grundlagen:** Hier werden alle relevanten Konzepte und Theorien die für das Verständnis der Arbeit notwendig sind, behandelt. Dies umfasst die Erklärung der CPU-Auslastung und die ökologischen Auswirkungen der IT-Nutzung sowie das Prinzip des User-in-the-Loop, welches zur Rückkopplung auf den Benutzer dient.

3. **Methodik:** In diesem Abschnitt wird der methodische Ansatz der Arbeit beschrieben, einschließlich der Verfahren zur Messung des Stromverbrauchs sowie dem Auslesen der CPU-Auslastung und der Auswahl der Regressionsmodelle.

4. **Durchführung der Messungen:** Dieses Kapitel beschreibt den Messaufbau sowie die Messverfahren. Es wird auch erläutert, wie die Daten für die Analyse vorbereitet und aufbereitet wurden.

5. **Ergebnisse:** Die in diesem Kapitel präsentierten Ergebnisse bieten einen detaillierten Überblick über die aus der Datenauswertung gewonnenen Erkenntnisse sowie die Ergebnisse der Regression.

6. **Rückführung der Ergebnisse:** Dieses Kapitel befasst sich mit der Rückkopplung der Ergebnisse auf den Nutzer sowie die Auswirkungen einzelner Prozesse auf das Klima.

7. **Diskussion:** Hier werden die Ergebnisse im Kontext der ursprünglichen Fragestellung und der gesetzten Hypothesen diskutiert. Es wird eine kritische Bewertung der Methodik vorgenommen, und die Bedeutung der Ergebnisse für die Praxis wird erörtert. Zudem werden Limitationen der Arbeit und Ansätze für auf dieser Arbeit aufbauende Entwicklungen diskutiert.

8. **Fazit und Ausblick:** Das abschließende Kapitel fasst die wichtigsten Erkenntnisse der Arbeit zusammen, reflektiert die erzielten Erkenntnisse in Bezug auf die Ziele der Arbeit und gibt einen Ausblick auf mögliche zukünftige Entwicklungen und Forschungsfelder.

9. **Anhang:** Der Anhang enthält ergänzende Informationen, Daten und Materialien, die für die Vollständigkeit der Arbeit relevant sind. Der vollständige Anhang zur Arbeit befindet sich auf CD und kann beim Erstgutachter eingesehen werden.

10. **Literaturverzeichnis:** Dieser Teil listet alle Quellen auf, die im Verlauf der Arbeit zitiert wurden.

1.3 Vorhandene Ansätze

Die Methode, den Stromverbrauch durch Analyse von Softwaredaten eines Computers zu bestimmen, ist hierbei nicht vollständig neu. Microsoft bietet bereits im Task Manager eine solche Funktion, bei der der Energieverbrauch jedes Prozesses angezeigt wird. Diese Angaben sind jedoch recht grob und erfolgen in einfachen Kategorien ohne Angabe von genauen Werten. Der Energieverbrauch wird in den folgenden Stufen dargestellt:

- gering
- mittel
- hoch
- sehr hoch

Obwohl diese Einteilungen einen ersten hilfreichen Überblick bieten, ermöglichen sie keine exakte Quantifizierung des Energieverbrauchs sowie eine Beschreibung der Auswirkungen auf das Klima.

Ein weiterer Ansatz zur Bestimmung des Energieverbrauchs einzelner Prozesse wird von Microsoft durch die Windows Energy Estimation Engine (E3) umgesetzt. Diese Technologie ermittelt den Energieverbrauch basierend auf den Daten des Computers und speichert diese Informationen in einer Datei, die über einen Kommandozeilenbefehl als Excel-Datei exportiert werden kann. Die Energieverbrauchswerte werden in Joule angegeben. Dieser Ansatz funktioniert jedoch nur auf Geräten mit Akkubetrieb, da die Berechnung des Stromverbrauchs auf der Grundlage der Akkunutzung erfolgt. Zudem ist das Auslesen und Auswerten der Datei aufgrund ihrer Komplexität für durchschnittliche Nutzer ungeeignet. [5]

2 Theoretische Grundlagen

2.1 User-in-the-Loop (UIL)

Das Prinzip vom User-in-the-Loop (UIL) basiert auf der Forschungsarbeit von Professor Schoenen und zielt darauf ab, eine gezielte Beeinflussung des Nutzers zu ermöglichen, um sein Verhalten hinsichtlich der Nutzung von Ressourcen wie beispielsweise Datenvolumen oder Energie anzupassen. Diese Methode basiert auf der Annahme, dass menschliche Nutzer zu den intelligentesten, aber auch unberechenbarsten Einheiten in einem Netzwerk gehören.

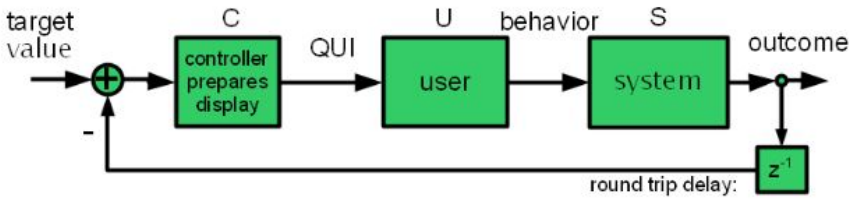

Abbildung 2.1: Prinzip vom User-in-the-Loop [25]

Wie in Grafik 2.1 zu erkennen, fungiert das Prinzip des User-in-the-Loop als geschlossener Regelkreis, wobei der Nutzer Teil des Systems ist. Der Grundgedanke hinter dem Prinzip ist es, dem Nutzer Vorschläge sowie Anreize zu schaffen, von seinem Standardverhalten abzuweichen. Dies geschieht, indem dem Nutzer über beispielsweise eine GUI sein Verhalten sowie die Konsequenzen hieraus aufgezeigt werden. Dieser nimmt diese Informationen auf und kann anhand dieser die Entscheidung treffen, von seinem aktuellen Verhalten, welches viele Ressourcen verbraucht, auf ein Verhalten mit weniger Ressourcenverbrauch zu wechseln. [25]

2.2 CPU-Auslastung und deren Messung

2.2.1 Bedeutung der CPU-Auslastung

Was ist CPU-Auslastung?
Die CPU-Auslastung ist ein Maß dafür, wie sehr der Prozessor eines Computers mit Aufgaben belastet ist. Sie wird in Prozent angegeben und zeigt an, wie viel der verfügbaren Rechenkapazität der CPU zu einem bestimmten Zeitpunkt genutzt wird. Eine hohe CPU-Auslastung bedeutet, dass der Prozessor intensiv arbeitet, was auf eine hohe Menge an laufenden oder komplexen Prozessen hinweist. Während eine niedrige Auslastung auf eine geringere Beanspruchung schließen lässt. [14]

Was ist Idle?
Idle bezeichnet den Zustand der CPU, wenn sie ungenutzt ist. Wenn keine Aufgaben ausgeführt werden müssen, geht die CPU in den Idle-Modus, um Energie zu sparen und die Effizienz zu erhöhen. Die Idle-Zeit ist somit ein Indikator für die Nicht-Auslastung der CPU. [14]

Welche Arten der Auslastung gibt es?
Es gibt verschiedene Arten der CPU-Auslastung, die betrachtet werden können:

- **User Time:** Die Zeit, die die CPU mit der Ausführung von Benutzerprozessen verbringt.
- **System Time:** Die Zeit, die für Betriebssystemaufgaben verwendet wird.
- **Idle Time:** Die Zeit, in der die CPU keine Aufgaben ausführt.

[14]

Was sind logische Kerne?
Logische Kerne, auch als virtuelle Kerne bekannt, sind eine Technologie, die es einer physischen CPU erlaubt, wie zwei separate Prozessoren zu funktionieren. Dies wird oft durch Hyper-Threading oder ähnliche Technologien erreicht, bei denen jedem physischen Kern des Prozessors zwei logische Prozessoren zugeordnet werden, wodurch die Bearbeitung von Prozessen und Threads effizienter gestaltet wird. [14]

Wie ändert sich die Frequenz der CPU?
Die Frequenz der CPU, oft als Taktgeschwindigkeit bezeichnet, kann sich dynamisch ändern, um den Energieverbrauch zu optimieren und die Wärmeentwicklung zu kontrollieren. Dies

geschieht durch Technologien wie Intel's Turbo Boost, die die Frequenz erhöhen, wenn hohe Leistung benötigt wird, und sie reduzieren, wenn die Last gering ist. [26]

2.2.2 Methoden zur Messung unter Windows

Unter dem Betriebssystem Windows gibt es unterschiedliche Möglichkeiten, die Auslastung der CPU sowie die Anteile der Prozesse an der CPU-Auslastung anzeigen zu lassen. Auf zwei davon soll hier genauer eingegangen werden, dies sind einmal das Auslesen der Werte über typeperf sowie über die Windows API. Grundlegend unterscheidet Windows hierbei zwischen zwei Arten von Leistungsindikatoren. Dies sind einmal Leistungsindikatoren, welche aus nur einem Messwert erstellt werden, und den Ratenindikatoren, diese benötigen zwei Messwerte zur Berechnung eines anzeigbaren Wertes. Die Leistungsindikatoren, welche die CPU-Auslastung des Prozessors sowie der einzelnen Prozesse anzeigen, sind hierbei Ratenindikatoren, da diese Counter nutzen, welche die Zeit messen, die die CPU aktiv sowie im Idle-Prozess verbracht hat und errechnen hieraus die CPU-Auslastung in Prozent. [12]

Der Leistungsindikator `Prozessor`, welcher für das Auslesen der Auslastung der gesamten CPU genutzt wird, nutzt hierfür den PERF_100NSEC_TIMER_INV-Zählertyp.
Die Formel zur Berechnung des Prozentsatzes aktiver Zeit, basierend auf dem PERF_100NSEC_TIMER_INV-Zählertyp, ist:

$$\text{Prozentsatz} = \left(1 - \frac{N_1 - N_0}{D_1 - D_0}\right) \cdot 100$$

Hierbei repräsentieren N_0 und N_1 die Zählerstände zu Beginn und Ende des Messintervalls und D_0 und D_1 die entsprechenden Zeitpunkte in 100-Nanosekunden-Einheiten. [20]

Der Leistungsindikator `Prozess`, welcher die Prozessorzeit für einen einzelnen Prozess berechnet, wurde ursprünglich für Systeme mit nur einem Prozessor entwickelt. In diesem Kontext kann die CPU-Auslastung eines Prozesses zwischen 0 und 100 Prozent variieren, wobei 100 Prozent eine vollständige Ausnutzung des Prozessors durch den Prozess anzeigen. Dieser Leistungsindikator nutzt den PERF_100NSEC_TIMER, welcher sich wie folgt berechnet:

$$\text{Prozentsatz} = \frac{(N_1 - N_0)}{(D_1 - D_0)} \cdot 100$$

wobei der Nenner D die gesamte verstrichene Zeit des Stichprobenintervalls darstellt und der Zähler N die Teile des Stichprobenintervalls repräsentiert, während derer die überwachten Komponenten aktiv waren. [1]

Bei Mehrprozessorsystemen, wie sie in dieser Arbeit untersucht werden, gestaltet sich die Berechnung der prozentualen Gesamtauslastung komplexer. In solchen Systemen kann die prozentuale Auslastung, die für einen einzelnen Kern berechnet wird, dazu führen, dass die summierte Auslastung des Prozesses auf allen Kernen den Wert von 100 Prozent übersteigt. Die maximale angezeigte CPU-Auslastung entspricht daher 100 Prozent multipliziert mit der Anzahl der logischen Kerne des Systems. Dies bedeutet, dass bei einem System mit 8 logischen Kernen die Gesamtauslastung bis zu 800 Prozent erreichen kann, wenn jeder Kern vollständig durch den Prozess ausgelastet ist.
Der Leistungsindikator `Prozess` verwendet hierbei den Instanznamen des Programms zur Berechnung der CPU-Auslastung. Dieser Instanzname ist hierbei nicht eindeutig bzw. kann mehrfach im System auftreten, was bei mehreren gleichnamigen Prozessen keine eindeutige Zuordnung zu einem Prozess zulässt. [12]

Auslesen mittels typeperf

`typeperf` ist ein Kommandozeilentool, das zur Erfassung von Leistungsindikatoren unter Windows verwendet wird. Mit diesem Tool können Leistungsdaten direkt in das Befehlsfenster ausgegeben oder in eine Protokolldatei geschrieben werden. Die allgemeine Syntax von `typeperf` lautet wie folgt:

```
typeperf <counter [counter ...]> [options]
```

Um beispielsweise die CPU-Auslastung aller Kerne zu messen, kann man `typeperf` mit dem spezifischen Leistungsindikator für die gesamte Prozessorzeit nutzen. Die Eingabe würde dann so aussehen:

```
typeperf "\Processor(_Total)\% Processor Time"
```

[9]

Auslesen mittels PDH-Funktionen

Eine alternative Methode zum Auslesen von Leistungsindikatoren unter Windows stellt die Verwendung der Performance Data Helper (PDH) API dar. Diese API ermöglicht das Sammeln und Auswerten aktueller Leistungsdaten und kann durch Programmiersprachen wie C++ genutzt werden, jedoch nicht durch Java. Um die PDH-Funktionen verwenden zu können, ist das Einbinden der Bibliothek `pdh.h` erforderlich. [8]

Im Vergleich zur Verwendung von `typeperf` bieten die PDH-Funktionen einen erheblich größeren Gestaltungsspielraum sowie eine erhöhte Flexibilität beim Auslesen von Leistungsindikatoren. Dies ermöglicht eine detailliertere und anpassbare Überwachung der Systemleistung.

2.2.3 Methoden zur Messung unter Linux

Zum Auslesen der Daten unter Linux stellt das Betriebssystem hier das `/proc`-Verzeichnis zur Verfügung. Hierbei handelt es sich um ein Pseudo-Verzeichnis, welches keinen Speicherplatz auf der Festplatte benötigt und sich lediglich im RAM, also dem Hauptspeicher, befindet. Dieses wird vom Kernel angelegt und beinhaltet Informationen zum System. Hierbei kann auf dieses Pseudo-Verzeichnis wie auf jedes Verzeichnis zugegriffen werden, dies allerdings nur lesend. Die für diese Arbeit nötigen Informationen liegen hierbei im Verzeichnis /proc/stat, dieses Verzeichnis beinhaltet alle Informationen zur CPU-Auslastung sowie den einzelnen Kernen dieser.
Ebenfalls von Interesse ist das Verzeichnis für die einzelnen Prozesse. Hierbei beinhaltet das Verzeichnis für jeden auf dem System laufenden Prozess ein eigenes Unterverzeichnis, welches als Namen die aktuelle Prozess-ID des jeweiligen Prozesses trägt. Programme wie ps oder top nutzen hierbei ebenfalls diese Verzeichnisse zum Erstellen von Werten wie der aktuellen CPU-Auslastung. [2]

Die Datei `/proc/stat` sowie das Verzeichnis für die einzelnen Prozesse liefern hierbei keine direkten Prozentangaben zur CPU-Auslastung. Stattdessen werden Rohdaten zur Verfügung gestellt, welche die Zeitdauer repräsentieren, die die CPU in verschiedenen Modi verbracht hat. Diese Zeiten werden in Einheiten von USER_HZ gemessen, welche $1/100$ Sekunden entsprechen.
Die relevanten Werte in `/proc/stat` sind wie folgt und kategorisiert:

- **User Mode:** Die Zeit, die Prozesse im User Mode (Nicht-Kernmodus) verbracht haben, ohne erhöhte Priorität durch `nice`.
- **Nice:** Die Zeit im User Mode, während der Prozesse eine niedrige Priorität hatten, was durch `nice` eingestellt wird.
- **System Mode:** Die Zeit, die im System- oder Kernel-Modus verbracht wurde, d. h., die Zeit, in der der Kernel Operationen für Prozesse durchführt.
- **Idle:** Die Zeit, in der die CPU nicht genutzt wurde. Dies entspricht der Zeit, die im Leerlaufprozess verbracht wurde.

Die Datei besteht hierbei aus mehreren Zeilen sowie Spalten, eine Zeile gibt hierbei die oben genannten Modi der CPU an, wobei diese der Reihenfolge nach sortiert sind. Die einzelnen Zeilen stehen hierbei für die unterschiedlichen Optionen, die angeboten werden, die CPU-Auslastung zu analysieren. Hierbei stehen in der ersten Zeile die Werte, die sich auf die gesamte CPU beziehen, die darunterliegenden Zeilen beziehen sich auf die einzelnen Kerne, welche somit die Auslastung jeweils nur für den einzelnen Kern liefern. [24]

2.3 Vorstellung der Regressionsanalyse

Die Regressionsanalyse ist ein statistisches Verfahren, das verwendet wird, um die Beziehung zwischen einer abhängigen Variablen y, auch Zielvariable genannt, und einer oder mehreren unabhängigen Variablen, bekannt als Kovariablen oder erklärende Variablen, zu untersuchen.

- **Definition und Ziel:** In der Regressionsanalyse wird versucht, die Zielvariable y auf Basis der Kovariablen X zu erklären. Der Grundgedanke ist, dass y zwar von X abhängt, aber nicht exakt durch X allein vorhergesagt werden kann, da zufällige Störungen u die Beziehung überlagern. Diese Störungen werden verursacht durch Messfehler sowie die Einflüsse von Faktoren hervorgerufen, welche nicht im Modell erfasst sind.

- **Mathematische Modellierung:** Mathematisch wird die Zielvariable y als Funktion der Kovariablen X plus einer zufälligen Störgröße u ausgedrückt als:

  $$y_i = \beta_0 + \beta_1 x_1 + \ldots + \beta_k x_k + u$$

  Hierbei ist $\beta_0, \beta_1, \ldots, \beta_k$ eine Reihe von Koeffizienten, die die Stärke und Art des Einflusses jeder Kovariablen auf y beschreiben. u ist die Störgröße, die als normalverteilt angenommen wird mit einem Erwartungswert von Null und einer konstanten Varianz.

- **Schätzung der Koeffizienten:** Die zentrale Aufgabe in der Regressionsanalyse besteht darin, die unbekannten Koeffizienten β zu schätzen, welche die Beziehung zwischen den Kovariablen und der Zielvariable erklärt. Dies geschieht mit der Methode der kleinsten Quadrate, welche die Summe der quadrierten Abweichungen der vorhergesagten Werte von den tatsächlichen Daten minimiert.

- **Überprüfung der Modellannahmen:** Nach der Schätzung der Koeffizienten ist es wichtig, die Modellannahmen zu überprüfen, um die Gültigkeit der Regression sicherzustellen. Dies umfasst die Prüfung auf Homoskedastizität, Unkorreliertheit der Residuen und Normalverteilung der Residuen.

[4]
Im einfachsten Fall, der einfachen linearen Regression, wird eine einzige unabhängige Variable verwendet, um die abhängige Variable vorherzusagen. Die Beziehung zwischen den Variablen wird durch eine gerade Linie dargestellt, die so durch die Datenpunkte gelegt wird, dass sie die Summe der quadrierten Abstände zwischen den beobachteten Werten und den durch das Modell vorhergesagten Werten minimiert. Diese Linie wird als Regressionsgerade bezeichnet und hat die allgemeine Form:

$$y = a + bx + u$$

wobei a und b reelle Zahlen sind, und u die zufälligen Fehler repräsentiert. Die Fehlervariable u umfasst hierbei alle zufälligen und unsystematischen Fehler, für die folgende Annahme gelten muss:

1. Der Erwartungswert $E(u)$ ist Null, was bedeutet, dass die Fehler sich im Mittel aufheben und somit keinen systematischen Einfluss auf y haben.

2. Die Kovarianz zwischen unterschiedlichen Fehlertermen in u ist Null. Eine Kovarianz von null bedeutet hierbei, dass die Änderungen in einer Störvariable nicht systematisch mit den Änderungen in der anderen Störvariable ist. Ist dies nicht gegeben, so deutet dies darauf hin, dass die Abweichung der Störvariable nicht zufällig ist.

3. Die Varianz von u ist konstant und gleich der Standardabweichung für alle Störterme. Ist diese Eigenschaft gegeben, so spricht man von einer Heteroskedastzität.

Unter diesen Annahmen ist der Erwartungswert von der Zielvariable gegeben durch $E(y) = a + bx$, und die Varianz von y ist konstant und gleich der Standartabweichung. Die Regressionsgerade repräsentiert somit den erwarteten Mittelwert der Zielvariable, was bedeutet, dass sie als Mittellinie für die Beobachtungen dient.
[4]

Das durch Regression erstellte Modell kann hierbei nicht die Realität vollständig abbilden, sodass lediglich eine möglichst gute Annäherung an die Realität erzeugt werden kann. Um die Güte dieser Annäherung zu untersuchen, müssen die oben genannten Bedingungen der Fehlervariable genauer untersucht werden. Hierzu wird der Begriff der Residuen eingeführt. Ein Residuum ist hierbei die Differenz zwischen einem beobachteten Wert und dem durch die Funktion des Modells vorhergesagten Wert. Formal ausgedrückt, wenn y_i der beobachtete

Wert und $\hat{y}_i$ der vom Modell vorhergesagte Wert für die i-te Beobachtung ist, dann ist das Residuum e_i gegeben durch:

$$e_i = y_i - \hat{y}_i$$

Mittels der Residuen ist nun eine Überprüfung der obigen Voraussetzungen für die Güte des Modells möglich. [4]

2.4 Bedeutung von Energieeffizienz und Nachhaltigkeit in der Informationstechnologie

2.4.1 Emissionsfaktor

Der Emissionsfaktor ist ein Wert, welcher die direkten CO2-Emissionen quantifiziert, die durch die Erzeugung einer Kilowattstunde Strom entstehen. Dieser Faktor ist besonders wichtig, da er dazu beiträgt, die Umweltauswirkungen der Stromproduktion zu bewerten. Da der in Deutschland erzeugte Strom aus mehreren Energiequellen stammt, variiert der CO2-Faktor für jeden Energieträger erheblich. Um also ein Gesamtbild der CO2-Emissionen zu erhalten, sind alle Energieträger, welche zur Stromproduktion in Deutschland beitragen, in der Berechnung mit einzuschließen. [3]

Zuordnung von Emissionen: Für die Berechnung des Emissionsfaktors wird angenommen, dass sämtliche Stromerzeugnisse, einschließlich eventueller Überschüsse, dem Land zugerechnet werden, in dem sie produziert werden. Dies sorgt dafür, dass die Verantwortung für die Emissionen dort liegt, wo der Strom erzeugt wird. [3]

Emissionsfaktoren verschiedener Energieträger: Die Emissionen aus der Stromerzeugung in Deutschland stehen direkt im Zusammenhang mit den verwendeten Energieträgern. Die folgende Tabelle zeigt die CO2-Emissionen pro Kilowattstunde für drei Hauptenergieträger, die in der deutschen Energieerzeugung genutzt werden.

Energieträger	CO2-Emission in g pro kWh
Erdgas	381
Steinkohle	835
Braunkohle	1137

Tabelle 2.1: CO2-Emissionen für Energieträger

[3]

Berechnung des Emissionsfaktors: Erneuerbare Energien, die in dieser Rechnung berücksichtigt werden, haben einen CO2-Koeffizienten von 0. Es ist jedoch wichtig zu beachten, dass diese Bewertung der erneuerbaren Energien die CO2-Emissionen, die bei Bau und Produktion der Anlagen anfallen, nicht miteinschließt. Der durchschnittliche Emissionsfaktor für Deutschland im Jahr 2022 wurde folgendermaßen berechnet:

$$\text{Emissionsfaktor} = \frac{\text{direkte CO2-Emissionen}}{\text{Stromverbrauch}}$$

Dieser Berechnung zufolge lag der Emissionsfaktor für Deutschland im Jahr 2022 bei 434 Gramm pro kWh.

[3]

Da jedes Land einen unterschiedlichen Anteil von erneuerbaren Energien besitzt und sich der Strommix jedes Landes unterschiedlich darstellt, weicht der Emissionsfaktor der gesamten Welt sowie für einzelne Länder von dem deutschen Faktor ab.
Der CO2-Faktor für die weltweite Betrachtung liegt für das Jahr 2022 bei 486 Gramm CO2-Ausstoß pro geleisteter Kilowattstunde. [23]

2.4.2 Stromverbrauch in der IKT

Der elektrische Energiebedarf in der Informations- und Kommunikationstechnik (IKT) machte im Jahr 2010 mehr als 10 Prozent des gesamten bundesweiten Energieverbrauchs aus. Aufgrund fortlaufender Innovationen und Optimierungen in der Leistungseffizienz wird erwartet, dass der Energiebedarf der IKT bis zum Jahr 2025 um 10 TWh auf insgesamt 46 TWh reduziert wird. In dieser Arbeit liegt der Fokus speziell auf dem Energieverbrauch von Computern und Laptops, die in zwei Hauptbereichen der IKT zum Einsatz kommen: an Arbeitsplätzen in deutschen Unternehmen und in privaten Haushalten.

- **Arbeitsplatz:** Nach einer Studie des Fraunhofer-Instituts waren im Jahr 2020 etwa 9,5 Millionen Desktop-PCs und 16,8 Millionen Laptops in deutschen Unternehmen im Einsatz. Prognosen für das Jahr 2025 gehen von einer Zunahme auf 7 Millionen Desktop-PCs und 20 Millionen Laptops aus. Im Jahr 2020 betrug die durchschnittliche Nutzungsdauer 6,5 Stunden täglich bei einer Standby-Zeit von 17 Stunden, bei einer jährlichen Nutzung von 220 Tagen. Für 2025 wird eine erhöhte Standby-Zeit von 17,5 Stunden erwartet, bei gleichbleibender Nutzungsdauer. Laut Berechnungen des Fraunhofer-Instituts wird der Gesamtenergiebedarf, der im Jahr 2020 bei 1329 GWh lag, bis 2025 auf 994 GWh sinken,

hauptsächlich bedingt durch eine Reduktion des Energiebedarfs der Systeme um etwa ein Drittel.

- **Haushalt:** In deutschen Haushalten waren 2020 etwa 15,5 Millionen Desktop-PCs und 28,5 Millionen Notebooks in Betrieb. Bis 2025 wird ein Rückgang der Desktop-PCs auf 12,9 Millionen und ein Anstieg der Notebooks auf 29,8 Millionen erwartet. Die Nutzungsmuster blieben unverändert mit einer angenommenen Nutzung von 356 Tagen im Jahr, da die Geräte privat verwendet werden. Die tägliche Nutzungsdauer wurde für 2020 auf 3,5 Stunden geschätzt, mit weiteren 20,5 Stunden im Standby- oder ausgeschaltetem Zustand. Der Gesamtenergiebedarf, der für das Jahr 2020 auf 1196 GWh geschätzt wurde, wird für 2025 auf 972 GWh prognostiziert.

[3]

2.5 Herausforderungen und Lösungsansätze

Die Messung der CPU-Auslastung sowie die Stromaufnahme stellen sowohl auf der Hardware- als auch auf der Softwareseite einige Herausforderungen dar, welche im Folgenden genauer erörtert werden.

Zeitliche Verschiebung der Messungen: Bei der Durchführung von zwei simultanen Messungen (einer softwarebasierten und einer mittels Multimeter) ist die zeitliche Gleichheit der Daten von entscheidender Bedeutung für die spätere Analyse. Jede Messung wird mit einem Zeitstempel versehen, um eine Zuordnung der Ergebnisse zu ermöglichen. Da jedes Gerät hierbei eine eigene Uhr zum Erstellen der Zeitstempel besitzt, können hierbei relative Abweichungen innerhalb der beiden Systemzeiten auftreten. Um dieses Problem zu lösen, wird ein Verfahren verwendet, das eine Synchronisation der Messdaten mittels einer analogen Triggerflanke, welche durch das Messgerät ausgewertet wird und eines nahezu zeitgleichen digitalen Signals, welches durch den Computer ausgewertet wird, ermöglicht. Diese Methode stellt sicher, dass die Zeitstempel beider Messgeräte präzise abgeglichen werden können.

CPU-Last erzeugen: Um zuverlässige und aussagekräftige Daten über den Stromverbrauch und die Leistung eines Computers unter verschiedenen Belastungsbedingungen zu erhalten, ist es notwendig, die CPU-Auslastung für den Zeitraum der Messung auf einem konstanten Level zu halten bzw. die CPU-Auslastung in möglichst viele Bereiche zu bringen, um diese in der späteren Auswertung gut verarbeiten zu können.

Hierzu wird spezielle Software eingesetzt, welche es ermöglicht, die Last der CPU künstlich hochzutreiben und zumindest näherungsweise auf einem eingestellten Level zu halten. Dieser Ansatz ermöglicht die Erstellung einer möglichst kontrollierten Testumgebung, in der die CPU-Auslastung systematisch variiert werden kann, um ein breites Spektrum an Leistungsdaten zu erfassen.

Unvorhersehbare CPU-Auslastung: Die CPU-Auslastung wird durch laufende Prozesse bestimmt, die vom Betriebssystem verwaltet werden. Der Nutzer hat keinen direkten Einfluss darauf, welche Prozesse aktiv sind und welche nicht. Obwohl externe Programme theoretisch eine konstante Last erzeugen können, kann das Hinzukommen weiterer Prozesse im Hintergrund die Messungen verfälschen. Um dieses Problem zu minimieren, werden für die Messungen Computer eingesetzt, die möglichst wenige Hintergrundprogramme aufweisen. Somit werden nur die für die Messung essenziellen Programme ausgeführt, um externe Störfaktoren so gering wie möglich zu halten.
Dies führt zur Verbesserung der Messergebnisse, da somit plötzliche Laständerungen innerhalb der CPU-Auslastung minimiert werden können.

Frequenzen in der Messung: Ein weiterer Punkt, welchen es zu beachten gibt, ist, dass das Messgerät nur in der Lage ist, Ströme bis zu einer bestimmten Frequenz innerhalb der vorgegebenen Genauigkeit zu erfassen. Da die Frequenzen der Ströme, welche durch das Schaltnetzteil des Computers hervorgerufen werden, jedoch weitgehend unbekannt sind, gilt es, diese zu erfassen, um eine Einordnung der Messwerte bezüglich ihrer Aussagekraft zu gewährleisten.

Hohe Einschaltströme: Bei der Messung des Computerstromverbrauchs muss die Spannungsversorgung zunächst getrennt und anschließend mit einem Messgerät zwischengeschaltet wieder eingeschaltet werden. Dies kann zu Anfangsströmen führen, die den eingestellten Messbereich des Geräts überschreiten. Der Grund dafür sind die hohen Ströme, die im Moment des Einschaltens auftreten, wenn elektronische Bauteile wie Kondensatoren sich aufladen. Diese Ströme sind typischerweise höher als die Ströme, die im regulären Betrieb fließen. Dies ist besonders wichtig zu beachten, um das Messgerät vor Überlastung zu schützen und einen möglichst geeigneten Messbereich einstellen zu können.
Hierzu wird der Messaufbau so angepasst, dass die hohen Einschaltströme keinen Einfluss auf das Messgerät haben.

3 Methodik

3.1 Beschreibung der verwendeten Software

3.1.1 C++

In dieser Bachelorarbeit wird C++ als die bevorzugte Programmiersprache für das Auslesen der Leistungsindikatoren unter Windows verwendet. Die Entscheidung für C++ basiert auf einer Reihe von Faktoren, die die Anforderungen an die Arbeit und die technischen Möglichkeiten berücksichtigen.
Erstens ermöglicht C++ einen direkten Zugriff auf die Windows-API (Application Programming Interface) sowie Systemressourcen. Dies ist entscheidend, um Leistungsindikatoren wie die CPU-Auslastung auszulesen und um Informationen zu den einzelnen Prozessen zu erhalten.
Zweitens bietet C++ eine hohe Leistung und Effizienz, die für die Echtzeitüberwachung und -messung von Leistungsindikatoren erforderlich ist. Durch die Möglichkeit der hardwarenahen Programmierung können hiermit Programme mit sehr geringer Ressourcennutzung geschrieben werden. Dies ist insbesondere für die Entwicklung der GUI nützlich, da dieses Programm dauerhaft im Hintergrund läuft und keine hohe CPU-Auslastung erzeugen darf, um den Computer hinsichtlich seiner Performance nicht zu beeinflussen.
Darüber hinaus bietet C++ die Möglichkeit des relativ einfachen Erstellens von grafischen Anwendungen. Die Verwendung von beispielsweise C würde sich hierfür weniger gut eignen. Die Möglichkeit, grafische Anwendungen zu erstellen und gleichzeitig die für diese Arbeit notwendigen APIs zu nutzen, macht C++ somit zu einer geeigneten Wahl für die Nutzung in dieser Arbeit.

3.1.2 Software für Lasterzeugung

Verwendet wurden die beiden Tools **CPU Expert** sowie **HeavyLoad**. Diese Programme wurden für Belastungstests der CPU entwickelt und erzeugen durch Berechnungen eine Last auf der CPU. Dabei kann die CPU-Auslastung, die das Programm erzeugt, gesteuert werden.

HeavyLoad: Hierbei kann die CPU-Auslastung, die durch das Programm erzeugt wird, in $\frac{1}{\text{Anzahl logischer CPUs}}$-Schritten gesteuert werden. Für jede Erhöhung der CPU-Auslastung wird ein neuer Thread geöffnet. Jeder Thread läuft auf einem Kern und belastet diesen vollständig. Mit diesem Programm ist es also lediglich möglich, einzelne Kerne der CPU voll auszulasten und nicht nur teilweise. Für einen Rechner, der mit 8 logischen Kernen arbeitet, lässt sich die CPU-Auslastung folglich in 12,5-Prozent-Schritten einstellen.

CPU Expert: Hierbei handelt es sich um ein Online-Tool, das im Gegensatz zu HeavyLoad keinen Download erfordert. Ebenso wie das Programm HeavyLoad kann hierbei die Last variabel eingestellt werden. Im Gegensatz zu HeavyLoad erfolgt die Auslastung der CPU jedoch aufgeteilt auf alle Kerne. Dies ist vorteilhaft, da somit auch der Stromverbrauch der CPU in Teillast der einzelnen Kerne erfasst werden kann.

3.1.3 Software Datenauswertung

In dieser Arbeit wird MATLAB aus mehreren Gründen für die Bearbeitung von Messwerten und die Erstellung von Regressionsgeraden verwendet.
MATLAB bietet eine umfangreiche Sammlung vorintegrierter Funktionen, die das Handling und die Analyse großer Datensätze erleichtern. Diese Funktionen umfassen Werkzeuge zur statistischen Analyse sowie zur Bearbeitung von Daten.
Ebenfalls zeichnet sich MATLAB durch seine Fähigkeit zur Visualisierung von Daten aus. Mit nur wenigen Codezeilen lassen sich komplexe Graphen und Diagramme erstellen, die es ermöglichen, Trends und Muster in den Daten leicht zu erkennen und zu interpretieren. Diese Visualisierungen sind nicht nur für die Analyse selbst von Bedeutung, sondern auch wichtig, um die Ergebnisse visuell darstellen zu können.

3.2 Beschreibung der verwendeten Hardware

Messgerät
Für die Messungen wurde das Multimeter OWON 18B verwendet, das sich besonders aufgrund seiner Bluetooth-Funktionalität eignet. Das Multimeter kann Messwerte via Bluetooth an ein Endgerät wie ein Handy oder einen Laptop senden. Die Daten liegen im CSV-Format vor und enthalten einen Zeitstempel im Format H:MM:SS. Dies ist besonders wichtig für den späteren Abgleich der Messungen mit den internen Aufzeichnungen der CPU-Auslastung. Die Übertragung per Bluetooth erfolgt dreimal pro Sekunde, was ausreichend ist, da nicht mehr als ein Messwert pro Sekunde generiert werden kann. Die maximale Anzahl an Messungen,

die durch das Multimeter übertragen werden können, beträgt 10.000 Samples. Die maximale Messzeit, die mit dem Multimeter realisiert werden kann, errechnet sich folgendermaßen:

$$T_{\text{max}} = \frac{\text{maximale Samples}}{\text{Anzahl Samples pro Sekunde}} = \frac{10000}{3} = 3300 \text{ Sekunden}$$

Die Messung von Strömen kann sowohl im Bereich des Gleichstroms (DC) als auch im Bereich des Wechselstroms (AC) von 0 A bis 20 A erfolgen. Die Angaben zur Auflösung sowie zur Genauigkeit der Messung lauten wie folgt:

AC-Messung

Einheit	Messbereich	Auflösung	Genauigkeit (% Wiedergabe + Digits)
mA	60,00 mA / 600,0 mA	0,01 mA	$\pm(1,0\% + 3)$
A	20,00 A	0,01 A	$\pm(1,5\% + 3)$

DC-Messung

Einheit	Messbereich	Auflösung	Genauigkeit (% Wiedergabe + Digits)
mA	60,00 mA / 600,0 mA	0,01 mA	$\pm(0,8\% + 2)$
A	20,00 A	0,01 A	$\pm(1,2\% + 3)$

Nach Angaben des Herstellers garantiert das Multimeter ein Echt-Effektivwertmessverfahren (True RMS) auch bei nicht sinusförmigen Signalverläufen im Frequenzbereich von 40 bis 1000 Hz.
Das vollständige Datenblatt des Messgeräts ist im Anhang auf CD zu finden.

Computer
Die Messung, die am Desktop für die Betriebssysteme Linux sowie Windows durchgeführt wird, erfolgt hierbei auf einem Dell-Rechner, der mit einem Intel Core i5-Prozessor betrieben wird. Das Netzteil hat eine Leistung von 180 Watt.

Laptop
Der für die Messungen genutzte Laptop ist ein Dell Inspiron, welcher unter Windows 10 läuft. Hierbei besitzt dieser eine Intel Core i7 CPU der 10. Generation. Der Laptop ist in der Lage, auch ohne Akku zu laufen, sofern ein ausreichend starkes Netzteil angeschlossen ist. Hierbei wird ein Netzteil mit einer Leistungsaufnahme von maximal 65 W eingesetzt, welches den Laptop mit einer Spannung von 19,5 V sowie einem maximalen Strom von 3,3 Ampere versorgt.

3.3 Erläuterung der Methodik zur Datenerfassung

3.3.1 Messung der CPU-Auslastung

Die Erfassung der CPU-Auslastung des Rechners unterscheidet sich zwischen den Betriebssystemen Windows und Linux wesentlich. Hierbei wird nun genauer auf die Methoden der Erfassung der CPU-Auslastung sowie anderer Werte unter den einzelnen Betriebssystemen eingegangen.

Auslesen der Daten unter Windows

In Abschnitt 2.2.2 wurde bereits dargelegt, dass es hauptsächlich zwei etablierte Methoden gibt, um die CPU-Auslastung unter dem Windows-Betriebssystem zu messen. Beide Ansätze bieten grundsätzlich denselben Funktionsumfang, der für die in dieser Studie beabsichtigten Messungen erforderlich ist. Jedoch offenbaren sich, insbesondere bei hoher CPU-Belastung, signifikante Unterschiede in der Performance der Methoden, die während der Messungen auffällig wurden.
Die erste Methode, die Nutzung von typeperf, zeigt bei einer CPU-Auslastung nahe 100 Prozent erhebliche Schwächen. Diese äußern sich vor allem in unzuverlässigen Messungen und großen Lücken in der Aufzeichnung der CPU-Daten. Dieses Phänomen lässt sich wahrscheinlich auf eine unzureichende Zuweisung von Rechenzeit für das Programm zurückführen. Unter solchen Bedingungen kann das Tool nicht regelmäßig genug auf die notwendigen Systemressourcen zugreifen, um eine lückenlose Datenerfassung zu gewährleisten.
Dem gegenüber steht die Messung der CPU-Auslastung mittels der Windows API in C++. Diese Methode erweist sich auch unter Bedingungen hoher CPU-Belastung als deutlich robuster. Zwar treten auch hier gelegentlich Lücken in der Datenspeicherung auf, diese sind jedoch wesentlich seltener und kleiner. Das bedeutet, dass die durch die Windows API gesammelten Daten trotz vereinzelter Aussetzer eine hohe Verlässlichkeit bieten und in der späteren Auswertung der Ergebnisse effektiv berücksichtigt werden können.
Somit wird für diese Arbeit das Auslesen der Daten mittels der Windows API angewandt.

Das Sammeln der Zählerdaten zur CPU-Auslastung erfolgt durch die Verwendung der Performance Data Helper (PDH)-Funktionen. Diese Funktionen sind Teil der PDH-API, die speziell dafür entwickelt wurde, Leistungsdaten von Windows-Systemen zu sammeln und zu speichern. Um diese Funktionen nutzen zu können, muss die PDH-Bibliothek, ‘pdh.lib‘, in das Projekt eingebunden werden. Der Prozess des Abrufens der Leistungsdaten umfasst mehrere Schritte, die jeweils durch spezifische PDH-Funktionen unterstützt werden:

- **Erstellen einer Abfrage**
  Die Funktion `PdhOpenQuery` wird verwendet, um eine neue Abfrage zu initialisieren. Diese dient als Container, in dem eine oder mehrere Leistungsindikatoren überwacht werden können. Die Funktion bereitet die Datenstruktur vor, die benötigt wird, um Leistungsdaten zu sammeln und zu verarbeiten. [19]

- **Hinzufügen der Leistungsindikatoren zur Abfrage**
  Nach dem Erstellen einer Abfrage ermöglicht die Funktion `PdhAddCounter` das Hinzufügen spezifischer Leistungsindikatoren. Jeder Indikator entspricht einem spezifischen Aspekt der Systemleistung, wie z.B. CPU-Auslastung, und wird durch eine eindeutige Pfadangabe identifiziert. Dieser Schritt ist notwendig, um die relevanten Daten für die Analyse auszuwählen. [15]

- **Erfassen der Leistungsdaten**
  Mit der Funktion `PdhCollectQueryData` werden die aktuellen Werte der hinzugefügten Indikatoren gesammelt. Diese Funktion aktualisiert alle Zähler in der Abfrage mit den neuesten verfügbaren Daten, was in einem regelmäßigen Intervall durchgeführt wird, um das kontinuierliche Auslesen der Zählerdaten zu ermöglichen. [17]

- **Ausgeben der Leistungsdaten**
  Die Funktion `PdhGetFormattedCounterValue` wird genutzt, um die gesammelten Daten in einem lesbaren Format auszugeben. Sie konvertiert die rohen Zählerdaten in ein Prozentformat. [18]

- **Schließen der Abfrage**
  Schließlich wird die Funktion `PdhCloseQuery` aufgerufen, um die Abfrage und alle zugehörigen Ressourcen zu schließen. Dies ist ein wichtiger Schritt, um Speicherlecks zu vermeiden und sicherzustellen, dass alle verwendeten Ressourcen ordnungsgemäß freigegeben werden, nachdem die Datenerfassung abgeschlossen ist. [16]

Die Überwachung der CPU-Auslastung erfordert die Sammlung von Ratenwerten. Bei der Verwendung von Ratenwerten ist es notwendig, zwei Datenstichproben zu sammeln, um mittels der Funktion `PdhGetFormattedCounterValue` Werte der CPU-Auslastung in Prozent zu erhalten. Die Implementierung dieser Abfrage lautet somit wie folgt und ist durch Microsoft vorgegeben:

- **Sammeln der Datenstichproben**
  Für die korrekte Erfassung der Ratenwerte muss zwischen den Aufrufen der Funktion

`PdhCollectQueryData` ein bestimmter Zeitabstand eingehalten werden. Microsoft empfiehlt hierfür eine Mindestzeit von einer Sekunde zwischen den Aufrufen, um zuverlässige Messwerte zu gewährleisten. Dieser zeitliche Abstand ist nötig, um dem Counter zu ermöglichen, die CPU-Auslastung korrekt zu berechnen.

- **Implementierung des Zeitabstands**
  Um den erforderlichen Zeitabstand zu gewährleisten, wird die Funktion `sleep()` mit einem Parameter von 1000 Millisekunden (ms) verwendet. Dies pausiert die Ausführung des aktuellen Threads, auf dem das Programm läuft, für mindestens eine Sekunde, wodurch sichergestellt wird, dass die Zeitspanne zwischen den einzelnen Stichproben eingehalten wird.

- **Auslesen der CPU-Auslastung**
  Nachdem die zweite Stichprobe gesammelt wurde, kann durch den Aufruf der `PdhGetFormattedCounterValue`-Funktion die CPU-Auslastung in Prozent ausgelesen werden. Dieser Wert repräsentiert die aktuelle Auslastung des Prozessors über das Intervall zwischen den beiden Stichproben.

- **Kontinuierliche Überwachung**
  Um die CPU-Auslastung über den Zeitraum der Messung zu überwachen, wird der gesamte Prozess des Sammelns und Auslesens der Daten in einer `for`-Schleife wiederholt. Dies ermöglicht eine fortlaufende Datensammlung, welche es erlaubt, über den gesamten Zeitraum der Messung Daten abzurufen.

[12]

Zur präzisen Zuordnung der CPU-Auslastungsdaten im späteren Vergleich mit den Stromverbrauchsmessungen ist es notwendig, jeden Messwert mit einem Zeitstempel zu versehen. Dies gewährleistet, dass die zeitliche Abfolge der Ereignisse klar nachvollziehbar und korrekt zugeordnet werden kann.

- **Erfassung des Zeitstempels:** Unmittelbar nach jeder Messung der CPU-Auslastung wird die aktuelle Systemzeit mittels der Funktionen `time()` und `localtime()` erfasst. Die Funktion `time()` liefert die Anzahl der Sekunden seit dem 1. Januar 1970 (Epoch), und `localtime()` konvertiert diesen Wert in die lokale Zeit unter Berücksichtigung der Zeitzone.

- **Speicherung der Daten:** Die erfassten Werte der CPU-Auslastung zusammen mit den entsprechenden Zeitstempeln werden anschließend im CSV-Format in eine Textdatei geschrieben. Dieses Format ermöglicht eine einfache Handhabung und Weiterverarbeitung der Daten mittels MATLAB.

Auslesen der Daten unter Linux

Da, wie in Abschnitt 2.2.3 beschrieben, es unter Linux nicht direkt möglich ist, über das /proc-Dateisystem die CPU-Auslastung in Prozent auszulesen, muss die CPU-Auslastung anhand der möglichen CPU-Zustände berechnet werden. Hierbei tragen alle Zustände der CPU, welche nicht Idle sind, zu einer CPU-Auslastung bei. Somit wird die CPU-Auslastung ermittelt, indem man die Zeiten, welche die CPU im Idle verbracht hat, mittels dem `/proc/stat`-Dateisystem extrahiert und diese in Relation zur Gesamtzeit setzt. Die folgende Formel wird verwendet, um die Auslastung in Prozent zu berechnen:

$$\text{CPU-Auslastung (\%)} = \left(1 - \frac{\Delta\text{Idle-Zeit}}{\Delta\text{Gesamtzeit}}\right) \cdot 100 \tag{3.1}$$

Die Auslese der Zeit, die das System im Zustand Idle verbringt, erfolgt hierbei gleich wie bei der Messung unter Microsoft zweimal. Durch Subtraktion der ersten Messung von der zweiten erhalten wir die Differenz der Zeiten für die beiden Stichproben. Besonders wichtig ist hier die Differenz der Idle-Zeit und der Gesamtzeit, die sich aus der Summe der Zeiten aller Prozessorzustände ergibt. Der Zeitabstand, der zwischen den Messungen gewählt wird, beträgt hierbei ebenfalls wie unter Windows eine Sekunde.
Da die Werte für die CPU-Auslastung sich im Gegensatz zum Windows-Betriebssystem in einem frei zugänglichen Dateisystem befinden, können diese einfach über Standard C++-Methoden ausgelesen werden. Der Programmablauf zum Berechnen sowie Auslesen der CPU-Werte lautet wie folgt:

- **Auslesen aller CPU-Zeiten**
  Das Auslesen der einzelnen CPU-Zeiten erfolgt hierbei durch das Öffnen des Dateisystems mittels der Funktion `ifstream()`, wobei das geöffnete Dateisystem mehrere Zeilen enthält, wobei die erste Zeile die Informationen für die gesamte CPU-Auslastung enthält. Über die Funktion `getline()` wird diese Zeile der Datei ausgelesen, da die ausgelesene Zeile mehrere Informationen enthält, welche separiert werden müssen, wird die ausgelesene Zeile über ein `istringstream` Objekt in seine einzelnen Bestandteile zerlegt. Die ausgelesenen CPU-Zeiten werden nun für den späteren Vergleich mit der

zweiten Stichprobe gespeichert. Anschließend wird das Dateisystem über die Funktion `close()` geschlossen.

- **Warten für nächste Abfrage**
  Nachdem die erste Datenstichprobe gesammelt wurde, wird der Thread, auf dem das Programm läuft, für eine Sekunde pausiert. Dies geschieht mittels der Funktion `sleep_for()` und dient dazu, einen zeitlichen Abstand der Zeiten für die CPU-Auslastung zu gewinnen, um die nächste Datenstichprobe sammeln zu können.

- **Erneutes Auslesen der CPU-Zeiten**
  Nachdem, wie oben beschrieben, die erste Datenstichprobe gesammelt wurde und der Thread für eine Sekunde blockiert wurde, kann nun mit dem gleichen Programmablauf wie oben die zweite Stichprobe gesammelt und gespeichert werden.

- **Berechnen der CPU-Auslastung in Prozent**
  Die beiden gesammelten Stichproben enthalten nun die Zeitwerte, welche die CPU in den unterschiedlichen Modi verbracht hat. Mittels der Formel 3.1 kann nun die CPU-Auslastung in Prozent berechnet werden. Hierbei wird für die Ermittlung der relativen Gesamtzeit zwischen den einzelnen Datenstichproben die Summe aus allen Modi der CPU gebildet und die einzelnen Gesamtzeiten im Anschluss subtrahiert.

Im Anschluss wird die errechnete CPU-Auslastung gleich wie bei der Ermittlung der CPU-Auslastung unter Windows im CSV Format in eine Textdatei gespeichert.

3.3.2 Messung des Stromverbrauchs

Zur Messung des Stromverbrauchs wird das in Abschnitt 3.2 beschriebene Messgerät verwendet. Hierbei werden zwei jeweils für die Situation am besten geeignete Messverfahren angewandt, welche im Folgenden genauer beschrieben werden.

Messung Tower PC

Da das Netzteil des Tower PCs mehrere Ausgänge für die Spannungsversorgung der einzelnen Komponenten des Computers besitzt, ist hierbei eine Messung auf der Gleichspannungsseite des Netzteils nicht optimal zu bewerkstelligen. Hierbei bietet sich die Messung des Stromverbrauchs auf der Wechselspannungsseite als bessere Alternative an. Da die Messung folglich bei einer Betriebsspannung von 230V AC vollzogen wird, gilt es hierfür den Schutz vor elektrischem Schlag zu beachten und diesen stets zu gewährleisten. Um diesen Schutz sicherzustellen, wurde

zum Zweck der Messung eine Messapparatur konstruiert, welche eine Berührung mit unter Spannung stehenden Teilen ausschließt, jedoch die Messung weiterhin zu gewährleisten. Diese ist im Abschnitt 9 in Abbildung 9.2 sowie 9.3 dargestellt. Ebenfalls sind ausschließlich isolierte Messleitungen zu verwenden, um ein versehentliches Berühren durch menschliches oder technisches Versagen auszuschließen. Ebenfalls ermöglicht die Messapparatur das Hinzufügen sowie Entfernen einer Brücke, welches das Messgerät anfangs überbrückt, um kurzzeitige Einschaltströme des Computers nicht über das Messgerät laufen zu lassen. Nachteil dieser Methode ist jedoch, dass Verluste, welche im Netzteil in Form von Wärme entstehen, mit in die Messung mit einfließen und diese somit beeinträchtigen können.

Messung Laptop

Die Spannungsversorgung eines Laptops kann, im Gegensatz zu einem Desktop-PC, einfach auf der Gleichspannungsseite getrennt werden. Daher bietet sich die Messung des Stromverbrauchs auf der Gleichspannungsseite des Netzteils an. Diese Methode hat mehrere Vorteile:

- **Sicherheit:** Das Netzteil wird auf der Sekundärseite mit einer Spannung von 19,4 Volt betrieben, was das Risiko eines elektrischen Schlages ausschließt. Bei der Arbeit mit elektrischen Komponenten ist es entscheidend, die Gefahren für Menschen und Material so gering wie möglich zu halten. Die Messung bei einer für den menschlichen Körper ungefährlichen Spannung ist daher vorzuziehen.
- **Bessere Messqualität:** Auf der Gleichspannungsseite des Netzteils treten keine stark pulsierenden Ströme auf, wie sie auf der Primärseite üblich sind. Dies verbessert die Genauigkeit der Messungen. Zudem sind Netzteile immer mit Verlusten verbunden, die zusätzlichen Strom verbrauchen, welcher die Messung verfälschen könnte. Bei der Messung auf der Gleichspannungsseite wird dieser Verluststrom nicht erfasst, was ebenfalls die Qualität der Messergebnisse verbessert.

Es gibt jedoch auch einen Nachteil bei dieser Methode: Auf der Gleichspannungsseite sind höhere Ströme zu erwarten als auf der Netzseitigen des Netzteils. Dies hat zur Folge, dass der Messbereich des Multimeters auf einen Bereich eingestellt werden muss, der eine geringere Auflösung hat. Laut Typenschild können Ströme bis zu 3,3 Ampere fließen, was die Einstellung des Multimeters auf den Ampere-Bereich mit einer Auflösung von 10 mA erfordert.

3.3.3 Synchronisation der Systemzeiten

Wie im vorherigen Abschnitt 2.5 beschrieben, besteht das Problem unterschiedlicher Systemzeiten zwischen dem Computer sowie dem Messgerät. Zur Lösung dieses Problems wurde ein C++-Programm geschrieben, welches auf Mausereignisse reagiert und den Zeitpunkt jedes Ereignisses im Format `HH:MM:SS` erfasst.
Zur Überwachung der Mausereignisse wurde die Windows-API-Funktion `SetWindowsHookEx` verwendet. Diese Funktion ermöglicht es, systemweite Hooks zu setzen, die auf bestimmte Ereignisse wie Mausaktionen reagieren.
`SetWindowsHookEx` wird mit folgenden Parametern aufgerufen:

- `WH_MOUSE_LL` - Der Hook-Typ, der für Low-Level-Mausereignisse steht.
- `MouseProc` - Ein Zeiger auf die Hook-Prozedur, die bei jedem Mausereignis aufgerufen wird.
- `NULL` - Ein Handle zum Modul, das die Callback-Funktion enthält. Für Low-Level-Hooks kann dieser Parameter `NULL` sein.
- `0` - Die Kennung des Threads, der den Hook empfängt. Ein Wert von `0` bedeutet, dass der Hook für alle Threads im System gilt.

[11]
Die Hook-Prozedur `MouseProc` wird jedes Mal aufgerufen, wenn ein Klick der linken Maustaste erkannt wird. Wird das Ereignis registriert, so wird die aktuelle Systemzeit in eine Textdatei abgespeichert.
Zur genauen Synchronisierung der Zeitstempel von Mausereignissen im Vergleich zur Systemzeit des Multimeters wird eine direkte Messung an der Maustaste implementiert. Diese Methode ist notwendig, da die Datenübertragung der Maus über ein Protokoll erfolgt, dass die unmittelbare Erkennung der Betätigung einer Taste durch das Multimeter nicht zulässt. Die Messung wird direkt an der linken Maustaste durchgeführt. Es wird das elektrische Verhalten der Taste unter folgenden Bedingungen untersucht:

- Im unbetätigten Zustand liegt an der Maustaste eine Spannung von 1,4 Volt an.
- Wird die Maustaste gedrückt, fällt das Potential auf 0 Volt.

Diese Änderung des Spannungsniveaus von 1,4 Volt auf 0 Volt, die sogenannte Flanke, wird vom Multimeter erfasst und kann zur präzisen Zeitmessung des Ereignisses verwendet werden.

Aus den Zeiten, welche in beiden Systemen beim Eintreten des Ereignisses gemessen werden, kann nun über Subtraktion der Zeiten die relative Zeitverschiebung der einzelnen Systeme ermittelt werden.

3.4 Erstellung des Regressionsmodells

Vor der Bildung eines Modells welches durch eine Regressionsgerade beschrieben wird, gilt es sich einen Überblick über die vorhandenen Daten zu verschaffen. Dies dient erstmal dazu die Messwerte auf Plausiblität zu prüfen, desweiteren kann anhand einer ersten Veranschaulichung der Daten bereits abgewogen werden welches Model für die Regression zugrunde gelegt werden kann.
Hierzu werden die folgenden Verfahren zum Analysieren der Messungen angewandt:

- **Darstellung der Daten:** Ein erster wichtiger Schritt ist die grafische Darstellung der gesammelten Daten in einem Streudiagramm oder auch Punktwolke genannt. Diese Darstellung erfolgt auf einer x-y-Ebene, wobei typischerweise die unabhängige Variable auf der x-Achse (in diesem Fall die CPU- Auslastung) und die abhängige Variable (gemessener Strom) auf der y-Achse aufgetragen wird. Dies ermöglicht eine visuelle Analyse der Daten, um erste Anhaltspunkte für mögliche Zusammenhänge zu erhalten.
- **Analyse des Streudiagramms:** Das Muster, dass durch das Streudiagramm gezeigt wird, kann erste Hinweise auf die Art des Zusammenhangs zwischen den Variablen geben. Erscheinen die Datenpunkte beispielsweise als schräg liegende Ellipse, kann dies auf einen linearen Zusammenhang hindeuten, was die Anwendung einer linearen Regression nahelegt. Liegen die Daten jedoch in Form einer Parabel vor, kann hierbei auf einen nicht linearen Zusammenhang geschlossen werden.

[4]
Ebenfalls ist die Anwendung von Boxplots ein nützliches Werkzeug, wenn es darum geht, einen ersten Eindruck von der Verteilung der Daten zu gewinnen sowie mögliche Ausreißer zu identifizieren. Diese grafische Methode ermöglicht es die Tendenzen und Streuungen der Daten grafisch zu erfassen und liefert Aufschluss über die wichtigen Kenndaten wie Minumum, Maximum, sowie arithmetischer Mittelwert, Standartabweichung und Median.

- **Darstellung der Daten:** In einem Boxplot werden die unabhängigen und abhängigen Variablen separat dargestellt. Jeder Boxplot visualisiert die zentrale Tendenz (Median), die Quartile (25% und 75% Perzentil) sowie die Spannweite der Daten, einschließlich der

sogenannten Whisker, die 1,5 mal den Quartilsabstand vom oberen bzw. unteren Quartil aus darstellen. Datenpunkte, die außerhalb dieser Whisker liegen, werden als Ausreißer markiert und separat angezeigt.

- **Analyse der Verteilung:** Boxplots ermöglichen eine schnelle Beurteilung der Symmetrie und Schiefe der Datenverteilungen. Eine symmetrische Verteilung wird im Boxplot durch einen zentralen Median und gleichmäßig verteilte Quartile angezeigt. Eine schiefe Verteilung ist erkennbar, wenn der Median nicht zentral liegt und die Längen der Whisker unterschiedlich sind. Ebenfalls können die Ausreißer in Ihrer Form sowie Anzahl hier sehr gut identifiziert werden.
- **Vergleich zwischen Gruppen:** Da es mehrere Messungen gibt, können diese nebeneinander Grafisch dargestellt werden, was einen Vergleich der Daten erlaubt um zu Identifizieren ob die Verteilung der Daten im Vergleich zu den anderen Messungen gleichmäßig ist oder ob die Messungen in ihrer Verteilung stark voneinander Abweichen.

[4]

Nachdem sich ein Überblick über die Daten verschafft wurde, kann anhand der gewonnenen Informationen eine erste Idee entwickelt werden, welche Art der Regression anzuwenden ist und inwieweit sich zuverlässig ein Modell hieraus erstellen lässt. Diese erste Annahme führt zum Erstellen der Regression, welche im Anschluss durch statistische Methoden auf die Güte überprüft werden muss.

3.4.1 Statistische Methoden zur Bewertung des Modells

Zur Überprüfung der Modellannahmen nach 2.3 kommen verschiedene statistische Verfahren zum Einsatz, welche die Güte des Modells bewerten.
Hierbei sind die Residuen entscheidend für die Diagnose des Modells, da sie Aufschluss darüber geben, wie gut das Modell die Daten anpasst. Wenn die Residuen systematische Muster aufweisen, deutet dies darauf hin, dass das Modell einige Aspekte der Datenstruktur nicht erfasst. Beispielsweise können Muster in den Residuen auf Probleme wie Nichtlinearität, Heteroskedastizität oder fehlende Variablen hinweisen. Zur Überprüfung der Residuen kommen in dieser Arbeit grafische Verfahren zum Einsatz, welche es erlauben das durch die Regression erstellte Modell zu bewerten.

Überprüfung auf Homoskedastizität

Zur Überprüfung auf Homoskedastizität nach 2.3 werden die standardisierten Residuen gegen die durch die Funktion vorhergesagten Werte (Regressionswerte) geplottet. Diese grafische Methode dient dazu, visuell zu beurteilen, ob die Varianz der Residuen konstant ist. Im Idealfall sollten die Punkte zufällig um die horizontale Achse (Nulllinie) verteilt sein, ohne erkennbare Muster oder systematische Strukturen.

- **Grafische Darstellung:** Auf der x-Achse werden die vorhergesagten Werte ($\hat{y}_i$) und auf der y-Achse die standardisierten Residuen ($e_i/\hat{\sigma}_i$) abgebildet.
- **Erkennen von Mustern:** Wenn die Residuen in etwa gleicher Breite um die x-Achse verteilt sind, deutet dies auf Homoskedastizität hin. Zeigen die Residuen jedoch ein erkennbares Muster, wie z.B. einen Trichter oder eine sich öffnende oder schließende Form, so liegt eine Heteroskedastizität vor. Dies bedeutet, dass die Varianz der Fehler mit den vorhergesagten Werten variiert.

[4]

Überprüfung auf Unkorreliertheit

Um die Annahme der Unkorreliertheit nach 2.3 zu überprüfen, kann man die Residuen gegen eine unabhängige Variable grafisch darstellen.

- **Grafische Überprüfung:** Die Residuen werden auf der y-Achse gegen eine unabhänige Variable auf der x-Achse geplotet. Wenn die Residuen um die Nulllinie ohne erkennbares Muster oder Trend streuen, deutet dies auf Unkorreliertheit hin. Dies bedeutet, dass die Residuen zufällig verteilt sind und keine interne Struktur aufweisen, die auf eine Verletzung der Modellannahmen hinweist.
- **Suche nach Mustern:** Sollten die Residuen Trends oder zyklische Muster zeigen, kann dies auf eine Korrelation der Fehlervariablen hinweisen. Dies kann Hinweise auf Fehlende Variablen geben, welche nicht in dem Modell beachtet wurden.

[4]

Überprüfung der Normalverteilungsannahme

Zur Überprüfung der Normalverteilung der Residuen nach Abschnitt 2.3 werden hierbei die Quantile der Residuen gegen die Quantile der Normalverteilung aufgetragen.

- **Quantil-Quantil-Diagramm (Q-Q-Diagramm):** Zur visuellen Überprüfung der Normalverteilungsannahme wird hierbei ein Q-Q-Diagramm verwendet. In diesem Diagramm werden die Quantile der Residuen gegen die Quantile einer Standardnormalverteilung aufgetragen.
- **Darstellung im Diagramm:** Auf der x-Achse des Q-Q-Diagramms werden die Quantile der Standardnormalverteilung angezeigt, während auf der y-Achse die Quantile der Residuen abgebildet werden. Wenn die Residuen normalverteilt sind, sollten sich die Datenpunkte entlang der Winkelhalbierenden (einer Geraden, die durch den Ursprung geht und einen Winkel von 45 Grad bildet) anordnen. Abweichungen von dieser Linie, wie eine Krümmung oder das Ausfransen an den Enden, deuten auf eine Abweichung von der Normalverteilung hin.

[4]

Bei der Bewertung des Modells ist hierbei zu beachten, dass diese Bedingungen in der Realität nicht ideal eingehalten werden können. Es gilt hierbei zu bewerten,inwieweit die in Abschnitt 2.3 gestellten Bedingungen eingehalten werden und ob diese zu mindestens näherungsweise erfüllt sind. Je geringer die oben aufgeführten Bedingungen erfüllt sind, desto ungenauer ist das aufgestellte Modell.

4 Durchführung der Messungen

4.1 Messaufbau

Wie in Abschnitt 3.3.2 beschrieben, werden für die Messungen am Desktop-PC sowie am Laptop zwei verschiedene Messaufbauten benötigt.

Desktop PC

Für die Messung am Desktop-PC wird der Strom vor dem Netzteil gemessen, hierbei wird das Multimeter zwischen der Steckdose und dem Netzteil in Reihe geschaltet. Um eine Überlastung des Multimeters durch hohe Einschaltströme des Netzteils zu vermeiden, wird das Messgerät anfangs durch eine Brücke überbrückt, sodass die Einschaltströme nicht über das Messgerät fließen. Diese Brücke kann nach dem Einschalten des Computers entfernt werden. Dadurch wird eine Messung im Milliampere-Bereich ermöglicht, was eine maximale Auflösung der Messergebnisse gewährleistet.

Laptop

Die Messung für den Laptop wird, wie in Abschnitt 3.3.2 beschrieben, auf der Gleichspannungsseite des Netzteils durchgeführt. Hierfür wird das Messgerät in den Amperemodus geschaltet und in Reihe mit dem Laptop verbunden. Da hierbei eine Spannung von 19,4 Volt anliegt, sind keine speziellen Maßnahmen bezüglich des Berührungsschutzes erforderlich. Um Messverfälschungen zu vermeiden, wird der Akku des Laptops im Anschluss für die Dauer der Messung temporär abgeklemmt.
Ebenfalls wird der Bildschirm des Geräts ausgeschaltet, um eine Verfälschung der Messung durch Aktivität des Bildschirms zu vermeiden.

4.2 Ablauf der Messung

Um eine qualitativ möglichst hochwertige Messung durchführen zu können, die es erlaubt, das spätere Regressionsmodell zu bilden sowie zu bewerten, wird hierbei auf die einzelnen Schritte

eingegangen, die zur Vorbereitung der Messung sowie deren Durchführung notwendig sind.

Bestimmung des Messbereichs

Da das Messgerät wie in Abschnitt 3.2 beschrieben eine Strommessung in zwei Messbereichen ermöglicht, gilt es vor Beginn der Messung den bestmöglichen Messbereich zu ermitteln. Dies ist der Messbereich, der die maximal mögliche Auflösung liefert, sodass eine möglichst genaue Messung vollzogen werden kann. Hierzu wird das Multimeter in den Messbereich Ampere geschaltet, da dies den größtmöglichen Messbereich abdeckt und somit Schäden durch zu hohe Ströme im Multimeter vermieden werden. Anschließend wird der maximale Strom, der bei der geplanten Messung am PC bzw. am Laptop zu erwarten ist, ermittelt. Die maximal zu erwartende Last ist hierbei die Last, die bei höchster CPU-Auslastung, also 100 Prozent, vorliegt. Diese Last wird durch das Einschalten der für die spätere Lasterzeugung herangezogenen Software auf höchste Stufe erzeugt. Die maximal auftretenden Ströme für den Laptop sowie den PC lauten hierbei wie folgt:

PC	0,37 Ampere
Laptop	2,2 Ampere

Tabelle 4.1: Maximale Stromaufnahme der Geräte

Da der Messbereich für das Multimeter in der Einstellung Milliampere Messungen bis 600 mA ermöglicht, kann für die Messung des Stroms am PC dieser Messbereich gewählt werden. Für die Messung am Laptop muss hingegen der Messbereich Ampere gewählt werden, welcher eine geringere Auflösung der Messung liefert.
Hierbei nicht berücksichtigt sind Einschaltströme, die kurzzeitig nach dem Einschalten des Computers entstehen, diese sind hierbei höher als die Ströme, die im normalen Betrieb fließen. Der Einschaltstrom des Computers wurde hierbei auf 1,8 Ampere ermittelt, dieser Strom fließt hierbei nur für eine sehr kurze Zeit und ist für den eigentlichen Verlauf der Messung nicht relevant. Dennoch übersteigt dieser Strom den Messbereich von 600 mA signifikant und könnte somit zum Ausfall des Messgeräts führen. Das Umgehen dieser hohen Ströme im Einschaltmoment wird hierbei im Folgenden nochmal behandelt.

Synchronisation der Systemzeiten

Wie im Abschnitt 3.3.3 beschrieben, müssen vor der eigentlichen Messung die Systemzeiten des Messgeräts sowie des Computers abgeglichen werden. Hierbei wird das dort beschriebene Verfahren zum Abgleich vor jeder Messung angewandt. Dieses Verfahren ist hier beispielhaft

für eine Messung in Abbildung 4.1 dargestellt, wobei die zeitlichen Messwerte für eine bessere Verarbeitung mittels MATLAB in Sekunden des Tages abgebildet werden.

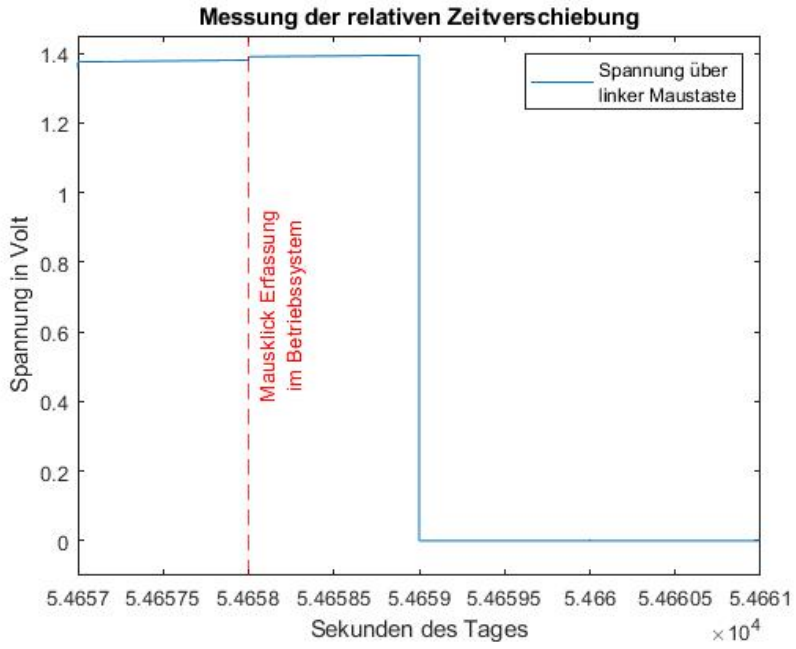

Abbildung 4.1: Darstellung der zeitlichen Verschiebung zwischen Messgerät und Systemzeit des Computers

Die in der Abbildung 4.1 dargestellte zeitliche Verschiebung von einer Sekunde kann nun als einfacher Offset in der späteren Auswertung berücksichtigt werden.
Hierbei ist der Abgleich der Systemzeiten vor jeder Messung durchzuführen. Dies ist notwendig, da die relative zeitliche Verschiebung der Zeiten über Stunden bzw. Tage hinweg um mehrere Sekunden schwanken kann.

Ermittlung der Frequenzen

Da die Messung an dem Desktop-PC vor dem Schaltnetzteil durchgeführt wird, sind hierbei, wie in Abschnitt 2.5 beschrieben, keine sinusförmigen Ströme zu erwarten. Da das verwendete Messgerät laut Herstellerangaben Ströme lediglich bis zu einer Frequenz von 1000 Hz mit der vom Hersteller angegebenen Genauigkeit darstellen kann, ist es notwendig, die Ströme mittels Fast Fourier Transformation (FFT) zu analysieren, um eine spätere Beurteilung der Messergebnisse zu ermöglichen.
Hierbei wurde mittels Oszilloskop eine FFT durchgeführt, die den Strom in die einzelnen

Frequenzen zerlegt und den jeweiligen Anteil der Frequenzen im Strom anzeigt.
Die Ergebnisse der FFT zeigen hierbei Ströme im zweistelligen Kilohertz-Bereich. Diese liegen somit höher als die Frequenzen, für welche der Hersteller eine Genauigkeit garantiert.
Es ist somit zu erwarten, dass bei der Messung an dem Desktop-PC Abweichungen innerhalb der Messung auftreten, die nicht in der vom Hersteller angegebenen Toleranz der Messungen liegen.

Durchführung der Messung
Da, wie oben beschrieben, die Einschaltströme des Computers die Ströme, welche bei der Messung fließen, übersteigen und dies zu einer Überlast im Multimeter führen kann, wird das Multimeter im Bereich des Hochfahrens des Computers anfangs überbrückt. Dies sorgt dafür, dass die Ströme, welche das Multimeter in dem eingestellten Messbereich überlasten könnten, nicht über das Multimeter, sondern über die Brücke fließen. Diese Brücke kann nach dem Hochfahren des Rechners entfernt werden, da die nun vorliegenden Ströme im Messbereich des Computers liegen. Ein solches Vorgehen ist bei der Messung für den Laptop hierbei nicht nötig, da der Messbereich aufgrund der höheren Ströme im DC-Bereich bereits so eingestellt ist, dass die Einschaltströme das Multimeter nicht überlasten.

Da nach dem direkten Einschalten des Computers viele Prozesse gestartet werden und die CPU hierdurch stark beansprucht wird, gilt es, nach dem Hochfahren einige Minuten zu warten, bis alle Prozesse nach dem Hochfahren abgeklungen sind und die CPU in ihrer Grundlast läuft.
Nachdem dies geschehen ist, wird das Programm zur Erzeugung der Last gestartet. Im Anschluss wird das Programm zur Messung der CPU-Auslastung, welches in Abschnitt 3.3.1 für Linux sowie Windows beschrieben ist, gestartet. Hierbei ist auf eine zeitliche Verzögerung der einzelnen Starts zu achten, da beim Programmstart eine kurzzeitig höhere CPU-Last eintritt als für die normale Messung vorgesehen.
Ebenfalls muss der Lüfter die Möglichkeit haben, anzulaufen, dies ist nötig, um die Messung in einem stationären Zustand durchzuführen.
Die Auslastung der CPU wird hierbei mit jeder Messung schrittweise erhöht, wobei die einstellbaren Abstände der Erhöhung der CPU-Auslastung hierbei durch die Programme zur Lasterzeugung reglementiert sind. Ebenfalls ist die CPU-Auslastung selbst in Zeiten, wo keine aktiv durch den Benutzer gesteuerten Programme laufen, nicht vollständig im Zustand `Idle`, was die Reglementierung der möglichen einstellbaren Zustände der CPU-Auslastung weiter erhöht.
Die Messungen werden hierbei für jeden einstellbaren Wert der CPU-Auslastung für 10 Mi-

nuten durchgeführt. Dies ist nötig, um später aussagekräftige Daten für die Erstellung der Regression zu gewinnen.

4.3 Datenaufbereitung und -analyse

Die Aufbereitung sowie Analyse der Messwerte erfolgt aus bereits beschriebenen Gründen mittels Matlab.

Datenaufbereitung

Um die Daten in einer späteren Analyse verwerten zu können, gilt es die Daten in mehreren Punkten aufzubereiten:

- **Mitteln der Strommessungen:** Da nach Abschnitt 3.3.1 das Auslesen der CPU-Auslastung nur einmal pro Sekunde erfolgen kann, das Messgerät jedoch drei Daten pro Sekunde überträgt, werden die Daten der Strommessung jeweils für eine Sekunde mittels des arithmetischen Mittelwerts so angepasst, dass nur ein Messwert pro Sekunde vorhanden ist. Das arithmetische Mittel ist hierbei dem Median vorzuziehen, da dieses, sofern keine großen Ausreißer zu erwarten sind, genauer ist.
- **Zeitlicher Abgleich:** Hierbei wird die vorher ermittelte zeitliche Verschiebung der Messwerte berücksichtigt, indem die ermittelte zeitliche Verschiebung als Offset berücksichtigt wird. Somit sind die Messwerte zeitlich abgeglichen, was es erlaubt, die Datenpaare unabhängig von der Zeit zu betrachten.
- **Bilden von Wertepaaren:** Da es in den Messungen bei hoher CPU-Auslastung trotz des Verwendens eines effizienten C++-Programms trotzdem zu kleinen Lücken innerhalb der CPU-Aufzeichnung kommen kann, werden hierzu nur die Messungen des Stroms berücksichtigt, welche zeitlich mit der Messung der CPU-Auslastung übereinstimmen. Dies ermöglicht es, jedem Messwert des Stroms einen eindeutigen Messwert der CPU-Auslastung zuzuordnen, was für eine Regression nötig ist.

Analyse der Daten

Um die Daten zu analysieren und die Regressionskoeffizienten zu bestimmen, wurden mehrere vorbereitende Schritte durchgeführt. Zu Beginn erfolgte nach Abschnitt 3.4 eine visuelle Inspektion der Daten, um ein grundlegendes Verständnis für die Verteilung und mögliche Ausreißer zu entwickeln.

Für die visuelle Darstellung der Daten wurden in MATLAB drei graphische Methoden verwendet. Die Funktion `boxplot()` wurde eingesetzt, um Boxplots zu erstellen, die eine schnelle Beurteilung der zentralen Tendenz und der Streuung der Daten ermöglichen. Diese Boxplots sind detailliert in Abschnitt 3.4 beschrieben. Zusätzlich wurde die Funktion `plot()` genutzt, um die Daten als Punktwolke darzustellen und so einen Überblick über die Messdaten zu gewinnen.
Zur Bewertung der Regressionsmodelle wurden neben der bereits beschriebenen Funktion `plot()` zur Visualisierung der Daten als Punktwolke auch die Funktion `qqplot()` eingesetzt. Die Funktion `qqplot()` in MATLAB stellt die Quantile der Residuen gegen die Quantile einer Normalverteilung dar.
Der Einsatz dieser Funktionen ermöglicht die nach 2.3 notwendige Bewertung der erstellten Regression
Für die Erstellung der Regression werden in MATLAB die Funktionen `polyfit` und `polyval` eingesetzt. Diese Funktionen ermöglichen es, Polynommodelle zu schätzen und auszuwerten.

- **polyfit**: Diese Funktion wird verwendet, um ein Polynom n-ten Grades zu schätzen, das am besten zu einer gegebenen Menge von Datenpunkten passt. Die Syntax der Funktion lautet `p = polyfit(x, y, n)`, wobei `x` und `y` die Datenvektoren darstellen und `n` den Grad des zu schätzenden Polynoms angibt. Der Rückgabewert `p` ist ein Vektor von Koeffizienten des Polynoms, beginnend mit dem Koeffizienten höchster Ordnung. [21]

- **polyval**: Nachdem das Polynom mit `polyfit` geschätzt wurde, kann `polyval` verwendet werden, um Werte des Polynoms zu berechnen. Die Funktion wird aufgerufen durch `y = polyval(p, x)`, wobei `p` der Vektor der Polynomkoeffizienten ist, der von `polyfit` zurückgegeben wurde und `x` die Punkte sind, an denen das Polynom ausgewertet werden soll. [22]

5 Ergebnisse

5.1 Darstellung der Messergebnisse

5.1.1 Desktop PC unter Windows

Wie in Abschnitt 3.4 beschrieben, wird sich als erstes ein Überblick über die Messdaten verschafft.

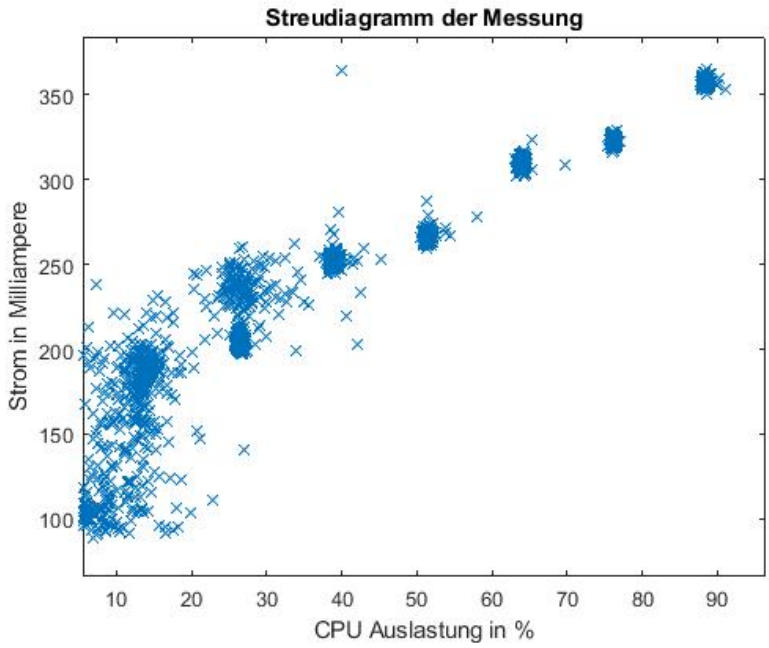

Abbildung 5.1: Streudiagramm der Messungen am PC unter Windows

In Abbildung 5.1 sind mehrere wichtige Punkte zu erkennen,

- **Verteilung der Messpunkte:** Betrachtet man die Messpunkte, so fällt auf, dass die Datenpunkte nicht alle Bereiche der CPU-Auslastung abbilden. Hierbei ist zu sehen, dass die Messpunkte sich jeweils in einem bestimmten Bereich der CPU-Auslastung

sammeln. Dies entspricht den Erwartungen, da die Last, welche durch das Programm zur Lastgenerierung erzeugt wird, schrittweise um 12,5 Prozent erhöht wurde.

- **Mögliche Abhängigkeiten:** Ebenfalls ist im Streudiagramm gut zu erkennen, dass der Stromverbrauch, welcher gemessen wurde, nicht linear zur CPU-Auslastung des Computers steht. Betrachtet man das Streudiagramm jedoch abschnittsweise, so ist im Bereich zwischen näherungsweise 20 Prozent und 100 Prozent sehr wohl ein möglicher linearer Zusammenhang zu erkennen. Ebenso in dem Bereich zwischen 0 und 20 Prozent.
- **Streuung:** Es ist ebenfalls zu beobachten, dass die Messwerte im Bereich der unteren CPU-Auslastung sehr stark streuen, im oberen Bereich jedoch sehr nahe beieinander liegen. Dieser Punkt wird jedoch bei der Betrachtung der Boxplots noch einmal klarer.

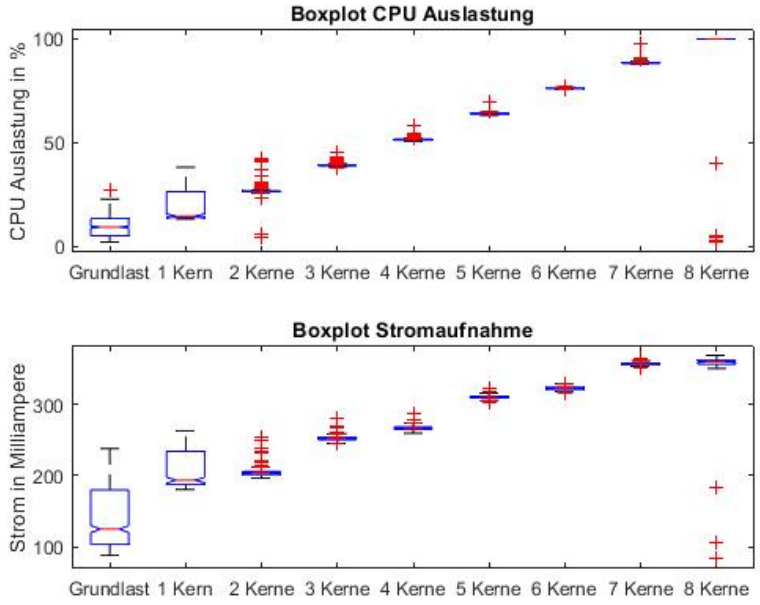

Abbildung 5.2: Boxplott der Messungen am PC unter Windows

Hierbei werden in Abbildung 5.2 die gemessenen Ströme sowie die CPU Auslastung für jede Messung dargestellt. Diese liefern, wie Abbildung 5.1 Aufschluss über die Verteilung sowie Plausibilität der Messungen.

- **Ausreißer:** Wie man der Abbildung 5.2 entnehmen kann, befinden sich die Ausreißer (hier rote Kreuze) überwiegend oberhalb des oberen Whiskers. Ebenso sind die Whisker

sowie die Standardabweichung in allen Fällen nach oben verschoben. Interpretiert heißt dies, dass die Werte von ihrer Anzahl her eher oberhalb des Mittelwertes als unterhalb liegen. Dies ist, wie in Abschnitt beschrieben, mit dem Ausführen von Hintergrundprogrammen durch das Betriebssystem zu begründen. Hierbei wird die Last, welche nur durch das Programm, welches für die Messung genutzt wurde, überschritten durch andere Prozesse, welche das Betriebssystem ausführt.

- **Verteilung:** Wie bereits in der Abbildung 5.1 zu erkennen, ist die Streuung der einzelnen Messungen nicht gleich. Hierbei ist bei den Messungen für zwei Kerne bis hin zu den Messungen für alle 8 Kerne nur eine sehr geringe Streuung sowohl in der Stromaufnahme als auch in der CPU Auslastung zu erkennen. Für die Messung unter Grundlast sowie für einen Kern sind jedoch höhere Streuungen in den Werten sowohl bei der CPU Auslastung als auch bei der Stromaufnahme zu erkennen. Insbesondere streut die Messung der Stromaufnahme für die Grundlast sehr stark, die Messung der CPU Auslastung hingegen signifikant weniger stark. Dies lässt auf einen hohen Einfluss der CPU Frequenz auf den Stromverbrauch schließen, da im unteren Bereich der CPU Auslastung die CPU Frequenz stark schwankt und nicht wie in den Messungen unter höherer Last relativ konstant bei 100 Prozent liegt.

Ebenfalls ist zu erkennen, dass bei steigender CPU Auslastung die Stromaufnahme des Rechners ansteigt, dass ist zu erwarten und bestätigt ebenfalls, dass die Messungen korrekt durchgeführt wurden.

5.1.2 Laptop unter Windows

Die Messergebnisse für die Messungen an dem Laptop unterscheiden sich hierbei stark von den Ergebnissen des Desktop PCs, bei der Betrachtung des Streudiagramms aus Abbildung 5.3 fallen hierbei folgende Punkte auf:

- Die Messergebnisse sind im Gegensatz zu dem Streudiagramm 5.1 wesentlich weniger zentriert und weisen eine höhere Bandbreite an CPU Auslastung auf. Was damit zusammenhängt, dass auf dem Laptop mehr Prozesse im Hintergrund laufen, welche nicht abgeschaltet werden können. Aus diesem Grund schwankt die CPU hier wesentlich mehr.
- Ebenfalls ist zu erkennen, dass die Steigung der Stromaufnahme kleiner ist als in der Messung unter dem Desktop PC, jedoch ist auch hier das Muster zu erkennen, dass

die Stromaufnahme von 12 Prozent bis 100 Prozent annähernd linear zur CPU Auslastung auftritt. Der Bereich von 0 bis 12 Prozent weicht hingegen wieder stark von dem vermuteten linearen Zusammenhang ab.

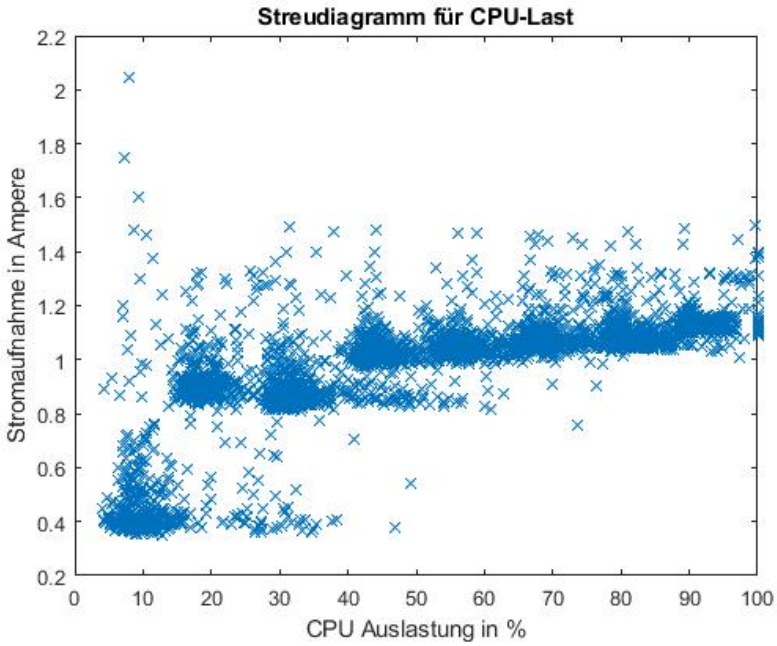

Abbildung 5.3: Streudiagramm der Messungen am Laptop unter Windows

Abbildung 5.4 präsentiert die Ergebnisse der Messung der Stromaufnahme am Laptop in Form von Boxplots. Auffällig ist die große Anzahl von Ausreißern, die vorwiegend oberhalb des Mittelwertes liegen. Dies kann auf Programme zurückgeführt werden, die im Hintergrund laufen und somit die geplante Last überschreiten.
Im Vergleich zu den Messungen am Desktop-PC zeigt sich, dass die Stromaufnahme in den Boxplots nicht in gleichem Maße ansteigt. Dies ist unter anderem auf die geringere Auflösung des Messbereichs zurückzuführen, der aufgrund der höheren Ströme beim Laptop gewählt werden musste.

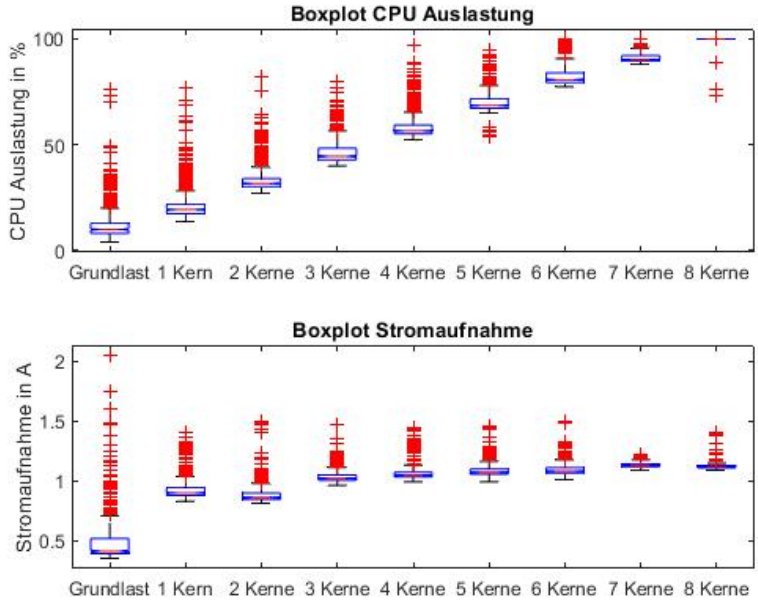

Abbildung 5.4: Boxplott der Messungen am Laptop unter Windows

5.1.3 Desktop PC unter Linux

In Abbildung 5.5 sind die Ergebnisse der Messung der Stromaufnahme des Computers unter dem Betriebssystem Linux dargestellt. Es zeigt sich ein deutlicher Unterschied zu den zuvor aufgeführten Messungen. Die Stromaufnahme steigt mit zunehmender CPU-Auslastung bis zu einem Maximum bei 30% stark an, fällt jedoch darüber hinaus wieder ab. Überraschenderweise ist die Stromaufnahme bei 100% CPU-Auslastung nahezu identisch mit der bei 20% Auslastung. Die maximale Stromaufnahme bei 30% CPU-Auslastung liegt unerwartet hoch, was möglicherweise auf eine Reduktion der CPU-Frequenzen bei höherer Auslastung zurückzuführen ist. Dies könnte ein Hinweis auf ein Problem mit dem Programm, das die Last erzeugt, oder mit dem Betriebssystem selbst sein.
Angesichts dieser widersprüchlichen Ergebnisse erscheint die Entwicklung eines Regressionsmodells ohne weitere Untersuchung der Ursachen für diese unerwarteten Messwerte wenig sinnvoll. Daher wird von einer weiteren Auswertung der Daten abgesehen.

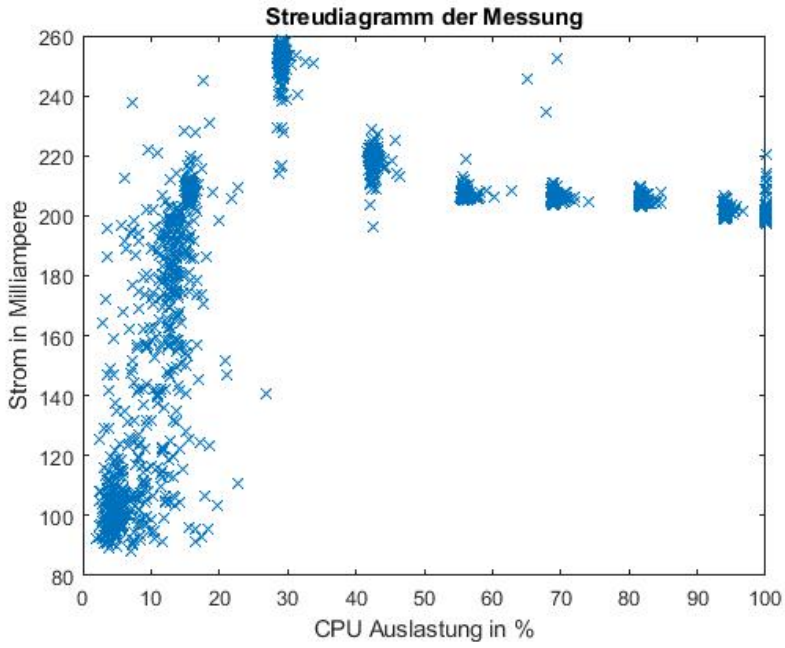

Abbildung 5.5: Streudiagramm der Messungen am Desktop PC unter Linux

5.2 Anwenden der Regression

Nach der ersten Betrachtung der Messwerte ist ersichtlich, dass eine einfache lineare Regression nicht zielführend ist. Nach der Betrachtung der Messwerte bietet sich als erster Ansatz eine nichtlineare Regression an, welche ein Polynom 2. oder 3. Grades erstellt.
Bei der Anwendung dieser Methode kommt es jedoch aufgrund von zu vielen Messpunkten zum Oszillieren der Funktion, was zu einer Verschlechterung des Ergebnisses führen kann. [21]
Da jedoch bei den Messungen unter Windows ab einer bestimmten CPU-Auslastung von einem linearen Verlauf ausgegangen werden kann, bietet sich hierbei die Möglichkeit der Erstellung einer Regression durch eine abschnittsweise definierte Funktion an. Hierbei werden die Messergebnisse in zwei Intervalle aufgeteilt, wobei jedes Intervall als lineare Funktion beschrieben werden kann.

5.2.1 Desktop PC unter Windows

Nach der ersten Betrachtung der Messwerte lassen diese einen annähernd linearen Verlauf im Bereich ab 20 Prozent CPU-Auslastung vermuten. Für den Bereich von 0 bis 20 Prozent CPU-Auslastung kann ebenfalls ein linearer Zusammenhang vermutet werden.

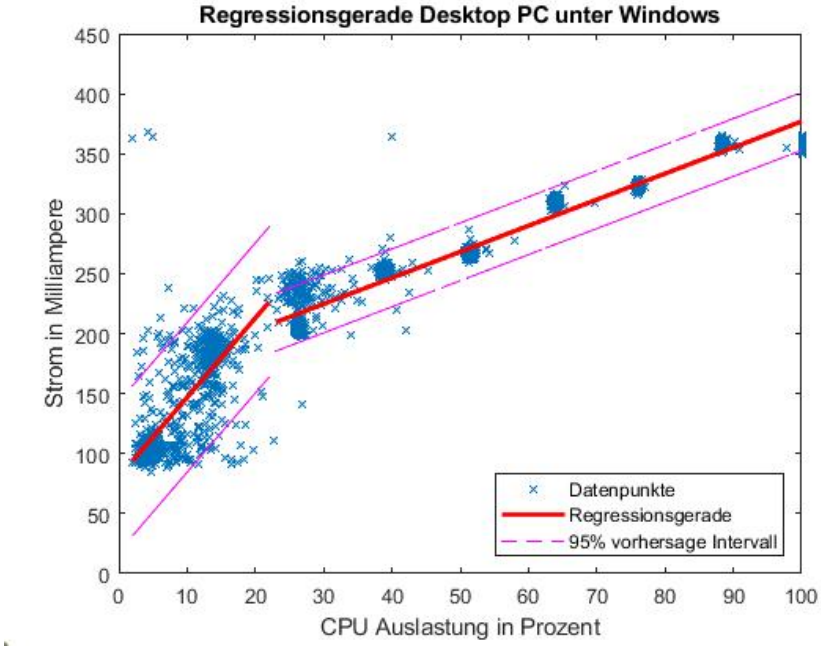

Abbildung 5.6: Lineares Regressionsmodell für den Desktop PC unter Windows

In Abbildung 5.6 wurde der Ansatz verfolgt das Regressionsmodell durch zwei Lineare Funktionen der Realität anzunähern. Die abschnittsweise definierte Funktion $f(x)_{DesktopPC}$ ist gegeben durch:

$$f(x)_{DesktopPC} = \begin{cases} 6,2764x + 85,5814 & \text{für } 0 \leq x \leq 20 \\ 2,1735x + 159,5917 & \text{für } 20 < x \leq 100 \end{cases} \tag{5.1}$$

Wobei $f(x)$ den Strom in Milliampere welcher vom Computer aufgenommen wird darstellt. Der Parameter x stellt hierbei die CPU-Auslastung in Prozent da.

5.2.2 Laptop unter Windows

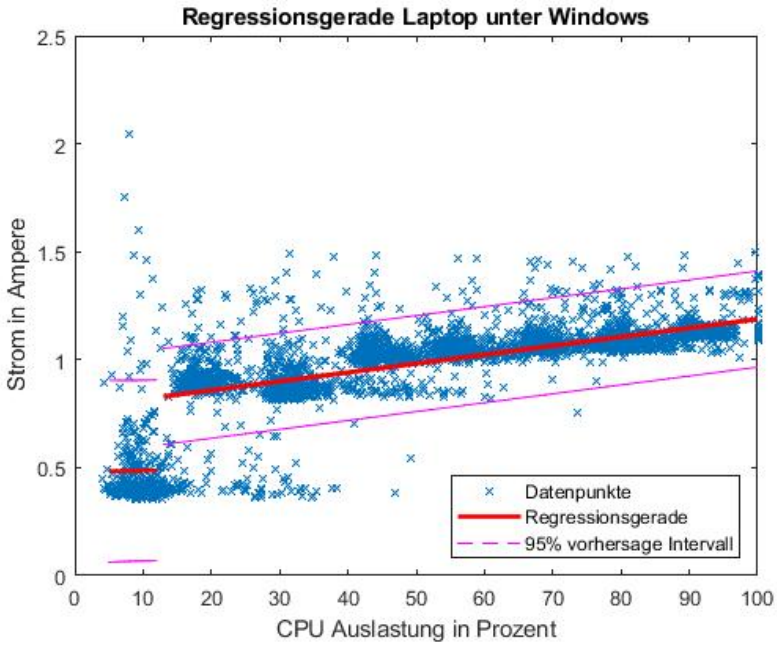

Abbildung 5.7: Lineares Regressionsmodell für den Laptop unter Windows

Gleich wie in Abbildung 5.6 wird hier der Ansatz einer Abschnittsweise definierten Funktion verfolgt. Die abschnittsweise definierte Funktion $f(x)_{Laptop}$ ist hierbei gegeben durch:

$$f(x)_{Laptop} = \begin{cases} 0,0006x + 0,4803 & \text{für } 0 \leq x \leq 12 \\ 0,0414x + 0,7761 & \text{für } 12 < x \leq 100 \end{cases} \tag{5.2}$$

5.3 Bewerten der Regression

Um die erstellten Regressionsmodelle nach ihrer Qualität bewerten zu können, werden die Methoden aus Abschnitt 3.4.1 angewandt.

5.3.1 Desktop PC unter Windows

Zur Überprüfung der Annahme einer Normalverteilung der Residuen werden QQ-Plots in den Abbildungen 5.8 und 5.9 dargestellt. Jede Abbildung repräsentiert dabei einen Abschnitt der abschnittsweise definierten Funktion. Abbildung 5.8 zeigt die Residuen für den Bereich von 0

bis 20 Prozent CPU-Auslastung. Hier ist erkennbar, dass die Residuen an den Rändern nach oben und unten leicht abweichen, was auf eine mögliche Verletzung der Normalverteilungsannahme in diesem Bereich hindeutet. Im Gegensatz dazu folgen die Residuen in Abbildung 5.9, die den Abschnitt von über 20 Prozent bis 100 Prozent CPU-Auslastung widerspiegelt, weitgehend der Winkelhalbierenden. Dies deutet darauf hin, dass die Residuen in diesem Abschnitt näherungsweise normalverteilt sind.
Aus diesen Beobachtungen lässt sich schließen, dass die Normalverteilungsannahme für die Residuen im ersten Bereich (0 bis 20 Prozent) möglicherweise nicht zutrifft, während sie im zweiten Bereich (über 20 Prozent bis 100 Prozent) als plausibl angesehen werden kann.

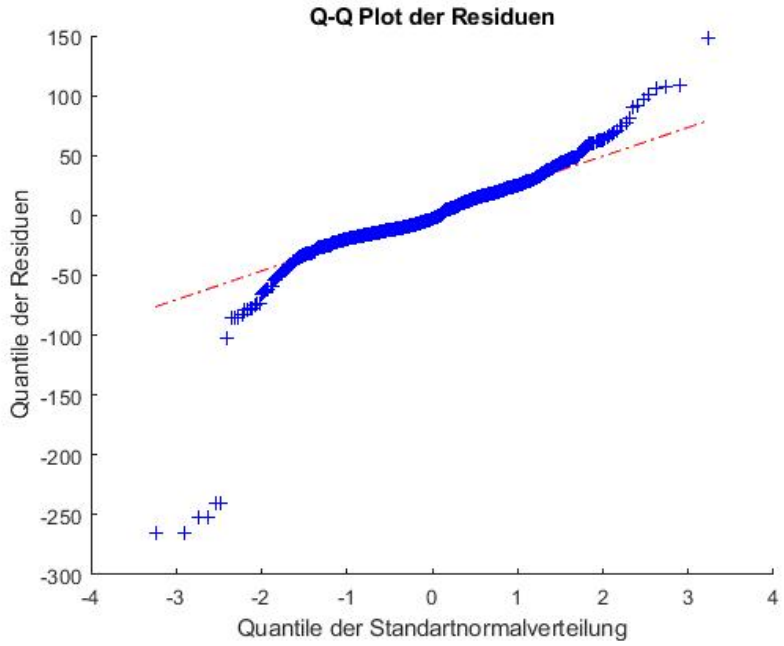

Abbildung 5.8: QQ-Plot der Residuen aus Abbildung 5.6 bis 20 % CPU-Auslastung

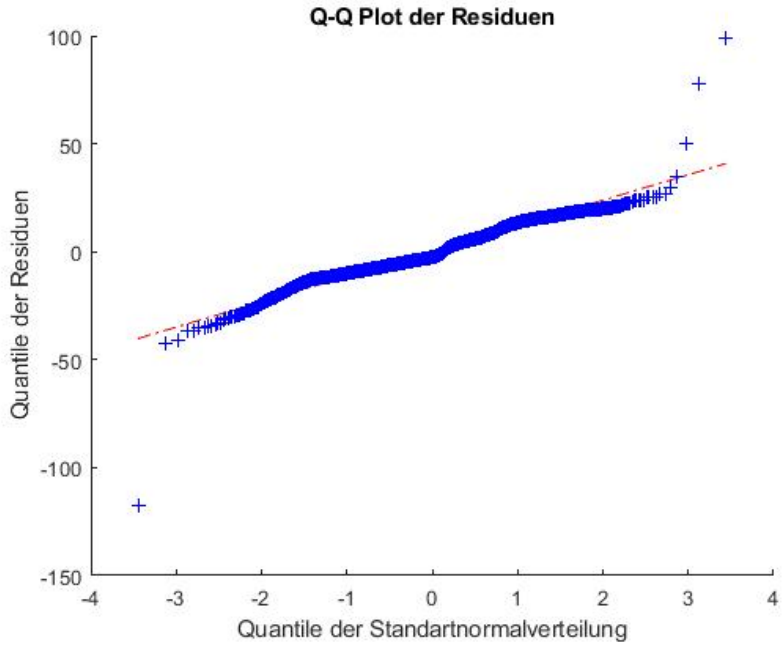

Abbildung 5.9: QQ-Plot der Residuen aus Abbildung 5.6 ab 20 % CPU-Auslastung

In Grafik 5.10, die zur Überprüfung der Homoskedastizität der Daten dient (beschrieben in Abschnitt 3.4.1), zeigen die Residuen für den Bereich bis 20 Prozent CPU-Auslastung keine erkennbaren Muster und schwanken um die Nulllinie. Dieses Verhalten unterstützt die Annahme der Homoskedastizität, da eine gleichmäßige Streuung der Residuen um die Nulllinie herum keine Anzeichen für zunehmende oder abnehmende Varianz erkennen lässt.
Für den Bereich ab 20 Prozent CPU-Auslastung sind die Residuen ebenfalls um die Nulllinie verteilt, ohne sichtbare Muster wie Trichterformen, die auf Heteroskedastizität hindeuten würden. Allerdings machen die größeren Abstände zwischen den Werten der CPU-Auslastung eine eindeutige Beurteilung der Homoskedastizität schwierig. Dennoch lässt das Fehlen von offensichtlichen Verstößen gegen die Homoskedastizität die vorsichtige Annahme zu, dass die Daten auch in diesem Bereich homoskedastisch sind.

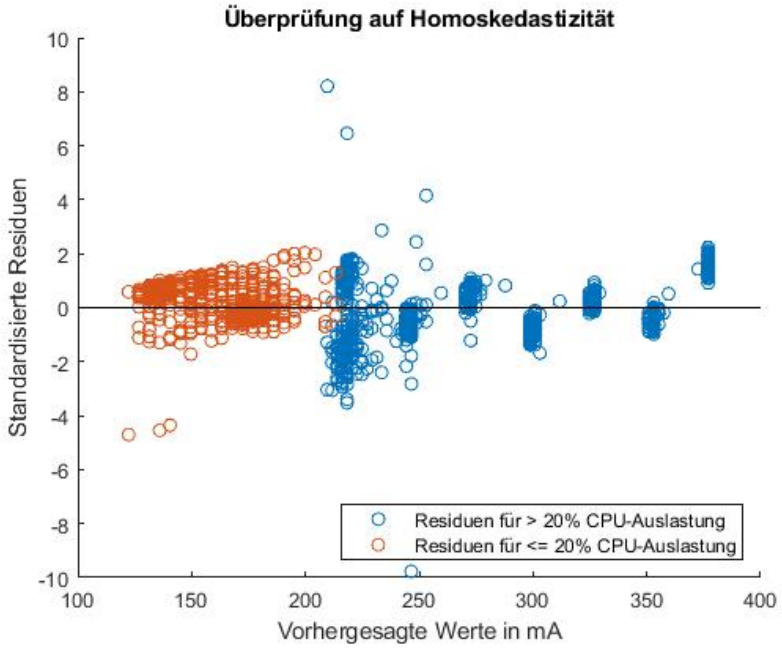

Abbildung 5.10: Abbildung der standardisierten Residuen aus Abbildung 5.6 gegen die Vorhergesagten Werte

In der Grafik 5.11, die gemäß Abschnitt 3.4.1 zur Überprüfung der Unkorreliertheit herangezogen wird, sind die Residuen des Modells für beide Abschnitte der abschnittsweisen Funktion 5.1 dargestellt.
Für den ersten Abschnitt der Funktion, der bis zu 20 Prozent CPU-Auslastung reicht, schwanken die Residuen effektiv um die Nulllinie. Dies deutet darauf hin, dass in diesem Bereich die Residuen keine systematischen Muster aufweisen und somit eine gute Unkorreliertheit annehmen lassen.
Im Gegensatz dazu zeigt der Bereich ab 20 Prozent CPU-Auslastung zwar ebenfalls Schwankungen der Residuen um die Nulllinie, jedoch sind die Abstände zwischen den einzelnen Residuenwerten aufgrund der höheren Abstände innerhalb der CPU-Auslastung größer. Dies erschwert eine eindeutige Beurteilung der Unkorreliertheit. Ebenfalls lässt sich ein Muster erkennen, wobei die Residuen abwechselnd über sowie unter der Nulllinie verlaufen. Diese bleiben jedoch in einem konstanten Rahmen, was ein guter Hinweis auf eine Unkorreliertheit ist.

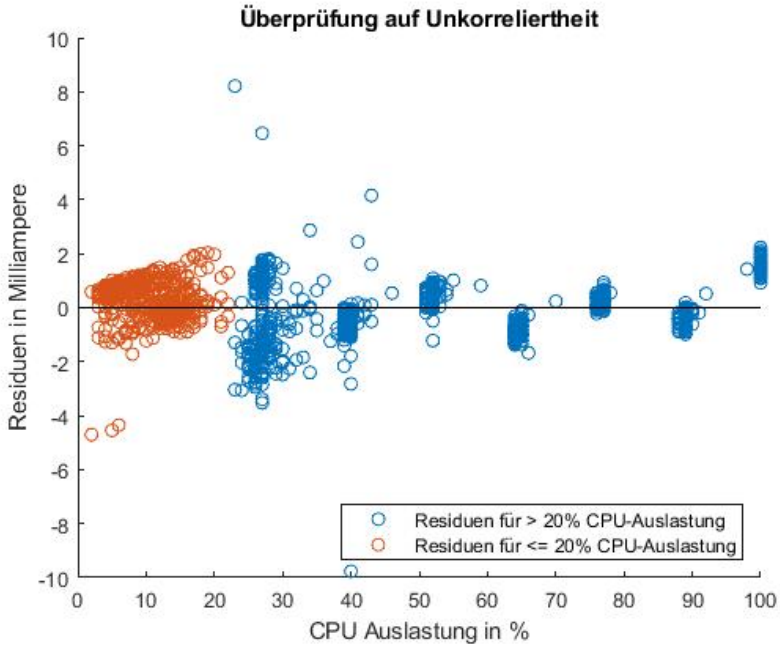

Abbildung 5.11: Abbildung der standardisierten Residuen aus Abbildung 5.6 gegen die unabhängige Variable

5.3.2 Laptop unter Windows

Die Grafiken 5.12 und 5.13 präsentieren die QQ-Plots der Residuen, die aus der abschnittsweise definierten Regressionsfunktion für die Messung des Laptops resultieren. In beiden Diagrammen ist eine deutliche Abweichung von der Winkelhalbierenden erkennbar. Diese Beobachtung deutet darauf hin, dass die Residuen nicht normalverteilt sind, wodurch die Annahme einer Normalverteilung der Residuen für dieses Modell nicht gewährleistet ist.

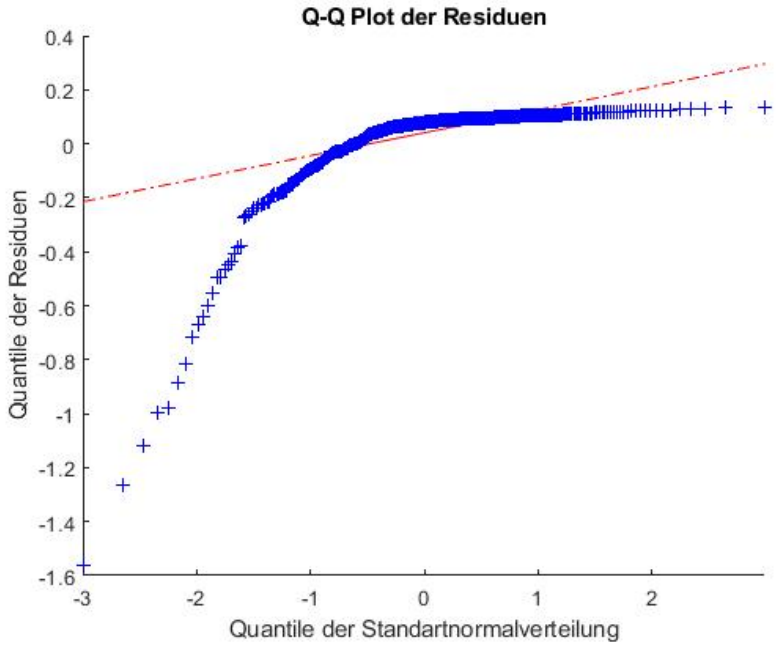

Abbildung 5.12: QQ-Plot der Residuen aus Abbildung 5.7 bis 12 % CPU-Auslastung

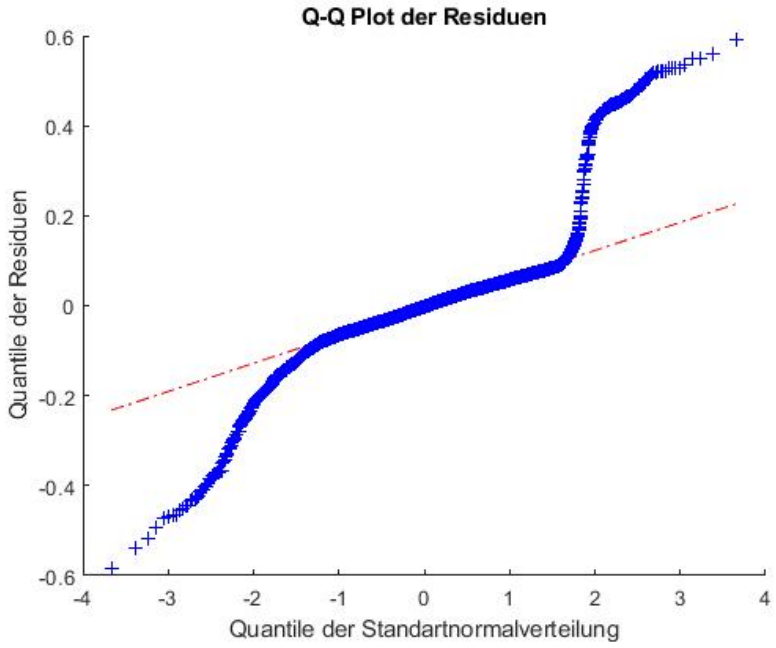

Abbildung 5.13: QQ-Plot der Residuen aus Abbildung 5.7 ab 12 % CPU-Auslastung

In Grafik 5.14, die zur Überprüfung der Homoskedastizität der Daten dient, ist der signifikante Abstand zwischen den Residuen der einzelnen Funktionsabschnitte gemäß 5.2 deutlich erkennbar. Besonders auffällig ist die geringe Ausdehnung des Funktionsabschnitts für CPU-Auslastungen kleiner oder gleich 12 Prozent nach links sowie rechts. Diese Beobachtung lässt sich durch die sehr geringe Steigung der Funktion in diesem Bereich erklären. Obwohl in diesem Segment starke Ausreißer existieren, schwanken die meisten Residuen um die Nulllinie, was eine generelle Stabilität der Varianz in diesem Bereich andeutet.
Für den Bereich über 12 Prozent CPU-Auslastung zeigen die Residuen ebenfalls Schwankungen um die Nulllinie, allerdings mit einer leichten Tendenz zum positiven Bereich, wobei die Abstände konstant bleiben. Im Übergang zwischen den Funktionsabschnitten treten zudem starke Ausreißer in den negativen Bereich auf. Diese Phänomene sind auf die abschnittsweise definierte Funktion zurückzuführen, bei der die Übergänge weniger linear verlaufen als bei den restlichen Werten. Aufgrund dieser Beobachtungen kann die Annahme der Homoskedastizität getroffen werden. Die gleichmäßige Verteilung der Residuen und die konsistenten Abstände über die verschiedenen Bereiche hinweg bestätigen, dass die Varianz der Residuen über den gesamten Bereich der CPU-Auslastung stabil ist.

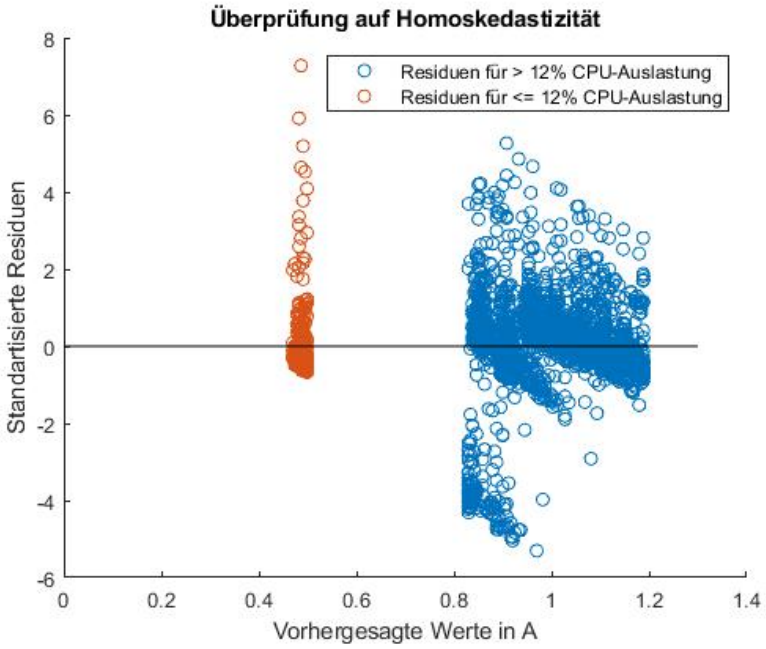

Abbildung 5.14: Abbildung der standardisierten Residuen aus Abbildung 5.7 gegen die Vorhergesagten Werte

Grafik 5.15 dient der Überprüfung der Unkorreliertheit der Residuen, wie in Abschnitt 3.4 beschrieben. Die Analyse zeigt, dass die Residuen generell um die Nulllinie schwanken, allerdings sind auffällige Ausreißer im positiven Bereich erkennbar, die im negativen Bereich nicht vorhanden sind. Diese Ausreißer verteilen sich gleichmäßig über den gesamten Bereich der CPU-Auslastung, ohne erkennbare Muster, die sich mit der CPU-Auslastung verändern würden. Eine Ausnahme bildet der Bereich zwischen 10 und 20 Prozent CPU-Auslastung, wo eine Gruppe signifikanter Ausreißer in den negativen Bereich auffällt. Dies kann auf die Eigenschaften der abschnittsweise definierten Funktion zurückgeführt werden, da der Übergangsbereich weniger Linearität aufweist als die Bereiche mit höherer CPU-Auslastung.

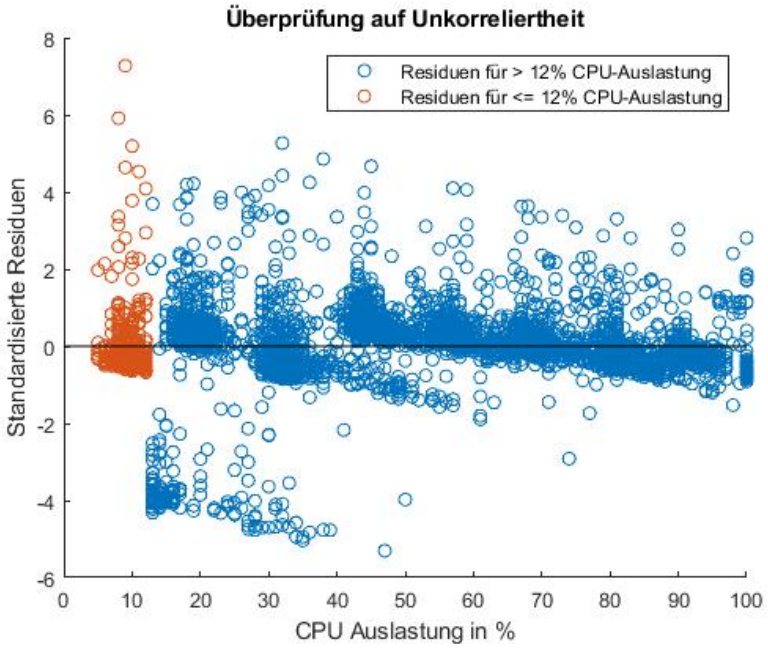

Abbildung 5.15: Abbildung der standardisierten Residuen aus Abbildung 5.7 gegen die unabhängige Variable

5.4 Fazit der Regressionsanalyse

Nachdem die erstellte Regression mittels der statistischen Methoden untersucht wurde, kann hierbei die Aussage getroffen werden, dass die erstellte Regression für den Desktop-PC in Bezug auf Genauigkeit sowie Robustheit besser ist als die für den Laptop.
Bei der Analyse ist vor allem die Normalverteilung der Residuen für die Regression, die für den Laptop erstellt wurde, nicht gegeben. Dies lässt den Schluss zu, dass die Residuen nicht

zufällig vorliegen und es somit einen nicht erkannten Faktor geben könnte.
Eine mögliche Ursache hierfür sind plötzlich im Hintergrund anlaufende Prozesse, welche das Messergebnis verfälschen könnten.

6 Rückführung der Ergebnisse

Folgendes Kapitel der Arbeit beschäftigt sich hierbei mit der Rückführung der Ergebnisse aus der Regressionanalyse im Kapitel 5. Das Wort Rückführung ist hierbei in mehrere Hinsicht wie folgt zu verstehen

- Rückführung auf einzelne Prozesse: Ziel ist es hierbei die Auslastung der CPU für einzelne Prozesse zu ermitteln und diese dann auf die Funktion der Regression anzuwenden. Hierbei wird sich auf das Auslesen der Prozesse unter dem Betriebssystem Windows konzentriert.
- Rückführung auf die Verwendeten Resourcen: Hierbei wird erklärt wie mittels des erstellten Modells des Computers auf den CO2 Verbrauch welcher hierdurch entsteht geschlossen werden kann.
- Rückführung der Ergebnisse auf den Nutzer: Dieser Teil Beschreibt die eigentliche Rückführung welcher dem Namen der Arbeit zugrunde liegt, hierbei wird nach dem Prinzip des User-in-the-Loop eine Grafische Rückführung der in den beiden oben beschriebenen Punkten erzeugt.

6.1 Rückführung des Stromverbrauch auf einzelne Prozesse

Um die in Abschnitt 5 erstellten Regressionsgeraden auf einzelne Prozesse anwenden zu können, muss zunächst die CPU-Auslastung, welche ein Prozess verursacht, ermittelt werden. Hierzu wird der in Abschnitt 2.2.2 beschriebene Performancecounter `Prozess` verwendet. Dieser verwendet ähnlich wie der Performancecounter `Prozessor` einen Rate-Counter, welcher die CPU-Auslastung, die durch den einen Prozess verursacht wird, ermittelt. Das Auslesen der CPU-Auslastung für einen einzelnen Prozess erfolgt somit ähnlich, wie in Abschnitt 3.3.1 für das Auslesen der gesamten CPU-Auslastung beschrieben. Die Abfrage der Werte wird zunächst initialisiert und anschließend sekündlich abgerufen.
Zum Initialisieren der Abfrage für einen Prozess, benötigt die Abfrage als Information den Namen des Prozesses.

Dadurch, dass der Counter für die Abfrage der CPU-Auslastung den Namen des Prozesses als Eingabewert benötigt, entsteht hier eine Limitation bezüglich des Auslesens spezifischer Prozesse. Anders als die Prozess-ID, welche im System nur einmal vorhanden ist und somit eine eindeutige Identifikation eines Prozesses ermöglicht, kann der Name eines Prozesses im Betriebssystem mehrfach vorkommen. Öffnet man beispielsweise das gleiche Programm mehrfach, so hat jedes dieser Programme eine individuelle Prozess-ID, der Name des Prozesses ist hierbei jedoch für alle Prozesse gleich. Somit kann die CPU-Auslastung für einen Prozess nur sicher ermittelt werden, wenn der Name des Prozesses einmalig im System ist. [10]

Des Weiteren muss beim Auslesen der CPU-Auslastung im Prozentformat über die API `PdhGetFormattedCounterValue`, im Gegensatz zum Auslesen der Daten für die gesamte CPU, das Flag `PDH_FMT_NOCAP100` hinzugefügt werden. Dieses hebt die Beschränkung für den Anzeigewert auf maximal 100 Prozent auf. [18]
Dies ist nötig, da der Counter, wie in Abschnitt 2.2.2 beschrieben, für ein Prozessor-System entwickelt wurde und die meisten heutigen Rechner über mehr als einen Kern verfügen, kann der abgefragte Prozentwert über 100 Prozent liegen. Um die korrekte CPU-Auslastung für einen Prozess zu ermitteln, muss der ermittelte Wert durch die Anzahl der auf dem System vorhandenen logischen Kerne geteilt werden. Hierzu wird mittels der Windows-Funktion `GetSystemInfo()` die Anzahl der Kerne ermittelt. Diese Funktion füllt eine Struktur vom Typ `SYSTEM_INFO`, die Informationen zur Hardware-Konfiguration des Systems enthält. Für das Auslesen der Anzahl der Kerne wird der Wert `.dwNumberOfProcessors` abgerufen und gespeichert.

Dieses Vorgehen ermöglicht es nun, die CPU-Auslastung zu bestimmen, die durch einzelne Prozesse verursacht wird. Um nun den hieraus resultierenden Stromverbrauch zu ermitteln, der durch die Prozesse verursacht wird, kann die mittels Regression erstellte Funktion herangezogen werden.
Dabei kann der ermittelte Prozentwert der CPU-Auslastung des einzelnen Prozesses in die Funktion eingesetzt werden, um den durch den Prozess verursachten Stromverbrauch zu ermitteln.

6.2 Rückführung der Ergebnisse auf den CO2-Verbrauch

In Abschnitt 2.4.1 wurde bereits erklärt, dass die Erzeugung von Strom einen signifikanten Beitrag zum CO2-Ausstoß leistet. Der Ausstoß von CO2, welcher für die Erzeugung von Strom entsteht, wird hierbei quantifiziert als Ausstoß von CO2 in Gramm pro geleisteter Kilowattstunde.
Hierbei ist die Kilowattstunde die geleistete Arbeit P, welche in einem gewissen Zeitraum verbraucht wurde und berechnet sich wie folgt:

$$P = W \cdot t$$

Die zur Berechnung der elektrischen Arbeit nötige elektrische Leistung W berechnet sich hierbei durch:

$$W = U \cdot I$$

Um nun den CO2-Ausstoß der CPU-Auslastung für das gesamte System sowie für einzelne Prozesse zu ermitteln, wird die in Abschnitt 5 ermittelte Funktion der Regressionsgerade herangezogen. Da diese den Strom, welcher durch die CPU verursacht wird, als Ergebnis liefert, kann durch Multiplikation des Ergebnisses der Funktion mit der für die Messung zugrunde liegenden Spannung die elektrische Leistung W berechnet werden.
Um nun von der Leistung auf die geleistete Arbeit zu schließen, wird noch die Zeit benötigt, in welcher die ermittelte CPU-Auslastung vorliegt. Da zwischen den Abständen der einzelnen Datenstichproben für die Ermittlung der CPU-Auslastung eine Sekunde vergeht und hierbei die mittlere CPU-Auslastung für diese Zeit angegeben wird, kann als Zeitbasis die Zeit zwischen den einzelnen Datenstichproben also eine Sekunde genommen werden. Somit erhält man die elektrische Arbeit, welche in der Zeit des Messintervalls geleistet wurde. Um nun auf die geleistete elektrische Arbeit für einen Zeitraum x zu schließen, muss folgende Berechnung durchgeführt werden:

$$P = \sum U \cdot I_n \cdot t_n \tag{6.1}$$

wobei U die Spannung, I_n der Strom welcher aktuell errechnet wurde und t_n die Zeit zwischen den einzelnen Messungen darstellt.

6.3 Rückführung der Ergebnisse auf den Nutzer

In diesem Abschnitt der Arbeit wird die Entwicklung einer grafischen Benutzeroberfläche erläutert, welche dem Nutzer Erkenntnisse der Arbeit sowie die Auswirkungen der von ihm

genutzten Programme auf das Klima aufzeigen. Die grafische Benutzeroberfläche wird hierbei für das Betriebssystem Windows entwickelt. Die Beschränkung auf Windows wird hierbei getätigt, da eine plattformübergreifende GUI zwar mittels geeigneter Programmier-Tools möglich ist, jedoch sind die Verfahren innerhalb der einzelnen Betriebssysteme zum Auslesen der relevanten Daten zu unterschiedlich, sodass eine plattformübergreifende Anwendung im Rahmen dieser Arbeit nicht sinnvoll ist.

6.3.1 Grundgedanke der grafischen Benutzeroberfläche

Der Gedanke hinter dem Einsatz einer GUI richtet sich hierbei nach dem Prinzip des User-in-the-Loop, dieses Prinzip, wie in Abschnitt 2.1 beschrieben, beschreibt hierbei einen geschlossenen Regelkreis. Dieser Regelkreis besteht hierbei aus dem Nutzer, welcher das System darstellt, welches geregelt werden soll, sowie einem Regler, welcher den Nutzer hinsichtlich seines Konsums bzw. seines Umgangs mit dem Klima regeln soll. Die Rückführung hierbei ist die Auswirkung auf den Stromverbrauch sowie das Klima, welche der Nutzer durch das Nutzen von Programmen verursacht.
Der Nutzer soll durch die grafische Benutzeroberfläche also ein Feedback seiner Auswirkungen auf das Klima erhalten, welches durch sein Handeln verursacht wird.

Funktionen der GUI

Um dem Nutzer ein Feedback seiner Handlungen auf die Auswirkungen des Klimas sowie seinen Stromverbrauch zu geben, gilt es die GUI so zu strukturieren, dass der Nutzer die für ihn notwendigen Informationen grafisch ansprechend präsentiert bekommt.
Ebenfalls muss ein Programm so konzipiert sein, dass es keinen negativen Einfluss auf die eigentliche Nutzung des Computers besitzt, hierbei ist vor allem der Punkt zu beachten, dass das Programm für eine dauerhafte Messung der Leistung dauerhaft ausgeführt werden muss. Das dauerhafte Laufen des Programms muss somit im Hintergrund gestaltet werden, sodass es dem Nutzer nicht auffällt, außer der Nutzer möchte die Informationen innerhalb des Programms abfragen.
Hierfür bietet sich die Programmierung der Anwendung als TrayIcon an, dabei handelt es sich um ein Programm, was im Hintergrund läuft und als Symbol nur im Tray-Bereich der Windows Taskbar sichtbar ist. Dies ermöglicht das kontinuierliche Laufen des Programms im Hintergrund ohne, dass es dem Nutzer auffällt, sogleich kann dieser Zugriff auf die Informationen bekommen, wenn er das Icon im Tray-Bereich betätigt.
Innerhalb der Nutzeroberfläche gilt es, die wichtigen Funktionen abzubilden, ohne dass die GUI hierbei überladen wirkt. Die hieraus entstehenden grafischen sowie funktionstechnischen

Anforderungen der GUI werden hierbei im Folgenden erläutert.

Funktionen der Grafische Anzeigen
Für eine effektive und nutzerfreundliche Darstellung der Informationen ist es wichtig, dass die Anzeige dieser Werte dynamisch und in einem angemessenen Zeitintervall aktualisiert werden. Dies stellt sicher, dass die Informationen stets aktuell sind, ohne den Nutzer zu überfordern oder es ihm zu erschweren, die Werte abzulesen.
Ein wichtiger Aspekt der Anwendung ist es, die durch die CPU-Belastung entstandenen Emissionen auf einzelne Prozesse sowie das gesamte System zurückzuführen. Um dies grafisch darzustellen, wird für die Prozesse und das gesamte System eine separate Anzeige für den Verbrauch und die Emissionen vorgesehen. Aufgrund der hohen Anzahl von Prozessen, die ein Computer typischerweise verwaltet, wäre eine Darstellung aller Prozesse zu unübersichtlich. Daher wird dem Nutzer eine Auswahlmöglichkeit über ein Menü angeboten, in dem alle Prozesse gelistet und zur einfacheren Navigation alphabetisch sortiert sind. Der Nutzer kann aus dieser Liste einen spezifischen Prozess auswählen, den er näher beobachten möchte.
Da der CO2-Emissionsfaktor, wie bereits dargestellt, regional stark variieren kann, ist es essenziell, dass Nutzer die Möglichkeit haben, die Region auszuwählen, deren CO2-Koeffizient als Grundlage für die Berechnungen verwendet werden soll. Diese Funktionalität ermöglicht es, die Berechnung der Emissionen an die spezifischen Gegebenheiten der gewählten Region anzupassen und somit genauere und relevantere Informationen bereitzustellen.

Bezüglich der Bedienbarkeit des Programms gilt es den Nutzer in eine gewohnte Atmosphäre zu setzen woraus folgt, dass alle Aktionen die der Nutzer tätigen kann nach bereits bekannten Prinzipien ablaufen sollen. Somit wird sich für die Implementierung der Interaktionen zwischen Nutzer und Programm an ihm bereits bekannte Programme gehalten.
Bei vielen Anwendungen, die ein TrayIcon verwenden, öffnet sich ein Fenster automatisch, wenn der Benutzer mit der Maus über das Icon fährt. Verlässt der Nutzer das Icon wieder, schließt sich das Fenster automatisch. Diese Funktion sollte auch in der Programmierung des Icons implementiert werden. Zusätzlich wird erwartet, dass sich das Fenster dauerhaft öffnet, sobald es durch einen einfachen Klick aktiviert wird. Anders als bei üblichen Desktop-Anwendungen, bei denen oft ein Doppelklick erforderlich ist, erwarten Nutzer bei Anwendungen in der Taskleiste, dass sich Programme durch einen einfachen Klick öffnen lassen.
Das Betätigen von Buttons oder dem Programm selbst wird hierbei so implementiert, dass dies mit einem einfachen Klick der Linken Maustaste geschehen kann. Dies klingt zwar erstmal sehr Trivial, jedoch ist dies der Grund es hierbei nochmal zu erwähnen. Bei der Programmierung

einer GUI ist die Implementierung von Ereignissen erstmal frei sodass auch das Betätigen eines Buttons mit der rechten Maustaste frei möglich ist, jedoch ist genau dies eine Aktion mit der kein Benutzer rechnet was zu einer erheblich erschwerten Bedienung des Programms führt. Für das Schließen einer Anwendung wird hierbei herkömmlich das **X** in der rechten oberen Ecke eines Programms gewählt, dies soll auch in dieser GUI möglich sein jedoch würde die Aufzeichnung der CPU Auslastung nach dem beenden des Programms nicht mehr möglich sein. Somit gilt es die GUI lediglich augenscheinlich zu schließen sodass der Nutzer diese nichtmehr bemerkt. Im Hintergrund soll die GUI hingegen weiter laufen um weiterhin das Aufzeichnen und berechnen der Werte zu ermöglichen.
Die genaue Erstellung der GUI sowie die für die Realisierung der Funktionen nötigen Schritte wird nun im folgenden Abschnitt beschrieben

6.3.2 Erstellung der grafischen Nutzeroberfläche

Grundlagen der Erstellung einer GUI

Die Erstellung einer grafischen Nutzeroberfläche in C++ basiert auf einem Grundgerüst, welches die Erstellung der Oberfläche sowie die späteren Nutzerspezifischen Aktionen ermöglicht. Auf dieses Grundgerüst wird im folgenden genauer eingegangen.

Registrierung der Fensterklasse

Eine Fensterklasse beschreibt die Eigenschaften eines Fensters, darunter auch, wie es auf verschiedene Nachrichten reagiert. Im Code wird zunächst eine Instanz von `WNDCLASS` definiert, die verschiedene Felder wie `lpfnWndProc` (die Fensterprozedur), `hInstance` (eine Instanz des Programms) und `lpszClassName` (der Klassenname) festlegt. Diese Klasse wird dann mit `RegisterClass` registriert, um sie im System zu verwenden.

Erstellung des Fensters

Nach der Registrierung der Fensterklasse wird das Fenster mit der Funktion `CreateWindowEx` erstellt. Diese Funktion benötigt mehrere Parameter, wie den Klassennamen, den Fenstertext, den Fensterstil und die Position und Größe des Fensters. Ist das Fenster erfolgreich erstellt, wird es mit `ShowWindow` angezeigt, basierend auf dem in `wWinMain` übergebenen Befehlszeilenparameter `nCmdShow`.

Nachrichtenschleife

Diese Schleife läuft kontinuierlich, solange das Programm ausgeführt wird und verarbeitet

alle Nachrichten (Events), die an das Fenster gesendet werden. Die Nachrichten werden mit `GetMessage` geholt, mit `TranslateMessage` übersetzt (zum Beispiel Tastendrücke) und schließlich mit `DispatchMessage` an die zuständige Fensterprozedur weitergeleitet.

Fensterprozedur
Die Fensterprozedur `WindowProc` ist eine Funktion in Windows-basierten Anwendungen, die alle Nachrichten verarbeitet, die an das Fenster gesendet werden. Sie reagiert auf eine Vielzahl von Nachrichten, die durch Benutzerinteraktionen oder das System ausgelöst werden. Die Fensterprozedur spielt eine Entscheidene Rolle für die interaktion des Benutzers mit dem Programm sowie für den Programmablauf selbst. Weshalb Folgend die für die Erstellung des Programms wichtigen Benutzeraktionen aufgelistet sind.
[8]

- **`WM_PAINT`**: Diese Nachricht wird gesendet, wenn der Teil des Fensters neu gezeichnet werden muss, beispielsweise nachdem es verdeckt war und nun wieder sichtbar ist. Die Behandlung dieser Nachricht erfolgt durch Aufrufen von `BeginPaint` und `EndPaint` und das Zeichnen erfolgt im dazwischenliegenden Block.
- **`WM_DESTROY`**: Diese Nachricht signalisiert, dass das Fenster geschlossen und die Anwendung beendet wird. `PostQuitMessage` wird in dieser Nachricht aufgerufen, um die Nachrichtenschleife zu beenden.
- **`WM_COMMAND`**: Wird verwendet, um Befehle Steuerelementen zu verarbeiten. Diese Nachricht wird gesendet, wenn der Benutzer eine Aktion ausführt, wie das Wählen eines Menüeintrags oder das Betätigen eines Buttons.
- **`WM_CREATE`**: Diese Nachricht wird nach der Erstellung des Fensters gesendet und wird verwendet, um das Erstellen von Steuerelementen durchzuführen.
- **`WM_CLOSE`**: Wird gesendet, wenn das Fenster über die Benutzeroberfläche oder durch einen Programm-Aufruf geschlossen wird. Diese Nachricht gibt die Möglichkeit, spezifische Ereignisse vor dem Schließen der Anwendung zu implementieren.
- **`WM_USER + 1`**: Dies wird genutzt um benutzerdefinierte Nachricht zu behandeln, die nicht durch Standard-Windows-Nachrichten abgedeckt sind.
- **`WM_TIMER`**: Diese Nachricht wird gesendet, wenn ein Timer-Intervall abläuft. Sie wird verwendet um periodische Ereignisse nach Ablauf eines Timers zu bearbeiten.

- **WM_MOUSEMOVE**: Diese Nachricht wird ausgelöst, sobald die Maus innerhalb der Grenzen des Fensters bewegt wird. Sie ermöglicht es der Anwendung, Mausbewegungen zu verfolgen und darauf zu reagieren.
- **WM_LBUTTONDOWN**: Gesendet, wenn der Benutzer die linke Maustaste innerhalb des Fensters drückt. Auf diese Nachricht kann im Anschluss benutzerspezifisch reagiert werden.

[8]

Die Nachrichten, die durch die Fensterprozedur `WindowProc` empfangen werden, werden mithilfe von `switch-case`-Anweisungen verarbeitet. Jeder Nachrichtentyp, wie `WM_PAINT`, `WM_DESTROY` oder `WM_COMMAND`, wird als ein separater Fall (`case`) innerhalb des `switch`-Blocks behandelt.
Im `switch`-Block prüft die Fensterprozedur den Nachrichtentyp, der durch den Parameter message der Funktion `WindowProc` übergeben wird. Abhängig von diesem Typ führt die Prozedur dann den spezifischen Code aus, der für die jeweilige Nachricht vorgesehen ist. [8]

Implementierung der Funktionen

Mittels der in Abschnitt 6.3.2 beschriebenen Funktionen erhält man bereits ein Programm welches ein leeres Fenster zeigt, dieses hat hierbei jedoch noch keinerlei Funktinonlität, die Erstellung der Funktionalität wird hier im folgenden beschrieben.
Als erstes wurde hierbei die Grafikelemente sowie Schaltflächen der GUI implementiert.

Erstellen der Grafikelemente

Zum einfügen der Grafischen Benutzerelemente in der GUI müssen diese als erstes erstellt werden. Hierbei wurden folgende Elemente in der GUI genutzt

- **Button** Er dient als Mechanismus für den Benutzer, eine Aktion auszulösen. Wenn der Benutzer auf den Button klickt, wird ein bestimmtes Ereignis oder ein Befehl ausgelöst. Hierbei kann der Button mit unterschiedlichen Beschriftungen gestaltet werden um dem Benutzer seine Eigenschaft besser zu verdeutlichen.
- **Combobox** Diese erstellt eine Dropdown-Liste von Optionen, aus der der Benutzer auswählen kann. Es kombiniert eine Textbox mit einer Liste, wobei der Benutzer eine Option aus der Liste auswählen kann. Diese Combobox wird hierbei für die Auswahl der Prozesse sowie des CO2 Koeffizienten genutzt.

Zur Erstellung dieser Elemente wird die Funktion `CreateWindowW` verwendet. Dabei werden als Eingabeparameter unter anderem der Typ der Schaltfläche, also ob es sich um eine Combobox oder einen Button handelt, vorgegeben. Des Weiteren werden die Position sowie die Größe der Schaltfläche als Eingabeparameter übermittelt. Um Aktionen, die innerhalb dieser Schaltfläche durch den Nutzer getätigt werden, zuordnen zu können, wird jeder Schaltfläche eine eindeutige ID zugewiesen. Da die Erstellung der Grafikelemente direkt beim starten des Programms geschehen soll, wird die Funktion `CreateWindowW` hierbei in der Benutzeraktion `WM_CREATE` aufgerufen.
Um die dem Nutzer anzuzeigenden Werte in der GUI darzustellen, wurde die Funktion `DrawValues()` implementiert. Innerhalb dieser Funktion werden die berechneten Werte durch String-Operationen mit den entsprechenden Einheiten und Bezeichnungen versehen, die dem Nutzer angezeigt werden sollen. Die formatierten Strings werden dann in der GUI positioniert, wofür die Funktion `TextOutW()` verwendet wird. Die `TextOutW()` Funktion ist ein Bestandteil der Windows-API und dient dazu, Strings an einer spezifizierten Position innerhalb des Fensters der GUI zu platzieren sowie anzuzeigen.
Die Platzierung der Elemente Erfolg hierbei durch die Zuweisung von x-y-Koordinaten wobei der Nullpunkt hierbei in der oberen linken Ecke der GUI liegt.

Die interaktiven Elemente der Benutzeroberfläche , die es dem Benutzer ermöglichen, mit der GUI zu interagieren, werden durch das Ereignis `WM_COMMAND` verarbeitet. Innerhalb dieses Ereignisses werden die den Elementen zugewiesenen IDs über `case`-Anweisungen abgefragt. Wenn der Nutzer ein solches Element anklickt, wird die für dieses GUI-Element programmierte Aktion ausgeführt.
Ein praktisches Beispiel hierfür ist die Schaltfläche, die dafür vorgesehen ist, alle aktuell auf dem Computer laufenden Prozesse anzuzeigen. Wenn der Nutzer auf diese Schaltfläche klickt, wird die Funktion `getRunningProcesses()` aufgerufen. Diese Funktion sorgt dafür, dass dem Nutzer ein stets aktuelles Abbild aller laufenden Prozesse bereitgestellt wird.

Das Anzeigen der GUI wird hierbei über das Ereignis `WM_USER+1` gesteuert. Dabei werden zwei Ereignisse abgefragt: `WM_MOUSEMOVE` und `WM_LBUTTONUP`. Über `WM_MOUSEMOVE` wird ermittelt, ob sich der Cursor über dem Icon befindet. Ist dies der Fall, wird die GUI angezeigt. Verlässt die Maus den Bereich der GUI, wird diese wieder versteckt. Durch `WM_LBUTTONUP` wird überprüft, ob das Icon mit der linken Maustaste angeklickt wurde. In diesem Fall erscheint das Icon dauerhaft, bis die GUI aktiv geschlossen wird. Für das Anzeigen und Verstecken der GUI wird dabei die Funktion `ShowWindow()` mit den Parametern `SW_SHOW` und `SW_HIDE`

verwendet.

Auswahl von Prozessen

Um dem Nutzer über die oben beschriebene Combobox eine Liste aller momentan auf dem Rechner laufenden Prozesse anzeigen zu können, wurde die Funktion `getRunningProcesses()` implementiert.
Die Hauptaufgabe dieser Funktion ist es, die Namen aller aktuell laufenden Prozesse zu erfassen, in einem geeigneten Datentyp zu speichern und diese als Rückgabewert zurückzugeben. Zur Realisierung dieser Funktionalität wird auf Funktionen der `Kernel32.lib` Bibliothek zurückgegriffen.

Zur Speicherung der einzelnen Prozessnamen werden folgende Funktionen aufgerufen:

- **`CreateToolhelp32Snapshot(TH32CS_SNAPPROCESS, 0)`**: Diese Funktion erstellt eine Momentaufnahme aller aktuell laufenden Prozesse sowie der Heaps, Module und Threads im System. Sie gibt ein geöffnetes Handle an diese Momentaufnahme zurück. Das Flag `TH32CS_SNAPPROCESS` wird spezifiziert, um sicherzustellen, dass die Momentaufnahme alle Prozesse im System umfasst. [6]
- **`Process32FirstW(hSnapshot, &pe32)`**: Diese Funktion wird verwendet, um Informationen über den ersten Prozess zu erhalten, der in der Systemmomentaufnahme gefunden wurde. Sie initialisiert die Struktur `pe32` mit den Daten des ersten Prozesses und gibt bei Erfolg TRUE zurück, was den Beginn des Durchlaufs durch die Prozessliste signalisiert. [7]
- **`Process32NextW(hSnapshot, &pe32)`**: Nachdem der erste Prozess abgerufen wurde, wird diese Funktion wiederholt aufgerufen, um Informationen zum nächsten Prozess in der Momentaufnahme zu erhalten. Für jeden weiteren Prozess aktualisiert sie die Struktur `pe32` und gibt TRUE zurück, solange weitere Prozesse in der Liste vorhanden sind. [13]

Hiermit wurde nun die Möglichkeit geschaffen, ein Abbild aller Namen, die für die Zeit des Snapshots zu erzeugen. Hierbei enden alle Prozesse mit der Endung `.exe`, dies ist ein Indikator dafür, dass es sich um ein ausführbares Programm handelt. Dies ist für die Darstellung der Programmnamen innerhalb der Combobox nicht weiter störend, da der Nutzer somit klar sieht, dass es sich um ein Programm handelt. Jedoch erlaubt die Syntax der Windows API, welche für das Erstellen der Abfrage der CPU-Auslastung zuständig ist, nicht, dass diese Endung verwendet wird. Für das Erstellen der Abfrage muss die Endung `.exe` somit entfernt werden.

Da die Prozesse, die über die Snapshot-Funktion ausgelesen wurden, nicht sortiert sind, ist hier noch eine alphabetische Sortierung der Namen durchzuführen. Dies erlaubt es dem Nutzer innerhalb der Combobox gezielt nach einzelnen Programmen suchen zu können.

Auslesen der Werte für Prozesse

Das Verfahren zum Auslesen der CPU-Auslastung für einzelne Prozesse wurde bereits in Abschnitt 6.1 dargelegt. Da das Überwachen von Prozessen durch den Nutzer aktiv gesteuert werden soll, muss der Name des Prozesses hierbei in die Abfrage integriert werden. Da die Abfrage den Counter, welcher initialisiert werden soll, als String verarbeitet, wird über folgenden Code der Name des Prozesses, der durch den Nutzer ausgewählt wird, in den Abfragestring eingefügt.

```
sprintf(counterPath, "\\Prozess(%s)\\Prozessorzeit (%%)", ProzessName)
```

Der zeitliche Versatz, welcher zur Berechnung der prozentualen CPU-Auslastung nötig ist, kann hierbei jedoch nicht wie in Abschnitt 3.3.1 über das Implementieren einer `sleep()`-Funktion implementiert werden. Dies würde den Thread des Programms stoppen, auf welchem die gesamte GUI läuft und somit wären für die Zeit, in welcher der Thread pausiert, keine Aktionen mit der GUI möglich. Der zeitliche Abstand wird hierbei durch das Verwenden von Timern realisiert, welche anfangs initialisiert werden. Die Abfrage des Timerereignisses erfolgt hierbei in der Benutzeraktion `WM_TIMER`, wobei das Timerereignis hierbei über die dem Timer zugewiesene ID zugeordnet wird.
Um die Möglichkeit zu schaffen, mehrere Prozesse als Gruppe betrachten zu können, wird hierfür für jeden zu überwachenden Prozess ein eigener Counter benötigt. Diese Counter werden in einem Vektor vom Typ HCOUNTER initialisiert, sodass mehrere Prozesse gleichzeitig erfasst werden können. Mittels einer `for`-Schleife wird durch die Liste der Prozesse in der Textdatei iteriert, um jeden Prozess der Counter-Abfrage hinzuzufügen. Das Auslesen der Werte für die Prozesse erfolgt dann analog zum Vorgehen bei einem einzelnen Prozess. Es wird erneut eine Schleife verwendet, um durch die einzelnen Counter zu iterieren und die Werte für jeden Prozess abzufragen. Um die Gesamt-CPU-Auslastung, die durch diese Prozesse verursacht wird, zu berechnen, werden die individuellen Auslastungswerte addiert. Das resultierende Gesamtergebnis wird anschließend durch die Anzahl der ermittelten logischen Kerne des Systems geteilt, um eine korrekte Durchschnittsauslastung zu erhalten. Dies ist sowohl für den Bereich mehrerer Prozesse als auch für nur einen Prozess nötig.

Berechnen der Anzeigewerte

Die Berechnung der Werte welche dem Nutzer angezeigt werden, werden mittels mehrerer

Funktionen implementiert. Die aktuelle Stromaufnahme wird durch die in 5 erstellten Regressionsgeraden berechnet. Hierbei wurden die mathematischen Funktionen im Programm implementiert. Da es sich bei der hier verwendeten Funktion um eine abschnittsweise definierte Funktion handelt, wird mittels Abfrage der CPU-Auslastung unterschieden, mit welchem Funktionsabschnitt die Stromaufnahme berechnet werden soll. Die geleistete Arbeit wird hierbei wie in Abschnitt 6.2 berechnet, hierbei wird die Formel 6.1 verwendet. Um eine dem Nutzer gerechte Aktualisierung zu gestalten, sind hierbei zeitliche Abstände der Aktualisierung zu beachten. Da die Daten jedoch trotzdem sekündlich abgefragt werden, werden diese über die Anzahl der Abfragen, welche zwischen den einzelnen Visualisierungsupdates vergehen, gemittelt, wobei der arithmetische Mittelwert genutzt wird. Die Berechnung dieser Werte kann hierbei direkt innerhalb des Timer-Events implementiert werden, welches für die Abstände des Sammelns der Datenstichproben genutzt wird.

Das erstelle TrayIcon ist hierbei abschließend in Abschnitt 9 als Grafik 9.1 abgebildet.

7 Diskussion

Folgender Abschnitt dient dazu die Arbeit sowie die angewandten Methoden abschließend zu Bewerten und zu diskutieren.

7.1 Bewertung der Methodik und der Ergebnisse

Die Messungen des Stromverbrauchs haben gezeigt, dass man anhand der CPU Auslastung den Stromverbrauch eines Rechners ermitteln kann. Die sehr unterschiedlichen Ergebnisse für die einzelnen Rechner zeigen jedoch, dass der Stromverbrauch für einen Rechner zwar durch die CPU Ermittelt werden kann, dies jedoch sehr individuell für einen Rechner ist. Das Modell der Regression für den Laptop ist somit nicht übertragbar auf einen anderen Rechner mit unterschiedlicher Hardware.
Die Methode CPU Auslastung durch externe Programme zu erzeugen ist eine gute Möglichkeit den Rechner in einer kontrollierten weise zu belasten um genaue Messwerte zu erhalten, jedoch sind die Abstufungen für die einzelnen Abschnitte der CPU Auslastung bei der verwendeten Software sehr hoch, hierbei wäre eine Software welche es ermöglicht die CPU Auslastung genauer zu steuern hilfreicher zur Erstellung eines besser angepassten Regressionsmodells.

Die Idee dem Nutzer die Ergebnisse nach dem Prinzip des User-in-the-Loop mittels eines Trayicons zu Präsentieren ist hierbei ein Guter Ansatz, da dieses Kontinuierlich Informationen liefern kann ohne den Nutzer hierbei bei der normale Tätigkeit am Rechner zu stören.

7.2 Diskussion der Bedeutung der Ergebnisse für die Praxis

Mittels der Rückführung der CPU-Auslastung auf den Stromverbrauch des Computers sowie der grafischen Darstellung dieser wird es dem Nutzer ermöglicht, den Effekt der von ihm genutzten Programme auf die Umwelt sowie auf den Stromverbrauch zu erkennen. Die Möglichkeit, einzelne Prozesse einem Stromverbrauch zuzuordnen, ist dahingehend sinnvoll, da Programme, die im Hintergrund und ohne Wissen des Benutzers ausgeführt werden, oft nur eine geringe CPU-Auslastung und somit einen geringen Stromverbrauch verursachen. Betrachtet man dies

jedoch auf die Gesamtzahl der Rechner, die in Deutschland sowohl privat als auch beruflich genutzt werden, so sind diese im Gesamtbild für einen hohen Verbrauch von Ressourcen verantwortlich.

7.3 Vorschläge für zukünftige Arbeiten

In dieser Arbeit wurde bereits demonstriert, dass es möglich ist, den Stromverbrauch eines Computers auf Basis der CPU-Auslastung zu ermitteln. Für zukünftige Forschungen wäre es interessant, die entwickelten Modelle um zusätzliche Faktoren zu erweitern. In dieser Arbeit wurde die CPU-Frequenz für das Betriebssystem Windows manuell abgelesen, da derzeit keine Möglichkeit besteht, diese programmatisch auszulesen. Da jedoch anzunehmen ist, dass die CPU-Frequenz einen bedeutenden Einfluss auf den Stromverbrauch hat, wäre ein weiterführender Ansatz, eine Methode zu entwickeln, um die Frequenz programmatisch zu erfassen oder zu berechnen.
Unter Linux ist das Auslesen der CPU-Frequenz direkt möglich, was einen vielversprechenden Ansatz bietet, das Modell durch die Integration der Frequenzen zu erweitern und so eine genauere Näherung des Stromverbrauchs durch die Regression zu ermöglichen.

Abseits des direkten Stromverbrauchs durch den Rechner gibt es in der Informations- und Kommunikationstechnik (IKT) noch viele Bereiche, in denen Energie verbraucht wird, ohne dass sich ein Großteil der Nutzer dessen bewusst ist. Ein bedeutender Aspekt ist hierbei die Nutzung des Internets, beispielsweise durch Streamingdienste, die einen hohen Energieverbrauch und daraus resultierenden CO2-Ausstoß verursachen. Ein möglicher Ansatz könnte darin bestehen, mittels der von Windows bereitgestellten APIs die Nutzung von Datenvolumen auszulesen und anhand dieser Daten den CO2-Ausstoß zu ermitteln, der dadurch entstanden ist. Diese Information könnte in die in dieser Arbeit erstellte grafische Benutzeroberfläche integriert werden, um dem Nutzer einen weiteren Aspekt der CO2-Emissionen, die durch sein Verhalten verursacht werden, aufzuzeigen.

8 Fazit und Ausblick

Mittels der angewandten Methoden wurde gezeigt, dass sich die CPU als zuverlässiger Indikator für den Stromverbrauch eines Computers heranziehen lässt. Die Möglichkeit einer grafischen Darstellung der Ergebnisse ist hierbei ein erster Schritt, dem Nutzer dahingehend zu sensibilisieren, dass Software ebenfalls einen Beitrag zum Klimawandel leistet und diese nicht nur ein abstraktes Objekt auf einem Rechner ist.
Aufbauend auf dieser Arbeit ist es interessant, die Auswirkungen einzelner Programme auf den CO2-Verbrauch genauer zu untersuchen. Hierbei müsste man über einen längeren Zeitraum hinweg einzelne Programme betrachten sowie Programme, die in gewisse Kategorien einteilbar sind, genauer untersuchen.
Beispielsweise ist das Einteilen von Programmen, welche durch das Betriebssystem Microsoft automatisch ausgeführt werden, sinnvoll, um über einen langen Zeitraum hinweg eine Betrachtung zu ermöglichen, die aufzeigt, wie hoch der CO2-Ausstoß allein durch das Betriebssystem ist.
Ansätze hierfür wurden bereits in dieser Arbeit geschaffen, indem das Programm ermöglicht, mehrere Programme gesammelt zu betrachten.
Die erstellte GUI stellt hierbei lediglich eine erste Idee dar, diese kann in Hinsicht auf Nutzerfreundlichkeit sowie Funktionen ergänzt werden. Hierbei wäre beispielsweise das Einbringen des jeweils aktuellen Strompreises nützlich, um dem Nutzer auch einen Preis, welchen dieser selbst zahlen muss, aufzuzeigen.

Abschließend lässt sich sagen, dass bei aller Anstrengung, einen Menschen zu einem umweltbewussten Leben zu bewegen, es am Ende der Mensch selbst sein muss, der die Wichtigkeit seiner Handlungen erkennt.

9 Anhang

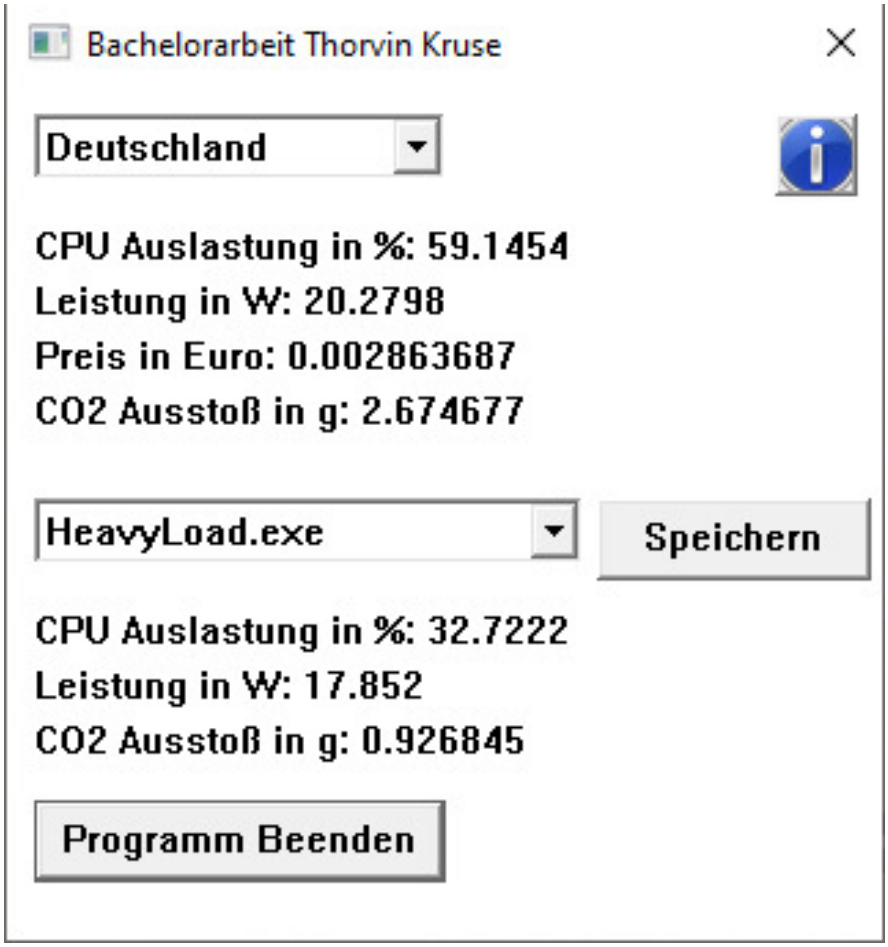

Abbildung 9.1: Abbildung des TrayIcons

Abbildung 9.2: Abbildung der Box für die Messung am Desktop PC

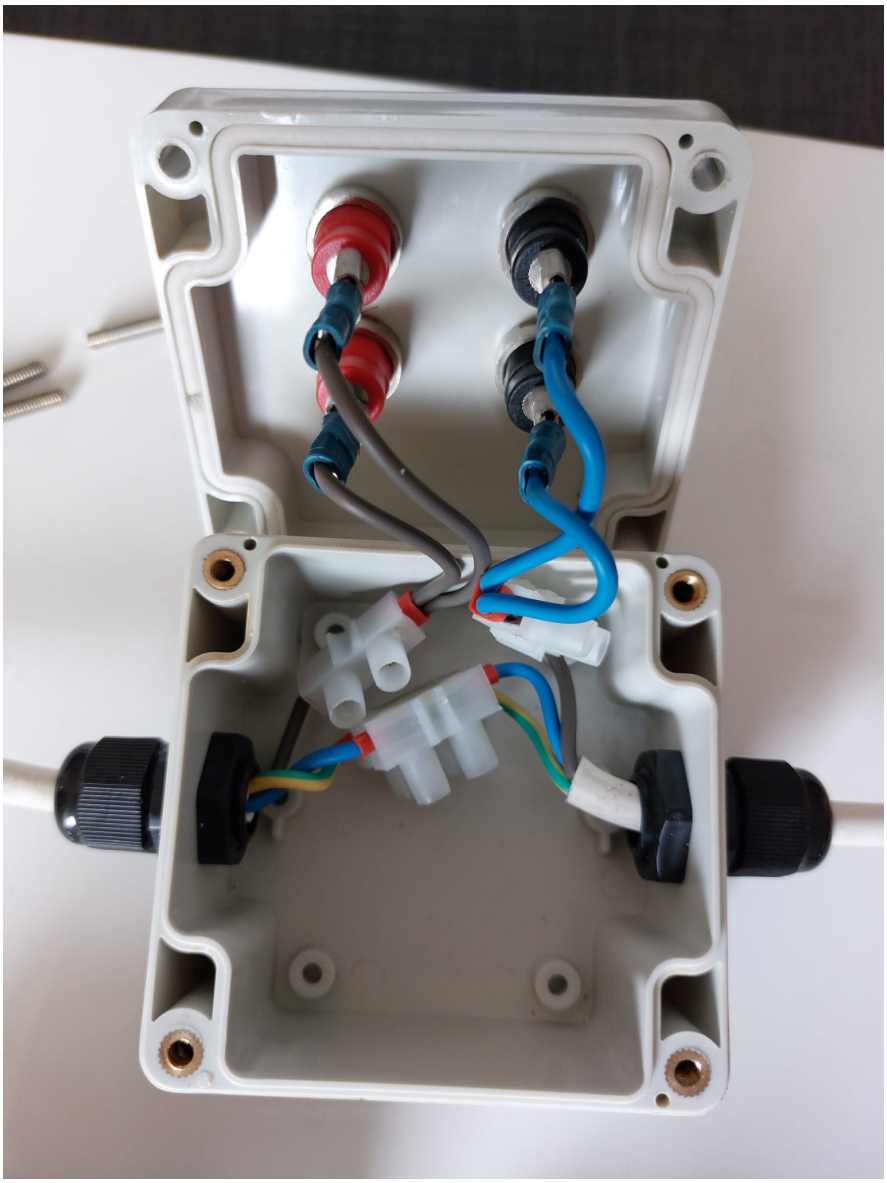

Abbildung 9.3: Abbildung des internen Aufbaus der Box für die Messung am Desktop PC

Literatur

[1] Archiveddocs. “PERF_100nsec_timer: Core services”. (8. Okt. 2009), Adresse: https://learn.microsoft.com/en-us/previous-versions/windows/it-pro/windows-server-2003/cc728274(v=ws.10) (besucht am 03. 04. 2024).

[2] J. Wolf, *Linux-UNIX-Programmierung Das umfassende Handbuch*, 3. Aufl. Galileo Computing, 2009, 1247 S., ISBN: 978-3-8362-1366-0.

[3] L. Stobbe, M. Proske, H. Zedel, R. Dr. Hintemann, J. Clausen und B. Severin, *Entwicklung des IKT-bedingten Strombedarfs in Deutschland*, 18. Nov. 2015. (besucht am 24. 03. 2024).

[4] I. Frost, *Einfache lineare Regression*. SpringerVS, 2018, 35 S., ISBN: 978-3-658-19732-2.

[5] S. Chamberlin. “Measuring your application power and carbon impact (part 1)”, Sustainable Software. (14. Sep. 2020), Adresse: https://devblogs.microsoft.com/sustainable-software/measuring-your-application-power-and-carbon-impact-part-1/ (besucht am 21. 05. 2024).

[6] karl-bridge-microsoft. “CreateToolhelp32snapshot function (tlhelp32.h) - win32 apps”. (13. Okt. 2021), Adresse: https://learn.microsoft.com/en-us/windows/win32/api/tlhelp32/nf-tlhelp32-createtoolhelp32snapshot (besucht am 07. 05. 2024).

[7] karl-bridge-microsoft. “Process32firstw function (tlhelp32.h) - win32 apps”. (9. Aug. 2022), Adresse: https://learn.microsoft.com/en-us/windows/win32/api/tlhelp32/nf-tlhelp32-process32firstw (besucht am 07. 05. 2024).

[8] Karl-Bridge-Microsoft. “Verwenden von Fensterprozeduren - Win32 apps”. (13. Juni 2023), Adresse: https://learn.microsoft.com/de-de/windows/win32/winmsg/using-window-procedures (besucht am 26. 03. 2024).

[9] Xelu86. “typeperf”. (15. Juni 2023), Adresse: https://learn.microsoft.com/de-de/windows-server/administration/windows-commands/typeperf (besucht am 21. 05. 2024).

[10] *Collecting Performance Data*, 23. Apr. 2024. Adresse: https://learn.microsoft.com/en-us/windows/win32/perfctrs/collecting-performance-data.

[11] jwmsft. "SetWindowsHookExA-Funktion (winuser.h) - Win32 apps". (13. März 2024), Adresse: https://learn.microsoft.com/de-de/windows/win32/api/winuser/nf-winuser-setwindowshookexa (besucht am 21. 05. 2024).

[12] Karl-Bridge-Microsoft. "Info zu Window-Prozeduren - Win32 apps". (8. Jan. 2024), Adresse: https://learn.microsoft.com/de-de/windows/win32/winmsg/about-window-procedures (besucht am 26. 03. 2024).

[13] karl-bridge-microsoft. "Process32nextw function (tlhelp32.h) - win32 apps". (22. Feb. 2024), Adresse: https://learn.microsoft.com/en-us/windows/win32/api/tlhelp32/nf-tlhelp32-process32nextw (besucht am 07. 05. 2024).

[14] Y. Pavel, I. Alex, R. Mark und S. David, *Windows Internals*, 1. Aufl., 2 Bde. dpunkt.verlag, 23. Mai 2024, Bd. 1, ISBN: 978-3-86490-538-4.

[15] *PdhAddCounterA function*, 23. Apr. 2024. Adresse: https://learn.microsoft.com/en-us/windows/win32/api/pdh/nf-pdh-pdhaddcountera.

[16] *PdhCloseQuery function*, 23. Apr. 2024. Adresse: https://learn.microsoft.com/en-us/windows/win32/api/pdh/nf-pdh-pdhclosequery.

[17] *PdhCollectQueryData function*, 23. Apr. 2024. Adresse: https://learn.microsoft.com/en-us/windows/win32/api/pdh/nf-pdh-pdhcollectquerydata.

[18] *PdhGetFormattedCounterValue function*, 23. Apr. 2024. Adresse: https://learn.microsoft.com/en-us/windows/win32/api/pdh/nf-pdh-pdhgetformattedcountervalue.

[19] *PdhOpenQueryA function*, 23. Apr. 2024. Adresse: https://learn.microsoft.com/en-us/windows/win32/api/pdh/nf-pdh-pdhopenquerya.

[20] *PERF_100NSEC_TIMER_INV*, 23. Apr. 2024. Adresse: https://learn.microsoft.com/en-us/previous-versions/windows/it-pro/windows-server-2003/cc757283(v=ws.10).

[21] *polyfit*, 23. März 2024. Adresse: https://de.mathworks.com/help/matlab/ref/polyfit.html.

[22] *polyval*, 23. März 2024. Adresse: https://de.mathworks.com/help/matlab/ref/polyval.html.

[23] "Carbon intensity of electricity generation", Our World in Data. (), Adresse: https://ourworldindata.org/grapher/carbon-intensity-electricity?tab=chart&country=EU-27~OWID_WRL (besucht am 22. 05. 2024).

[24] "proc(5) - Linux manual page". (), Adresse: https://man7.org/linux/man-pages/man5/proc.5.html (besucht am 26. 03. 2024).

[25] R. Schoenen, G. Bulu, A. Mirtaheri und H. Yanikomeroglu. "User in the loop". (), Adresse: https://userintheloop.org/ (besucht am 21. 05. 2024).

[26] V. I. Turbo-Boost-Technik, u. d. P. z. b. W. erklären um deinen und W. D. Geht. "Was ist Intel® Turbo-Boost-Technik?", Intel. (), Adresse: https://www.intel.com/content/www/de/de/gaming/resources/turbo-boost.html (besucht am 21. 05. 2024).

Hiermit versichere ich, dass ich die vorliegende Arbeit ohne fremde Hilfe selbständig verfasst und nur die angegebenen Hilfsmittel benutzt habe.

Hamburg, 24. Mai 2024 Thorvin Kruse