Anwendung von KI und ML zur Messdatenverarbeitung aus Energieversorgungskomponenten

Bachelorarbeit

Sergej Lamert - 727245
Fakultät angewandte Informatik

24. Dezember 2020

Inhaltsverzeichnis

Abbildungsverzeichnis

1 Einleitung

Buck Wandler sind heutzutage in aller Munde, sie stellen beispielsweise Spannung in Notebook Prozessoren und Ladegeräten zu verfügung, Regeln den Strom an Stepper-Motoren. Durch die weitläufigen Anwendungsfälle, ist es besonders in der heutigen Zeit, in der Künstliche Intelligenz und Machine Learning Algorithmen immer wichtiger werden, zu evaluieren, inwieweit sich solche Wandler für Machine Learning Anwendungen eignen, wenn Daten über Strom, Spannung, Leistung und Temperatur bekannt sind. Deshalb ist das Thema dieser Arbeit die Analyse eines Hybrid-Buck-Wandlers. Im Folgenden wird solch ein Wandler hinsichtlich seiner Anwendbarkeit für Themen im Bereich Maschine Learning und Zustandsüberwachung analysiert werden. Die Analyse erfolgt dabei in mehreren Schritten. Zuerst werden Daten erhoben, während der Wandler an unterschiedliche statische Lasten angeschlossen ist, um einen Überblick über das Verhalten des Wandlers zu gewinnen. Anschließend wird der Wandler an eine dynamische Last angeschlossen und es werden periodische Signale gemessen. Die gemessenen Daten werden dann in Echtzeit mit bereits bekannten Daten verglichen und ebenfalls gegen andere, parallel laufende Wandler, abgestimmt. Das Letztendliche Ziel dieser Arbeit ist es, zu evaluieren, wie präzise und zuverlässig die Daten sind und inwieweit diese sich für Anwendungen in den am Anfang genannten Bereichen eignen. Da der vorliegende Wandler über einen Mikrocontroller mit I2C Schnittstelle verfügt, aus welchem Strom, Spannungs und Temperaturdaten erhoben werden können, wird das Erfassen der Daten durch das vorhandene Protokoll stark vereinfacht und sehr portierbar gemacht, was es ermöglicht die Daten aus beliebigen Platformen (Win 10, Linux, etc.) oder einfach durch andere Mikrocontroller (z.B. Raspberry Pi) auszulesen.

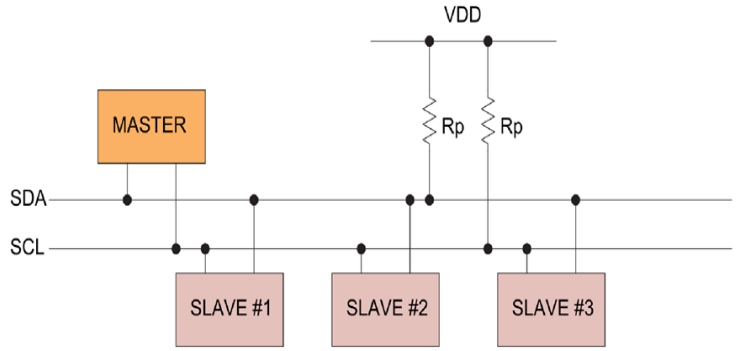


Abbildung 1: I2C Bus Beispiel Quelle: https://www.analog.com/en/technical-articles/i2c-primer-what-is-i2c-part-1.html

2 Theoretischer Hintergrund

Dadurch bedingt, das diese Arbeit sich an vielen Konzepten aus der Signaltheorie, Elektrotechnik und Informatik bedient, ist es sinnvoll vorab einige Begriffe und Konzepte zu definieren. Aus diesem Grund, werden im Folgenden das I²C Protokoll, verwendete Software, sowie die genutzte Hardware näher erläutert werden.

2.1 Grundlagen zum verwendeten Hybrid-Buck-Wandler

2.2 Grundlagen I2C

I2C steht für Inter-Integrated Circuit und wurde von Philips Semiconductors 1982 entwickelt. Es handelt sich dabei um einen seriellen Datenbus im Master-Slave Stil. Dieses Protokoll setzt auf zwei Leitungen für den Datenaustausch, davon ist ein Kanal explizit für die Daten vorgesehen und der andere Kanal für den Takt. Der Takt beträgt im klassischen 100 KHz oder 400 KHz.
Es ist zu erkennen, dass es eine Master -und mehrere Slave Komponenten gibt. Der Master gibt allen Slaves vor, was sie zu senden haben, und wie schnell sie es tun sollen. Des Weiteren ist zu erkennen, das die beiden Leitungen SDA und SCL an einem Pull-Up Widerstand angeschlossen sind. Dieser dient dazu, die Leitungen, wenn weder Master noch Slave sendet, auf die Versorgungsspannung zu schalten, sodass keine undefinierten Zustände und die Leitung im unbenutzten Zustand eine logische 1 besitzt. Die Kommunikation erfolgt durch Adressen, so besitzt z. B. jeder Sensor eine I2C Slave Adresse, über die ein Master auf diese zugreifen kann. Das auslesen eines Slaves erfolgt per Registeradressen, so hat beispielsweise ein beliebiger I²C fähiger Sensor ein Register, in dem bestimmte Werte

gespeichert sind. In dieser Arbeit handelt es sich bei dem I2C Slave um einen Mikrocontroller auf dem Buck-Wandler. Dieser Slave ist in der Lage Daten zum Strom der Eingangs -und Ausgangsspannung und Temperatur auszugeben. Der Master ist dabei entweder ein PC, welcher per PicKit (I2C auf USB) damit kommuniziert, oder ein Raspberry Pi, welcher direkt per I2C mit dem Netzteil verbunden werden kann.

2.3 Hintergrund der verwendeten Platformen, Programmiersprachen und Bibliotheken

Für die statischen Tests wurde Windows 10 als Platform und C als Programmiersprache gewählt, da allein diese mit dem PICkit kompatibel sind.
Für die dynamischen Tests wurde ein Raspberry Pi 4 (4 GB) mit der Programmiersprache Python für Datenerfassung- und verarbeitung verwendet.

2.4 Grundlagen Neuronale Netze

Als Klassifizierungsalgorithmus wird ein simples Neuronales Netz verwendet. Neuronale Netze gehören zur Gruppe vom SSupervised Learning"(dt. "Betreutes Lernen"). Das Konzept von diesem Algorithmus ist es, dem Model einen Datensatz und die dazugehörige Lösung zu geben. Ausgehen davon passt das Neuronale Netz seine Parameter an. Dieser Vorgang wird im Folgenden formell erläutert.
Zur durchführung von einigen simplen Testfällen unter Punkt 3.3 ist es notwendig eines der Grundlegenden Konzepte von Machine Learning zu nennen: Neuronale Netze. Neuronale Netze sind in ihrem Grundgedanken dem Konzept des menschlichen Gehirns nachempfunden jedoch in vielerlei hinsicht anders. In Abb. 2 erkennt man ein Diagram eines Neuronalen netzes. Die Kreise repräsentieren die Neuronen, die Kante stellen Verbindungen unter den Neuronen dar. Es ist zu beachten, das in diesem Model die Neuronen von einer Schicht N vollvermascht mit den Neuronen einer Schicht N+1 sind. Des Weiteren ist zu erkennen, dass der dargestellte Graph drei Schichten (engl. Layers) besitzt. Der Input Layer dient dabei als Ausgangspunkt, an dieser Stelle wird in jedes Neuron ein numerischer Wert eingespeist. Alle Werte im Input Layer zusammengenommen ergeben einen Vektor V.
Nachfolgend kommt das Hidden Layer ("Versteckte Schicht"), durch Matrixmultiplikation vom Input Vektor und der Gewichtungsmatrix wird der neue Vektor für das Hidden Layer berechnet. Es zeigt sich, dass durch einbringen eines Hidden Layers nichtlinea berechnet werden (konkretisieren + Quelle angeben)
Abschließend erfolgt eine weitere Matrixmultiplikation des Hidden Layers

und seinen Weights, um das Output Layer zu generieren. Das Output Layer gibt, wie der Name schon sagt, die Ausgangswerte der Berechnung. Bei Klassifizierungsaufgabe sind die Werte im Output Layer Warscheinlichkeiten für die jeweiligen, zu klassifizierenden Klassen. Die Anzahl der Neuronen im Output Layer ist gleich der Anzahl der Klassen

In Abb. x sind Neuronen durch die leeren Kreise abgebildet, im Detail sieht ihre Struktur folgendermaßen aus:

Die Formel für Feedforward:

Die Formel für die Sigmoid Funktion:

$$s(z) = \frac{1}{1 + e^{-z}} \tag{1}$$

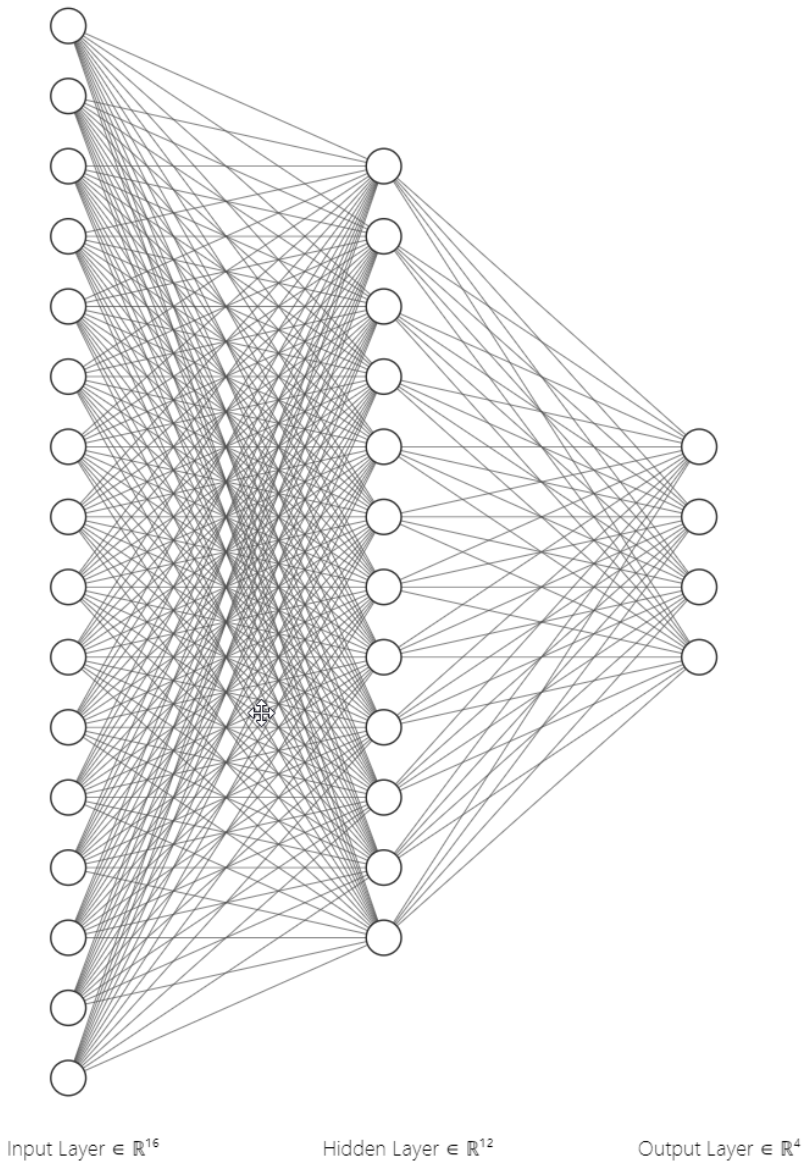


Abbildung 2: Darstellung eines Neuronalen Netztes

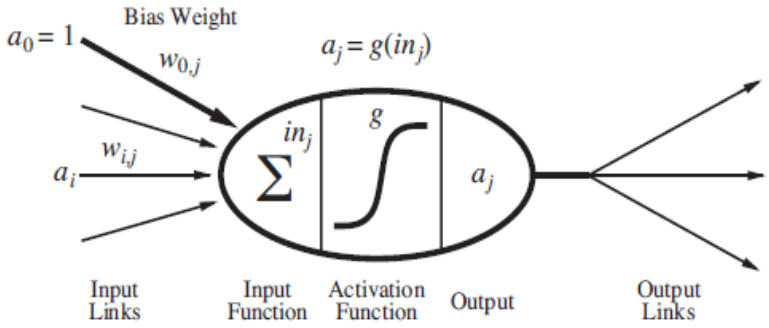


Abbildung 3: Darstellung eines Neurons Artifical Intelligence P. 728

2.5 Statische Datenerfassung

Die statische Datenerfassung dient dem Zweck, die Funktionalität der Wandler unter einer konstanten Last zu überprüfen. Es wurden zwei simple Testfälle aufgebaut. Bei beiden handelt es sich um Parallelschaltungen von 4R7 Ohm Widerständen. Der erste Fall schaltet zwei davon parallel, der zweite drei. Durch das Anwenden der Ohmschen Gesetzes, kann ein theoretischer Stromfluss und somit auch die theoretische Leistungsaufnahme berechnet werden. Der Wandler wird durch ein Netzteil mit 24 Volt Ausgangsspannung betrieben. Diese 24 Volt werden von dem Wandler auf 12 V herabgesetzt. Somit kann der Strom berechnet werden, der durch die Widerstände fließt:

$$\frac{U}{R} = I \tag{2}$$

$$\frac{12V}{4,7Ohm} * 2 = 5,10A \frac{12V}{4,7Ohm} * 3 = 7,65A$$

(3)

Da es sich in diesen Testfällen um statische Lasten handelt, sind keine hohen Abtastraten erforderlich, deshalb wird zur Datenerfassung ein PicKit mitsamt I^2C Schnittstelle verwendet und kein Raspberry Pi.
Des Weiteren ist zu erwähnen, dass mehrere Wandler getestet wurden, einer davon mit welcher fehler. Bei der Messung der Daten, zeigt sich, dass sowohl Temperatur als auch Ausgangsstrom, dieses Wandlers unter gleicher Last höher waren als die der anderen, was man in Abb. X erkennen kann

2.6 Dynamische Datenerfassung

Das Ziel der dynamischen Datenerfassung ist es, zu evaluieren, wie der Wandler sich unter Lasten verhält, die sich mit der Zeit ändern. Sowie mögliche Abweichungen zwischen den einzelnen Wandlern selbst. Der Testaufbau beinhaltet den Hybridwandler, einen Stereoverstärker, welcher per Wandler gespeist wird und in den ein Audiosignal eingespeist wird. Als Signal wird hierbei zu begin ein Sinussignal verwendet, welches mit dem Programm Audacity generiert wird (https://www.audacityteam.org/). Der erste Testschritt ist dabei die Überprüfung, wie hoch die maximale Frequenz eines Signals sein kann, sodass kohärente Daten erzeugt werden können. Die maximale Frequenz kann unter berücksichtigung der Nyquist

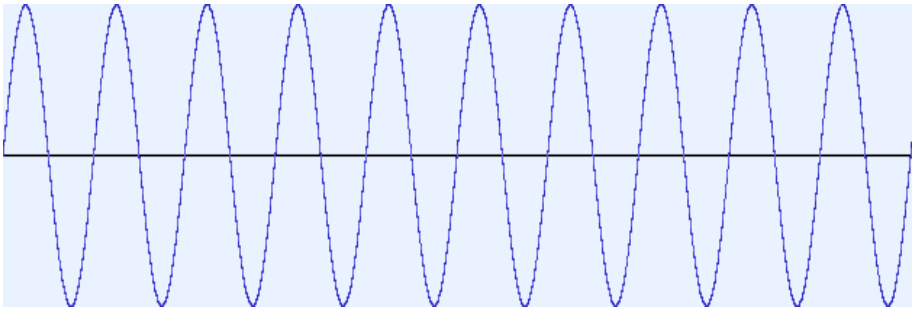


Abbildung 4: In Audacity generiertes 10 Hz Sinussignal

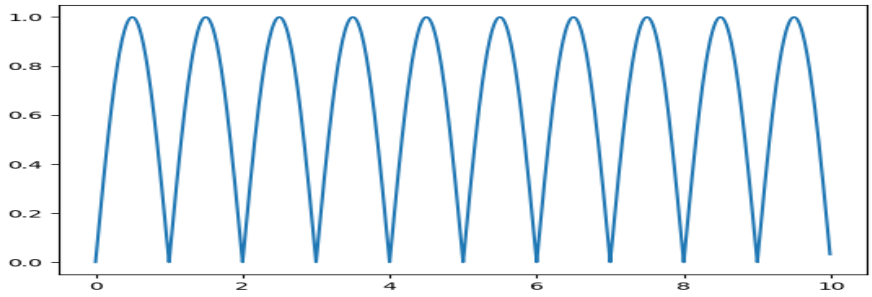


Abbildung 5: Effektive Frequenz eines 10 Hz Signals ist 20 Hz, weil, weil es für den Verstärker egal ist ob + oder - Signal

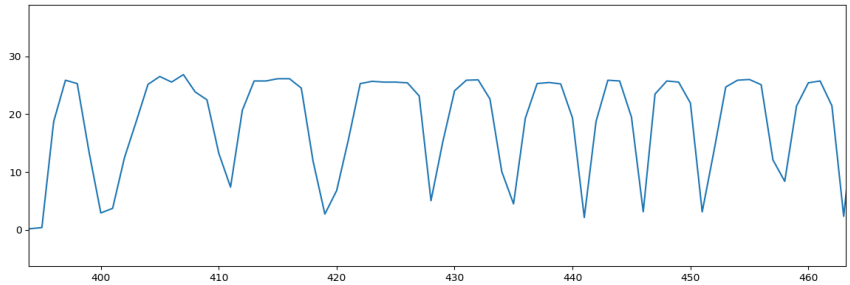


Abbildung 6: Tatsächlich gemessene Leistung des Wandlers

Frequenz theoretisch hergeleitet werden oder durch praktische Tests empirisch bestimmt werden.
In Abb. 6 ist der ungefilterte Leistungsverlauf des Wandler dargestellt. Es ist ersichtlich, dass das 10 Hz Sinussignal in Abb. 5, dem gemessenen Sinussignal entspricht. Der unsaubere Verlauf der Leistung kann auf Rauschen innerhalb der Hardware und auf die Abtastrate zurückzuführen sein.

$$f_{\text{nyquist}} = 1/2 * f_{\text{abtastrate}} \tag{4}$$

Die Abtastrate kann berechnet werden, da die Parameter des I²C Protokolls bekannt sind. Zeitliche Abweichungen die innerhalb des Codes entstehen sollen hier vernachlässigt werden.
Es werden pro Parameter 2 Byte an Daten angefragt, somit ist pro Transfer eine Bitmenge von 1+8+1+8+1+8+1+1 = 29 bits erforderlich Bei einer Taktrate von 80 KHz ist das folglich eine Bitrate von T = 1 / 800000 = 0,0000125 s == 1 bit , -¿ 29 Bit = 0,0000125 *29 = 0,003625 pro Parameter *4 alle Parameter = 0,00145 s alle, 1/ 0,00145 = 689 Hz... Experimentelle Werte zeigen, dass die maximale Abtastfrequenz bei ca. 100 Hz liegt, somit ist die Nyquistfrequent des Signals 50Hz.

Die Dynamische Datenerfassung erfolgt durch eine dynamische Last und einen Stereoverstärker (https://www.conrad.de/de/p/conrad-components-stereo-verstaerker-bausatz-9-v-dc-12-v-dc-18-v-dc-35-w-2-1216582.html) an dem ein Audiosignal von einem Raspberry Pi angeschlossen werden. Es werden Audiosignale mit anfänglich geringer Frequenz und es wird so lange getestet, bis die maximal sinnvoll erfassbare frequenz erreicht wurde, die gemessen werden kann.

Die linken Wert repräsentieren die Zeitstellen in Sekunden, die Rechten sind die Signaldaten, welche dann per Programm zu einer Kurve interpoliert werden.

2.6.1 Echtzeitmonitoring

2.6.2 Laufzeitmonitoring

2.6.3 Vergleichsmonitoring mehrerer Wandler im laufenden Betrieb

2.7 Simple Testfälle im Bereich Machine learning

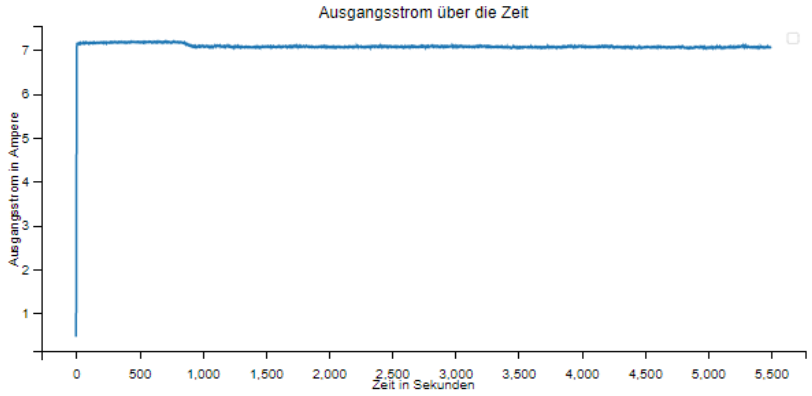


Abbildung 7: Ausgangsstrom über die Zeit mit Filter

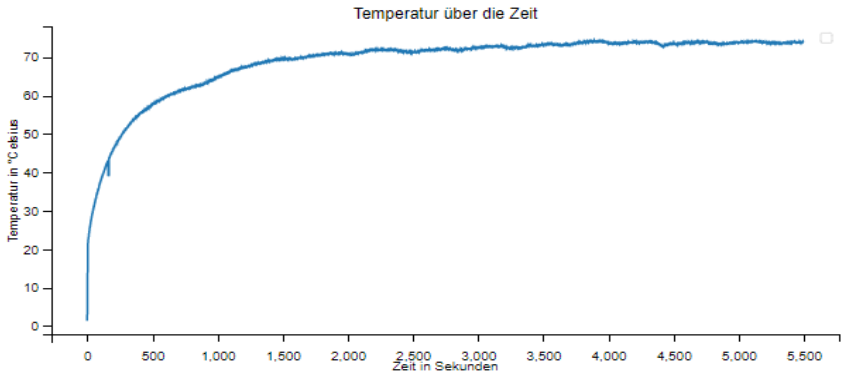


Abbildung 8: Temperatur über die Zeit

3 Ergebnisse

In Abb. x ist der Ausgangsstrom geplotet gegen die Zeit zu sehen. Der erste Graph ist eine darstellung der Daten ohne Frequenz filter, der zweite mit. Der Frequenzfilter ist für die visualisierung notwendig, da der Wandler eine Standardabweichung von mehreren Milliampere besitzt. Für weiteres wird nun immer eine Visualisierung mit Frequenzfilter verwendet insofern nicht anders genannt. Es ist in Abb. x zu erkennen, das der Strom nach einer Gewissen Zeit absackt, es liegt nahe, dass dies an der steigenden Temperatur assoziiert ist,

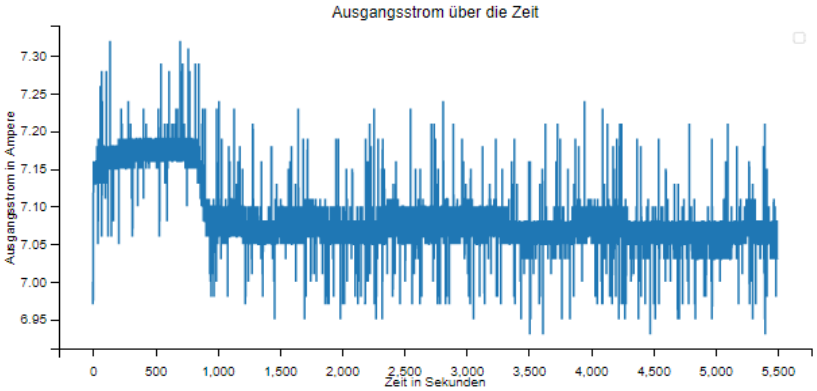


Abbildung 9: Temperatur über die Zeit

Bei der Temperatur ist erkenntlich, das diese anfangs Rasant ansteigt und nach etwa 1600 Sekunden auf eine Endtemperatur hinkonvergiert. Es ist bei T = 250 Sec erkenntlich, das der Graph einen ausreiser hat, dies hat zwei ursachen: die 1. und fundamentalste ist, dass der I2C Mikrocontroller auf dem Board in intervallen den Wert -20.51 ausgibt, dadurch, dass dies dann entsprechend ebenfalls frequenzgefiltert wird, ergibt sich so eine kleine auslenkung. hlEs steht außer Frage, dass gelegentliche Sprünge zu negativen Werten, durchaus kritisch für das System sein können, wenn die Daten weiterverarbeitet und abhängig von diesen Daten Aktionen durchgeführt werden sollen. Daher werden Daten, während sie verarbeitet werden, dahingehen überprüft, ob solche Fehlerfälle auftreten, und sofort beseitigt. Das beseitigen kann dabei mit mehreren Methoden erfolgen :

Diese Abbildung zeigt einen der heißesten Punkte auf dem Bord, wie der Abbildung zu entnehmen ist, zeigt sich eine Temperatur von 87° C. Dies ist eine diskrepanz von 10°C zu der maximalen Temperatur, die der Sensor aufgenommen hat. Es ist offensichtlich, dass der Sensor nicht die vollen 87° C messen kann, weil er nicht in der nahen Umgebung des Hotspots ist. Im Folgenden wird versucht die Temperaturdiskrepanz durch einführen eines Offset wertes zu minimieren, um möglichst realistische Daten zu erheben.

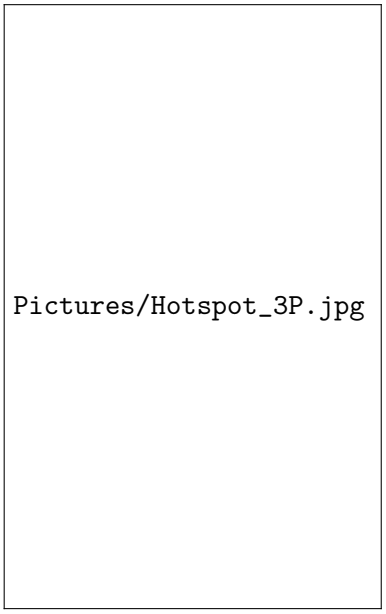
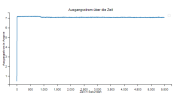
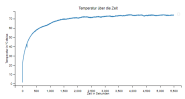


Abbildung 10: Hotspot aufnahme per Wärmebildkamera



(a) A subfigure



(b) A subfigure

Abbildung 11: A figure with two subfigures

4 Literaturverzeichnis