Malte Seldschopf

Entwicklung eines Messsystems zur Bestimmung des Phasenwinkels der Netzspannung

Bachelorarbeit

Technische Hochschule Köln,

Cologne Institut for Renewable Energy,

3. Juni. 2019

Betreuer: Prof. Dr. Eberhard Waffenschmidt

Ko-Referent: Prof. Dr. Johanna May

Cologne Institute for Renewable Energy

Technology Arts Sciences TH Köln

Erklärungen

Name: Malte Seldschopf Matrikel-Nummer: 11101222

Erklärung zum eigenständigen Verfassen

Hiermit erkläre ich an Eides statt, dass ich die vorliegende Arbeit selbständig verfasst habe. Ich habe keine anderen außer den von mir angegeben Quellen und Hilfsmitteln verwendet
Die Arbeit wurde bisher in gleicher oder ähnlicher Form keiner anderen Prüfungsbehörde vorgelegt und auch nicht veröffentlicht.

Köln, 3.Juni 2019

Malte Seldschopf

Erklärung zur Veröffentlichung

Ich bin damit einverstanden, dass meine Abschlussarbeit ausgeliehen werden darf. Sie darf von meinem Betreuer im Internet veröffentlicht werden.

Köln, 3.Juni 2019

Malte Seldschopf

Erklärung zu den Bildrechten

Die verwendeten Bilder sind alle von mir selbst angefertigt oder aus frei zugänglichen Quellen genommen.

Köln, 3.Juni 2019

Malte Seldschopf

Danksagung

Mein besonderer Dank gilt Herrn Prof. Dr. Waffenschmidt, der es mir ermöglicht hat, meine Bachelorarbeit im Labor für elektrische Energietechnik anfertigen zu können. Zudem danke ich ihm für sein stets offenes Ohr und seine konstruktiven Ratschläge.

Außerdem möchte ich mich bei meinen Kommilitonen im Labor und aus meiner Lerngruppe bedanken, die stets für eine gute Arbeitsatmosphäre sorgten und mich im Labor mit ihren Erfahrungen unterstützten.

Ich danke zudem meiner Familie und meinem Freunden, darunter ganz besonders meinen Eltern, welche mich während der Dauer meines gesamten Studiums unterstützt haben.

Kurzfassung

Diese Arbeit beschäftigt sich mit der Entwicklung eines Messsystems zur Bestimmung des Phasenwinkels der Netzspannung und bezieht sich dabei auf die entwickelte Software des von mir im vergangenen Januar durchgeführten Praxisprojekts „Erstellung eines Phasor-Messgerätes“. In diesem Praxisprojekt wurden die Entwicklungsplatine ausgewählt und der zur Messung benötigte Quellcode programmiert. Der Programmcode des Praxisprojekts beinhaltet den Zugriff auf die für die Messungen benötigten Schnittstellen, die Abspeicherung von Daten auf der SD-Karte und die Auswertung der GNSS-Datensätze. Im weiteren Verlauf dieser Arbeit wird auf die zusätzlich verwendete Hardware eingegangen, insbesondere den Spannungswandler, welcher die Netzspannung von 230V in eine Wechselspannung zwischen 0,72V und 2,62V umwandelt. Eine detaillierte Beschreibung, des verwendeten Gehäuses bildet den nächsten Arbeitsschritt. Danach wird der zusätzliche Programmcode dargestellt sowie die anschließende Messung mit dem Gerät und die damit verbundene Auswertung der Daten.

Abstract

This work is about the development of a measuring system for the determination of the phase angle of the mains voltage and refers to the developed software of the practical project "Creation of a Phasor measuring device" which I finished last January. In this practical project, the development board was selected and the source code for the measurement was programmed. The program code of the practical project contains the access to the interfaces which are needed for the measurements, the storage of data on the SD card and the evaluation of the GNSS data sets. In the further course of this work, the additional hardware that is used will be explained, as example, the voltage converter, which converts the mains voltage of 230V into an AC voltage of 0.72V and 2.62V. The next step is a detailed description of the housing which is used. Then the additional program code is represented as well as the following measurement with the device and the resulting evaluation of the data.

Inhaltsverzeichnis

Kurzfassung I

Abstract ..... **Fehler! Textmarke nicht definiert.**

Inhaltsverzeichnis II

1. Einleitung 1

2. Haupteil 3

2.1. Theorie und Stand der Technik 3

2.1.1. Phasor-Messgeräte 3

2.1.2. Transformatoren 4

2.1.3. Arbeitsspeicher 5

2.1.4. Netzrückwirkung 6

2.2. Hardware 7

2.2.1. Entwicklungsplatine 7

2.2.2. Zeiterfassungsmodul 8

2.2.3. Netzteil 8

2.2.4. Spannungswandler 9

2.3. Fertigung des Messsystems 21

2.3.1. Gehäuse 21

2.3.2. Einschnitte 22

2.3.3. Befestigungsplatte 23

2.3.4. Messsystem 24

2.4. Software 25

2.4.1. Referenzzeitermittlung 27

2.4.2. Konfigurationsmöglichkeiten 31

2.4.3. Startpunkt der Messung 35

2.4.4. Berechnung 37

2.4.5. Abspeicherung 57

2.4.6. Wiederaufnahme 61
2.5. Messwertaufnahme 62
2.5.1. Identische Positionen 62
2.5.2. Simulationsmessung unter Last 65
2.5.3. Amplitudenabweichungen 67
3. Schlussteil 68
3.1. Fazit 68
3.2. Ausblick & Diskussion 69
Anhang 71
Abbildungsverzeichnis 71
Tabellenverzeichnis 72
Formelverzeichnis 73
Literaturverzeichnis 74
Gesamter Schaltplan 75

1. Einleitung

Zum Verständnis einer Analyse der Netzstabilität und zur Vertiefung von Kenntnissen der elektrischen Energietechnik sowie der Programmierung der Mikrocontroller ist das Thema dieser Bachelorarbeit sehr geeignet. Mikrocontroller haben mich durch ihre Vielfältigkeit seit etwa zehn Jahren fasziniert und in mir den Wunsch geweckt, meine Erfahrungen in diesen Fachbereich zu erweitern. Das von mir im letzten Januar entwickelte Praxisprojekt „Erstellung eines Phasor-Messgerätes“ beinhaltet bereits ein anfängliches Kennenlernen einer Mikrocontrollerumgebung und erste Versuche, einen solchen Controller zu programmieren. Die vorliegende Bachelorarbeit baut also auf dem genannten Praxisprojekt auf.

Diese Bachelorarbeit zeigt zunächst die benötigen Bestandteile der Hardware und deren Verschaltung untereinander. Danach wird dargestellt, wie der dazugehörige Programmcode implementiert, wie die Hardware in ein Gehäuse eingebaut und wie schließlich die Funktionalität im Niederspannungsnetz getestet und ausgewertet wird.

Jeweils zwei identische Messgeräte werden gebaut, welche zusammen das Messsystem ergeben. Diese beiden Messgeräte werden vor und hinter einer Last angeschlossen und sollen anhand der Differenz des Phasenwinkels über die zu messende Last Aufschluss geben. Darüber hinaus wird ausgewertet, wie sich der Phasenwinkel über Distanz verändert und welche Auskünfte dadurch über die Netzstabilität gewonnen werden können.

Im ersten Kapitel werden die theoretischen Grundlagen und der aktuelle Stand der Technik erläutert. Dieses Kapitel beinhaltet grundlegende Informationen, die zur Verständlichkeit der folgenden Kapitel notwendig sind.

Das zweite Kapitel beschäftigt sich mit der verwendeten Hardware. Die Komponenten sowie die Argumente für ihre Verwendung werden vorgestellt und beschrieben. Bei der Fertigstellung des Messsystems wurden mehrere Komponenten zur Spannungsumwandlung verwendet, d.h. verschiedene Spannungswandler. Der Begriff Spannungswandler ist der Oberbegriff für spezifische Arten von Spannugswandlern, die ich brauchte: z.B. Transformatoren, Spannungsregler und Spannungsteiler. Auch die Fehler, die sich bei der Auswahl und dem Einsatz der Hardware ergaben, sollen dargestellt werden. Der ganze erste misslungene Lösungsversuch wird beschrieben, analysiert und bildlich festgehalten.

Die Fertigung des Messsystems wird im dritten Kapitel beschrieben. Dabei soll auf die Verschaltung, auf den Einbau in das Gehäuse und auf die verwendeten Materialen eingegangen werden.

Im vierten Kapitel ist die Software Gegenstand der Arbeit. Der implementierte Programmcode wird erklärt und Schritt für Schritt kommentiert.

Die Auswertung der gemessenen Daten ist das Thema des fünften Kapitels. Dort wird dargestellt, wie die Daten in Excel umgewandelt werden und welche Schlussfolgerungen aus ihnen gezogen werden können.

2. Haupteil

2.1. Theorie und Stand der Technik

2.1.1. Phasor-Messgeräte

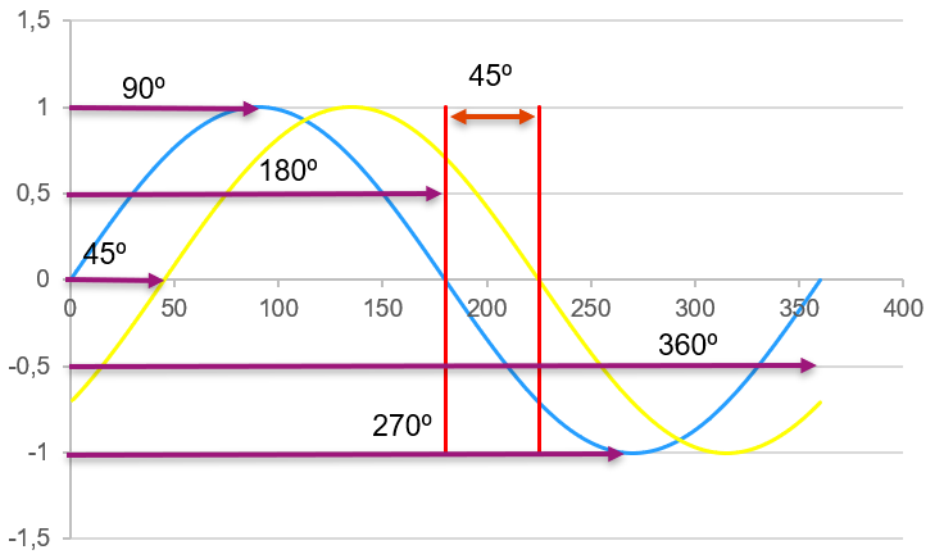

Abbildung 1 - Phasenlagen und Phasendifferenz [1]

Phasor-Messgeräte sind Messgeräte für sinusförmige Signale, deren Amplitude und Phasenlage sie messen (Abbildung 1). Sie werden in der Niederspannungstechnik und in Smart-Grids eingesetzt und messen in kurzen Zeitabständen den Zustand der Wechselspannung. Informationen zu anderen Abschlussarbeiten über Phasor-Messgeräte sind nicht vorhanden oder nicht auffindbar. Die verschiedenen Phasor-Messgeräte unterscheiden sich durch ihren Bezug zu ihrem Referenzwert. Nicht jeder Hersteller gibt die Referenz an. Häufig wird jedoch, wie bei der Fertigung meines Messsystems, mit einer hochpräzisen Zeitreferenz, dem GNSS, synchronisiert. Dadurch erweitert sich die Lageerfassung über großflächige Stromnetzwerke hinweg. Durch die Genauigkeit der verwendeten Hardware unterscheidet sich mein Messsystem geringfügig von handelsüblichen Phasor-Messgeräten. Die Hersteller von Phasor-Messgeräten bieten eine Vielzahl von Optionen. Alle diese Geräte ermitteln den Phasenwinkel, die Frequenz und die Amplitude. Häufig werden diese Messgeräte in Smart-Grids eingesetzt, um die Netzspannung in der Niederspannungstechnik zu kontrollieren und nachzuregeln. Meistens bieten Phasor-Messgeräte zudem Schnittstellen für Sensoren an, welche z.B. Temperaturen und Luftfeuchtigkeit messen können. Eine übersichtliche Benutzeroberfläche ist auch häufig vorhanden. Sie erleichtert für die Benutzer das Differenzieren von Anforderungen und gewährleistet damit einen leichteren und effizienteren Umgang. [2]

2.1.2. Transformatoren

Ein Transformator wandelt Wechselstrom auf eine höhere oder niedrigere Spannungsebene um. Trafos eignen sich zu verlustarmer Energieübertragung, da auf einem hohen Spannungsniveau geringe Stromstärken mit entsprechend geringen Verlusten transportiert werden können.
Das Kernelement eines Transformators ist ein aus Stahlblech aufgeschichteter Kern, der einen magnetischen Kreis bildet. Auf jeweils einer von zwei gegenüberliegenden Seiten (Primär-, und Sekundärseite) befindet sich mindestens eine Wicklung, wobei es sich um Drähte handelt, welche mit unterschiedlicher Windungszahl um den Stahlkern gewickelt sind, also um Spulen.
Die Primärspule des Transformators empfängt eine Wechselspannung. Dadurch wird im Stahlkern ein magnetisches Wechselfeld induziert. Dieses sich ständig ändernde Magnetfeld induziert in der Sekundärspule eine Wechselspannung. Dabei ist diese an der Sekundärseite ausgehende Spannung eine Wechselspannung mit der gleichen Frequenz wie die Wechselspannung an der Primärseite. Das Verhältnis von Eingangs- und Ausgangsspannung wird durch das Verhältnis der Windungszahlen von Primär- und Sekundärspule bestimmt. [3] [4]

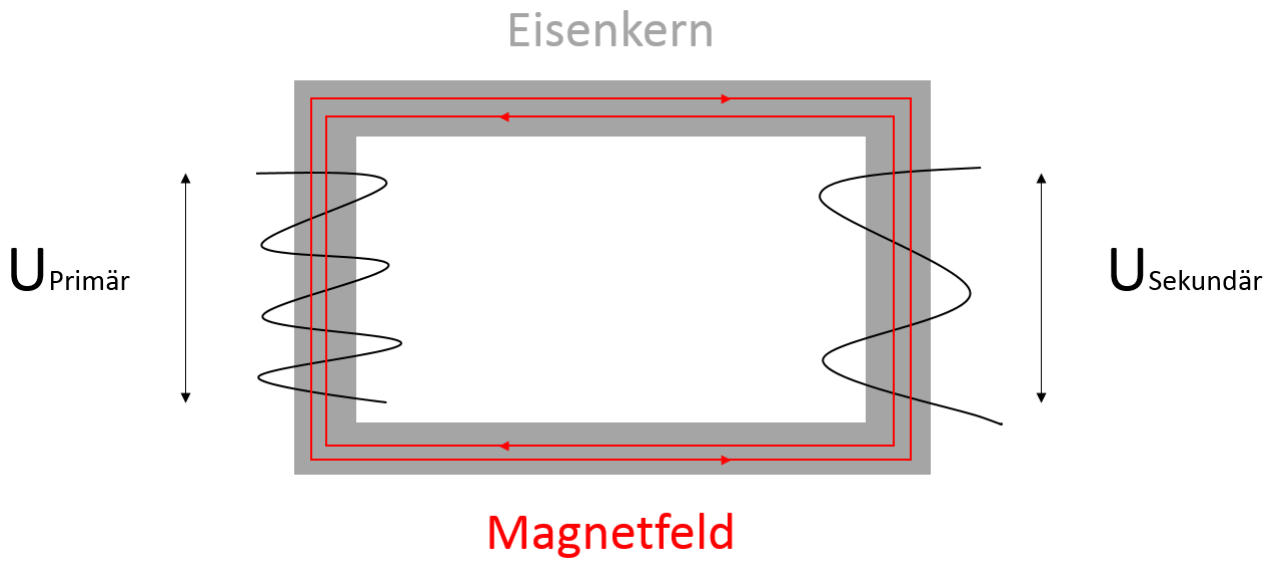

Abbildung 2 - Transformator - Funktionsweise [1]

2.1.2.1. Magnetische Sättigung

Ein Material ist magnetisch gesättigt, wenn sich die magnetische Flussdichte nicht mehr proportional zum Strom verhält. Ab diesem Punkt steigt die magnetische Flussdichte nur noch sehr langsam an. In einem Transformator, wo sich, wie gesagt, Spulen mit unterschiedlicher Windungszahl an einem Eisenkern gegenüberstehen, bestimmt die magnetische Permeabilität des Eisenkerns die Effizienz der Umwandlung der elektrischen Spannung. Ein Transformator mit einer geringen magnetischen Permeabilität erreicht die Sättigung schnell. Das führt zu einer ineffizienten Umwandlung von elektrischer in magnetische Energie. [5]

2.1.3. Arbeitsspeicher

Der Arbeitsspeicher eines Mikrocontrollers ist ein Speicher, der die gerade auszuführenden Programmteile und die dazu benötigten Daten enthält. Der Prozessor hat auf den Arbeitsspeicher unmittelbar Zugriff. Deshalb beeinflussen dessen Leistungsfähigkeit und Größe in wesentlichem Maße die Leistungsfähigkeit des gesamten Mikrocontrollers. Der verwendete Mikrocontroller, PIC32MZ2048EFG, enthält 512 kB SRAM. SRAM bedeutet static random-asccess memory, auf Deutsch „Statistischer Direktzugriffsspeicher“. Dort werden die Berechnungen des Prozessors abgespeichert. Die Daten, welche sich im Arbeitsspeicher befinden, gehen bei Abschaltung der Betriebsspannung verloren. Im Unterschied zu dem dynamischen Arbeitsspeicher (DRAM) benötigt SRAM kein periodisches Auffrischen zur Vermeidung von Datenverlusten, sondern er behält seine Dateninformation, solange die Betriebsspannung anliegt.
Der Arbeitsspeicher wird im Programm zurzeit zu 99 % ausgereizt. Das passiert hauptsächlich durch die Abspeicherung der aufgenommenen Spannungskurve. Zurzeit werden 120.000 Werte vom Datentyp Integer abgespeichert. Ein Integer besitzt eine Größe von 32 Bit. Dies entspricht einer Größe von 4 Bytes. Bei 120.000 aufgenommenen Messwerten wird der Arbeitsspeicher mit 480.000 Bytes gefüllt. Dies ergibt eine Auslastung von 93,75 % ((480 ÷ 512) × 100). Die restlichen 6,75 % werden von Hilfsvariablen und der Bildung der Regressionsgeraden beansprucht. Aus diesem Grund würde der Mikrocontroller bei einer Erhöhung der Messwerte den Programmcode nicht weiter ausführen können. [6]

2.1.4. Netzrückwirkung

Netzrückwirkung ist der Oberbegriff für Spannungsänderungen, Spannungseinbrüche, Spannungsschwankungen, Flicker und Oberschwingungen. Sie entstehen meistens vermehrt durch den Einsatz von Leistungselektronik. [7]

2.1.4.1. Oberschwingungen

Oberschwingungen sind Wellen, deren Frequenz um ein Vielfaches höher als die Frequenz der Grundschwingung ist. Solche Oberschwingungen entstehen durch unterschiedliche Einflussfaktoren und verzerren die elektrische Spannung, die im Idealfall in einer perfekten Sinuskurve verläuft, in der die Polarität gleichmäßig zwischen positiven und negativen Werten hin- und herwechselt. Durch Betriebsmittel mit einer nichtlinearen Kennlinie, wie z.B. Transformatoren, lässt sich eine Überlagerung von verschiedenen sinusförmigen Wellen kaum vermeiden.
Infolge dieser Oberschwingungen entsteht ein Rauschen, wodurch sich die Genauigkeiten der Sinuskurven verändern. Eine genaue Messung wird erschwert und erfordert eine hohe Präzision, um aussagekräftige Ergebnisse zu erzielen. [8] [9]

2.1.4.2. Spannungseinbrüche

Spannungseinbrüche führen für einen kurzen Zeitraum zu einem Rückgang oder völligen Ausfall der Effektivspannung. Sie entstehen durch das Anlaufen von großen Lasten am betroffenen Standort, durch Verbraucher im selben Stromkreis oder wegen Fehler in anderen Teilnetzen. Ein plötzliches Absinken des Spannungseffektivwerts der genormten Spannung auf einen Wert zwischen 90 % und 1 % entspricht einen Spannungseinbruch. Dagegen zählen kleinere Spannungsaussetzer, über 90 % und unter 100 %, nicht als Spannungseinbrüche, haben aber Auswirkung auf die Netzstabilität. In der Regel werden die meisten Spannungseinbrüche nicht erkannt, doch entstehen durch sie wirtschaftliche Schäden, welche größer als die von Unterbrechungen sind. [10]

2.2. Hardware

Zur Erstellung des Messsystems wurde folgende Hardware ausgewählt. Als Rechen- Mess- und Verwaltungssystem wurde eine Entwicklungsplatine der Firma „Digilent“ verwendet. In dieser Platine, dem ChipKIT Wi-Fire, werkelt ein Prozessor der Firma „Microchip“, der PIC32MZ2046EFG.
Zur Zeiterfassung wurde das GNSS 5-click der Firma „MikroElektronika“ verwendet. Auf den GNSS 5-click befindet sich der Prozessor NEO-M8, ein GNSS-Modul der Firma „u-blox“.
Die aufgenommene Messspannung wird mit einem selbst konstruierten Spannungswandler so umgewandelt, dass der Mikrocontroller die eingehende Netzspannung messen kann.
Der Mikrocontroller und die Komponenten an dem Spannungswandler werden mit einem Netzteil der Firma „Mean Well“ mit Strom versorgt. Der Mikrocontroller wandelt die anliegende Spannung um und versorgt dadurch das Zeiterfassungsmodul mit der benötigten Betriebsspannung.

2.2.1. Entwicklungsplatine

Abbildung 3 - Entwicklungsplatine [1]

Die gewählte Entwicklungsplatine bietet eine Reihe von unterschiedlichen Anwendungen. Für das dargestellte Projekt wurden der Analog-Digital-Wandler (AD), die sichere digitale Speicherkarte (SD-Karte), der Universelle-Asynchrone-Empfänger-Überträger (UART) und der Universelle Serielle Bus (USB) verwendet. Die Messung findet über den AD-Wandler statt. Die Zeiterfassung wird über das UART-Modul empfangen. Der Prozessor verarbeitet die eingehenden Daten und speichert sie über die SD-Karte ab. Die USB-Schnittstelle dient zur alternativen Übertragung der Messwerte. Diese Möglichkeit ist vorhanden, wurde jedoch in diesem Programm nicht implementiert. [11]

2.2.2. Zeiterfassungsmodul

Das Zeiterfassungsmodul dient der Ermittlung des Zeitpunkts, ab dem die Messung gestartet werden soll. Die Erfassung der Daten erfolgt mit einem globalen Satellitennavigationssystem (GNSS). Das GNSS-Modul nutzt den gleichzeitigen Empfang von bis zu drei GNSS-Systemen. Die genutzten Satellitennavigationssysteme sind GPS (USA), Galileo (Europa), GLONASS (Russland) und BeiDou (China). Dadurch vervielfältigt sich die Möglichkeit, die benötigten Daten abzugreifen. Das Zeiterfassungsmodul kann über mehrere Schnittstellen Daten transportieren. In dem hier vorgestellten Gerät findet der Datentransport über die UART-Schnittstelle statt. [12]

Abbildung 4 - GNSS-Modul [1]

2.2.3. Netzteil

Das Netzteil (Abbildung 5) wandelt Wechselspannung zwischen 85 V und 264 V in eine Gleichspannung von 12 V um. Es kann eine maximale Leistung von 30 W liefern, was einem Strom von maximal 2,5 A entspricht. Dieses Netzteil unterstützt eine geringe Leistungsaufnahme im lastfreien Betrieb von unter 0,1 W und eignet sich dadurch für das hergestellte Messsystem, da dessen Leistungsaufnahme sehr gering ist. Mit einem hohen Wirkungsgrad von bis zu 90 % erfüllt das Netzteil die weltweiten Bestimmungen zum maximalen Energieverbrauch. [13]

Abbildung 5 - Netzteil [1]

2.2.4. Spannungswandler

Zur Spannungsmessung muss die Netzspannung des deutschen Stromnetzes an die Spannung des AD-Wandlers im verwendeten Mikrocontroller angepasst werden. Die Netzspannung des deutschen Stromnetzes besitzt einen Effektivwert von 230 V. Der AD-Wandler des Mikrocontrollers kann nur Spannungen zwischen 0 V und 3,3 V messen. Also muss der Spannungswandler die Wechselspannung von 230 V auf die Mikrocontrollerspannung absenken.

Zunächst wird ein Spannungsteiler aus zwei Widerständen verwendet, welcher die Netzspannung um den Faktor 0,11 verringert. Anschließend wird die Spannung mithilfe eines Transformators vermindert. Das Spannungsübersetzungsverhältnis beträgt 38,33 V zu 1 V. An der Sekundärseite des Transformators wird eine Gleichspannungsquelle angeschlossen, welche als Offset fungiert und die von dem Transformator ankommende Wechselspannung mit ihren negativen Spannungsanteilen um 1,65 V anhebt und somit die Spannung auf ein positives Spannungsniveau bringt. Am Ende der Spannungsumwandlung liegt dadurch an dem AD-Wandler des Mikrocontrollers ein Spannungsbereich zwischen 0,72 V und 2,58 V an. Abbildung 6 stellt links den Schaltplan dar und rechts zeigt sie ihn fertig aufgebaut.

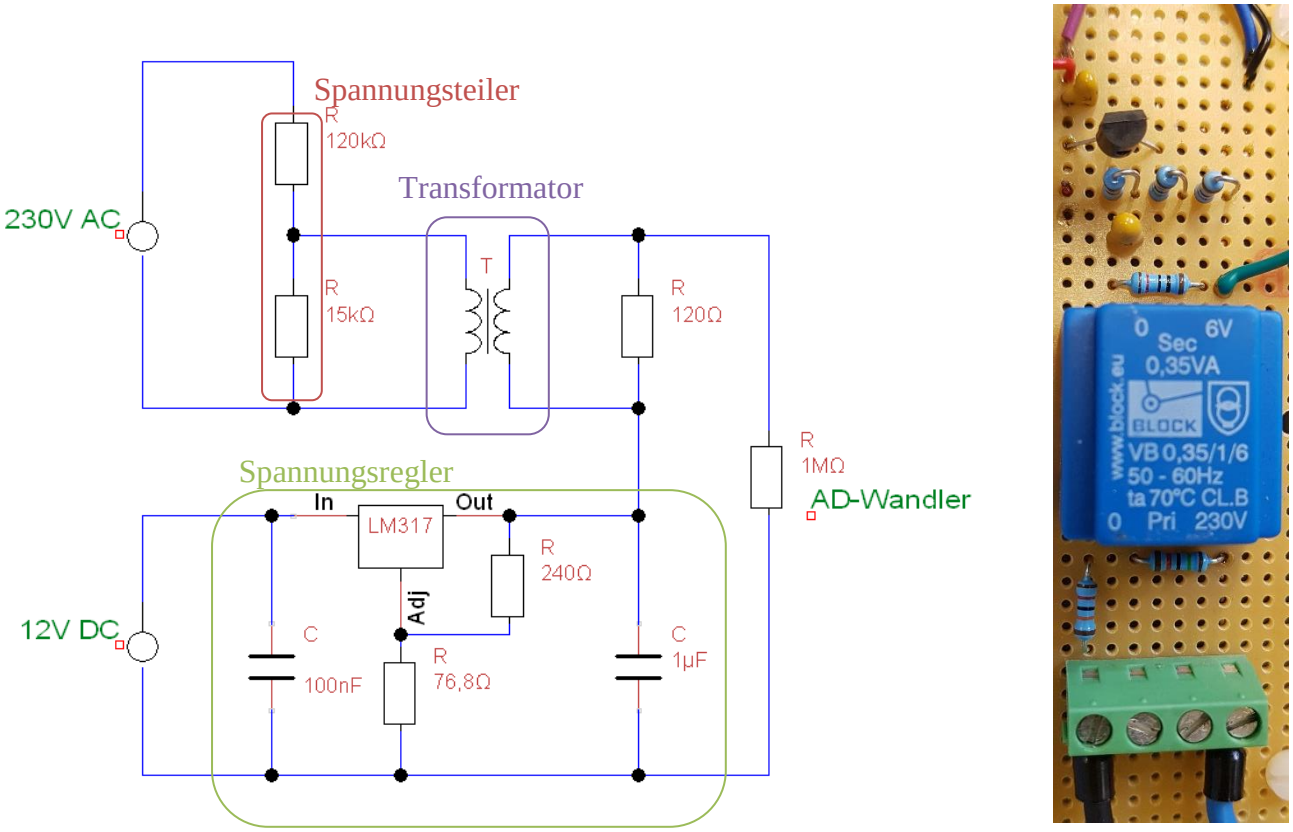

Abbildung 6 - Spannungswandler [1]

2.2.4.1. Spannungsteiler

Die Netzspannung beträgt im Scheitelpunkt $\pm 325\,V$ und durchläuft, wie im Kapitel 2.2.4 beschrieben, hinter der Spannungsquelle einen Spannungsteiler. Dort wird die Wechselspannung mit dem Spannungsteiler-Faktor 0,11 multipliziert, der sich mithilfe von Formel 1 bestimmen lässt.

$$\frac{U2}{U} = \frac{R2}{(R1+R2)} = \frac{15k\Omega}{120k\Omega+15k\Omega} \approx 0{,}11 \qquad (1)$$

Abbildung 7 - Spannungsteilerkurven (Türkis & Lila) & Netzspannung (Gelb) [1]

Der hier beschriebene Vorgang läuft in beiden Messgeräten ab. Durch den Spannungsteiler ergibt sich jeweils eine Spannung von maximal ± 36 V ($230\text{ V} \times \sqrt{2} \times 0{,}11$), welche an der Primärseite des Transformators anliegt. Abbildung 7 zeigt die Netzspannung in Gelb. Sie erreicht effektiv ungefähr 230V und der Scheitelwert liegt bei ±325V ($230\,V \times \sqrt{2}$). Die Kurven der Spannungsteiler sind in Türkis und Lila dargestellt. Da diese Kurven direkt übereinander liegen, sieht man fast nur die türkisfarbene Kurve. Der Spannungsteiler bewirkt im Vergleich zur Netzspannung eine Phasenverschiebung von 3 ms bis 4 ms. Da die Phasenverschiebungen der beiden Spannungsteilerkurven nahezu identisch sind, führt dies zu keiner Verfälschung der Messwerte.

2.2.4.2. Printtransformator

Damit eine möglichst verlustarme Transformation stattfinden kann, wird zur Herabstufung der Spannung ein Netztransformator verwendet. Das Messgerät sollte möglichst klein und kostengünstig gebaut werden. Aus diesen Gründen wurde der folgende Netztransformator der Firma „Block“ gewählt.

Abbildung 8 - Printtransformator [1]

Dieser Printtransformator hat an der Primärseite eine Bemessungseingangsspannung von 230 V AC. Die Bemessungsausgangsspannung an der Sekundärseite beträgt 6 V AC und die Bemessungsleistung maximal 0,35 W. Der Transformator besitzt an der Sekundärseite nur einen Ausgang. Mithilfe der Durchsteckmontage wurde der Trafo auf der Leiterplatte angebracht. Er hat eine Breite von 22,7 mm, eine Höhe von 15 mm und eine Tiefe von 22mm. Die Funktionsweise eines Transformators wurde in Kapitel 2.1.1 beschrieben. Diese Beschreibung ist auf den hier dargestellten Transformator anwendbar. Das Übersetzungsverhältnis des Transformators von der Primär- zur Sekundärseite beträgt ungefähr 38,33V zu 1V und wurde mit der folgenden Formel berechnet. [14]

$$\text{Übersetzungsverhältnis} = \frac{Bemessungseingangsspannung}{Bemessungsausgangsspannung} \qquad (2)$$

$$Bemessungseingangsspannung = 230V$$

$$Bemessungsausgangsspannung = 6V$$

$$\text{Übersetzungsverhältnis} = 38{,}33333333$$

2.2.4.3. Offset

Die Offset-Spannung wird eingefügt, damit, wie schon in Kapitel 2.2.4 erklärt, der Mikrocontroller die negative Halbwelle der Netzspannung nicht empfängt. Sie ist eine Gleichspannung, welche eine Spannungsquelle in einen anderen Spannungsbereich anhebt oder senkt. In dem hier entwickelten Messsystem wird die Offset Spannung genutzt, um die transformierte Wechselspannung auf ein positives Potential anzuheben. Der Mikrocontroller sollte keine negativen Spannungen einlesen, weil er dadurch beschädigt und auf Dauer zerstört würde.

In dem entwickelten Messsystem wird ein Offset verwendet, der die Spannung um 1,65 V anhebt. Die benötigte Offset-Spannung wird mit einem Spannungsregler realisiert. Dafür wurde der einstellbare Spannungsregler „LM317LZ“ der Firma „Texas Instruments“ gewählt. Mit ihm können Spannungen zwischen 1,2 V und 37 V eingestellt werden. Am Eingang des Spannungsreglers befindet sich ein Kondensator mit einer Kapazität von 100 nano Farad, welcher die eingehende Spannung stabilisiert. Um am Ausgang Schwingungen zu vermeiden, wurde parallel zu den Widerständen der Kondensator C2 geschaltet. Dessen Kapazität beträgt 1 µF. Die Kapazitäten der Kondensatoren wurden nach Vorgabe des Datenblatt gewählt. Die Ausgangsspannung lässt sich mithilfe von Formel 3 konfigurieren.

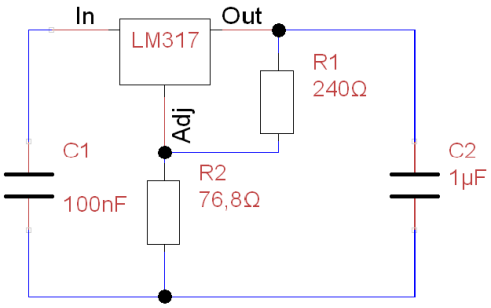

Abbildung 9 - Spannungsregler [1]

$$Uout = 1{,}25V \times (1 + \frac{R2}{R1}) \tag{3}$$

Über den Widerstand R2 ist der Spannungsregler konfigurierbar. Für den Widerstand R1 werden nach Vorgabe im Datenblatt 240 Ω verwendet. Die 1,25 V Spannung, welche in Formel 3 angegeben ist, kommt durch die benötigte Betriebsspannung des Spannungsreglers zustande. Da der Spannungsregler keine kleineren Spannungsbereiche anheben kann, wurde das Netzteil (Kapitel 2.2.3) überdimensioniert. [15]

Mit dem einstellbaren Spannungsregler „LM317LZ“ wurde ein Bauteil ausgewählt, das eine möglichst konstante Gleichspannung erzeugt. Bei einer ungenauen Gleichspannung würde das Spannungsrippel der transformierten Wechselspannung verfälscht werden.

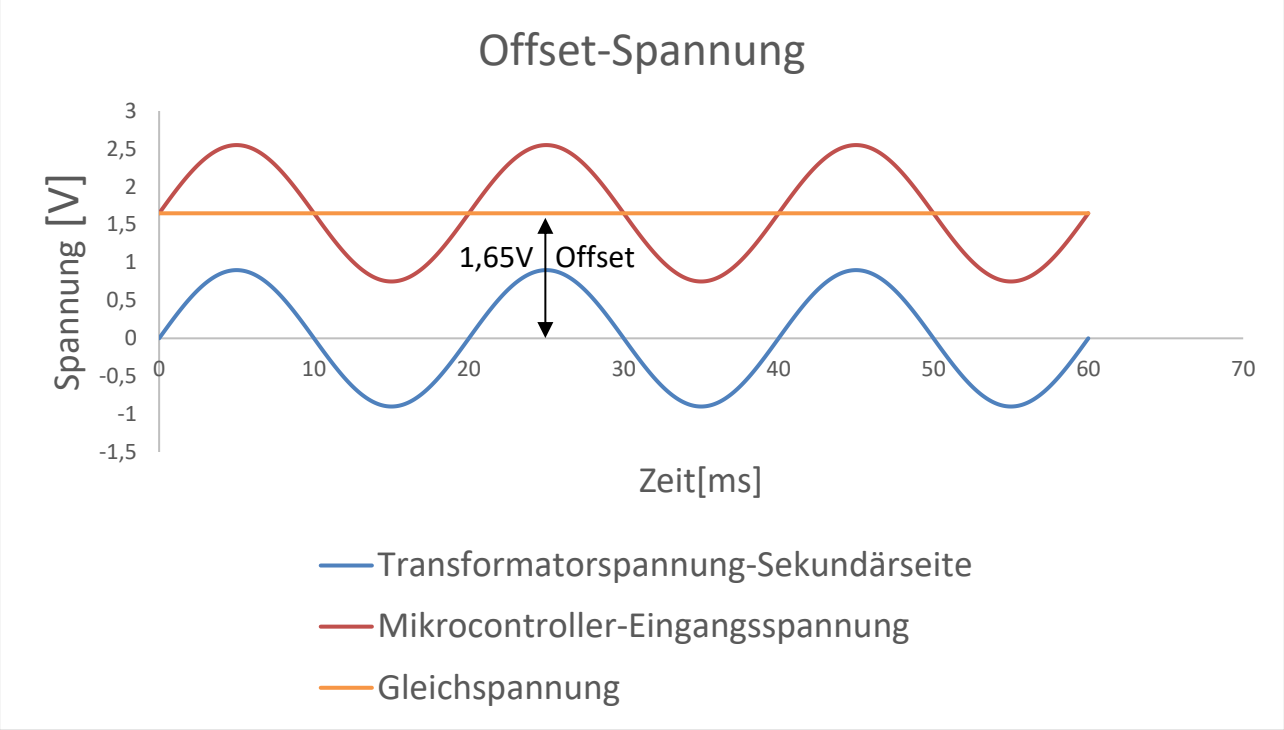

Abbildung 10 - Auswirkungen der Offset-Spannung [1]

Abbildung 10 veranschaulicht, welche Auswirkung die Offsetspannung auf die transformierte Wechselspannung hat und wie der Offset die Spannung anhebt. In Abbildung 11 wird das vom Spannungsregler erschaffene Spannungsrippel im zeitlichen Verlauf von einer Millisekunde dargestellt, ebenso die steigende Flanke. Das Spannungsrippel ist zwar in Abbildung 10 eingeflossen, jedoch wegen der Skalierung nicht zu erkennen. Die Auswirkungen auf die Messung sind so geringfügig, dass man sie vernachlässigen kann.

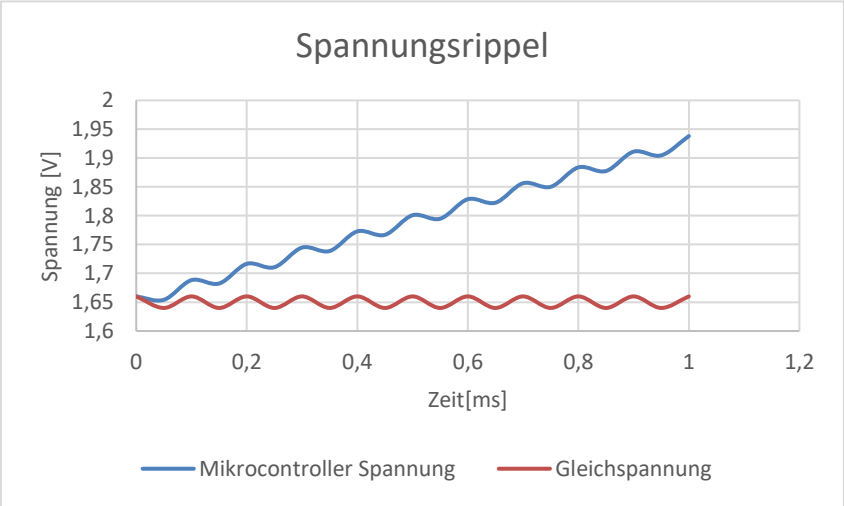

Abbildung 11 - Rippel des Spannungsreglers [1]

2.2.4.4. Fehlerquellen

Während der Entwicklung des Spannungswandlers traten unerwartet nach und nach verschiedene Probleme auf. Der erste, misslungene Lösungsansatz sah folgendermaßen aus: Ein handelsüblicher Printtransformator sollte die Netzspannung von 230 V auf 6 V vermindern. Anschließend sollte ein Spannungsteiler die Spannung von 6 V auf ca. 1,25 V absenken. Um den Mikrocontroller nur positive Spannungen messen zu lassen, wurde zur Anhebung der negativen Anteile der Wechselspannung eine Offset-Spannung geplant. Dafür sollte die Gleichspannungsquelle des Mikrocontrollers, welche 3,3 V beträgt, verwendet werden. Mit einem gleichmäßigen Spannungsteiler sollte dann die Spannung auf 1,65 V gesenkt und an die gemessene Spannung angelegt werden. Der Messbereich, den der Mikrocontroller damit empfangen würde, sollte zwischen 0,4 V und 2,9 V liegen. Abbildung 12 zeigt diesen ursprünglichen Plan als Schaltbild.

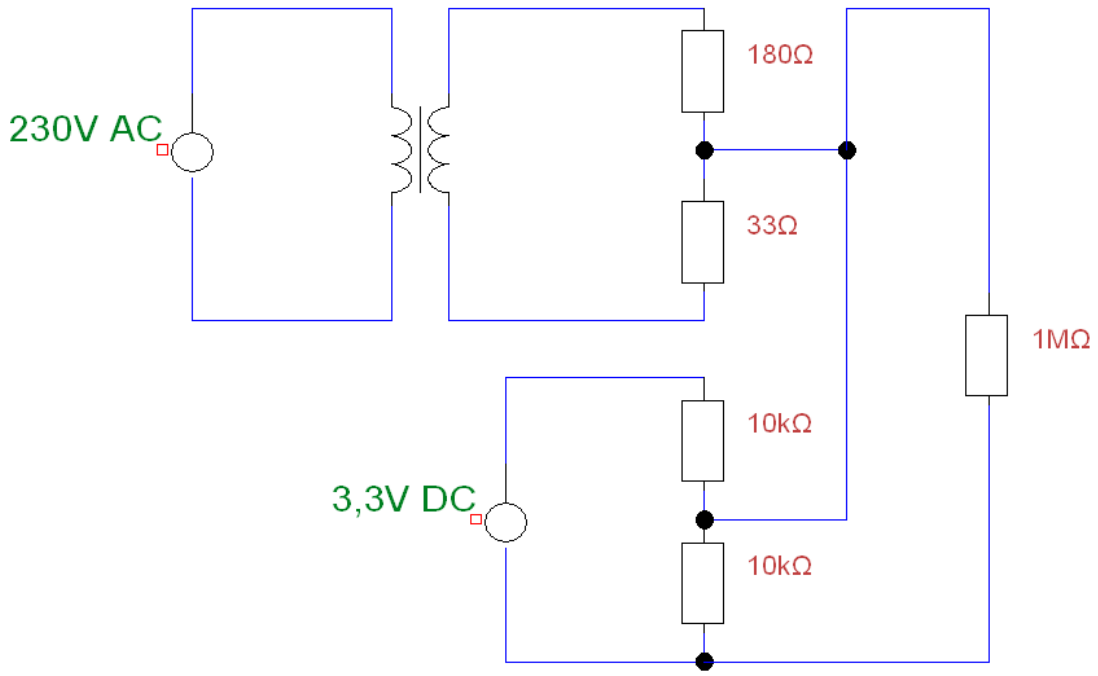

Abbildung 12 - Schaltplan Spannungswandler erster Versuch [1]

2.2.4.4.1. Sättigung

Im Experiment nach dem 1. Lösungsversuch stellte sich zunehmend klarer heraus, dass der angefertigte Spannungswandler die gewünschten Ergebnisse nicht erbringen konnte. Auf der Offset-Spannung, die von dem Mikrocontroller abgegriffen werden sollte, lag ein starkes Rauschen, das die Messwerte dauerhaft verfälschte. Mit einem Spannungsregler (Abbildung 9) konnte dieses Rauschen nahezu vollständig reduziert werden und betrug nur noch ±10 mV.

Die Eigenschaften des im ersten Lösungsversuch zunächst allein verwendeten Transformators erlaubten es nicht, den gewünschten Effekt zu erzielen. Abbildung 13 und Abbildung 14 beschreiben den Spannungsverlauf an den Sekundärseiten der Transformatoren im Leerlauf die sich in den folgenden Abbildungen überlagern.

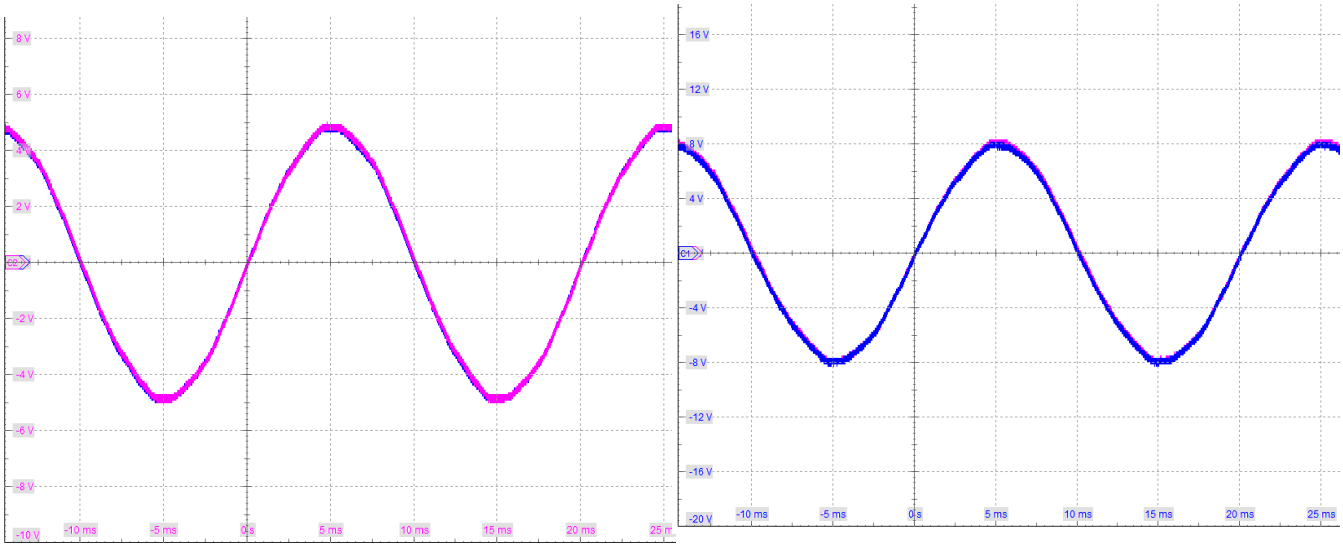

Abbildung 13 - Transformator im Leerlauf (Primärseite links 60 V, Primärseite rechts 110 V) [1]

Abbildung 13 zeigt den Spannungsverlauf des Transformators bei 60 V und 110 V. Wie in Abbildung 13 zu sehen, wird bei 60V ein Sinus transformiert, der dem Idealwert sehr nahekommt. Bei 110V wird immer noch ein sauberer Sinus übertragen.

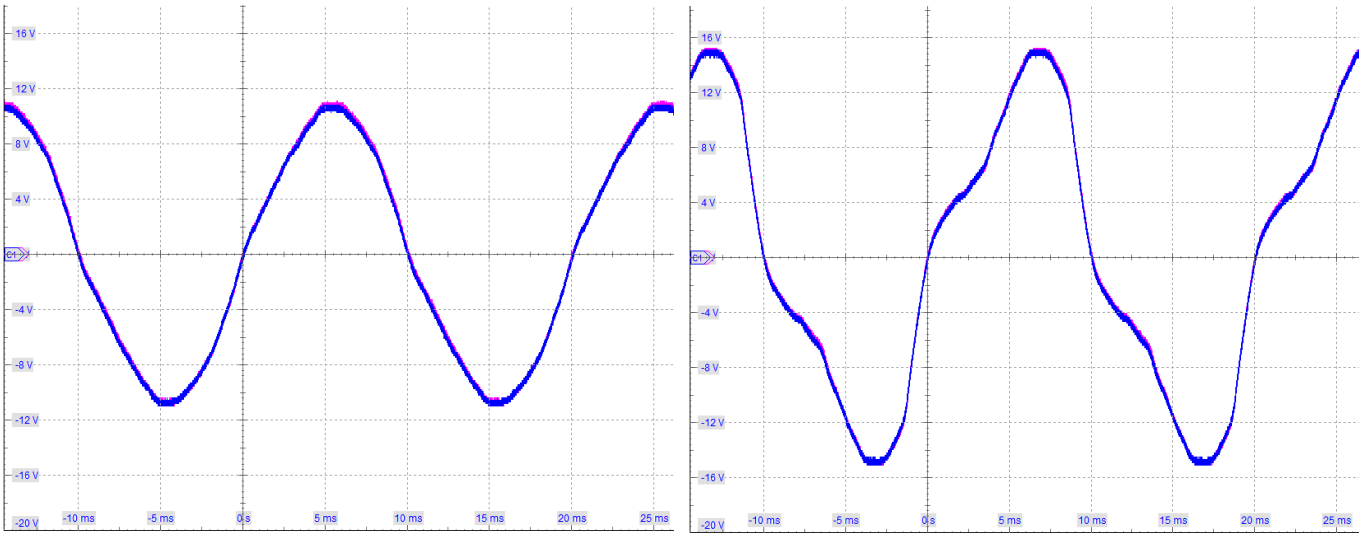

Abbildung 14 - Transformator im Leerlauf - Primärseite links 150V, Primärseite rechts 230V [1]

Abbildung 14 lässt erkennen, dass der Transformator langsam in Sättigung geht. Die linke Kurve veranschaulicht, wie der Sinus bei einer Belastung von 150 V langsam unförmig wird. Rechts ist schließlich die Transformatorkurve unter der Belastung von 230 V gezeigt. Auf den ersten Blick ist zu erkennen, dass kein reiner Sinus mehr übertragen wird. Dieses Ergebnis ist mithilfe der magnetischen Sättigung (Kapitel 2.1.2.1) zu erklären.

Im weiteren Verlauf des Experimentierens nach dem ersten Lösungsversuch verdeutlichte sich bei genauerer Analyse in den steigenden und fallenden Flanken, welche die nachfolgenden Abbildungen (Abbildung 15, Abbildung 16 und Abbildung 17) zeigen, noch ein Problem In diesen Abbildungen können die beiden verwendeten Transformatoren direkt miteinander verglichen werden. Pro Dekade in x Richtung vergehen 20 µs. In y Richtung ändert sich die Spannung um 40 mV pro Dekade. Abbildung 15 veranschaulicht die steigende und die fallende Flanke bei 60 V. Dabei ist ein Phasenversatz von einer Dekade festzustellen, der einem Zeitunterschied von 20 µs entspricht.

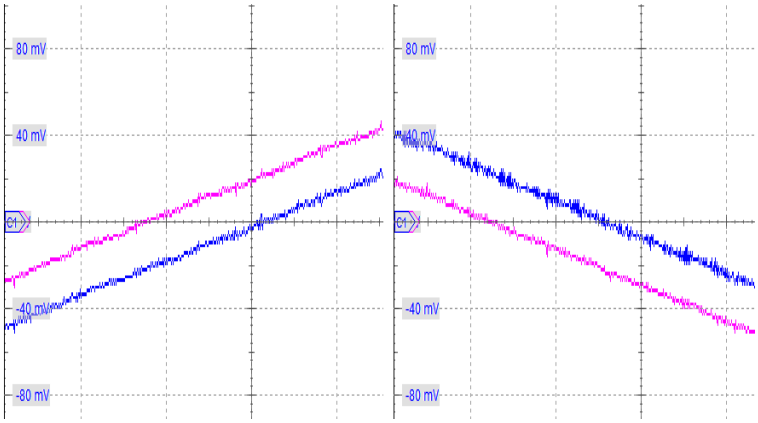

Abbildung 15 - Steigende & Fallende Flanke bei 60V [1]

Abbildung 16 zeigt die steigende und die fallende Flanke bei 110 V. Dort ist kein unterschiedlicher Phasenversatz festzustellen. Die Phasen verlaufen nahezu identisch und unterscheiden sich maximal um 1 µs bis 2 µs.

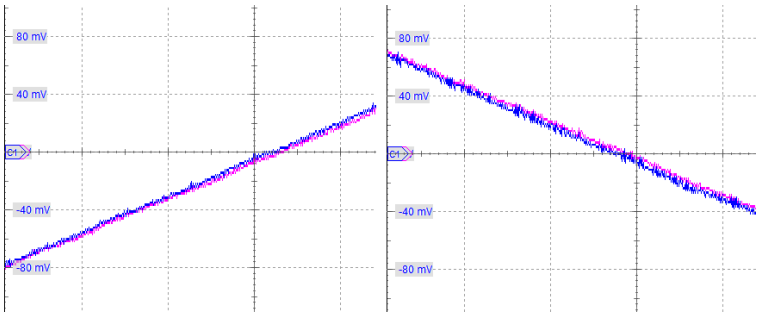

Abbildung 16 - Steigende & Fallende Flanke bei 110V [1]

Abbildung 17 stellt die Flanken bei 150 V dar. Dort hat sich der Phasenversatz wieder um eine Dekade verschoben.

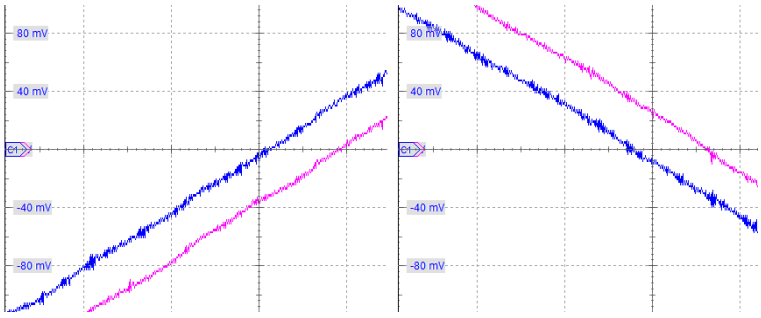

Abbildung 17 - Steigende & Fallende Flanke bei 150V [1]

Insgesamt lassen die drei Abbildungen erkennen, dass sich der Phasenversatz bei unterschiedlichen Spannungen verändert. In Abbildung 15 zeigt sich die pinkfarbene Spannungsflanke zeitlich vor der blauen, in Abbildung 16 stimmen die Spannungskurven zeitlich überein, wogegen in Abbildung 17 die blaue Spannungsflanke der pinkfarbenen vorauseilt.

Die Spannungsflanken der beiden Transformatoren wechseln im zeitlichen Versatz unvorhersehbar zwischen ±20 µs hin und her. Damit sich aber bei einer kontinuierlichen Messung keine Messfehler ergeben, muss dieser Phasenversatz immer konstant sein. Nur dadurch könnte man in der Software-Implementierung den Phasenversatz problemlos berücksichtigen. Da sich der Phasenversatz jedoch als unkalkulierbar erwies, wurde erwogen, ob er toleriert werden könne oder ob es noch weitere Möglichkeiten gibt, diese Ungenauigkeit zu beseitigen. Aus Abbildung 14 geht jedenfalls hervor, dass bei maximaler Spannung der Sinus des Spannungsverlaufs, den die beiden Transformatoren liefern, nicht mehr in reiner Form darzustellen ist.

Die Lösung des Problems wurde mit einem Spannungsteiler realisiert, welcher die Netzspannung um den Faktor 0,11 verringert. Wie bereits in Kapitel 2.2.4.1 gezeigt, ergibt sich dadurch an der Sekundärseite des Transformators eine Spannung von $\pm 940\ mV$ $(36V/38{,}33\ \approx\ \pm 940mV)$.

Abbildung 18 - Spannungskurven: Netz (Gelb), Sekundärseiten der Trafos (lila & türkis) [1]

Nach Absenkung der Netzspannung durch den Spannungsteiler zeigt Abbildung 18 die Netzspannung und die Spannungen, welche an den Sekundärseiten der Transformatoren anliegen. Allerdings ist die türkisfarbene Spannungskurve kaum zu erkennen, weil sie von der lilafarbenen Spannungskurve fast ganz überlagert wird. Dabei gibt die gelbe Kurve die Netzspannung an. Im Vergleich zu dem Spannungsverlauf aus Abbildung 14 ist die Verformung des Sinus damit nun behoben. Die gesättigten Spannungskurven, die in Abbildung 14 im Vergleich zu Abbildung 13 dargestellt werden, lassen sich durch den Spannungsteiler in Sinusform bringen, da nun an den Primärseiten der Transformatoren nicht mehr Spannungen von über 60 V auftreten.

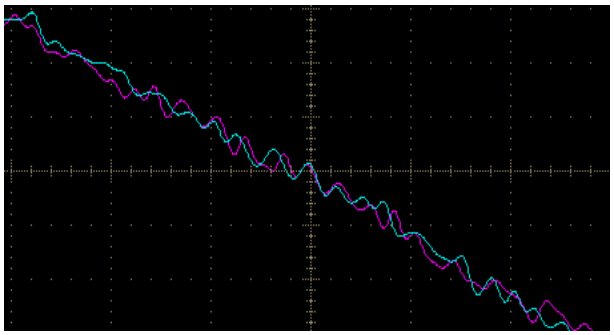

Abbildung 20 - Fallende Flanke [1]

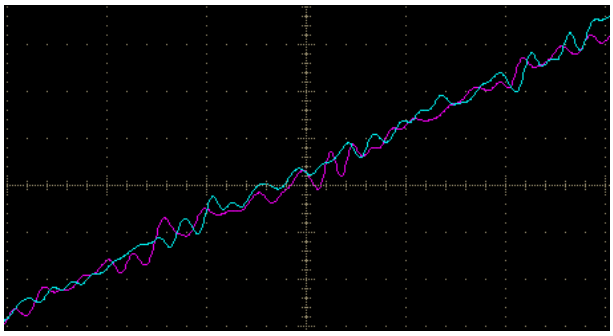

Abbildung 19 - Steigende Flanke [1]

In Abbildung 20 und Abbildung 19 werden dann die steigenden und fallenden Flanken der Ausgangsspannungen beider Trafos (Abbildung 18) detaillierter dargestellt.

In y-Richtung verändert sich die Spannung pro Dekade um 10mV, in x-Richtung verändert sich die Zeit um 40µs pro Dekade. Es ist zu erkennen, dass sich dort kein signifikanter Unterschied im Phasenversatz mehr feststellen lässt. So kann die Phasenverschiebung, die in den Abbildung 15, Abbildung 16 und Abbildung 17 veranschaulicht wurde, durch den Spannungsteiler, eliminiert werden.

2.2.4.4.2. Erhitzung

Ein weiterer Test sollte die Reaktion der Transformatoren unter Hitzeeinfluss klären. Dazu wurden beide Printtransformatoren mit einem Trenntrafo an der Primär- und einem Oszilloskop an der Sekundärseite verbunden. Anschließend wurde ein Transformator mit einem Heißluftfön erhitzt. Die Temperatur des erhitzten Transformators lag zwischen 70 und 90 Grad. Die gemessene Spannungskurve ist in Abbildung 21 dargestellt.

Abbildung 21 - türkisefarbene Kurve: Trafo Normalzustand, lilafarbene Kurve: Trafo erhitzt, gelbe Kurve: Netzspannung [1]

Abbildung 21 lässt deutlich erkennen, dass der erhitzte Transformator sein Übersetzungsverhältnis verändert. Die Amplitude des erhitzten Transformators hat sich um ca. 300 mV verändert. Nach Abkühlung des Transformators pendelte sich das alte Übersetzungsverhältnis wieder ein. Die Umwandlung der beiden Transformatoren erfolgte wieder identisch.

2.2.4.4.3. Schlussfolgerung

Die beiden Transformatoren der beiden Messgeräte erwiesen sich ohne ergänzenden Spannungsteiler als nicht geeignet, unter zunehmender Belastung, ab 150 V (Abbildung 14), das eingehende Signal von der Form her identisch zu halten. Vermutlich wurden bei der Herstellung solcher Trafos Bauteile von minderwertiger Qualität verbaut, wodurch eine geringere magnetische Permeabilität zu einer schnellen Sättigung führt, wie in Kapitel 2.1.2.1 beschrieben. Die verwendeten Transformatoren verändern aber die eingehende Spannung wie sie im Datenblatt angegeben ist.
Für die meisten Zwecke reicht diese Funktion aus. Geräte, die im Niederspannungsnetz betrieben werden, arbeiten selten mit einer Wechselspannung von 230 V. Die Transformatoren senken die Spannungen nahezu verlustfrei auf ein niedrigeres Potential. Die Geräte aus dem Niederspannungsnetz werden häufig mit einer Gleichspannung betrieben. In der Regel wird die eingehende Spannung mit diesen Transformatoren auf die benötigte Spannung gesenkt. Anschließend wird mit einem Gleichrichter die Wechselspannung gleichgerichtet. Dabei ist es für einen Gleichrichter unbedeutend, ob die eingehende Wechselspannung einem perfekten Sinus entspricht. Der Gleichrichter inventiert die negativen Halbwellen und glättet die Spannung mithilfe von Kondensatoren auf das gewünschte Spannungsniveau.
Auch der rotierende Flankenwechsel, wie er in den Abbildung 15, Abbildung 16 und Abbildung 17 beschrieben wurde, ist durch Toleranzen in den Bauteilen der verwendeten Transformatoren zu begründen. Für die oben beschriebenen Gleichrichtungen von Wechselspannungen sind rotierende Flankenwechsel nebensächlich und verändern nichts an der resultierenden Gleichspannung.

2.3. Fertigung des Messsystems

2.3.1. Gehäuse

Abbildung 22 - Gehäuse [1]

Die Hardware wurde in einem grauen, wasserdichten Abzweigkasten der Firma „ABB“ verbaut. Das Gehäuse, welches in Abbildung 22 dargestellt ist, wird mit einem durchsichtigen Deckel verschlossen, wodurch sich die innen befindlichen Komponenten beobachten lassen. Das Material des Gehäuses besteht aus Thermoplast. Thermoplaste sind Kunststoffe, welche sich in einem bestimmten Temperaturbereich verformen lassen. Solange nicht durch Überhitzung die thermische Zersetzung des Materials beginnt, kann dieser Vorgang beliebig oft wiederholt werden. Die Abzweigdose ist an der längeren Seite 220 mm und an der schmaleren 170 mm lang und besitzt eine Höhe von 80 mm. Da in diesem Gehäuse genug Platz für die zu verwendende Hardware ist, wurde das Gehäuse als ausreichend empfunden und bestellt [16].

2.3.2. Einschnitte

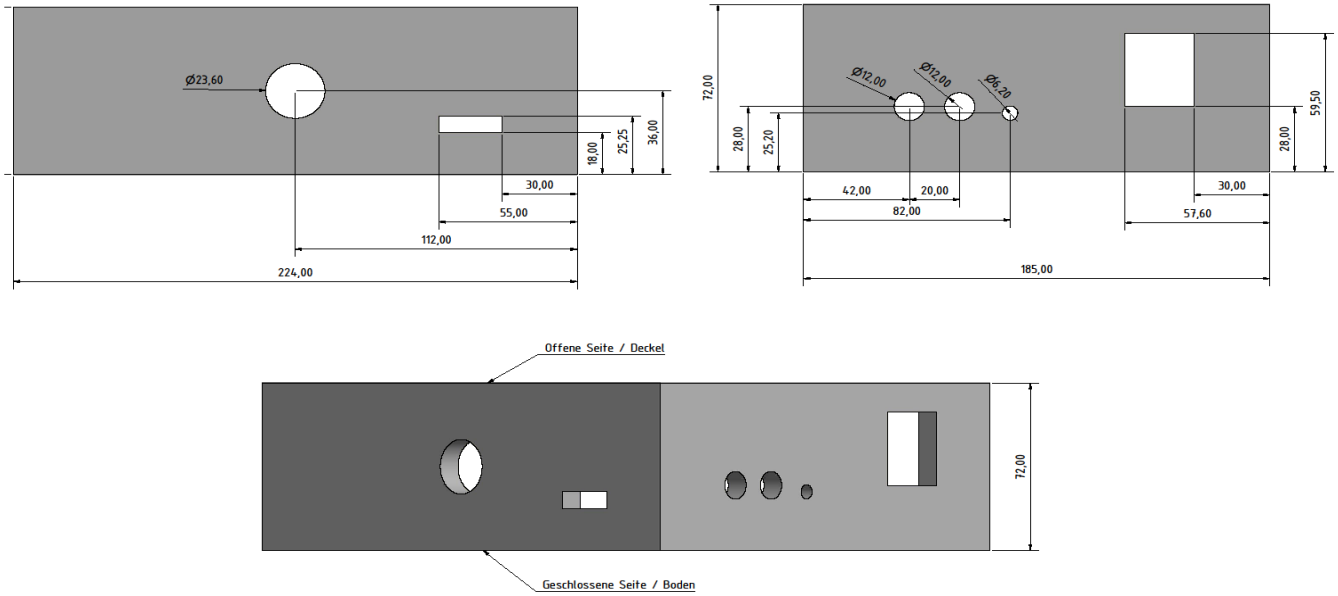

Abbildung 23 - Einschnitte [1]

Abbildung 23 zeigt die technische Zeichnung, welche angefertigt wurde, um die darin verzeichneten Einschnitte und Bohrungen auszuführen. Die Einschnitte wurden entsprechend der Größe der bestellten Komponenten bemessen. Die in Abbildung 24, Abbildung 25 und Abbildung 26 dargestellten Verbindungsschnittstellen veranschaulichen die eingebauten Komponenten. Abbildung 24 zeigt die verwendeten Buchsen, die USB-Buchse und die IEC-Buchse. Die USB-Buchse wurde zur Verbindung des Mikrocontrollers mit einem Computer eingelassen. Die IEC-Buchse versorgt das Netzteil mit Spannung und beinhaltet eine Sicherung, welche bei einem Kurzschluss durchbrennt. In Abbildung 25 sind die koaxiale Verlängerung für die GNSS-Antenne und die Mikro-SD-Karten-Verlängerung für die SD-Karten-Schnittstelle zu sehen. Durch die Verlängerung ist ein direkter Einschub am Gehäuse vorhanden, worüber die Schnittstellen verbunden werden können. Abbildung 26 zeigt links die USB-Verlängerung, die den Mikrocontroller mit der in Abbildung 24 dargestellten USB-Buchse verbindet.

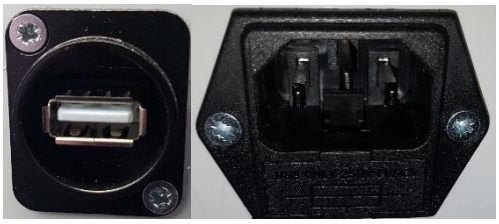

Abbildung 24 - USB&IEC - Buchse [1]

Abbildung 25 - SD- & Antennen-Verlängerung [1]

Abbildung 26 - USB-Verlängerung [18] & Messbuchse [1]

Links sieht man eine Sicherheitsbuchse, über welche die Messungen erfolgen. Hier wird die zu messende Wechselspannungsquelle angeschlossen.

2.3.3. Befestigungsplatte

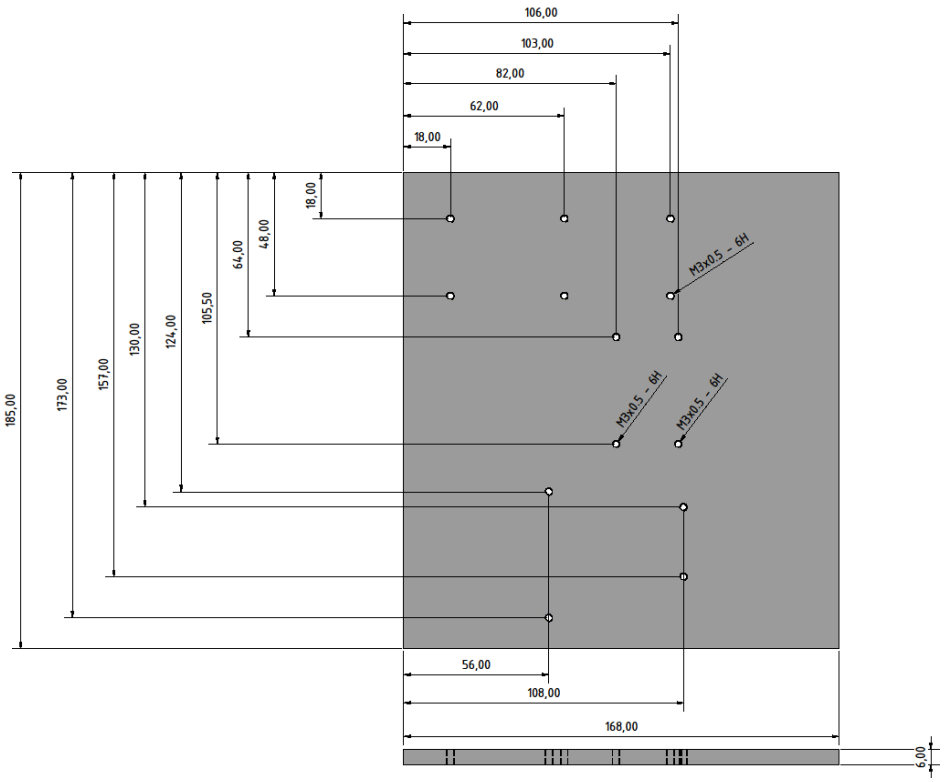

Abbildung 27 - Befestigungsplatte [1]

Um die verwendete Hardware in dem Gehäuse zu fixieren, wurde eine Acrylplatte angefertigt, worauf der Spannungswandler, das GNSS-Modul, der Mikrocontroller und das Netzteil befestigt wurden. Entsprechend der Bohrlöcher der Module wurden die in Abbildung 27 dargestellten Bohrungen platziert. In die Bohrungen wurde jeweils ein M3-Gewinde geschnitten. Die Platte wurde mit 6 mm Dicke der Gewindelänge der Abstandbolzen angepasst. Die Wahl fiel auf 10 mm lange Abstandhalter aus Nylon, welche die Module 10 mm über die Platte anheben. Die Platte passt passgenau in das Gehäuse und wurde mit ihm durch selbstschneidende Schrauben an den Eckpunkten verbunden

2.3.4. Messsystem

Abbildung 28 zeigt das fertiggestellte Messsystem. Dort sind die entwickelten Messgeräte einzeln untereinander zu sehen. Die Zuordnung der Komponenten wird farbig veranschaulicht. Im blau markierten Rechteck ist der Spannungswandler aus Kapitel 2.2.4 zu sehen. Das rote Rechteck stellt das Zeiterfassungsmodul aus Kapitel 2.2.2 dar. Rechts daneben, im grünen Rechteck befindet sich die Entwicklungsplatine aus Kapitel 2.2.1. Im oberen Sektor des Geräts, im lilafarbenen Rechteck ist das Netzteil (Kapitel 2.2.3) befestigt.

Abbildung 29 stellt ein Messgerät aus der Seitenperspektive dar. Dort sind die verbauten Schnittstellen zu sehen, welche in Kapitel 2.3.2 beschrieben wurden. Außerdem sieht man den Deckel des Gehäuses, welcher den Benutzer daran hindern soll, in das eingeschaltete Messgerät zu fassen. Dieser ist notwendig, da im eingeschalteten Gerät eine Netzspannung anliegt, welche bei Berührung einen tödlichen Stromschlag versetzen kann. Zur Programmierung muss der Deckel jedoch geöffnet werden, damit man auf die Programmierschnittstelle mit dem Programmieradapter zugreifen kann.

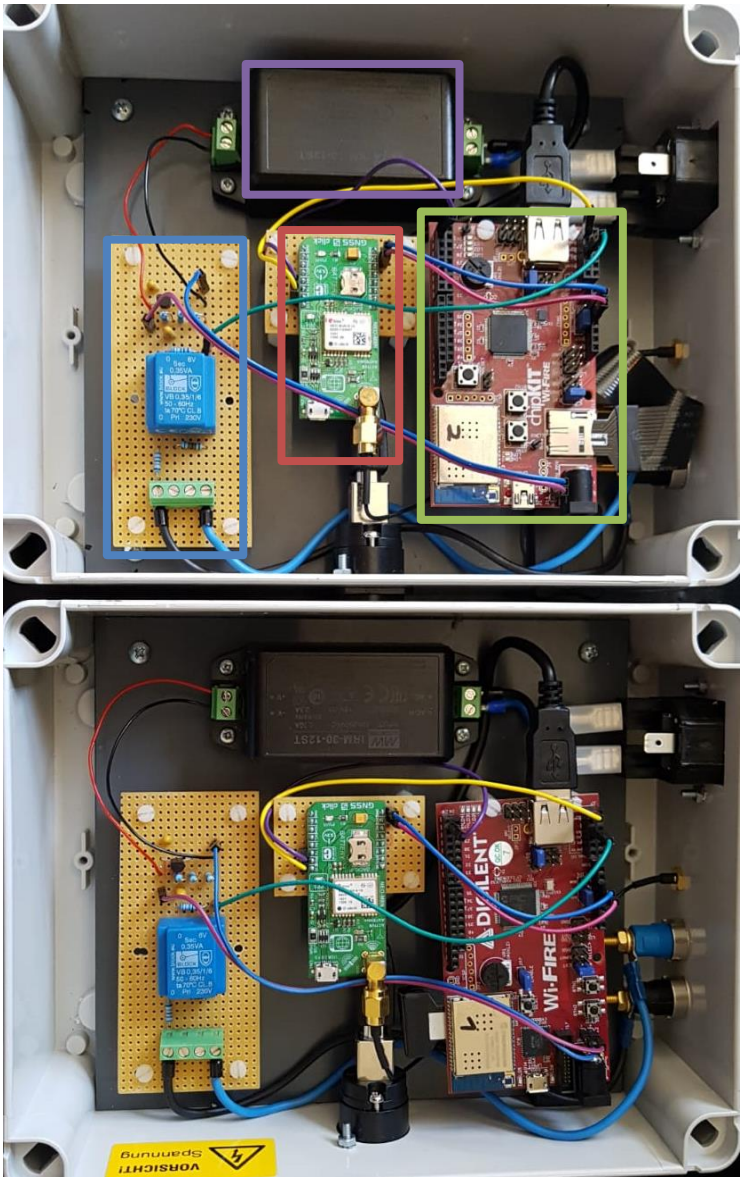

Abbildung 28 - Beide Messgeräte bei geöffnetem Deckel. Zusammen bilden sie das Messsystem [1]

Abbildung 29 - Seitenansicht eines Messgeräts [1]

2.4. Software

Das Programm wurde in drei Dateien erstellt. In der Hauptdatei (Main.c) befindet sich der Programmablauf, der in Gang kommt, sobald das Gerät mit Strom versorgt wird (Zeile 1 bis 23).

```
1.  void main(){

            //Initialisiert die Systemeinstellungen - z.B. Taktfrequenz der peripheren Buse
2.          SYSTEM_Init();
            //Initialisiert die Interrupts - UART4, ADC1, IC2
3.          INTERRUPT_Init();
            //Initialisiert die Timer für IC2, ADC1 - T2(IC2), T5(ADC1
4.          TIMER_Init();
            //Initialisiert die SD-Karte - SPI3
5.          SD_Init();
            //Initialisiert das GNSS-Modul - UART4
6.          UART4_Init();
            //Initialisiert das Spannungsmessungsmodul - ADC1
7.          ADC1_Init();
            //Initialisiert das Input Capture Modul - IC2
8.          IC2_Init();
            //Hauptschleife - Wird permanent ausgeführt

            /* Endlosschleife */
9.          while (1){
            //Die Bedingung ist erfüllt, wenn der GNSS-Datensatz komplett ist
10.                 if(a == AnzGNSS){
11.                         UART_Read();   //Wertet den GNSS-Datensatz aus
12.                 }
            //Die Bedingung ist erfüllt, wenn die Anzahl der aufzunehmenden Messwerte erreicht worden ist.
13.                 if(e == anz){
            //Schaltet das Messinterrupt aus.
14.                         IEC1bits.ADCD1IE = 0;
            //Vergleicht welche Speicherform in der Header Datei "defines.h" ausgewählt worden ist.
15.                         if (Darstellung_SD == 0){
            //Speichert die berechneten Werte auf die SD-Karte
16.                                 calculate();
17.                         }
18.                         else{
            //Speichert die Rohwerte auf die SD-Karte,
19.                                 SD_DataSave_RohWerte();
20.                         }
21.                 }
22.         }

23.}
```

Die verwendeten Module, die zur Messung nötigt sind, werden zwischen Zeile 2 und 8 initialisiert. Die Funktionen zur Initialisierung befinden sich in der zweiten Datei, SYSTEM.c. Sie wurden im Bericht zum Praxisprojekt, mit Ausnahme des Input-Capture-Moduls (Zeile 8), detailliert beschrieben. Nach der Initialisierung startet die in Zeile 9 bis 22 dargestellte Hauptschleife, deren Prozess solange durchgeführt wird, bis man das Gerät vom Stromnetz trennt. In der Hauptschleife wird die Funktion „UART_Read()“ aufgerufen, sobald der Datensatz des GNSS- Moduls, welcher durch ein Interrupt aufgenommen wird, vollständig ist (Zeile 10 bis 12). In der Funktion „UART_Read()“ werden die Uhrzeit und das Datum aus dem Datensatz gefiltert. Wenn der Messzeitpunkt erreicht ist, wird in der Funktion eine Kennzeichnung gesetzt. Diese Kennzeichnung löst in dem Input-Capture-Modul eine Bedingung aus, welche daraufhin das Messinterrupt startet. Sobald die Anzahl der zu messenden Messpunkte aufgenommen ist (Zeile 13), wird das Messinterrupt gestoppt (Zeile 14). Im nächsten Schritt wird verglichen, welcher Speichermodus ausgewählt wurde. Der Speichermodus lässt sich in der dritten Datei einstellen (defines.h). Es sind zwei verschiedene Speicherformen möglich. Die erste Speicherform (Zeile 15) beinhaltet die Umrechnung der Messwerte. Die aufgenommenen Messwerte werden in den folgenden Kapiteln (2.4.4 & 2.4.5.1) genauer erklärt, berechnet und anschließend auf der SD-Karte abgespeichert (Zeile 16). Die zweite Speicherform (Zeile 18, Kapitel 2.4.5.2) speichert die Messwerte direkt auf der SD-Karte ab. Sie eignet sich zur Fehlerquellenermittlung, da dort die Ursprungswerte direkt miteinander verglichen werden können.

2.4.1. Referenzzeitermittlung

Zur Ermittlung der Phasenverschiebung zwischen den Messgeräten ist es notwendig, einen Referenzzeitpunkt zu wählen, welcher für beide Messgeräte identisch ist. Dazu wurden zwei Varianten erstellt, welche auf die eingehende GNSS-Taktflanke reagieren und daraufhin die Messung starten. Die ursprüngliche Variante wurde mit einem ADC-Modul und einem Timer realisiert. Die zweite Variante wurde mit dem Input-Capture-Modul des Mikrocontrollers verwirklicht.

2.4.1.1. 1. Methode: ADC & Timer Modul

Mit einem Timer und folgendem Software-Code wurde die ursprüngliche Variante umgesetzt.

```
        /* Initialisierung von Timer 2 - Zählt die Zeit zwischen der GNSS-Puls-pro-Sekunde-Flanke und
        der ersten Nullstelle */
        //Der Timer greift auf PBCLK3 zu. PBCLK entspricht 100MHz.
24.     T2CONbits.TON = 0;          //Timer wird zur Konfiguration ausgeschaltet.
25.     T2CONbits.TCKPS = 0b101;    //Timer Frequenz wird durch 32 geteilt.
26.     PR2 = 62499                 //Es wird eine Periode von 20ms definiert.
27.     T2CONbits.TON = 1;          //Timer wird eingeschaltet.
```

Der Timer wurde gemäß der maximalen Zeit konfiguriert, welche erreicht werden kann, bis die erste Nullstelle in Erscheinung tritt. Darum wurde der Timer mit der bekannten Netzfrequenz versehen. Diese entspricht 50 Hz. Innerhalb einer Periode müssen in der Netzspannung zwei Nullstellen aufzufinden sein. Durch Setzung des PR-Bits (Zeile 26) lässt sich die Frequenz mithilfe von Formel 4 genau einstellen. In der Zeile 27 wird der Timer aktiviert.

$$PRI = \frac{PBCLK}{PS * fg} - 1 \qquad (4)$$

PRI = Perioden-Register, $PBCLK$ = Peripherer-Bus-Takt (Peripher-Bus-Clock), PS = Vorteiler (Prescaler), fg = Frequenz gesucht

```
        /* Vergleicht, ob der eingehende GNSS-Puls eingeht */
28.     if (ADCDATA0 > 4000){
29.             IEC1bits.ADCD0IE = 0;       //Interrupts von ADC0 werden deaktiviert.
30.             IEC1bits.ADCD1IE = 1;       //Interrupts von ADC1 werden aktiviert.
31.             TMR2 = 0;                   //Timer2-Variable wird auf null gesetzt.
32.     }
```

Der in Zeile 28 bis 32 beinhaltete Programmcode befindet sich in der ADC0-Interrupt-Service-Routine. Sie empfängt die eingehende GNSS-Puls-pro-Sekunde-Flanke, sobald Interrupts für das ADC0-Modul aktiviert werden. In Zeile 28 befindet sich die Abfrage, welche den eingehenden Zeitimpuls misst. Sobald die Bedingung erfüllt ist, wird in Zeile 29 das ADC0-IEC-Bit deaktiviert. Durch diesen Schritt werden Interrupts, welche die eingehende GNSS-Puls-pro-Sekunde-Flanke messen, deaktiviert. Damit wird ein Konflikt zwischen den folgenden Interrupts vermieden. Anschließend wird in Zeile 30 das ADC1-IEC-Bit aktiviert. Im ADC1-Modul wird die Messung aufgenommen. Die Zählervariable von Timer 2 wird in Zeile 31 auf 0 gesetzt. Ab diesen Zeitpunkt läuft die Referenzzeit.

```
    /* Speichert das Resultat */
33.         E[e] = ADCDATA1;
34.         if(((E[e] == 2047) || (E[e] == 2048)) && (dt == true)) {
35.             h = TMR2;
36.             dt = false;
37.         }
```

Die ADC1-Interrupt-Service-Routine führt die Spannungsmessung durch und speichert die gemessenen Werte des Datentyps Integer in Zeile 33, im Array E. Nach jedem gespeicherten Wert fragt in Zeile 34 eine Bedingung ab, ob sich dort eine Nullstelle befindet. In der Bedingung wird nach dem binären Ergebnis der Nullstelle gefragt. Sobald eine Nullstelle gefunden ist, wird in Zeile 35 die Referenzzeit in der Variablen „h“ gespeichert und die Abfrage aus Zeile 34 fragt mit der Booleschen Variablen „dt“, ob die Referenzzeit schon gespeichert ist. Die Nullstelle, nach der die Bedingung (Zeile 34) fragt soll der Offsetspannung entsprechen. Die Offset-Spannung beträgt 1,65 V. Dies entspricht einer binären Darstellung von 2047,5. Ohne den Offset würde die Spannung hinter dem Transformator maximal zwischen ±1,25 V hin und her schwingen. Durch den Offset wird die Spannung um 1,65 V angehoben. Die Nullstelle liegt somit nun bei 1,65 V und die Spannung schwingt zwischen 0,4 V und 2,9 V ($\pm 1{,}25\ V + 1{,}65\ V$) hin und her. Da die Nullstellen-Abfrage für die Berechnung des Phasenwinkels und zur präzisen Messauswertung sehr genau sein muss, wurden die Ursprungswerte der Spannungsmessung ausgewertet.
Durch folgende Formel wurde die binäre Darstellungsform der Offset-Spannung bestimmt.

$$\frac{gesuchte\ Spannung}{Mikrocontroller{-}Spannung} \times maximale\ binäre\ Auflösung = gesuchte\ binäre\ Auflösung \qquad (5)$$

Die maximale binäre Auflösung des AD-Wandlers des PIC32MZ2048EFG beträgt 12 Bit. Damit können maximal 4096 (2^{12}) Werte dargestellt werden. Der Mikrocontroller arbeitet mit einer Spannung von 3,3 V. Durch diese Werte wird der Ursprungswert der offset-Spannung ermittelt. Die Ursprungswerte des AD-Wandlers werden im Datentyp Integer gespeichert, der nur ganzzahlige Werte aufnehmen kann. Um die Nullstelle zu ermitteln, wird deshalb in der Bedingung (Zeile 34) der Bereich zwischen 2047 und 2048 gewählt.

2.4.1.2. 2. Methode: Input-Capture-Modul

Die 2. Methode nimmt die eingehende GNSS-Taktflanke mit dem Input-Capture-Modul auf. Dieses sogenannte IC-Modul löst ein Interrupt aus, sobald eine fallende und/oder steigende Flanke am deklarierten Pin erscheint.

```
    /* Input Capture 2*/
38. void IC2_Init(){

39.         ANSELE. = 0;                    //Port E wird als digital deklariert.
40.         TRISEbits.TRISE3 = 1;           //Deklariert Pin 29 als Input.
41.         IC2R = 0b0110;                  //0101 = RPE3 - digital Pin 29.

            //Schaltet das Input-Capture-Modul aus und alle Einstellmöglichkeiten auf Bit 0.
42.         IC2CON = 0;

            //011 = Simple Capture Event mode every rising edge.
43.         IC2CONbits.ICM0 = 1;
44.         IC2CONbits.ICM1 = 1;

45.         IC2CONbits.ICTMR = 1;           //Timer 2 wird vom IC Modul genutzt.
46.  I      C2CONbits.ON = 1;               //Schaltet IC Modul ein.

47. }
```

Das IC Modul greift auf den 29 Pin des Mikrocontrollers zu. Die nötigen Zugriffsberechtigungen werden in Zeile 39 bis 41 dargestellt. In Zeile 42 werden die Bits des IC-Moduls standardmäßig auf 0 gesetzt. Die Zeilen 43 und 44 definieren, dass das IC-Modul das IC-Interrupt auslöst, sobald eine steigende Flanke eingeht. In Zeile 45 wird konfiguriert, auf welchen Timer das IC-Modul zugreift. Anschließend wird in Zeile 46 das IC-Modul aktiviert.

```
    /*IC2 - GNSS Impuls pro Sekunde -Input Capture Interrupt */
48. void __ISR_AT_VECTOR(_INPUT_CAPTURE_2_VECTOR, IPL6SRS) input_capture_2_Interrupt(){

            //Sorgt dafür, dass die Zeit ermittelt werden kann.
49.         if(a>AnzGNSS){
50.                 a=0;
51.         }

            //Die Messung wird gestartet, wenn die festgelegte Uhrzeit erreicht worden ist.
52.         if(string_compare == true){
53.                 string_compare = false;
54.                 IEC1bits.ADCD1IE = 1;   //Schaltet das Messinterrupt an.
55.         }

56.         LED_LD4_INV()           //Lässt LED4 ein und ausgehen pro IC.
57.         time_IC2 = IC2BUF;      //Notwendig für Input Capture Interrupt.
58.         TMR5 = 0;               //Setzt das Timer 5 Register auf null.

            //Die Interrupt-Kennzeichnung wird zurückgesetzt, weiterer Interrupt-Empfang wird ermöglicht.
59.         IFS0bits.IC2IF = 0;
60. }
```

Die IC-Interrupt-Service-Routine wird jedes Mal bei eingehender GNSS-Taktflanke ausgeführt. In Zeile 49 bis 51 werden die eingegangenen GNSS-Daten zurückgesetzt. Dadurch wird gewährleistet, dass zu jeder Sekunde ein neuer GNSS-Datensatz empfangen werden kann. Zwischen Zeile 52 und 55 findet die Bedingungsabfrage statt, welche die Messung startet. Die Boolesche Variable „string_compare“ befindet sich in der Funktion, welche die Uhrzeit ermittelt, und wird auf „true“ gesetzt, wenn die vordefinierte Uhrzeit erreicht wird. Anschließend wird bei der nächsten eingehenden GNSS-Taktflanke die Messung mit Setzung des Messinterrupts in Zeile 54 gestartet. Durch Zeile 56 geht bei eingehendem Impuls im dauernden Wechsel die 4. LED des Mikrocontrollers an und aus. Zeile 57 speichert die Zeit des IC-Moduls ab. In Zeile 58 wird das Timer-Modul auf null gesetzt. Dadurch wird sichergestellt, dass beide Messgeräte den Messwert mit der selben Taktfrequenz aufnehmen.

2.4.2. Konfigurationsmöglichkeiten

```
61. #define SYS_FREQ 200000000          // Running at 200MHz.
62. #define PBCLK SYS_FREQ/2            // Timer-Frequenz = 100MHz.
63. #define AnzGNSS 707                 // 707 Zeichen pro empfangenen GNSS-Datensatz.

    /* veränderbare Werte */
64. #define AbtastFrequenz 1000000      //1MHz.
    //Speichert die Anzahl der eingestellten Werte in RAM ab. Sobald die Anzahl erreicht ist, wird auf
    die SD-Karte gespeichert.
65. #define AnzahlWieVielePerioden 5
    //Zurzeit wird alle 10 Sekunden eine Aufnahme gestartet. Daraus folgen 360 Zugriffe auf die SD
    Karte pro Stunde.
66. #define anzTime 6
    //0 = Verarbeitete Werte werden auf der SD-Karte gespeichert,
    1 = Rohwerte werden auf der SD-Karte gespeichert.
67. #define Darstellung_SD 0
    //Zur Kalibrierung ist entsprechend des Anschlusses Gerät 1 oder 2 auszuwählen.
68. #define Geraet 2

    /* feste Werte */
69. #define anzPerioden (AnzahlWieVielePerioden+1)
70. #define Pmal2 (AnzahlWieVielePerioden*2)
71. #define (AbtastungProPeriode AbtastFrequenz/50)
72. #define (anz AbtastungProPeriode*anzPerioden)
73. #define anzString 20      //Maximale Zeichen Anzahl der Strings, welche abgespeicherten werden.
    //Anzahl der Werte, aus denen die Regressionsgerade gebildet wird.
74. #define Reg_anz (anz/(5*anzPerioden))
75. #define Messwerte (Reg_anz/2)
```

Der Programmkopf enthält verschiedene Einstellmöglichkeiten. In Zeile 61 wird die Systemfrequenz definiert, mit der der Mikrocontroller arbeitet. Alle weiteren Konfigurationen sind von der Systemfrequenz abhängig. Zeile 62 definiert die Frequenz der internen peripheren Busse im Mikrocontroller. Wie in Zeile 62 dargestellt, beträgt die Bus-Frequenz immer die Hälfte der Systemfrequenz. In Zeile 63 wird die Anzahl der GNSS-Werte definiert. Da ein GNSS-Datensatz immer aus 707 Werten besteht, wurde die Länge entsprechend konfiguriert Die Definitionen in den Zeilen 61 und 63 sowie 69 bis 74 sollten nicht verändert werden, da auf ihnen weitere Berechnungen beruhen und es bei einer Veränderung zu Konflikten kommen kann. Der in den Zeilen 64 bis 73 dargestellte Programmcode ist variabel und kann, je nach Bedürfnis, verändert werden. Zeile 64 stellt die Frequenz ein, mit der der Mikrocontroller die Spannung abtastet. In Zeile 65 wird definiert, wie viele Perioden pro Zeitimpuls aufgenommen werden. Zeile 74 gibt an, aus wie vielen Werten die Regressionsgerade gebildet werden soll.

Die Variable „Messwerte“ aus Zeile 75 schreibt vor, mit welchen Messwerten die Berechnungsfunktion anfangen soll. Falls sich unter den ersten aufgenommenen Messwerten eine unpräzise Nullstelle befindet, würde die Funktion Regression nicht ordnungsgemäß ablaufen, da sie viele Werte sowohl vor und wie nach dem Erscheinen dieser Nullstelle auswertet. Möglicherweise würde die Funktion bei einer zu frühen Nullstelle auf Werte zuzugreifen, welche nicht existieren, wodurch es zu Konflikten im Mikrocontroller käme. Aus diesem Grund ist die Variable an die Anzahl der Werte, aus denen die Regressionsgerade gebildet wird, gekoppelt.
Eine Veränderung der festen Werte führt in der Regel zu Konflikten mit systeminternen Prozessen. Vor allem der Arbeitsspeicher würde bei einer Erhöhung derart an seine Grenzen stoßen, dass eine Programmierung durch die Programmierumgebung verhindert würde. In Kapitel 2.1.3 ist beschrieben, wie hoch der Arbeitsspeicher maximal belastet werden kann.

2.4.2.1. Abtastfrequenz

Die Abtastfrequenz wird in Zeile 64 definiert. Um die Mindestabtastfrequenz zu ermitteln, wird von folgender Annahme ausgegangen: Die Netzspannung liefert eine Wechselspannung von $\pm 325{,}27\ V$ $(\sqrt{2} * 230\ V)$. Dazwischen liegen ca. 650 Spannungswerte. Um die maximale Genauigkeit des A/D-Wandlers zu definieren, muss der maximale Wertebereich durch die maximale Auflösung geteilt werden $(650\ V/4096)$. Das entspricht einer Auflösung von ca. 158,7 mV pro Bit. Der Spannungswandler wurde so dimensioniert, dass die Spannung, welche am Mikrocontroller anliegt, unterhalb der maximalen Mikrocontroller-Spannung und oberhalb der negativen Spannungsanteile liegt. In einem Spannungsbereich von -329 V bis +334 V liegt der Wertebereich hinter dem Spannungsteiler und dem Offset zurzeit zwischen den 898 und 3249 Bit. Dazwischen liegen 663 $(329 + 334)$ Spannungswerte in Volt und 2351 $(3249 - 898)$ binäre Werte. Daraus erschließt sich eine Genauigkeit von 282mV $(663/2351)$ pro Bit.

$$maximale\ Auflösung = \frac{Spannungsbereich}{Spannungswerte} \tag{6}$$

$$Spannugsbereich = 663\ V,\ Spannungswerte = 2351,$$

$$maximale\ Auflösung = 282\ mV$$

In Bezug auf die Feststellung, dass die maximale Änderung des vorliegenden Spannungswandlers 282 mV pro Bit beträgt, sollte die Frequenz so eingestellt sein, dass sie mit einer Abtastrate definiert wird, welche mindestens der Genauigkeit von 282 mV entspricht. Mithilfe des Dreisatzes wurde die minimale Abtastfrequenz für eine linear ansteigende und fallende Wechselspannung bestimmt.

Tabelle 1 – Minimale Abtastfrequenz [1]

Spannung/V	Zeit/s	Frequenz/Hz
653	0,02	50
0,282	8,6371E-6	115,7801E+3

$$Zeitänderung = \frac{Spannungsänderung \times maximale\ Zeit}{maximale\ Spannung} \tag{7}$$

$$maximale\ Zeit = 20\ ms, maximale\ Spannungsänderung = 653\ V$$

$$Spannungsänderung = 282\ mV$$

$$minimale\ Abtastfrequenz = \frac{1}{Zeitänderung} \tag{8}$$

$$Zeitänderung = 8{,}637\ \mu s$$

Bei einer linearen Steigung wie in Abbildung 30 steigt oder fällt die Spannung alle 8,367 µs um 282 mV und wechselt ihre Polarität, sobald ein Maximum oder Minimum erreicht wird.

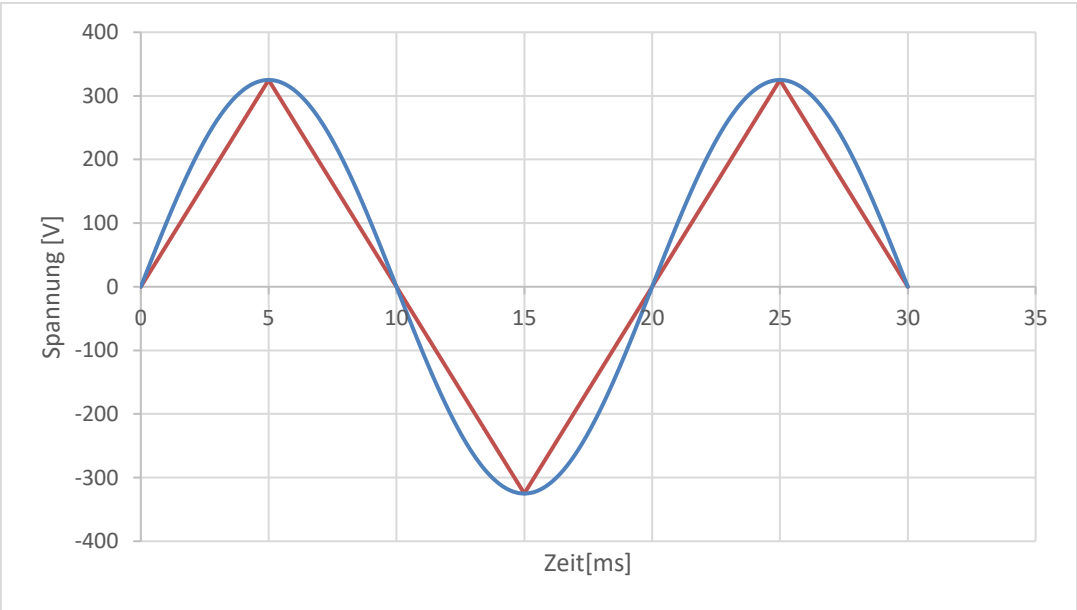

Abbildung 30 - Dreieckspannung & Sinusspannung [1]

Im Mittelwert hat die Netzspannung denselben Spannungsanstieg wie die Dreieckspannung, jedoch ändert sich die Steigung entsprechend der Messstelle. Deshalb ist für eine möglichst genaue Darstellung der gemessenen Datenkurve eine hohe Abtastfrequenz notwendig, für welche die berechneten 115 kHz nicht ausreichen. Durch unberechenbare Netzrückwirkungen (Kapitel 2.1.4.1) kommt es zu schnellen unkontrollierbaren Spannungswechseln, die oftmals die Spannung um mehr als 278mV pro 8,637 µs erhöhen. Abbildung 31 zeigt eine Nahaufnahme der Netzspannung.

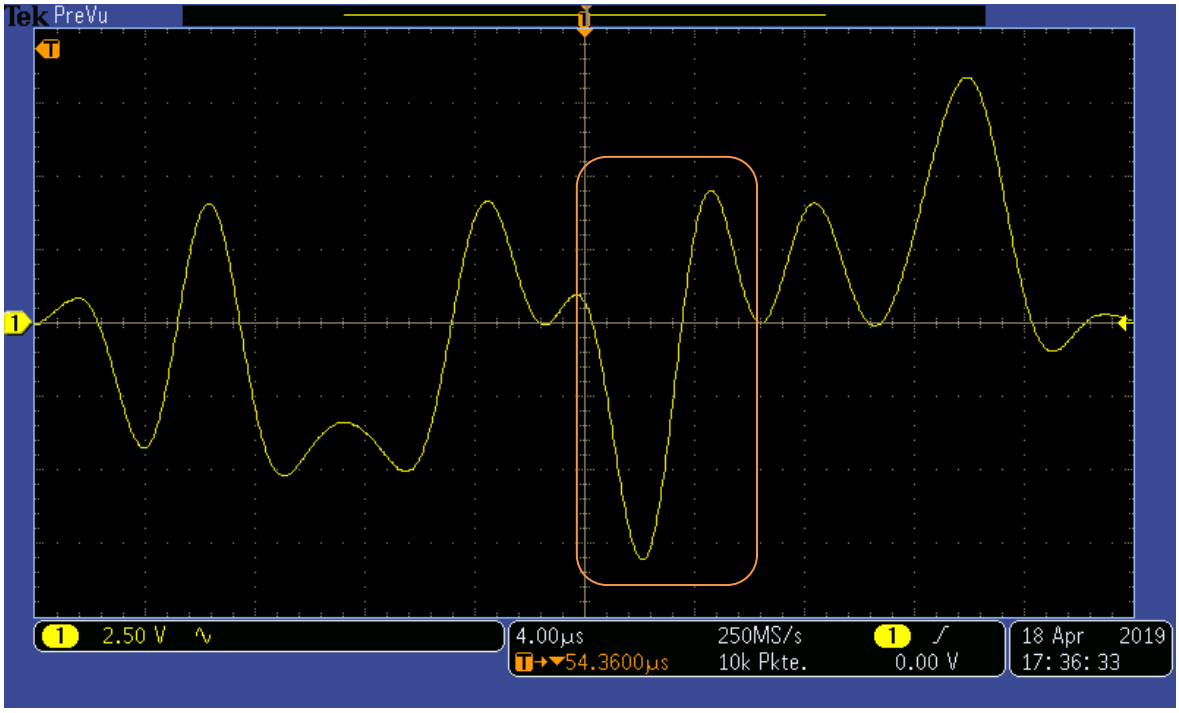

Abbildung 31 - Schwingungen in der Netzspannung [1]

Abbildung 31 verdeutlicht, dass eine viel höhere Frequenz notwendig ist, um die Netzspannung detailliert aufzunehmen. In x-Richtung findet eine Zeitänderung von 4 µs pro Dekade statt. In y-Richtung ändert sich die Spannung um 2,5 V pro Dekade. Innerhalb der orangefarbenen Markierung sieht man, wie sich die Netzspannung um bis zu 12 V pro 4 µs ändert. Diese Spannungsschwankung lässt sich mit der berechneten Frequenz nicht wiedergeben. Aus diesem Grund wurde ein Kompromiss zwischen einer hohen Frequenz und der Anzahl der aufgenommenen Perioden gewählt, welcher den Arbeitsspeicher zu fast 100 % ausreizt. Die Wahl fiel auf eine Abtastfrequenz von 1 MHz und fünf Perioden pro empfangenem Referenzzeitpunkt. Dadurch werden 20.000 Messungen pro Periode durchgeführt. Die Abtastgenauigkeit beträgt eine µs. Mit dieser Genauigkeit lassen sich die Spannungsschwankungen darstellen. Eine detailgetreue Auswertung der Netzspannung ist damit gewährleistet.

2.4.3. Startpunkt der Messung

Die Messungen werden zu einem genauen Startzeitpunkt begonnen. Dieser ist von der eingehenden Zeit abhängig und wird mit folgendem Code implementiert.

```
76. unsigned char gnss_time[6];                    //Hier wird der Messzeitpunkt gespeichert.
77. unsigned char gnss_time_max[6] = "235950";     //Maximale Uhrzeit.

78. void change_time(){

            /* Sonderform 23:59:50 Uhr */
79.         if ((gnss_time[0] == 2) && (gnss_time[1] == 3) && (gnss_time[2] == 5) && (gnss_time[3] == 9)
            && (gnss_time[4] == 5)){
80.                 for (c=0;c<6;c++) {
81.                         gnss_time[c] = 0;
82.                 }
83.         }
            /* Sonderform 09:59:50 Uhr oder 19:59:50 Uhr*/
84.         if ((gnss_time[1] == 3) && (gnss_time[2] == 5) && (gnss_time[3] == 9) && (gnss_time[4] == 5)){
85.                 gnss_time[0] = gnss_time[0] + 1;
86.                 gnss_time[1] = gnss_time_max[5];
87.                 gnss_time[2] = gnss_time_max[5];
88.                 gnss_time[3] = gnss_time_max[5];
89.                 gnss_time[4] = gnss_time_max[5];
90.         }
            /* Sonderform 00:09:50 Uhr, Erhöhung in einzelnen Stunden */
91.         if ((gnss_time[2] == 5) && (gnss_time[3] == 9) && (gnss_time[4] == 5)){
92.                 gnss_time[1] = gnss_time[1] + 1;
93.                 gnss_time[2] = gnss_time_max[5];
94.                 gnss_time[3] = gnss_time_max[5];
95.                 gnss_time[4] = gnss_time_max[5];
96.         }
            /* Sonderform 00:01:50 Uhr, Erhöhung alle 10 Minuten */
97.         if ((gnss_time[3] == gnss_time_max[3]) && (gnss_time[4] == gnss_time_max[4])){
98.                 gnss_time[2] = gnss_time[2] + 1;
99.                 gnss_time[3] = gnss_time_max[5];
100.                gnss_time[4] = gnss_time_max[5];
101.        }
            /* Sonderform 00:00:50 Uhr, Erhöhung in einzelnen Minuten */
102.        if (gnss_time[4] == gnss_time_max[4]){
103.                gnss_time[3] = gnss_time[3] + 1;
104.                gnss_time[4] = gnss_time_max[5];
105.        }
            /*Wenn keine der oberen Bedingungen zutrifft, Erhöhung alle 10 Sekunden */
106.        else {
107.                gnss_time[4] = gnss_time[4] + 1;
108.        }
109. }              //Funktionsende
```

Tabelle 2 - Funktionsvorgang, Bestimmung des Messzeitpunktes [1]

Wenn	Dann	Zeile
23:59:50	00:00:00	79 - 83
22:59:50	23:00:00	91 - 96
21:59:50	22:00:00	
20:59:50	21:00:00	
19:59:50	20:00:00	84 - 90
18:59:50	19:00:00	91 - 96
17:59:50	18:00:00	
16:59:50	17:00:00	
15:59:50	16:00:00	
14:59:50	15:00:00	
13:59:50	14:00:00	
12:59:50	13:00:00	
11:59:50	12:00:00	
10:59:50	11:00:00	
09:59:50	10:00:00	84 - 90
08:59:50	09:00:00	91 - 96
07:59:50	08:00:00	
06:59:50	07:00:00	
05:59:50	06:00:00	
04:59:50	05:00:00	
03:59:50	04:00:00	
02:59:50	03:00:00	
01:59:50	02:00:00	
00:59:50	01:00:00	
00:49:50	00:50:00	97 - 101
00:39:50	00:40:00	
00:29:50	00:30:00	
00:19:50	00:20:00	
00:09:50	00:10:00	
00:00:50	00:01:00	102 - 105

Diese im vorigen Code beschriebene Funktion (Zeile 76 bis 109) dient der korrekten Erfassung des Uhrzeitformats. Tabelle 2 soll diesbezüglich die Verständlichkeit erleichtern. Die Funktion, welche ausgelöst wird, sobald der definierte Messzeitpunkt erreicht ist, fragt in dem in Tabelle 2 dargestellten Format die Uhrzeit ab. In Zeile 76 sieht man, wie die Variable, welche angibt, wann die Messung stattfinden soll, deklariert wird. Zeile 77 gibt den spätesten möglichen Messzeitpunkt eines Tages an, wenn man mindestens alle 10 Sekunden eine Messung starten möchte. Darauf wird in den Zeilen 78 bis 109 die Funktion zur Änderung des Messzeitpunktes aufgerufen. Diese Funktion wird nach jeder abgeschlossenen Messung aufgerufen. Zurzeit ist die Funktion so eingestellt, dass alle 10 Sekunden eine Messung durchgeführt wird. Die Funktion inkrementiert die Variable, welche die 10er Sekunden erfasst, in jedem Durchgang um eins. Das geht gut, solange die Darstellung nicht die 50. Sekunde überschreitet, d.h. die 60. Sekunde erreicht, womit diese Darstellung im Zeitformat ungültig und deshalb nicht auffindbar wäre. In der Tabelle ist dieser Sachverhalt im grün markierten Feld dargestellt. Eine weitere Messung würde ab diesem Zeitpunkt nicht mehr stattfinden. Aus diesem Grund wird in den Zeilen 102 bis 105 definiert, dass sich die nachfolgende Anzeige, welche die einzelnen Minuten darstellt, um eins erhöht und die Darstellung der 10er Sekunden auf null gesetzt wird. Die Sonderform für die Änderung der 10er Darstellung der Minuten ist in der Tabelle gelb markiert und exemplarisch nur für eine Stunde abgebildet. Für die einzelnen Stunden wurde der grau markierte Bereich gewählt. Bei einer Änderung der 10er Darstellung des Stundebereichs tritt die rot markierte Darstellung in Funktion. Der letzte Sonderfall wird in der blau markierten Zeile gekennzeichnet. Sobald der Tag vorbei ist, wird der Zeitpunkt zur Messaufnahme von 23:59:50 auf 00:00:00 gesetzt. Falls keine der oben genannten Sonderformen auftritt, wird die Zeit um 10 Sekunden erhöht, was mit der Funktion in den Zeilen 106 bis 108 konfiguriert wird.

2.4.4. Berechnung

Der gemessene Datensatz der Netzspannung muss zur Weiterverarbeitung ausgewertet werden. Dafür sind mehrere Berechnungsschritte notwendig, welche in diesem Kapitel aufgelistet und abgearbeitet werden.

2.4.4.1. Kalibrierung

Da die Messgeräte Abweichungen voneinander hatten, war es notwendig die Geräte zu kalibrieren. Dafür wurden die Ursprungswerte beider Messgeräte miteinander verglichen.

2.4.4.1.1. Regression

Das Rauschen, das auf der Netzspannung lag, konnte nicht ganz behoben werden. Bei näherem Hinsehen, gleichsam unter einer sehr starken Lupe, zeigt sich dieses Rauschen am Verlauf der Messkurve. Abbildung 32 zeigt den winzigen Spannungskurvenabschnitt im Nullstellenbereich, nicht als Abschnitt einer ruhigen Linie, sondern als Spannungsrippel (englisch: ripple = Kräuselung, Kräuselwelle) mit vielen starken Schwingungen und entsprechend vielen Nullstellen. Zur Ermittlung einer präzisen Nullstelle muss aus den Messwerten des Nullstellenbereichs eine Regressionsgerade errechnet werden. Abbildung 32 zeigt eine durch ein Spannungsrippel gezogene Regressionsgerade

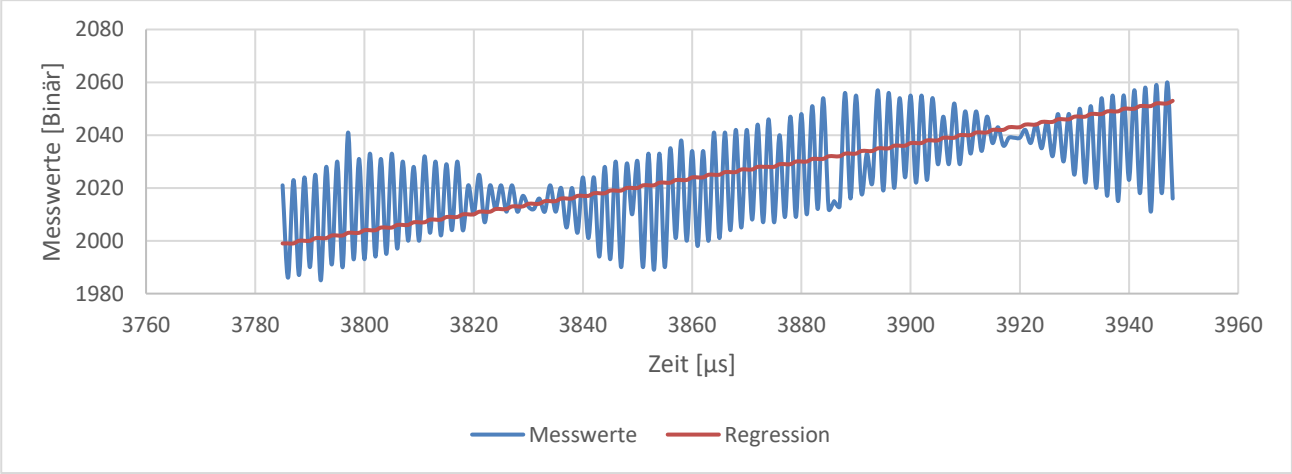

Abbildung 32 - Regressionsgerade durch die Messkurve [1]

Ohne diese Regressionsgerade waren die Abweichungen der Messgeräte voneinander sehr deutlich. Der Phasenwinkel war teilweise um ±10 º versetzt. Durch die Regressionsgerade (Abbildung 32) wurde die Phasenwinkelungenauigkeit auf unter ±0,5 º Unterschied zwischen den Messgeräten gesenkt.

```
            /* Bildet eine Regressionsgerade an den Nulldurchgängen */
110. void regression_values(){
111.        for (j=Messwerte; j<anz; j++){
112.            if(((E[j] >= 2045) && (E[j] <= 2050))){
113.            regression();
114.            j = j+Reg_anz;
115.            }
116.        }
117. }
```

Der Programmcode (Zeilen 110 bis 117) dient der Erstellung der Regressionsgeraden im Umfeld der Nullstelle. Dafür durchsucht eine Zählerschleife (Zeilen 111 bis 116) alle Messwerte, nach dem ersten Nulldurchgang ab. Sobald dieser Messwert gefunden wird, welcher größer gleich 2045 und kleiner gleich 2050 ist (Zeilen 112), wird die Funktion regression() (Zeile 113) aufgerufen. In dieser Funktion werden die Schwingungen im Nullstellenbereich als eine Linie dargestellt und abgespeichert. Anschließend wird die Zählerschleife (Zeilen 111 bis 116) um die Anzahl der veränderten Werte erhöht (Zeile 114). Dadurch wird vermieden, dass die Zählerschleife die regressierten Werte nach einem Nulldurchgang weiter durchsucht.

2.4.4.1.2. Regressionsgerade

Angesichts der Schwingungen der Netzspannung, welche exemplarisch in Abbildung 33 dargestellt sind, ist es ,wie gesagt, nötig, zur möglichst genauen Ermittlung der jeweiligen präzisen Nullstelle eine Regressionsgerade aus den Messwerten des Spannungsrippels zu bilden, welches die Nullstelle umgibt.

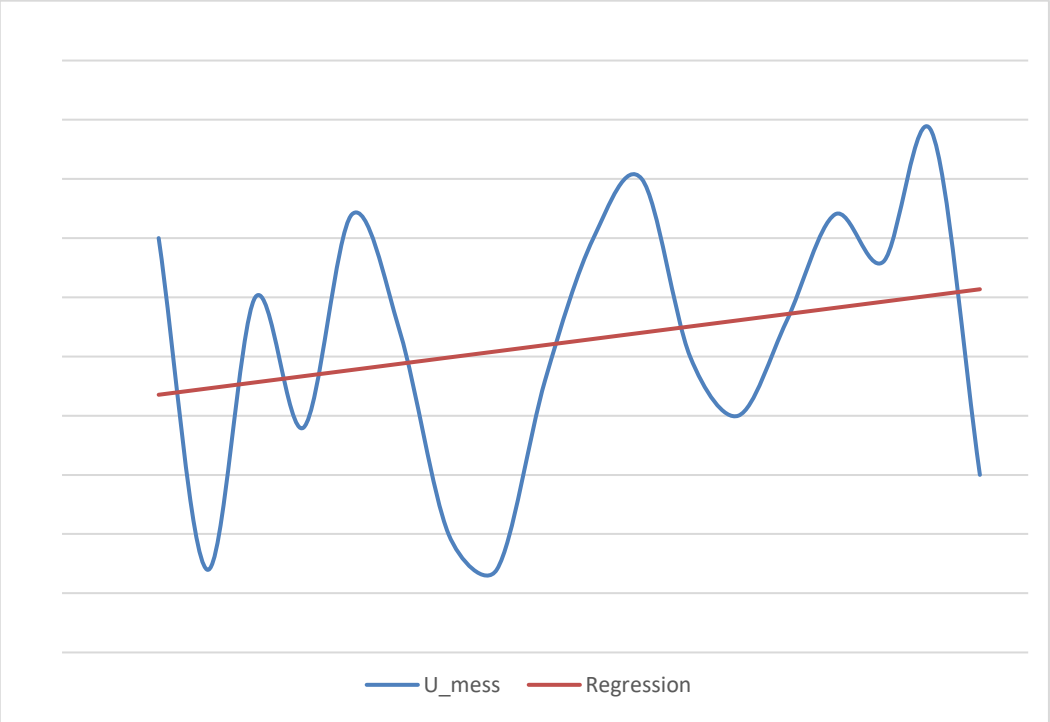

Abbildung 33 - Beispiel, Erstellung einer Regressionsgeraden. [1]

Eine Regression ist eine statistische Methode, um einen Datensatz mit schwankenden Werten als eine Gerade darzustellen. Abbildung 33 ist ein Beispiel, wie die Regressionsgerade eines Datensatzes mit schwingenden Werten gebildet wird. Zur Bildung der Regressionsgeraden muss die Steigung (Formel 9) der Messwerte bestimmt werden.

$$b = \frac{\sum(x-\bar{x})(y-\bar{y})}{\sum(x-\bar{x})^2} \tag{9}$$

Aus der Steigung wird nun der Achsenabschnitt (Formel 10) bestimmt.

$$a = \bar{y} - b\bar{x} \tag{10}$$

Mit diesen beiden Formeln (Formel 9 und Formel 10) lässt sich nun die Regressionsgerade (Formel 11) berechnen.

$$Ry = a + b \times y \tag{11}$$

Mithilfe der Regressionsgeraden lassen sich nun exakte, gleichmäßig steigende Werte der schwankenden Netzspannung darstellen. Dadurch ist eine höchst präzise Frequenzermittlung möglich. Der folgende Programmcode beschreibt die Bestimmung des Schnittpunktes der x-Achse mit der Regressionsgeraden.

```
118. void regression(){

119.        anz_steigung_start = j-(Reg_anz/2);
120.        anz_steigung_stop = j+(Reg_anz/2);

            /* Mittelwert x-Achse (Messwerte) */
121.        Reg_i = 0;
122.        for(z = anz_steigung_start; z < anz_steigung_stop; z++){
123.                Reg_i = Reg_i + E[z];
124.        }
125.        Reg_f = (float)Reg_i;
126.        average_fx = Reg_f/Reg_anz;

            /* Mittelwert y-Achse (Anzahl) */
127.        Reg_i = 0;
128.        for(z = anz_steigung_start; z < anz_steigung_stop; z++){
129.                Reg_i = Reg_i + z;
130.        }
131.        Reg_f = (float)Reg_i;
132.        average_fy = Reg_f/Reg_anz;

            /*xi, x – x (Mittelwert) */
133.        c=0;
134.        for(z = anz_steigung_start; z < anz_steigung_stop; z++){
135.                xi[c] = (float)E[z];
136.                xi[c] = xi[c] - average_fx;
137.                c++;
138.        }

            /*yi, y – y (Mittelwert) */
139.        c=0;
140.        for(z = anz_steigung_start; z < anz_steigung_stop; z++){
141.                yi[c] = (float)z;
142.                yi[c] = yi[c] - average_fy;
143.                c++;
144.        }

            /* Summe aus xi × yi */
145.        Sum_Xi_Mal_Yi = 0;
146.        for(z = 0; z < Reg_anz; z++){
147.                yi[z] = xi[z] * yi[z];
148.                Sum_Xi_Mal_Yi = Sum_Xi_Mal_Yi + yi[z];
149.        }
```

```
                /* Summe aus xi^2 */
150.            Sum_Xi_Quadrat = 0;
151.            for(z = 0; z < Reg_anz; z++){
152.                    xi[z] = xi[z] * xi[z];
153.                    Sum_Xi_Quadrat = Sum_Xi_Quadrat + xi[z];
154.            }

        /*Steigung = Summe (xi × yi) / Summe (xi^2)*/
155.    Steigung = (Sum_Xi_Mal_Yi/Sum_Xi_Quadrat)/10;
156.
        /*Achsenabschnitt = Mittelwert x-Achse - Steigung × Mittelwert y-Achse*/
157.    Achsenabschnitt_f = average_fx - (Steigung * average_fy);
158.
        /*Regressionswerte*/
159.    for(z = anz_steigung_start; z < anz_steigung_stop; z++){
160.    E[z] = Achsenabschnitt_f + (Steigung * z);
161.    }
162. }
```

In Zeile 119 und 120 wird definiert, mit wie vielen Werten die Regressionsgerade gebildet wird. Durch diese Zahl werden in den Zählerschleifen die Start- und Endpunkte definiert. Die Variable „Reg_anz“ wird in Kapitel 2.4.1.1, Zeile 31 konfiguriert und gibt an, mit wie vielen Messwerten die Regressionsgerade gebildet wird. Zurzeit wird die Regressionsgerade aus einem fünftel der aufgenommenen Messwerte errechnet. Durch die Anzahl der Werte, welche zur Bildung der Regressionsgeraden genutzt werden, wird die Genauigkeit ihres Schnittpunktes mit der x-Achse definiert. In den Zeilen 121 bis 132 wird der Mittelwert der Messwerte und der Mittelwert der Anzahl der Messwerte gebildet. Dies beiden Rechnungen werden zur Bestimmung der Steigung (Formel 9) und für den Achsenabschnitt (Formel 10) benötigt. Die Zeilen 133 bis 154 implementieren Berechnungsschritte, welche benötigt werden, um die Steigung zu bestimmen. Aus den Zeilen 133 bis 138 wird $(x - \bar{x})$ für jeden einzelnen Messwert berechnet. Die Zeilen 139 bis 144 zeigen den Berechnungsschritt $(y - \bar{y})$ für die Anzahl der Messwerte. In den Zeilen 145 bis 149 wird die vorausgegangen Berechnung multipliziert $(x - \bar{x})(y - \bar{y})$ und die Summe daraus gebildet $(\sum(x - \bar{x})(y - \bar{y}))$. In den Zeilen 150 bis 154 wird folgender Berechnungsschritt durchgeführt $\sum(x - \bar{x})^2$. Zeile 155 zeigt die Bestimmung der Steigung aus den Zwischenwerten, welche die Anzahl der Werte zwischen den berechneten Nulldurchgängen darstellen (Formel 9). Mithilfe der Steigung und der Mittelwerte folgt nun in Zeile 157 die Berechnung des Achsenabschnitts (Formel 10). Anschließend werden die bisherigen Messwerte in den Zeilen 159 bis 161 mit der regressierten Werten überschrieben. Dieser Vorgang wird für alle weiteren Nulldurchgänge durchgeführt. Die regressierten Werte werden im weiteren Verlauf des Programms zur Frequenzbestimmung und zur Ermittlung des Referenzzeitpunkts benötigt.

2.4.4.1.3. Anpassung

```
/* Die Abweichungen von Messgerät 2 werden an Messgerät 1 angepasst */
163.  void device_choice(){
164.      if (Geraet == 2){
              /* Steigende Flanke bestimmen */
165.          for(e = 0; e < anz; e++){
166.              if(E[e] >= E[e+100]){
167.                  e = e +100;
168.              }
169.              else{
170.                  E[e] = E[e] - 34;
171.              }
172.          }
              /* Fallende Flanke bestimmen */
173.          for(e = 0; e < anz; e++){
174.              if(E[e] >= E[e+100]){
175.                  e = e +100;
176.              }
177.              else{
178.                  E[e] = E[e] - 4;
179.              }
180.          }
181.      }
182.  }
```

Die Aufnahme der Ursprungswerte der Messkurve haben ergeben, dass die Messwertabweichungen sich bei steigenden (Zeilen 166 bis 168) und fallenden (Zeilen 174 bis 176) Flanken signifikant voneinander unterscheiden. Die Messwerte wurden so kalibriert, dass die maximale Phasenwinkelabweichung an derselben Messstelle höchsten 0,3 ° beträgt. Dafür wurden die Messwerte von Messgerät zwei (Zeile 164) an dieser Stelle verändert. Bei den steigenden Flanken wird von jedem Messwert die Zahl 34 abgezogen (Zeile 170) und auch bei den fallenden Flanken verringern sich die Messwerte durch Subtraktion einer nach dem anderen: je um die Zahl 4 (Zeile 178). In der „defines.h“-Datei muss beim Umprogrammieren das angeschlossene Gerät entsprechend ausgewählt werden (Zeile 68), da sonst die entsprechende Kalibrierung nicht stattfindet.

2.4.4.2. Abspeicherung

Damit der Arbeitsvorgang möglichst übersichtlich und nachvollziehbar ist, wurde die Abspeicherung der aufgenommenen Datensätze durch folgendes „Struct“ realisiert. Dieses ist in den Zeilen 183 bis 199 dargestellt.

```
183. struct{

184. char GNSS_DATE[6];                                          //Datum
185. char GNSS_TIME[anzTime][6];                                 //Zeit
186. float PositiveAmplitude[anzTime][anzPerioden];              //Positive Amplituden
187. float NegativeAmplitude[anzTime][anzPerioden];              //Negative Amplituden
188. float Zwischenwerte[anzTime][anzPerioden*2];                //Messanzahl zwischen den Nullstellen
189. float Frequenz[anzTime][anzPerioden];                       //Frequenz
190. float DeltaT[anzTime];                                      //Δ t
191. float Phasenwinkel[anzTime];                                //Phasenwinkel
192. float Phasenwinkel_average;                                 //Phasenwinkel Mittelwert

193. char PositiveAmplitude_String[anzTime][anzPerioden][anzString]; //Positive Amplituden - Zeichenkette
194. char NegativeAmplitude_String[anzTime][anzPerioden][anzString]; //Negative Amplituden - Zeichenkette
195. char Frequenz_String[anzTime][anzPerioden][anzString];  //Frequenz - Zeichenkette
196. char DeltaT_String[anzTime][anzString];                 // Δ t - Zeichenkette
197. char Phasenwinkel_String[anzTime][anzString];           //Phasenwinkel - Zeichenkette
198. char Phasenwinkel_average_String[anzString];            //Phasenwinkel Mittelwert - Zeichenkette

199. }results;
```

Das „Struct“ speichert alle Variablen ab, welche der Messung entnommen werden können. Im Programmschnipsel sieht man mehrere Variablen vom Datentyp Float und Char. Die Aufnahme der Zeit und des Datums erfolgt mit den Variablen in den Zeilen 184 und 185. Die Berechnungen werden mit Variablen des Datentyps Float (Zeile 186 bis 192) durchgeführt. Zur Abspeicherung auf der SD-Karte werden diese in den Datentyp Char (Zeile 193 bis 198) umgewandelt. Die Bibliotheken, welche den Zugriff auf die SD-Karte ermöglichen, speichern nur Daten vom Datentyp Char oder Integer ab. Da der Datentyp Integer keine Kommastellen darstellen kann, war es erforderlich, den Datentyp Char zu verwenden. Mit diesem Datentyp lässt sich eine genaue Darstellung gewährleisten.
Die Anzahl der aufgenommenen Zeitpunkte und Perioden wurde variabel gestaltet. Dazu wurden mehrdimensionale Arrays genutzt, welche je nach Konfigurationseinstellungen eine mehr oder weniger große Anzahl von Perioden pro Tag aufnehmen können.

2.4.4.3. Nullstelle

Die Nullstellen der Netzspannung müssen zur Frequenz-Ermittlung aus dem Datensatz herausgefiltert werden. Dafür wurde folgender Code impliziert.

```
        /* Nulldurchgang bestimmen */
200.    PUFFER = false;         //Boolesche Hilfsvariable
201.    k = 0;                  //Zählervariable - Integer
        //Geht den aufgenommenen Datensatz durch.
202.    for (j=Messwerte; j<anz; j++){
            //Bei 2047,5 befindet sich eine Nullstelle.
203.        if((E[j] == 2048) && (PUFFER == false)){

                //Werden benötigt, um die Frequenz zu ermitteln.
204.            results.Zwischenwerte[p][k] = Nullstelle;
205.            PUFFER = true;
206.            k++;
207.        }
            /*Wird benötigt, damit keine direkt folgenden Nullstellen durch
            Spannungsschwankungen aufgenommen werden*/
            //Abfrage nach einem kleineren Wert, bei fallender Flanke
208.        if(E[j] <= 1500){
209.            PUFFER = false;
210.        }
            //Abfrage nach einem größeren Wert, bei steigender Flanke
211.        if(E[j] >= 2600){
212.            PUFFER = false;
213.        }
214.    }
```

Die Berechnung der Nullstellen erfolgt durch eine Zählerschleife (Zeile 202 bis 214), welche den gesamten Datensatz nach einem Wert durchsucht, welcher gleich 2048 ist. Die Ermittlung der Nullstellen erfolgt also mit derselben Bedingung, welche in Kapitel 2.4.1.1 (Seite 28) dargestellt ist. Diese Ermittlung der Nullstelle durch die Zählerschleife ist nur deshalb genau genug, da am Anfang der Berechnungsfunktion die Funktion regression_value() aufgerufen wurde. In dieser Funktion werden die Rippelschwingungen um den Nulldurchgang herum als lineare Linie dargestellt, die nicht viele Schnittpunkte mit der x-Achse bildet, sondern nur einen, welcher die gesuchte Nullstelle ist. Nun findet eine hochpräzise Abfrage statt und das Ergebnis wird in Zeile 204 abgespeichert.

Die Boolesche Variable „PUFFER“ aus Zeile 200 befindet sich auch in der Nullstellen-Bedingung (Zeile 203) und bewirkt, dass nachdem eine präzise Nullstelle ermittelt und gespeichert worden ist, nachfolgende Nullstellenschwankungen nicht aufgenommen werden.
Dazu wird die Variable in Zeile 205 mit dem Wert „true“ deklariert. Die Boolesche Variable wird erst wieder auf „false“ gesetzt, wenn der aufgenommene Messwert kleiner gleich 1500 (Zeile 208 bis 210) oder größer gleich 2600 (Zeile 211 bis 213) ist. Mit Formel 12 lässt sich der binäre Wert in Volt umrechnen. Die Spannung ist um ± 400 mV vom Nulldurchgang entfernt. Ab diesem Zeitpunkt ist es ausgeschlossen, dass ein weiterer Nulldurchgang unkontrolliert aufgenommen wird.

$$\frac{gesuchte\ binäre\ Auflösung}{maximale\ binäre\ Auflösung} \times maximale\ Mikrocontroller\ Spannung = gesuchte\ Spannung \quad (12)$$

In Zeile 204 wird nun die Anzahl der erfolgten Messungen in einer Variablen abgespeichert. Die Variable „k“ aus Zeile 201 bewirkt, dass die Zwischenwerte nicht überschrieben, sondern einzeln abgespeichert werden. Dazu wird die Variable in Zeile 206 inkrementiert. Die Erhöhung befindet sich in der Nullstellen-Bedingung (Zeile 202 bis 214).

2.4.4.4. Frequenz

```
/* Frequenz berechnen */
//Wird durch die Abtastrate und die Nullstellen bestimmt.
215.        c=0;
216.        for (j=0; j<anzPerioden; j++){
            //Die Anzahl der Werte zwischen den steigenden und fallenden Nulldurchgängen wird ermittelt.
217.                results.Frequenz[p][j] = results.Zwischenwerte[p][c+2] – results.Zwischenwerte[p][c];
                    //Die Abtastfrequenz wird durch die Anzahl der Nullwerte geteilt, Ergebnis reale
                    Netzfrequenz der Periode.
218.                results.Frequenz[p][j] = AbtastFrequenz/results.Frequenz[p][j];
219.                c = c+2;
220.        }
```

Die Bestimmung der Frequenz erfolgt mithilfe der Zwischenwerte, welche in dem vorangegangenen Kapitel 2.4.4.3 bestimmt wurden. Mit den variablen Zwischenwerten wird die Anzahl der aufgenommenen Messwerte gespeichert, an denen eine präzise Nullstelle gefunden wurde. Bei einer Frequenz von 1MHz ($1/1MHz = 1\mu s$) werden pro Netzspannungsperiode (20ms) 20.0000 ($20ms/1\mu s$) Messwerte gespeichert.

$$Differenz = Zwischenwert\ 2 - Zwischenwert\ 1 \tag{13}$$

$$Frequenz = \frac{AbtastFrequenz}{Differenz} \tag{14}$$

Die Funktion speichert mithilfe einer Zählerschleife (Zeile 216 bis 220) alle Frequenzen für jede Periode ab. In Zeile 217 bis 217 geschieht die Differenzberechnung, wie in Formel 13 beschrieben. Die Differenzberechnung erfolgt durch die Nullstellen c0 und c2, welche in Abbildung 34 dargestellt sind. Die Frequenzbestimmung aus Formel 14 wird mit der Differenz aus Formel 13 und der definierten Abtastfrequenz aus Kapitel 2.4.1.1, Zeile 64 berechnet.

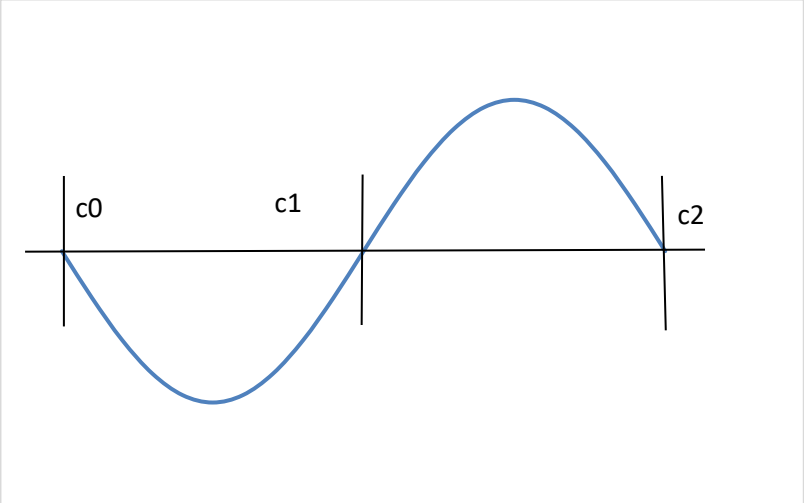

Abbildung 34 - Grafische Darstellung - Nullstellenermittlung [1]

In dem Programmcode wird die Formel in den Zeilen 218 und 218 dargestellt. Die Variable „c“ aus Zeile 215 erhöht sich in jedem Zählerschleifendurchgang um 2 (Zeile 219).
Durch diese Variable werden die Zwischenwerte der darauf folgenden Perioden definiert. Die Variable definiert, mit welchen Zwischenwerten man die Frequenz berechnet. Abbildung 34 stellt graphisch dar, wie die Zwischenwerte bestimmt und abgespeichert werden. Die Frequenz wird immer aus den ersten gefundenen präzisen Nullstellen von zwei aufeinanderfolgenden Perioden gebildet (c0 und c2). Die zweite Nullstelle der Periode (c1) wird nicht weiter benötigt, da durch sie nur die halbe Frequenz bestimmt werden kann.

2.4.4.5. Amplituden

Zur Ermittlung der Amplituden wurde folgendes Annäherungsverfahren programmiert. Die Bestimmung der positiven und der negativen Amplituden befindet sich in der Funktion „calculate“. Sie läuft in zwei voneinander getrennten Vorgängen ab, die in ihrem Funktionsprinzip jedoch identisch sind. Der grundlegende Unterschied liegt darin, dass bei der Bestimmung der positiven Amplituden ein Maximum gesucht wird, bei der der negativen Amplituden dagegen ein Minimum.

```
        /* Positive Amplitude bestimmen */
221.    m = 0;
222.    for (j = 0; j< anz; j++) {                          //Hauptzählerschleife.
223.            maxValue = 3000;
224.            if(E[j] >= 3100) {                  //Bedingung größer gleich 295 V.
225.                    for(c = j; c<j+5000; c++){  //Verschachtelte Zählerschleife.
226.                            if(E[c] >= maxValue) {  //Vergleich, aktuellen mit vorherigem
                                                        Wert.
227.                                    maxValue = E[c];//Speichert den größten gefundenen
                                                        Wert ab.
228.                            }
229.                    }
230.            j=c;            //Überschreibt die Anzahl der durchgegangenen Werte in
                                Hauptzählerschleife.
231.            }
232.            if (maxValue > 3000) {  //Bedingung ist dann erfüllt, wenn die verschachtelte Schleife
                                        beendet ist.
233.            results.PositiveAmplitude[p][m] = (float)maxValue;//Abspeicherung des Maximus im
                                                                  Struct.
234.            m++;
235.            }
236.    }                                           //Ende Hauptzählerschleife.
```

```
        /* Negative Amplitude bestimmen */
237.    n = 0;
238.    for (j=0; j<anz; j++) {                                  //Hauptzählerschleife.
239.            maxValue = 1100;
240.            if(E[j] <= 1000) {                       //Bedingung kleiner gleich - 295 V.
241.                    for(c = j; c<j+5000; c++){       //Verschachtelte Zählerschleife.
242.                            if(E[c] <= maxValue) {   //Vergleich, aktueller mit vorherigem
                                                         Wert.
243.                                    maxValue = E[c];//Speichert den kleinsten gefundenen
                                                         Wert ab.
244.                            }
245.                    }
246.            j=c;            //Überschreibt die Anzahl der durchgegangen Werte in
                                Hauptzählerschleife.
247.            }
248.            if (maxValue < 1100) {   //Bedingung ist dann erfüllt, wenn die verschachtelte Schleife
                beendet ist.
249.            results.NegativeAmplitude[p][n] = (float)maxValue;//Abspeicherung des Minimums im
                                                                  Struct.
250.            n++;
251.            }
252.    }                                                        //Ende Hauptzählerschleife.
```

Zu Beginn der Bestimmung der positiven und der negativen Spannungsamplituden wird jeweils abgefragt, ob der gemessene Wert größer als 295 V ist oder kleiner als – 295 V. 295 V entspricht der binären Darstellung 3100 und – 295 V entspricht 1000. Die Umrechnung der binären Darstellung in Volt erfolgt mit der in Kapitel 2.4.4.6. dargestellten Formel 15. Die negativen und die positiven Amplituden befinden sich in zwei verschiedenen Zählerschleifen. Die positiven Amplituden werden in den Zeilen 221 bis 236 bestimmt, die negativen Amplituden in den Zeilen 237 bis 252. Die beiden im Aufbau identischen Amplituden-Bestimmungs-Funktionen bestehen aus zwei ineinander verschachtelten Zählerschleifen. In der jeweiligen Hauptzählerschleife wird jeder aufgenommene Messwert geprüft, ob er größer als 295 V (Zeile 224) oder kleiner als – 295 V (Zeile 240) ist. Sobald sich ein Wert zeigt, der dem entspricht, startet die in der Hauptzählerschleife verschachtelte Zählerschleife (Zeile 225 oder 241), welche die nächsten 5000 Werte nacheinander vergleicht. Wert für Wert wird überprüft, ob er als neues „maxValue“ infrage kommt (Zeilen 226 bis 228 oder 242 bis 244). Nach dem Vergleich der 5000 Werte ist der zuletzt gefundene „maxValue“-Wert der Wert der Amplitude. Er wird abgespeichert (Zeile 227 oder Zeile 242). Sobald die Zählerschleife die 5000 Werte verglichen hat, ist ihr Programm erfüllt. Nun wird der zuletzt verglichene Wert aus der verschachtelten Zählerschleife mit der aktuellen Messstelle aus der Hauptzählerschleife ersetzt (Zeile 230 oder 246), damit die Hauptzählerschleife die nächste Spannungsamplitude ermitteln kann. Die nächste if-Abfrage (Zeilen 232 bis 235 oder 248 bis 251) befindet sich in der Hauptzählerschleife

und ermöglicht die Abspeicherung von „maxValue“ aus der verschachtelten Zählerschleife. Die if-Abfrage ist nur erfüllt, wenn die verschachtelte Zählerschleife abgeschlossen ist, da nun der neue „maxValue“ größer oder kleiner als der vorher festgelegte Maximalwert aus der Hauptzählerschleife ist. Der als Integer abgespeicherte Wert („maxValue“) wird in eine Variable von Datentyp Float umgewandelt und dauerhaft in einer Variablen aus dem „Struct“ abgespeichert. Durch die spätere Umrechnung in die Spannungsamplituden ist dies notwendig, da die Uwandlungsfaktoren mit Kommazahlen berechnet werden und dies sich nicht mit einer Variablen von Datentyp Integer umsetzen lässt, da dort nur Ganzzahlen, also Zahlen ohne Kommastellen, abgespeichert werden können. Die Variablen „m“ und „n“ erhöhen die Arrays, worin die Spannungsamplituden gespeichert werden (Zeile 234 oder 250). Sie werden vor Beginn dieser Schleife auf null gesetzt (Zeile 221 oder 237), was eine Zuordnung ermöglicht. Dies ist notwendig, da zu jedem Zeitpunkt, wo eine Messung erfolgt, mehrere Spannungsperioden aufgenommen werden können und alle abgespeichert werden.

2.4.4.6. Umrechnung

Die im Kapitel 2.4.4.5 ausgewerteten Amplituden werden zu anschaulicherer Darstellung umgerechnet. Wie in Kapitel 2.4.4.5 werden die negativen und die positiven Amplituden je einzeln behandelt. Die Umrechnung der positiven Amplituden erfolgt zwischen Zeile 254 und 258, die der negativen in den Zeilen 259 bis 263. Die Umrechnungen erfolgen in einer Zählerschleife (Zeilen 253 bis 264), welche von der Anzahl der festgelegten Perioden abhängig ist.

```
        /* Binäre Werte in Wechselspannung umrechnen */
253.    for (j=0; j<anzPerioden; j++) {

            /* Umrechnung der positiven Amplitude in Netzspannung */
            //Berechnung, welche Spannung am Mikrocontroller anliegt
254.        results.PositiveAmplitude[p][j] = results.PositiveAmplitude[p][j]/4095;
255.        results.PositiveAmplitude[p][j] = results.PositiveAmplitude[p][j]*3.3;
            //Offset – Spannung
256.        results.PositiveAmplitude[p][j] = results.PositiveAmplitude[p][j]- 1.65;
            //Übersetzungsverhältnis des Transformators
257.        results.PositiveAmplitude[p][j] = results.PositiveAmplitude[p][j]*38.3333;
            //Spannungsteilerverhältnis
258.        results.PositiveAmplitude[p][j] = results.PositiveAmplitude[p][j]/0.11111;
```

```
                /* Umrechnung der negativen Amplitude in Netzspannung */
                //Berechnung, welche Spannung am Mikrocontroller anliegt
259.            results.NegativeAmplitude[p][j] = results.NegativeAmplitude[p][j]/4095;
260.            results.NegativeAmplitude[p][j] = results.NegativeAmplitude[p][j]*3.3;
                //Offset – Spannung
261.            results.NegativeAmplitude[p][j] = results.NegativeAmplitude[p][j]- 1.65;
                //Übersetzungsverhältnis des Transformators
262.            results.NegativeAmplitude[p][j] = results.NegativeAmplitude[p][j]*38.3333;
                //Spannungsteilerverhältniss
263.            results.NegativeAmplitude[p][j] = results.NegativeAmplitude[p][j]/0.11111;

264.        }
```

Die Umrechnung erfolgt, wie in Kapitel 2.2.4 beschrieben, mit den Faktoren, welche im Spannungswandler festgelegt worden sind, und sie verläuft rückwärts. Zuerst wird die gemessene Spannung vom Datentyp Float durch die maximale oder minimale Spannung, welche der Mikrocontroller auswerten kann, dividiert (Zeilen 254 bis 254 oder Zeilen 259 bis 259). Der A/D-Wandler des PIC32MZ hat eine 12 Bit hohe Auflösung. Mit 12 Bit lassen sich maximal 4096 (2^{12}) verschiedene Werte darstellen. Anschließend wird die binäre Darstellungsform in die Mikrocontroller-Spannung umgerechnet. Dazu wird der dividierte Wert mit der maximalen oder minimalen Mikrocontroller-Spannung multipliziert (Zeilen 255 bis 255 oder Zeilen 260 bis 260) Die maximale Spannung, welche der Mikrocontroller auswerten kann, beträgt 3,3 V, was der binären Darstellung von 4095 ($2^{12} - 1$) entspricht. Damit wurde die Spannung bestimmt, welche am Mikrocontroller anliegt. Im weiteren Verlauf wird die Offset-Spannung durch Subtraktion nullifiziert (Zeilen 256 bis 256 oder Zeilen 261 bis 261). Anschließend folgt nacheinander die Elimination von zwei Relationen durch zwei verschiedene Rechenoperationen: Das Übersetzungsverhältnis der Transformatoren durch Multiplikation (Zeilen 257 bis 257 oder Zeilen 262 bis 262) und das Spannungsteiler-Verhältnis durch Division (Zeilen 258 bis 258 und Zeilen 263 bis 263). Damit sind die Netzspannungen in den Variablen abgespeichert. Entsprechend der vordefinierten Anzahl der Perioden wird nun dieser Vorgang so oft durchgeführt, bis alle Amplituden abgespeichert sind. In der folgenden Formel wird die Umrechnung zusammengefasst.

$$\hat{U} = \frac{\left(\left(\frac{\mu C - Wert}{\mu C - Max} \times \mu CSp\right) - Offset\right) \times T\ddot{U}}{SpTV} \tag{15}$$

$$\hat{U} = Scheitelwert,\ \ \mu C - Wert = gemessene\ Spannung\ in\ Bit,$$

$$Offset = Spannungsteiler = 1{,}65\ V,\ \mu C - Max = maximale\ Spannung\ in\ Bit\ = 4095,$$

$$\mu CSp = maximale\ Microcontroller - Spannung = 3{,}3\ V,$$

$$T\ddot{U} = Trafo\ddot{u}bersetzungsverhältnis = 38{,}33,\ SpTV = Spannungsteilerverhältnis = 0{,}11$$

2.4.4.7. Referenzzeitberechnung

Zur Referenzzeitermittlung (Δt) haben sich zwei unterschiedliche Methoden ergeben. Diese werden in den folgenden Kapiteln dargestellt.

2.4.4.7.1. 1. Methode: ADC & Timer-Modul

Die erste Methode beinhaltet die Zeit zwischen der GNSS-Taktflanke und dem ersten Nulldurchgang, welche mithilfe der in Kapitel 2.4.4.7 vorgestellten Funktion bestimmt wurde. Mit Formel 16 in Zeile 265 wurde die aufgenommene Zeit ins Sekunden-Format umgerechnet.

```
    /* Zeit zwischen GNSS-Taktflanke und erstem Nulldurchgang = Delta t */
265. results.DeltaT[p] = (float)h / 3125000;
```

$$\Delta t = \frac{vergangene\ Zeit}{Netzfrequenz \times maximale\ Zeit} \tag{16}$$

$$\Delta t = Zeit\ zwischen\ Refferenzzeitpunkt\ und\ \text{dem}\ ersten\ Nulldurchgang$$

$$vergangene\ Zeit = die\ vergangene\ Zeit\ als\ Integer\ dargestellt$$

$$Netzfrequenz = 50\ Hz$$

$$maximale\ Zeit = 62.499 = abhängig\ von\ der\ Timer\ Zeitdauer, hier\ 20\ ms$$

In der Variablen „h“, die sich in dem Messinterrupt befindet (Kapitel 2.4.1.1, Seite 28), wird der Referenzzeitpunkt abgespeichert. Damit wird die Vergleichbarkeit mit dem anderen Messgerät ermöglicht. Das Periodenregister des Timers wird mit einer maximalen Zeit von 62.499 Bit (Zeile 25 und 26) initialisiert. Die maximale Zeit wurde mit Formel 4 berechnet. Die Frequenz wurde auf 50 Hz definiert, welche einem Zeitraum von 20 ms entspricht. Der binäre Wert wird zur Berechnung des Phasenwinkels in Millisekunden umgerechnet. Dazu muss er durch 3.124.950 dividiert werden. Diese Zahl ergibt sich durch die Rückrechnung der erwarteten binären Zeit durch die erwartete Zeit in metrische Einheiten ($62499/20\ ms$). Aus der binären Zeit und dem berechneten Wert von 3.124.950 ist die Zeit ins metrische Einheitensystem umzurechnen. Da ein Timer seinen konfigurierten Zeitablauf in Ganzzahlen misst, die Umrechnung aber eine Kommazahl enthält, wird die Variable „h“ in einen Float umgewandelt, einen Datentyp, welcher zur Berechnung von Gleitkommazahlen imstande ist.

2.4.4.7.2. 2. Methode: Input-Capture-Modul

Mit der zweiten Methode wurde, die Zeit zwischen der GNSS-Taktflanke und dem Erscheinen der ersten Nullstelle durch die Nullstellenbestimmung aus Kapitel 2.4.4.3 bestimmt. Mithilfe von Formel 17 in Zeile 266 wurde die aufgenommene Zeit ins Sekunden-Format umgerechnet.

```
/* Zeit zwischen GNSS-Taktflanke und erstem Nulldurchgang = Delta t */
266.    results.DeltaT[p] = ((results.Zwischenwerte[p][0] – Messwerte)/ (AbtastFrequenz/1000));
```

$$\Delta t = \frac{erste\ Nullstelle - Zeit\ nach\ dem\ Start\ der\ Messung}{AbtastFrequenz\ /1000} \tag{17}$$

Das Δ t entspricht der Anzahl der Messwerte in ms. Die Anzahl der aufgenommenen Messwerte zwischen dem Start der Messung und der ersten präzisen Nullstelle entspricht der vergangenen Zeit in µs. Um das Δ t in ms umzurechnen, wird der Wert dazwischen mit der Division der Abtastfrequenz durch den Faktor 1000 dividiert (Siehe Formel 17). Diese Methode, die Zeitbestimmung durch Auswertung der Messwerte zu erzielenm, ist relativ statisch.

Die andere vorgestellte Funktion (Kapitel .2.4.4.7.1., Seite 28, Zeile 35) ist dynamischer, da sie die Zeit während der Messwertaufnahme ermittelt. Da durch das Rauschen auf der Messkurve Abweichungen entstehen, kommt es zu unkalkulierbaren Messwertdifferenzen. Die statische Methode erwies sich als zuverlässiger und wurde deshalb genutzt. Wenn kein Rauschen auftritt, sollte die andere Methode nutzbare Werte liefern.

2.4.4.8. Phasenwinkel

Der Phasenwinkel ist das Produkt aus Frequenz und Zeit zwischen der GNSS-Taktflanke und dem ersten Nulldurchgang. Zur Ermittlung der Phasendifferenz des Messsystems wird der Phasenwinkel mithilfe des Programmcodes aus Zeile 267 und der Formel 18 berechnet.

```
        /* Phasenwinkel berechnen */
        // Phi = 360 Grad * Frequenz * Δt
267.    results.Phasenwinkel[p] = 360 * results.Frequenz[p][0] * results.DeltaT[p];
```

$$\varphi = 360° \times ermittelte\ Frequenz \times \Delta t \tag{18}$$

2.4.4.9. Umwandlung

Zwei Schritte sind notwendig, damit die Daten überschaubar auf der SD-Karte abgespeichert werden können. Diese Schritte werden nun nacheinander dargestellt.

2.4.4.9.1. String

Die berechneten Werte müssen in den Datentyp String umgewandelt werden, damit sie auf der SD-Karte abgespeichert werden können. Die Umwandlung wird mit folgendem Code realisiert:

```
        /* Float Daten in Strings umwandeln, zur Abspeicherung */
268.    for(j=0; j<anzPerioden; j++){
269.            sprintf(results.PositiveAmplitude_String[p][j], "%f", results.PositiveAmplitude[p][j]);
270.            sprintf(results.NegativeAmplitude_String[p][j], "%f", results.NegativeAmplitude[p][j]);
271.            sprintf(results.Frequenz_String[p][j], "%f", results.Frequenz[p][j]);
272.            sprintf(results.DeltaT_String[p], "%f", results.DeltaT[p]);
273.            sprintf(results.Phasenwinkel_String[p], "%f", results.Phasenwinkel[p]);
274.    }
```

Die Funktion „sprintf()“ ist in der Headerdatei „stdio.h“ deklariert und schreibt formatiert in einem String. In der Klammer steht an erster Stelle die Datei, in welcher der formatierte String abgespeichert werden soll. Die zweite Stelle gibt den Datentyp an, welcher umgewandelt wird.

Zuletzt steht die Datei, auf die zugegriffen wird. Dafür wurden im „Struct“ (Kapitel 2.4.4.1) neue Variablen vom Datentyp Char erstellt (Zeilen 193 bis 197). In den Zeilen 268 bis 274 wandelt eine Zählerschleife die berechneten Ergebnisse in Strings um. Die berechneten Werte sind vom Datentyp Float und werden deshalb an der zweiten Stelle in der „sprintf()“ Funktion mit einen „%f“ markiert [17].

2.4.4.9.2. Punkt in Komma

```
/* Punkt in Komma umwandeln */
275.    for(j=0; j<anzString; j++){
276.            for(i=0; i<anzString; i++){
277.                    if (results.PositiveAmplitude_String[p][j][i] == 46){           //46 = Punkt
278.                            results.PositiveAmplitude_String[p][j][i] = 44;         //44 = Komma
279.                    }
280.                    if (results.NegativeAmplitude_String[p][j][i] == 46){           //46 = Punkt
281.                            results.NegativeAmplitude_String[p][j][i] = 44;         //44 = Komma
282.                    }
283.                    if (results.Frequenz_String[p][j][i] == 46){                    //46 = Punkt
284.                            results.Frequenz_String[p][j][i] = 44;                  //44 = Komma
285.                    }
286.            }
287.    }

288.    for(j=0; j<anzString; j++){
289.            if (results.DeltaT_String[p][j] == 46){                                 //46 = Punkt
290.                    results.DeltaT_String[p][j] = 44;                               //44 = Komma
291.            }
292.            if (results.Phasenwinkel_String[p][j] == 46){                           //46 = Punkt
293.                    results.Phasenwinkel_String[p][j] = 44;                         //44 = Komma
294.            }
295.    }
```

Ein Mikrocontroller arbeitet mit einem Punkt als Dezimaltrennzeichen. Da die heutigen Textverarbeitungsprogramme einen Punkt als Trennzeichen nach Tausendern und ein Komma als Dezimaltrennzeichen verstehen, ist es für die weitere Analyse notwendig, an der Dezimaltrennstelle Punkt auf Komma umzuändern. Die Umwandlung erfolgt mit dem oben dargestellten Code (Zeile 275 bis Zeile 295). Dabei wird für jedes Zeichen der berechneten Werte geprüft, ob dort ein Punkt als Zeichen steht. Wenn das der Fall ist, wird die entsprechende Stelle jeweils mit einem Komma überschrieben. Der Vorgang ist abgeschlossen, sobald alle Werte verglichen sind.

2.4.4.10. Abspeicherung von Zeit & Datum

```
        /* Speichert die Zeit ab */
296.    for (j=0; j<6; j++){
297.            results.GNSS_TIME[p][j] = gnss_time[j];
298.    }

        /* Speichert das Datum ab */
299.    for (j=0; j<6; j++){
300.            results.GNSS_DATE[j] = gpsdate[j];
301.    }
```

Zwischen Zeile 296 und 301 erfolgt die Abspeicherung der Zeit und des aktuellen Datums. Die Abspeicherung der Zeit erfolgt in der Zählerschleife zwischen Zeile 296 und 298. In den Zeilen 299 bis 301 wird in einer anderen Zählerschleife das Datum abgespeichert. Auf die in diesem Code abgespeicherten Werte kann man in den Abspeicherungsfunktionen zugreifen.

2.4.4.11. Bedingung zur Abspeicherung & Erhöhung

```
        /* Vergleicht ob die Anzahl der aufzunehmenden Messungen erreicht ist */
302.    if(p == anzTime-1){             //Speichert die Messung
303.            phasor_average();
304.            SD_DataSave_calculate();
305.    }

306.    else{                           //Führt weitere Messungen aus
307.            p++;
308.            start();
309.    }
```

Am Ende der Berechnungsfunktion wird verglichen, ob die festgelegte Anzahl der aufgenommenen Perioden erreicht wurde (Zeile 302). Wenn dies der Fall ist, werden die Funktionen „phasor_average()“ und „SD_DataSave_calculate()“ aufgerufen. Die Funktion „phasor_average()“ bildet den Mittelwert aus allen aufgenommen Phasenwinkeln. Die Funktion „SD_DataSave_calculate()“ führt die Abspeicherung der berechneten Werte auf die SD-Karte durch.

In Zeile 307 wird die Variable „p“ um eins erhöht. Die Variable „p“ dient der Zwischenspeicherung der gemessenen, umgerechneten Werte im Arbeitsspeicher. Dadurch ist es möglich, den Zugriff auf die SD-Karte zu verringern. Wenn nun die festgelegte Anzahl der aufgenommenen Perioden noch nicht erreicht ist, inkrementiert sich die Variable, welche die Anzahl der Perioden definiert, um eins und die Funktion (Zeile 308), welche das Gerät für die nächste Messaufnahme vorbereitet, wird aufgerufen. In dieser Funktion wird das UART-Modul aktiviert und der nächste Messzeitpunkt definiert. Dadurch ist eine fortwährende Aufnahme von Messwerten gewährleistet.

2.4.4.12. Mittelwertbildung

```
310. phasor_average(){

311.        results.Phasenwinkel_average = 0;
            // Addiert die Phasenwinkel
312.        for (j=0; j<anzTime; j++){
313.                results.Phasenwinkel_average = results.Phasenwinkel_average + results.Phasenwinkel[j];
314.        }
            // Teilt die addierten Phasenwinkel durch die Anzahl der Phasenwinkel
315.        results.Phasenwinkel_average = results.Phasenwinkel_average/anzTime;
            // Wandelt das Dateiformat um
316.        sprintf(results.Phasenwinkel_average_String, "%f", results.Phasenwinkel_average);
            // Ändert das Dezimale Trennzeichen
317.        for (j=0; j<anzString; j++){
318.                if (results.Phasenwinkel_average_String[j] == 46){          //46 = Punkt
319.                        results.Phasenwinkel_average_String[j] = 44;        //44 = Komma
320.                }
321.        }

322. }
```

In der Funktion „phasor_average()“ wird der Mittelwert aus allen aufgenommen Phasenwinkeln gebildet. Die Bildung des Mittelwerts findet in den Zeilen 312 bis 315 mit jedem davor aufgenommenen Phasenwinkel statt. Zeile 316 wandelt die Zahl in das Datenformat String um. Zwischen den Zeilen 317 und 320 werden die dezimalen Trennzeichen, wie in Kapitel 2.4.4.9.2 erklärt, umgewandelt

2.4.5. Abspeicherung

Die Abspeicherung der Messwerte erfolgt mit der Funktion „SD_DataSave_calculate“ oder „SD_DataSave_RohWerte“. In der „defines.h“-Datei wird in Zeile 67 festgelegt, welche der beiden Funktionen am Ende einer Messung gestartet wird.

2.4.5.1. Berechnete Werte

Die Funktion „SD_DataSave_calculate()“ befindet sich in der „calculate()“-Funktion und wird ausgelöst, sobald die Umrechnung abgeschlossen ist. Durch diese Funktion werden die umgerechneten Werte in der SD-Karte abgespeichert. Mit folgendem Code wurde die Funktion implementiert:

```
323. void SD_DataSave_calculate(){
324.        SaveData[0] = gpsdate[0];        //Die erste Ziffer des Tages
325.        SaveData[1] = gpsdate[1];        //Die zweite Ziffer des Tages
326.        SaveData[2] = gpsdate[2];        //Die erste Ziffer des Monats
327.        SaveData[3] = gpsdate[3];        //Die zweite Ziffer des Monats
328.        SaveData[4] = gpsdate[4];        //Die dritte Ziffer des Jahres
329.        SaveData[5] = gpsdate[5];        //Die vierte Ziffer des Jahres

            //Wird als TXT Datei abgespeichert
330.        SaveData[6] = 46;                //‘.‘
331.        SaveData[7] = 116;               //'t'
332.        SaveData[8] = 120;               //'x'
333.        SaveData[9] = 116;               //'t'

334.        f_open(&file, SaveData, FA_OPEN_APPEND | FA_WRITE);

335.        for(p=0; p <= anzTime-1; p++){
                /* In der ersten Zeile wird die Zeit gespeichert, wann die Messung starten soll. */
336.            f_printf(&file, "Zeit\n");
337.            f_printf(&file, "%c", gpstime[0]);
338.            f_printf(&file, "%c", gpstime[1]);
339.            f_printf(&file, "%c", 58);          //‘:‘
340.            f_printf(&file, "%c", gpstime[2]);
341.            f_printf(&file, "%c", gpstime[3]);
342.            f_printf(&file, "%c", 58);          //‘:‘
343.            f_printf(&file, "%c", gpstime[4]);
344.            f_printf(&file, "%c", gpstime[5]);
345.            f_printf(&file, "\n");
```

```
                /* In der zweiten Zeile wird die Frequenz gespeichert. */
346.            for(j=1; j <= AnzahlWieVielePerioden; j++){
347.                    f_printf(&file, "%d.Frequenz;", j);
348.            }
349.            f_printf(&file, "\n");
350.            for(j=0; j < AnzahlWieVielePerioden; j++){
351.                    f_printf(&file, "%s; ", results.Frequenz_String[p][j]);
352.            }
353.            f_printf(&file, "\n");

                /* In der dritten Zeile werden die negativen Amplituden gespeichert. */
354.            for(j=1; j <= AnzahlWieVielePerioden; j++){
355.                    f_printf(&file, "%d.Negative Amplitude;", j);
356.            }
357.            f_printf(&file, "\n");
358.            for(j=0; j < AnzahlWieVielePerioden; j++){
359.                    f_printf(&file, "%s; ", results.NegativeAmplitude_String[p][j]);
360.            }
361.            f_printf(&file, "\n");

                /* In der vierten Zeile werden die positiven Amplituden gespeichert. */
362.            for(j=1; j <= AnzahlWieVielePerioden; j++){
363.                    f_printf(&file, "%d.Positive Amplitude;", j);
364.            }
365.            f_printf(&file, "\n");
366.            for(j=0; j < AnzahlWieVielePerioden; j++){
367.                    f_printf(&file, "%s; ", results.PositiveAmplitude_String[p][j]);
368.            }
369.            f_printf(&file, "\n");

                /* In der fünften Zeile wird Delta T gespeichert. */
370.            f_printf(&file, "\tDelta T\n");
371.            f_printf(&file, "\n", results.DeltaT_String[p]);

                /* In der sechsten Zeile wird der Phasenwinkel gespeichert. */
372.            f_printf(&file, "\tPhasenwinkel\n");
373.            f_printf(&file, "\n", results.Phasenwinkel_String[p]);

                /* Die Datei wird geschlossen. */
374.            f_printf(&file, "\n\n\n");          //Drei Zeilenumbrüche
375.        }

376.        f_close(&file);

            /* Sobald alles gespeichert worden ist, werden die Variablen, welche zur Ausführung erneuerter
            Messungen benötigt werden zurückgesetzt. */
377.        LED_LD1_INV()
378.        p = 0;
379.        start();
380. }
```

Die Funktion startet ab Zeile 323 und endet mit Zeile 380. In den Zeilen 325 bis 333 wird ein Array beschrieben, welcher den Namen der Datei bestimmt, in der die Messwerte abgespeichert werden. Die Dateien werden in den Zeilen 324 bis 329 immer mit dem aufgenommenen Datum versehen. In den Zeilen 330 bis 333 wird der Datentyp der Datei bestimmt. Die Messwerte werden als Text-Datei (txt) abgespeichert. In Zeile 334 wird die Datei auf der SD-Karte erstellt und zum Beschreiben freigegeben. Die darauf folgende Zählerschleife von Zeile 335 bis 375 speichert die aufgenommen Perioden ab und ist von der im Kapitel 2.4.1.1 eingestellten Anzahl der Perioden abhängig. Nun wird die erste Zeile in der Textdatei durch die Zeilen 336 bis 344 mit dem Zeitpunkt der Messung versehen. In Zeile 345 findet ein Zeilenumbruch statt. In der zweiten und dritten Zeile wird die Frequenz gespeichert. Mit dem Code aus den Zeilen 346 bis 352 wurde das umgesetzt. Dasselbe Verfahren findet in Zeile 353 bis Zeile 373 für die anderen ermittelten Werte statt. In Zeile 374 erfolgt ein dreifacher Zeilenumbruch. Dadurch sind die Messungen voneinander zu unterscheiden. In Zeile 376 wird der Zugriff auf die SD-Karte deaktiviert, diese Schließung ist das Pendant zu der Öffnung der SD-Karte in Zeile 334. Danach wird in Zeile 377 LED 1 von dem Entwicklungsboard invertiert. Dadurch ist visuell zu erkennen, wann Daten auf der SD-Karte gespeichert wurden. Zeile 378 setzt die Variable „p“ auf null zurück. Dadurch können die nächsten Messungen zugeordnet werden. Die letzte Zeile, Zeile 379, ruft die Funktion „start()“ auf. Hier werden alle nötigen Schritte eingeleitet, um neue Messungen aufzunehmen.

2.4.5.2. Ursprungswerte

Die folgende Funktion dient der Abspeicherung der Messwerte in der Ursprungsform:

```
381.        SaveData[0] = gpsdate[0];        //Die erste Ziffer des Tages
382.        SaveData[1] = gpsdate[1];        //Die zweite Ziffer des Tages
383.        SaveData[2] = gpsdate[2];        //Die erste Ziffer des Monats
384.        SaveData[3] = gpsdate[3];        //Die zweite Ziffer des Monats
385.        SaveData[4] = gpsdate[4];        //Die dritte Ziffer des Jahres
386.        SaveData[5] = gpsdate[5];        //Die vierte Ziffer des Jahres

            //Wird als TXT Datei abgespeichert.
387.        SaveData[6] = 46;                //'.'
388.        SaveData[7] = 116;               //'t'
389.        SaveData[8] = 120;               //'x'
390.        SaveData[9] = 116;               //'t'

391.        f_open(&file, SaveData, FA_OPEN_APPEND | FA_WRITE);

392.        for(j=0; j < anz; j++){
393.                f_printf(&file, "%d",E[j]);
394.                f_printf(&file, "\n");
395.        }

396.        f_printf(&file, "\n\n\n");

397.        f_close(&file);

            /* Sobald alles gespeichert worden ist werden die Variablen, welche zur Ausführung erneuerter
            Messungen benötigt werden zurückgesetzt. */
398.        LED_LD1_INV()
399.        start();
400. }
```

In der Dateierstellung, der Namenzuordnung und der Wiederaufnahme ist diese Funktion mit der Funktion, welche in Kapitel 2.4.5.1 erklärt worden ist, identisch. Ein Unterschied besteht allerdings in den Zeilen 392 bis 395. Dort werden, entsprechend der Anzahl der aufgenommenen Messwerte die Ursprungswerte abgespeichert (Zeile 393). Die nächste Zeile zeigt einen Zeilenumbruch (Zeile 394), durch den jedem Messwert seine eigene Zeile zuzuordnen ist. Diese Funktion erleichtert es, die Ursachen und Ungenauigkeiten zu finden, die sich möglicherweise beheben lassen.

2.4.6. Wiederaufnahme

Sobald die Abspeicherung komplett ist, wird die Messwertaufnahme zurückgesetzt und die Funktion start() ausgeführt. Ab diesen Zeitpunkt sind weitere Messungen möglich. Die Funktion befindet sich jeweils am Ende der Abspeicherungsfunktionen sowie am Ende der Berechnungsfunktion. Hier wird sie aufgerufen, solange noch nicht alle Perioden aufgenommen sind (Zeile 379).

```
401. void start(){
402.        a = 0;
403.        e = 0;
404.        change_time();          // Ändert den Messzeitpunkt
405.        IEC5bits.U4RXIE = 1;    // Starte das GNSS-Modul
406. }
```

Die Variablen „a“ (Zeile 402) und „e“ (Zeile 403) werden zurückgesetzt und als Zählervariablen eingesetzt. Die Variable „a“ dient dem GNSS-Datensatz und „e“ den aufzunehmenden Spannungsmesswerten. In Zeile 404 wird die Funktion „change_time()“ aus Kapitel 2.4.3 aufgerufen. Die Funktion ändert die Uhrzeit, zu der eine Messung stattfindet. Zeile 405 aktiviert das UART4-Interrupt. Durch das UART-Modul werden die GNSS-Daten empfangen, welche die Uhrzeit liefern. In der UART-Funktion befindet sich die Kennzeichnung, welche die Spannungsmessung aktiviert.

2.5. Messwertaufnahme

Nachdem das Gerät vollständig aufgebaut, programmiert und kalibriert war, wurden Messungen im Niederspannungsnetz getätigt. Die hier dargestellten Messwerte wurden exemplarisch zusammengefasst. Üblicherweise werden pro gestarteter Messung mehrere Perioden hintereinander aufgenommen. Zur Veranschaulichung ist hier immer nur die erste Periode dargestellt. In der ersten Periode sind die Referenzzeit und der Phasenwinkel abgebildet. In den darauf folgenden Perioden dagegen nicht, weil nur die Referenzzeit in der ersten Periode für die Bildung des Phasenwinkels genutzt wird. Bevor die Daten auf der SD-Karte gespeichert werden, wird der Mittelwert aus allen gemessenen Perioden gebildet und am Anfang einer Messaufnahme gespeichert. Dadurch werden die Differenzen der Phasenwinkel gemittelt. Die Messwerte werden als eine Textdatei auf der SD-Karte gespeichert. und befinden sich alle in einer Spalte. Um die Perioden voneinander zu trennen, wurde ein Semikolon als Trennzeichen verwendet. In Excel muss zur übersichtlichen Darstellung die Funktion „Text in Spalten“ aufgerufen werden. Dort wird als Trennzeichen das Semikolon ausgewählt. Damit wird jede Periode ihrer eigenen Spalte zugeordnet.

2.5.1. Identische Positionen

2.5.1.1. Minimale Abweichung

Tabelle 3 – Messwertaufnahme: identische Position, minimale Abweichung [1]

Zeit	Differenz (Gerät 1 minus Gerät 2)	Zeit
17:05:30	00:00:00	17:05:30
1.Frequenz [Hz]		1.Frequenz [Hz]
50,017506	0,010006	50,0075
1.Negative Amplitude [V]		1.Negative Amplitude [V]
-292,340118	2,502167	-294,842285
1.Positive Amplitude [V]		1.Positive Amplitude [V]
315,415894	-13,901123	329,317017
Delta t [ms]		Delta t [ms]
1,755	0,004	1,751
Phasenwinkel [°]		Phasenwinkel [°]
31,601063	0,078339	31,522724

Tabelle 3 zeigt die berechneten Messwerte mit einer minimalen Abweichung der Messgeräte an. Die Phasenwinkeldifferenz liegt unter 0,1 º. Bei 30 aufgenommen Messungen tritt fünfzehnmal eine minimale Abweichung auf. Somit ergibt sich eine minimale Abweichung bei 50 % der Messwerte.

2.5.1.2. Mittlere Abweichung

Tabelle 4 – Messwertaufnahme: identische Position, mittlere Abweichung [1]

Messgerät 1	Differenz (Gerät 1 minus Gerät 2)	Messgerät 2
Zeit		Zeit
17:04:50	00:00:00	17:04:50
1.Frequenz [Hz]		1.Frequenz [Hz]
49,997501	0,007499	49,990002
1.Negative Amplitude [V]		1.Negative Amplitude [V]
-329,595032	-3,336243	-326,258789
1.Positive Amplitude [V]		1.Positive Amplitude [V]
317,084045	-14,457184	331,541229
Delta t [ms]		Delta t [ms]
3,35	0,008	3,342
Phasenwinkel [°]		Phasenwinkel [°]
60,296982	0,153012	60,14397

Tabelle 4 zeigt die berechneten Messwerte bei einer mittleren Abweichung der Messgeräte an. Die Phasenwinkeldifferenz beträgt unter 0,2 º. Bei 30 aufgenommen Messungen tritt diese mittlere Abweichung acht Mal auf. Somit ergibt sich eine mittlere Differenz bei 26,7 % der Messwerte.

2.5.1.3. Maximale Abweichung

Tabelle 5 – Messwertaufnahme: identische Position, maximale Abweichung [1]

Messgerät 1	Differenz (Gerät 1 minus Gerät 2)	Messgerät 2
Zeit		Zeit
17:05:10	00:00:00	17:05:10
1.Frequenz [Hz]		1.Frequenz [Hz]
50,010002	0,049969	49,960033
1.Negative Amplitude [V]		1.Negative Amplitude [V]
-330,707123	-5,838471	-324,868652
1.Positive Amplitude [V]		1.Positive Amplitude [V]
316,249969	-16,125306	332,375275
Delta t [ms]		Delta t [ms]
4,327	0,012	4,315
Phasenwinkel [°]		Phasenwinkel [°]
77,901588	0,29367	77,607918

Tabelle 5 zeigt die berechneten Messwerte bei einer maximalen Abweichung der Messgeräte auf. Die Phasenwinkeldifferenz beträgt unter 0,3 º. Bei 30 aufgenommen Messungen tritt die maximale Abweichung sieben Mal auf. Somit ergibt sich eine maximale Differenz bei 23,3 % der Messwerte.

2.5.1.4. Zusammenfassung

Die Tabellen zeigen die minimalen, mittleren und maximalen Abweichungen der berechneten Messwerte, aus den berechnet wurde, mit welcher Wahrscheinlichkeit sie auftreten können. Ausschlaggebend für diese Differenzen sind die Frequenzen und das Δ t.

Tabelle 6 – Mittelwert des Phasenwinkels aus einer Messwertaufnahme [1]

Gemittelter Phasenwinkel		
17:04:50	Bis	17:05:40
Phasenwinkel - Mittelwert [°]	Differenz (Gerät 1 minus Gerät 2)	Phasenwinkel - Mittelwert [°]
80,763496	0,115066	80,64843

Tabelle 6 zeigt innerhalb einer Minute die gemittelten Phasenwinkel der Messgeräte eins und zwei im Vergleich. Durch die gemittelten Phasenwinkel werden die Differenzen, welche durch das Rauschen entstehen, verringert. Damit ist eine genauere Phasenwinkelbestimmung pro Abspeicherung auf der SD-Karte gegeben.

2.5.2. Simulationsmessung unter Last

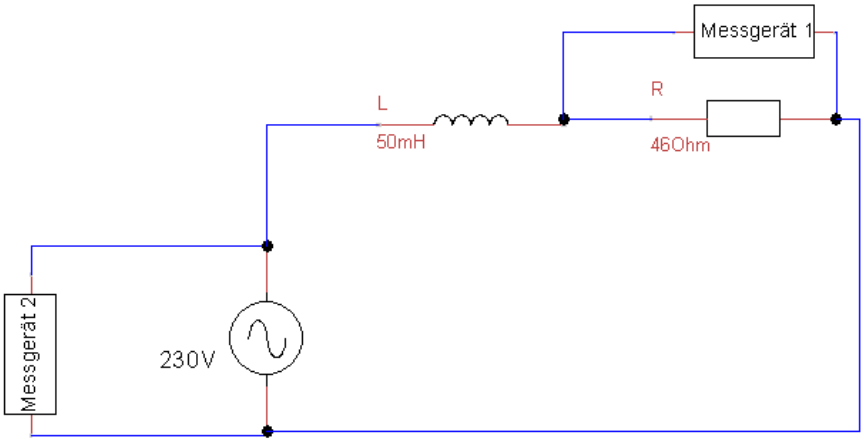

Abbildung 35 - Simulationsmessung, Schaltplan

Das Messsystem wurde unter Last getestet. Der Versuchsaufbau besteht aus einer Spule mit 50 mH und einem Schiebewiderstand, welcher anfangs auf 46 Ω eingestellt wurde. Abbildung 35 stellt den Messaufbau als Schaltplan dar.

$$Phasendifferenz\ \varphi = arctan\frac{Uim}{Ure} \tag{19}$$

$$Spannung\ Immaginärteil = \ Uim = I\ \times j\omega L \tag{20}$$

$$Spannung\ Realteil = Ure = I \times R \tag{21}$$

Die Spule ist für einen maximalen Strom von 5 A ausgelegt. Der Widerstand der Spule wurde bei der Berechnung vernachlässigt. Mit Formel 21 wurde mit dem maximalen Strom der Widerstandswert für den Schiebewiderstand bestimmt $(230\ V/5\ A)$. Mithilfe des Stroms, der durch die Spule fließt, und der Induktivität wurde mit Formel 20 die Spannung, welche über der Spule abfällt, berechnet. Über der Spule liegt somit eine Spannung von ca. 78,5 V $(5\ A \times 2 \times \pi \times 50\ Hz \times 50\ mH)$ an. Mit Formel *19* wurde aus den beiden anliegenden Spannungen ein Phasenwinkel von 32,9 º berechnet.
Der Messaufbau simuliert eine Leitung, an deren Ende ein Verbraucher angeschlossen ist. Da die Induktivität anhand der Leitungslänge zunimmt, findet darüber eine Phasenwinkelverschiebung der Spannung statt. Aus diesem Grund kann man eine entsprechend lange Leitung mit einer Induktivität gleichsetzen. Sobald die Last angeschaltet wird, fließt ein dementsprechend hoher Strom über die Leitung. Durch die über der Leitung abfallende Spannung entsteht ein Leistungsabfall, welcher in Form von Wärmeenergie an die Umwelt abgegeben wird.

Tabelle 7 - Messwertaufnahme, Simulationsmessung unter Last [1]

Gemittelter Phasenwinkel am 29.05.2019		
Gerät 1		Gerät 2
11:15:40	Bis	11:16:30
Phasenwinkel - Mittelwert [°]	Differenz (Gerät 1 minus Gerät 2)	Phasenwinkel - Mittelwert [°]
103,808922	32,992058	70,816864

Tabelle 7 zeigt den Mittelwert des Phasenwinkels von sechs gemessenen Phasenwinkeln, welche im 10 Sekunden-Takt von 11:15:40 bis 11:16:30 aufgenommen wurden. Die gemessene Phasenwinkeldifferenz ist mit der berechneten identisch.

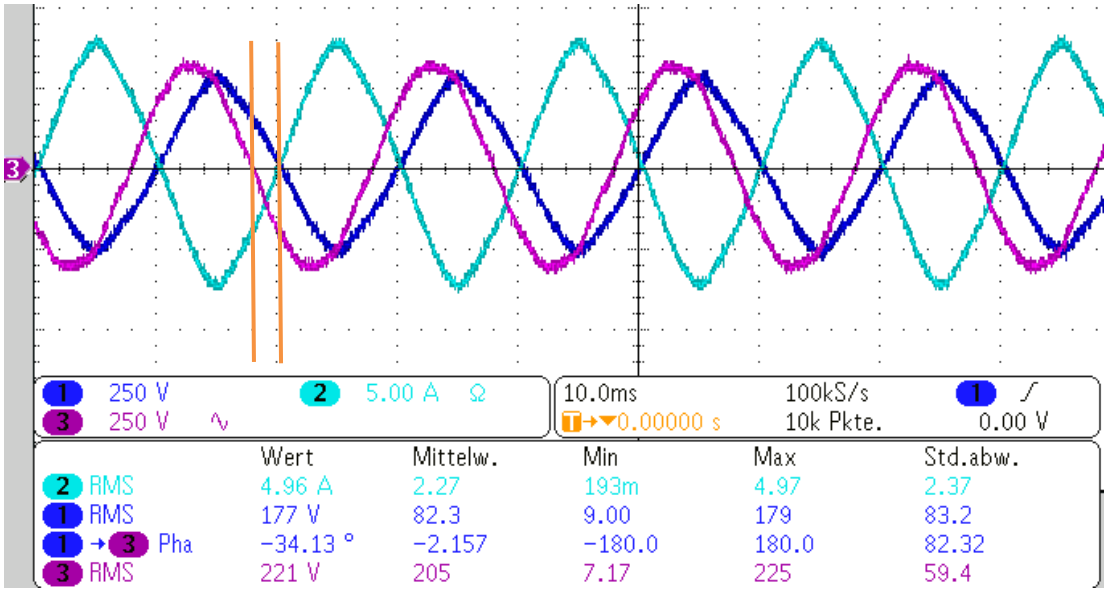

Abbildung 36 – Simulationsmessung, Oszilloskopaufzeichnung

Abbildung 36 zeigt die Ergebnisse der Simulationsmessung durch ein Oszilloskop aufgenommen. Mithilfe des Oszilloskops wurde der Strom, welcher durch den Schiebewiderstand fließt, auf 5 A eingestellt. Dadurch wurde auch der Widerstand der Spule zur Strombestimmung berücksichtigt. Der Strom ist türkisfarben dargestellt. Die blaue Kurve zeigt die Spannung von Messgerät eins an, welches die Spannung über den Schiebewiderstand gemessen hat. Die lila Kurve stellt die Netzspannung dar. Die Phasenwinkeldifferenz auf dem Oszilloskop wird mit -34,13 º angegeben und ist zwischen den beiden orangefarbenen Strichen dargestellt.

2.5.3. Amplitudenabweichungen

Durch Oberschwingungen, welche teilweise schneller als die Abtastfrequenz wechseln, befindet sich auf den Amplituden ein noch stärkeres Messwertrauschen als zwischen den Nulldurchgängen. An beiden Controllern entstehen zu unterschiedlichen Zeiten hohe und tiefe Werte. Eine Regressionsgerade durch dieses Rippel ist kaum zu berechnen, da es dort kaum eine Steigung gibt, welche benötigt wird, um eine solche Grade zu ermitteln. Abbildung 37 verdeutlicht die angesprochenen Schwankungen, welche probeweise aufgenommen wurden, um die Ermittlung der Amplituden zu bestimmen.

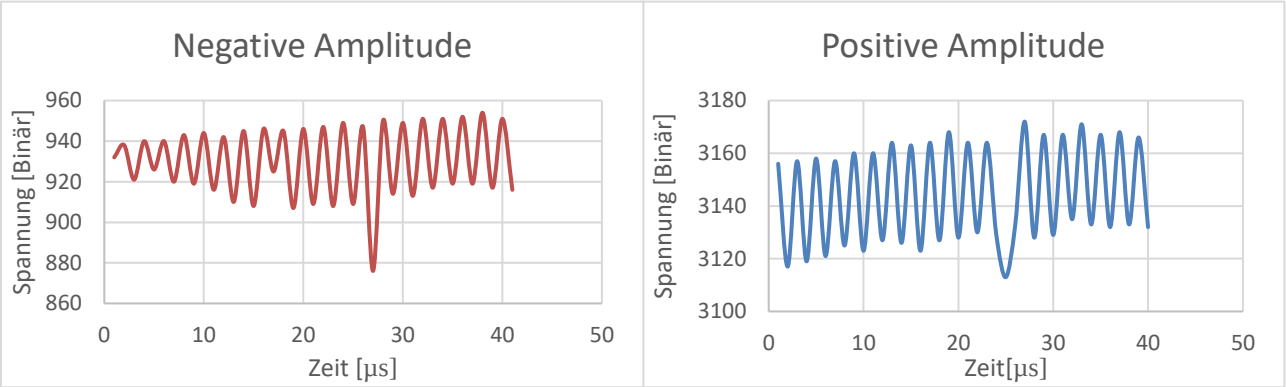

Abbildung 37 - Spannungsamplituden im Vergleich

Eine weitere Ursache der Abweichungen sind vermutlich die Mikrocontrollertoleranzen. In den Messwerttabellen (Seite 63 und 64) weichen die positiven Amplituden am stärksten voneinander ab. Es wurde vermutet, dass einer der Spannungswandler die Werte positiven Flanke verfälscht. Um auszuschließen, dass die Spannungswandler unterschiedlich transformieren, wurden die Mikrocontroller mit einem Funktionsgenerator getestet. Das Ergebnis zeigt, dass Controller zwei, wie die Tabellen auch zeigen, höhere positive Spannungsamplituden als Controller eins erfasst. Da die Amplituden zur Phasenwinkel-Bestimmung nicht benötigt werden, wurde dort nicht weiter nachgearbeitet.

3. Schlussteil

3.1. Fazit

Ziel dieser Bachelorarbeit war die Entwicklung eines Messsystems, welches den Phasenwinkel der Netzspannung bestimmt. Zu diesem Zweck wurden zwei Messgeräte gebaut, welche mit einem Mikrocontroller und einen Spannungswandler die Netzspannung abtasten. Der Mikrocontroller tastet die Netzspannung mit einer Frequenz von 1 MHz ab, was einen Messwert pro µs ermöglicht. Dadurch können 20.000 ($1\,MHz/50\,Hz$) Messwerte pro Spannungsperiode aufgenommen werden. Das Programm wurde so gestaltet, dass sich die Abtastfrequenz bei unterschiedlichen Anforderungen in der Konfigurationsdatei umändern lässt. Zur Referenzzeitermittlung (Δ t) wird der Zeit-Impuls eines GNSS-Moduls genutzt. Die Abweichungen der Zeit-Impulse befinden sich im 10er Nanosekunden Bereich. Der Spannungswandler verringert die Netzspannung um den Faktor 0,00287 und hebt die umgewandelte Spannung um 1,65 V an. Dadurch wurden die negativen Spannungsanteile der Netzspannung in einen positiven Spannungsbereich angehoben. Dies war notwendig, da der Mikrocontroller nur Spannungen über 0 V und unter 3,3 V unbeschadet verarbeiten kann. Somit ergeben sich am Messpunkt der Mikrocontroller Spannungen von 0,72 V bis 2,58 V (±325 V×0,00287+1,65 V). Das Rauschen auf der Messspannung wurde mit einer Regressionsgeraden durch die Nulldurchgänge und einer Kalibrierung des zweiten Messgerätes reduziert, welches an das erste angeglichen wurde. Zusätzlich wird der Mittelwert des Phasenwinkels aus den Messwerten gebildet, welche sich, bevor die Messwerte auf die SD-Karte gespeichert werden, im Arbeitsspeicher befinden.

Durch die Messwertaufnahme wurde ermittelt, dass der Messfehler zwischen den Messgeräten maximal 0,3 º Phasenwinkeldifferenz-pro-Periode aufweist. Das Ziel, herauszufinden, wie sich der Phasenwinkel über Distanz verändert, wurde mit der Simulationsmessung aus Kapitel 2.5.2 dargestellt. Diese hat ergeben, dass sich der Phasenwinkel über Distanz verändert. Die Phasenwinkeldifferenz ist von der Induktivität der Leitung abhängig sowie von dem durchfließenden Strom. Das Ergebnis des Messsystems ist identisch mit dem berechneten Wert. Das Messsystem kann also zur Phasenwinkelmessung zuverlässig genutzt werden.

3.2. Ausblick & Diskussion

Während der Entwicklung des Messsystems sind unterschiedliche Fehler zu Tage getreten, welche nicht behoben werden konnten. Im Experiment zeigte sich, das bei einer Trennung der Mikrocontrollerspannung von der Offsetspannung das Rauschen der Messspannung deutlich verringert werden konnte. Dazu wurden die beiden Spannungen von zwei separaten Netzteilen erzeugt. Als eine mögliche Fehlerquelle wurde in Betracht gezogen, dass der Mikrocontroller bei der Messwertaufnahme in schnellen Zeitabständen Strom zieht, was zu kleinen Spannungseinbrüchen führt. Diese Spannungseinbrüche kann die Offsetspannung nicht schnell genug ausgleichen, weswegen dort ein minimales Rauschen entsteht.
Zur Lösung dieses Problems wurden mehrere Methoden getestet, unter der Bedingung, dass der Mikrocontroller und die Offsetspannung, wieder wie im entwickelten Messsystem, vor Nutzung der Regressionsgeraden, Zugriff auf dasselbe Netzteil hatten:

1. Ein 5 V Spannungsregler vor dem 1,65 V Spannungsregler
2. Einen DC-DC Wandler vor dem Mikrocontroller
3. Testen, wie das Rauschen auf niedrigere Spannung reagiert
4. Ein 1000 µF Elektrolytkondensator parallel zur Ausgangsspannung des Netzteils

Die Korrekturversuche haben das Problem nicht gelöst. An dieser Stelle könnte weiter nachgeforscht werden.
Eine andere mögliche Ursache für das Rauschen könnte ein Massenkonflikt der Messspannung mit der Versorgungsspannung sein. Durch die Offsetspannung werden diese beiden Spannungen miteinander verbunden. Mit einer galvanischen Trennung, z.B. mit einem Optokoppler, könnte vielleicht Abhilfe geschaffen werden.
Ein einfaches Verfahren, durch eine Beseitigung des Rauschens die Messgenauigkeit zu verfeinern, ist, wie das Experiment zeigte, der Einbau eines Netzteils, welches die Offsetspannung und den Mikrocontroller voneinander trennt. Diese Trennung wurde in dem entwickelten Messgerät noch nicht realisiert.

Weitere Mittel und Wege, das Rauschen zu unterbinden, wurden überlegt, ohne realisiert werden zu können. Vielleicht müsste eine andere Art von Spannungswandler konzipiert werden, welcher die zu messende Spannung anders absenkt als in dem hier dargestellten Gerät. Zumal es nicht unbedingt sinnvoll ist, einen Spannungsteiler zur Verminderung der Netzspannung zu nutzen, da dort die Verlustleistungen viel höher als bei einem Transformator sind. Ein Transformator, welcher die Netzspannung gleichmäßig transformiert, könnte, anders als der in Kapitel 2.2.4.4.1 beschriebene, das Problem vielleicht beheben.

Im Kapitel 2.2.4.4.2 wurde in einem anderen Experiment bei einer Erhitzung eines Transformators, eine Veränderung des Trafoübersetzungsverhältnisses festgestellt. Um dem entgegenzuwirken, könnte man einen Temperatursensor an dem Mikrocontroller anbringen, welcher in der Berechnung das veränderte Trafoübersetzungsverhältnis berücksichtigt. Von hier aus müsste die Messwertverfälschung bei Überhitzung genauer untersucht werden, um dann dementsprechend den Mikrocontroller zu kalibrieren.

Weiterhin ist es möglich die Messwerte nicht in der SD-Karte, sondern über die USB-Schnittstelle oder das WLAN-Modul direkt am Computer auszuwerten. Dazu müsste der entsprechende Code in der Software implementiert werden. Über diese Schnittstellen könnte man möglicherweise auch die Konfigurationsmöglichkeiten ändern, sodass der Controller nicht umprogrammiert werden müsste, um Einstellungen minimal zu verändern. Andernfalls könnte man eine die Programmierschnittstelle, welche zum Programmieren des Controllers benötigt wird, verlängern, sodass man von außen Zugriff darauf bekäme und ein Öffnen das Gehäuses vermeiden könnte.

Bei einer Weiterentwicklung des Messgeräts könnte man die Absicherung der Messwerte durch den Einbau einer Batterie verbessern, damit bei einem Stromausfall die Stromversorgung nicht unterbrochen wird. Denn eine solche Unterbrechung, würde die Daten vernichten, aus denen u.a. die Ursache des Stromausfalls abgelesen werden könnte.

Die Simulationsmessung aus Kapitel 2.5.2 könnte auch unter realen Bedingungen getestet werden, um festzustellen, wie sich der Phasenwinkel über weite Entfernungen verändert. Um solche Veränderungen des Phasenwinkels festzustellen, müssten kilometerlange Leitungen genutzt werden.

Anhang

Abbildungsverzeichnis

ABBILDUNG 1 - PHASENLAGEN UND PHASENDIFFERENZ [1] 3
ABBILDUNG 2 - TRANSFORMATOR - FUNKTIONSWEISE [1] 4
ABBILDUNG 3 - ENTWICKLUNGSPLATINE [1] 7
ABBILDUNG 4 - GNSS-MODUL [1] 8
ABBILDUNG 5 - NETZTEIL [1] 8
ABBILDUNG 6 - SPANNUNGSWANDLER [1] 9
ABBILDUNG 7 - SPANNUNGSTEILERKURVEN (TÜRKIS & LILA) & NETZSPANNUNG (GELB) [1] 10
ABBILDUNG 8 - PRINTTRANSFORMATOR [1] 11
ABBILDUNG 9 - SPANNUNGSREGLER [1] 12
ABBILDUNG 10 - AUSWIRKUNGEN DER OFFSET-SPANNUNG [1] 13
ABBILDUNG 11 - RIPPEL DES SPANNUNGSREGLERS [1] 13
ABBILDUNG 12 - SCHALTPLAN SPANNUNGSWANDLER ERSTER VERSUCH [1] 14
ABBILDUNG 13 - TRANSFORMATOR IM LEERLAUF (PRIMÄRSEITE LINKS 60 V, PRIMÄRSEITE RECHTS 110 V) [1] 15
ABBILDUNG 14 - TRANSFORMATOR IM LEERLAUF - PRIMÄRSEITE LINKS 150V, PRIMÄRSEITE RECHTS 230V [1] 16
ABBILDUNG 15 - STEIGENDE & FALLENDE FLANKE BEI 60V [1] 16
ABBILDUNG 16 - STEIGENDE & FALLENDE FLANKE BEI 110V [1] 17
ABBILDUNG 17 - STEIGENDE & FALLENDE FLANKE BEI 150V [1] 17
ABBILDUNG 18 - SPANNUNGSKURVEN: NETZ (GELB), SEKUNDÄRSEITEN DER TRAFOS (LILA & TÜRKIS) [1] 18
ABBILDUNG 19 - STEIGENDE FLANKE [1] 18
ABBILDUNG 20 - FALLENDE FLANKE [1] 18
ABBILDUNG 21 - TÜRKISEFARBENE KURVE: TRAFO NORMALZUSTAND, LILAFARBENE KURVE: TRAFO ERHITZT, GELBE KURVE: NETZSPANNUNG [1] 19
ABBILDUNG 22 - GEHÄUSE [1] 21
ABBILDUNG 23 - EINSCHNITTE [1] 22
ABBILDUNG 24 - USB&IEC - BUCHSE [1] 22
ABBILDUNG 25 - SD- & ANTENNEN-VERLÄNGERUNG [1] 22
ABBILDUNG 26 - USB-VERLÄNGERUNG [18] & MESSBUCHSE [1] 22
ABBILDUNG 27 - BEFESTIGUNGSPLATTE [1] 23
ABBILDUNG 28 - BEIDE MESSGERÄTE BEI GEÖFFNETEM DECKEL. ZUSAMMEN BILDEN SIE DAS MESSSYSTEM [1] 24
ABBILDUNG 29 - SEITENANSICHT EINES MESSGERÄTS [1] 24
ABBILDUNG 30 - DREIECKSPANNUNG & SINUSSPANNUNG [1] 33
ABBILDUNG 31 - SCHWINGUNGEN IN DER NETZSPANNUNG [1] 34

ABBILDUNG 32 – REGRESSIONSGERADE DURCH DIE MESSKURVE [1] 37
ABBILDUNG 33 - BEISPIEL, ERSTELLUNG EINER REGRESSIONSGERADEN. [1] 39
ABBILDUNG 34 - GRAFISCHE DARSTELLUNG - NULLSTELLENERMITTLUNG [1] 46
ABBILDUNG 35 – SIMULATIONSMESSUNG, SCHALTPLAN 65
ABBILDUNG 36 – SIMULATIONSMESSUNG, OSZILLOSKOPAUFZEICHNUNG 66
ABBILDUNG 37 – SPANNUNGSAMPLITUDEN IM VERGLEICH 67
ABBILDUNG 38 – SCHALTPLAN DER MESSANORDNUNG 75

Tabellenverzeichnis

TABELLE 1 – MINIMALE ABTASTFREQUENZ [1] 33
TABELLE 2 - FUNKTIONSVORGANG, BESTIMMUNG DES MESSZEITPUNKTES [1] 36
TABELLE 3 – MESSWERTAUFNAHME: IDENTISCHE POSITION, MINIMALE ABWEICHUNG [1] 62
TABELLE 4 – MESSWERTAUFNAHME: IDENTISCHE POSITION, MITTLERE ABWEICHUNG [1] 63
TABELLE 5 – MESSWERTAUFNAHME: IDENTISCHE POSITION, MAXIMALE ABWEICHUNG [1] 63
TABELLE 6 – MITTELWERT DES PHASENWINKELS AUS EINER MESSWERTAUFNAHME [1] 64
TABELLE 7 - MESSWERTAUFNAHME, SIMULATIONSMESSUNG UNTER LAST [1] 66

Formelverzeichnis

Formelbezeichnung:	*Seite:*
1. Spannungsteiler	10
2. Transformator-Übersetzungsverhältnis	11
3. Spannungsregler	12
4. Timer-Perioden-Register	27
5. Volt in Bit	28
6. Auflösung AD-Wandler	32
7. Zeitänderung	33
8. Frequenz	33
9. Steigung	39
10. Achsenabschnitt	39
11. Regression	40
12. Bit in Volt	45
13. Differenz der Messwerte	46
14. Frequenz aus Messwerten	46
15. Übersetzungsformel	50
16. Zeitreferenz – Δ t	51
18. Phasenwinkel - φ	53
19*. Phasenverschiebung*	65
20. Induktiver Blindwiderstand	65
21. Ohmsches Gesetz	65

Literaturverzeichnis

[1] Seldschopf, Malte: Eigene Darstellung - 2019

[2] Lipinski, Klaus: Phasor-Messgerät – September 2014
https://www.itwissen.info/Phasor-Messgeraet-phasor-measurement-unit-PMU.html

[3] Amprion: Transformator – Oktober 2018
https://www.amprion.net/%C3%9Cbertragungsnetz/Technologie/Transformator/

[4] Leitner, Ernst und Finckh, Ulrich: Transformator - November 2018
https://www.leifiphysik.de/elektrizitatslehre/transformator-fernubertragung

[5] Schmitt, Franz-Josef: Magnetische Sättigung - 2011
https://www.supermagnete.de/magnetismus/magnetsaettigung

[6] Sebastian, Dworatschek: Einführung in die Datenverarbeitung – Wien, De Gruyther, 1976

[7] Schulz, Detlef: Netzrückwirkungen - Theorie, Simulation, Messung und Bewertung
VDE VERLAG, 2004

[8] Chapman, David: Oberschwingungen – Mai 2002
https://www.kupferinstitut.de/fileadmin/user_upload/kupferinstitut.de/de/Documents/Shop/Verlag/Downloads/Anwendung/Elektrotechnik/Leonardo/5.1_SpannungsEinbrueche.pdf

[9] Voltus: Oberschwingungen - 2019
https://www.voltus.de/blog/oberschwingungen-ursachen-und-auswirkungen/

[10] Chapman, David: Spannungseinbrüche - April 2002
https://www.kupferinstitut.de/fileadmin/user_upload/kupferinstitut.de/de/Documents/Shop/Verlag/Downloads/Anwendung/Elektrotechnik/Leonardo/3.1_Oberschwingungen.pdf

[11] Digilent: ChipKIT Wi-FIRE, Datenblatt - Juli 2016
https://reference.digilentinc.com/_media/chipkit_wifire/chipkit_wi-fire_rm_revc.pdf.

[12] MikroElektronika: GNSS 5 Click, Datenblatt - 2017
https://static5.arrow.com/pdfs/2017/10/15/1/18/11/227/mikroe_/manual/pgurl_gnss-5-click.pdf

[13] MeanWell: Netzteil, Datenblatt - November 2016
https://docs-emea.rs-online.com/webdocs/15c7/0900766b815c75ca.pdf

[14] BLOCK: Printtransformator, Datenblatt - 2019
https://catalog.block-trafo.de/prodvardatasheet/393256-459148_DE

[15] Fairchild: Spannungsregler, Datenblatt - Dezember 2012
https://docs-emea.rs-online.com/webdocs/10e2/0900766b810e250b.pdf

[16] ABB: Gehäuse - 2019
https://docs-emea.rs-online.com/webdocs/0af8/0900766b80af841f.pdf

[17] TutorialsPoint: C-Funktionen, Sprintf - 2019
https://www.tutorialspoint.com/c_standard_library/c_function_sprintf.htm

[18] Wikimedia: USB-Kabel - November 2008
https://de.wikipedia.org/wiki/Universal_Serial_Bus

Gesamter Schaltplan

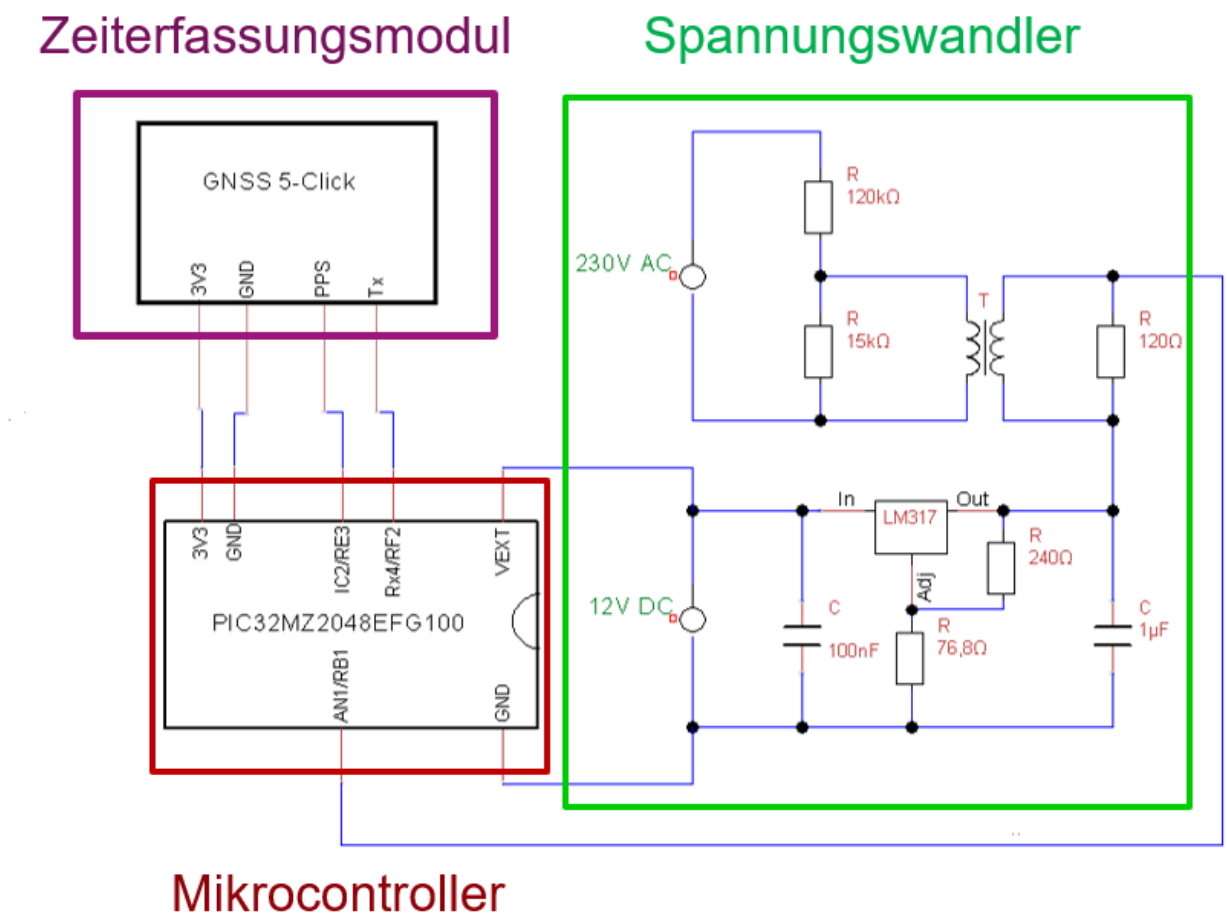

Abbildung 38 - Schaltplan der Messanordnung