Technology
Arts Sciences
TH Köln

**Diese Arbeit wurde vorgelegt am Institut für Elektrische Energietechnik**

# Vergleich und Auswertung von Verfahren der Frequenzmessung am Beispiel der Netzfrequenzmessung

Implementierung und Simulation ausgewählter Verfahren

**Bachelorarbeit**

von

**Semih Karagözoglu**

| | |
|---|---|
| Erstprüfer: | Prof. Dr. Eberhard Waffenschmidt |
| Zweitprüfer: | Prof. Dr. Beate Rhein |
| Tag der Abgabe: | 15. Januar 2018 |

# Erklärung

Ich versichere hiermit, dass die vorliegende Bachelorarbeit, bis auf die Betreuung und Hilfestellung durch das Institut für Elektrischen Energietechnik, von mir eigenhändig und ohne fremde Hilfe angefertigt wurde. Die verwendete Literatur ist vollständig aufgeführt.

Köln, den 05. September 2017

# Kurzfassung

Die vorliegende Arbeit behandelt den Vergleich und die Auswertung von Verfahren der Netzfrequenzmessung und damit zusammenhängend die mikrocontroller-basierte Implementierung und Simulation der vorgestellten Verfahren. Damit besteht das Ziel darin ausgewählte vorhandene Verfahren der Netzfrequenzmessung und Netzfrequenzanalyse am Beispiel der Netzfrequenzmessung programmiertechnisch und elektrotechnisch zu Implementieren und somit einem Vergleich zu unterziehen.

Zunächst werden die vorhandenen Verfahren konzeptionell vorgestellt. Dabei wird auf das Allgemeine Verfahren der Netzfrequenzmessung eingegangen, welcher sich der Signalumformung bedient und anschließend werden Verfahren der Netzfrequenzanalyse dargestellt.

Im Zuge der Simulation der Verfahren werden zunächst die Vorschaltung für eine mikrocontroller-basierte Lösung mit dem Ziel der Signalanalyse getestet. Während die Frequenz über das erste Verfahren direkt berechnet werden kann, da es aus einen sinusförmigen Signal in ein Rechtecksignal umgewandelt wird. Mit dem zweiten Verfahren erfolgt die Frequenzbestimmung hingegen mittelbar aus der Signalanalyse, während die Qualität des Signals beibehalten wird. Damit einhergehend müssen also effektive Algorithmen getestet werden die für die Signalanalyse herangezogen werden können und konzeptionell vorgestellt wurden. Basierend auf den Simulationen der Schaltungen sowie Algorithmen können somit die ausgewählten Verfahren implementiert werden.

Die Arbeit endet mit einem Fazit und Vergleich der implementiert Verfahren, dabei wird auf die Effektivität der Verfahren eingegangen und zudem weitere Implementierungsmöglichkeiten in Ausblick gestellt, da eine mikrocontroller-basierte Lösung einfach und effektiv ausbaufähig ist.

# Abstract

The present work deals with the comparison and evaluation of power frequency measurement methods and the related microcontroller based implementation and simulation of the presented techniques. Thus, the aim is to implement selected existing methods of network frequency measurement and network frequency analysis using the example of the network frequency measurement programming technology and electrical engineering and thus subject to a comparison.

First, the existing procedures are conceptually presented. In this case, the general method of network frequency measurement is discussed, which makes use of the signal conversion and then procedures of the network frequency analysis are shown.

In the course of the simulation of the methods, first the pre-circuit for a microcontroller-based solution with the aim of signal analysis will be tested. While the frequency can be calculated directly via the first method, since it is converted from a sinusoidal signal into a square wave signal. On the other hand, with the second method, the frequency determination is made indirectly from the signal analysis while keeping the quality of the signal. Along with this, effective algorithms must be tested which can be used for signal analysis and have been conceptually presented. Based on the simulations of the circuits as well as algorithms, the selected methods can thus be implemented.

The work concludes with a conclusion and comparison of the implemented procedures, while considering the effectiveness of the procedures and also further implementation possibilities in view, since a microcontroller-based solution is easy and effectively expandable.

# Inhaltsverzeichnis

Seite

**Erklärung** ..... **iii**

**Kurzfassung** ..... **vii**

**Abstract** ..... **ix**

**Inhaltsverzeichnis** ..... **xi**

**Abbildungsverzeichnis** ..... **xiii**

**Tabellenverzeichnis** ..... **xv**

**1 Einleitung** ..... **1**

1.1 Motivation und Zielsetzung ..... 1

1.2 Aufbau der Arbeit ..... 2

**2 Verfahren der digitalen Frequenzmessung** ..... **3**

2.1 Allgemeine Vorgehensweise ..... 3

2.2 Analoge Vorschaltung ..... 5

2.2.1 Tiefpassfilterung ..... 5

2.2.2 Einweggleichrichter und Spannungslimitierung ..... 6

2.2.3 Schmitt-Trigger ..... 7

2.3 Mikrocontroller-basierter Zählalgorithmus ..... 10

2.4 Phasenregelkreis (PLL) ..... 11

**3 Verfahren der digitalen Frequenzanalyse** ..... **13**

3.1 Allgemeine Vorgehensweise ..... 13

3.2 Analoge Vorschaltung ..... 15

3.2.1 Bandpassfilter ..... 16

3.2.2 Nicht-invertierender Verstärker ..... 17

3.2.3 A/D-Wandler ..... 18

3.3 Fast Fourier Transformation (FFT) ..... 20
3.4 Sinus Fit Algorithmus ..... 24
3.5 Autokorrelation ..... 26

**4 Simulation ausgewählter Verfahren ..... 27**

4.1 Dimensionierung der Vorschaltungen ..... 28
4.1.1 Schaltung 1 ..... 28
4.1.2 Schaltung 2 ..... 30
4.2 Simulation der Vorschaltungen ..... 32
4.2.1 Schaltung 1 in Multisim ..... 32
4.2.2 Schaltung 2 in Multisim ..... 34
4.3 Testen der Algorithmen ..... 36
4.3.1 Sinus Fit Algorithmus in Scilab ..... 37
4.3.2 Fast Fourier Transformation (FFT) in Scilab ..... 42
4.3.3 Autokorrelation in Scilab ..... 46

**5 Implementierung ausgewählter Verfahren ..... 47**

5.1 Frequenzzählung ..... 47
5.2 Messdatenerfassung mit dem Mikrocontroller ..... 49
5.3 Sinus Fit Algorithmus ..... 53
5.4 Fast Fourier Transformation ..... 55
5.5 Autokorrelation ..... 56

**6 Fazit ..... 57**

6.1 Zusammenfassung und Vergleich ..... 57
6.2 Fazit und Ausblick ..... 61

**Literaturverzeichnis ..... 63**

**Abkürzungsverzeichnis ..... 66**

**Anhang ..... 69**

# Abbildungsverzeichnis

Seite

Abbildung 2-1:Sinus mit Rauschen ..... 3

Abbildung 2-2:Digitalsignal im Bezug zur Zeitbasis ..... 4

Abbildung 2-3: RC-Glied 1. Ordnung ..... 6

Abbildung 2-4:Einweggleichrichter und Z-Diode ..... 7

Abbildung 2-5:Optokoppler und Schmitt-Trigger ..... 8

Abbildung 2-6: Struktogramm für den Zählalgorithmus ..... 10

Abbildung 2-7:Blockschaltbild eines Phasenregelkreises ..... 11

Abbildung 2-8:Fehlersignal des Phasendetektors ..... 12

Abbildung 3-1: Abgetasteter Sinus ..... 13

Abbildung 3-2:Bandpassfilter ..... 16

Abbildung 3-3:Verstärkerschaltung ..... 17

Abbildung 3-4:3-Bit A/D-Wandlung ..... 19

Abbildung 3-5:FFT Schmetterlingsgraph ..... 22

Abbildung 3-6:Struktogramm für den Sinus Fit Algorithmus ..... 25

Abbildung 3-7:Autokorrelation eines Sinus mit Rauschen ..... 26

Abbildung 4-1: Vorbereitung der Signalverarbeitung ..... 27

Abbildung 4-2:Bodediagramm des Tiefpassfilters ..... 33

Abbildung 4-3:Ausgangssignal der Schaltung 1 ..... 33

Abbildung 4-4:Bodediagramm des Bandpassfilters ..... 34

Abbildung 4-5:Mittelwertbildung zweier Signale ..... 35

Abbildung 4-6:Ausgangssignal der Schaltung 2 ..... 36

Abbildung 4-7:Frequenzschätzung Teil 1 ..... 37

Abbildung 4-8:Frequenzschätzung Teil 2 ..... 38

Abbildung 4-9: Sinussignal in Scilab ..... 39

Abbildung 4-10:Sinussignal mit 50Hz und 50 Perioden ..... 42

Abbildung 4-11:FFT Diagramm für das Sinussignal mit 50Hz ..... 43

Abbildung 5-1:Vorschaltung für die Rechteckumformung ..... 47

Abbildung 5-2:Ausgangssignal von Schaltung 1 ..... 48

Abbildung 5-3:Ausgabe am seriellen Monitor ..... 49

Abbildung 5-4:Vorschaltung für die Messdatenerfassung ..... 50

Abbildung 5-5:Ausgangssignal von Schaltung 2 ..... 50

Abbildung 5-6:Skript zum Einlesen der Messwerte ..... 51

Abbildung 5-7:Dargestellte Messung des Mikrocontrollers ..... 52

Abbildung 5-8:Horizontale Ausgabe der Messwerte in Putty ..... 53

# Tabellenverzeichnis

Seite

Tabelle 3-1: Effizienz der FFT ..... 23

Tabelle 4-1: Dimensionierung von Schaltung 1 ..... 29

Tabelle 4-2:Dimensionierung von Schaltung 2 ..... 31

Tabelle 4-3:Sinus Fit mit Frequenzschätzung ..... 40

Tabelle 4-4:Mittlere Fehlerquadrat für Messwerte ohne Rauschen ..... 41

Tabelle 4-5:Sinus Fit für Funktionen mit Rauschen ..... 41

Tabelle 4-6:Sinus Fit mit FFT für Funktionen mit Rauschen ..... 44

Tabelle 4-7:FFT mit Ausgabe des Frequenz-Bins und Delta f ..... 45

Tabelle 4-8:FFT mit Erhöhung der Auflösung ..... 45

Tabelle 4-9:Autokorrelation bei Signalen mit und ohne Rauschen ..... 46

Tabelle 4-10:Autokorrelation mit höherer Abtastung ..... 46

Tabelle 5-1:Sinus Fit für reale Messwerte ..... 54

Tabelle 5-2:FFT für reale Messungen ..... 55

Tabelle 5-3:Autokorrelation und Frequenzbestimmung der Messungen ..... 56

Tabelle 6-1:Test 1 Eingabesignal weist kein Rauschen auf ..... 59

Tabelle 6-2:Test 2 Eingabesignal weist Rauschen auf ..... 60

Tabelle 6-3:Test 3 Eingabe stammen aus Messdatenerfassung ..... 60

# 1 Einleitung

## 1.1 Motivation und Zielsetzung

Das Verbundnetz weist regelmäßige Abweichungen der Netzfrequenz auf, welches durch Überangebot bzw. Unterangebot zustande kommt. Wird vom Erzeuger mehr elektrische Leistung angeboten als aufgenommen werden kann, steigt die Netzfrequenz. Da sich diese Unregelmäßigkeiten aber im Rahmen von 0,2 Hz aufhalten, können diese Abweichungen gut geregelt werden. Trotz dessen kam es im Jahre 2006 ausgelöst durch einen Stromausfall in Europa zu einem Unterangebot an Leistung und somit zu einem starken Abfall der Netzfrequenz. Auch der Ausbau der Erneuerbaren Energien fordert immer höhere Anforderungen an die Überwachung vom Verbundnetz im Hinblick auf die Netzfrequenz. Durch herkömmliche mechanisch-analoge, elektronische sowie digitale Messmethoden kann eine solche Überwachung durchgeführt werden. Dabei spielt die digitale Messtechnik eine vorrangige Rolle um automatisierte Frequenzmessungen durchführen zu können.

Daher werden in dieser Arbeit herkömmliche Verfahren der digitalen Frequenzmessung und Frequenzanalyse vorgestellt und verglichen. Dabei wird mit Hilfe der Methoden der Signalverarbeitung auf die Problemstellung der Frequenzmessung bei überlagerten Signalen näher eingegangen und verschiedene Algorithmen zur Frequenzmessung ausgesucht und verglichen. Die ausgewählten Algorithmen werden dann in nächsten Schritt durch Simulation und programmiertechnisch mit Hilfe eines Mikrocontrollers am Beispiel der Netzfrequenzmessung implementiert. Ziel der Arbeit ist es die ausgewählten Verfahren im allgemeinem vorzustellen und basierend auf den vorgestellten Konzepten eine Lösung zu implementieren.

## 1.2 Aufbau der Arbeit

In der vorliegenden Bachelorarbeit werden in den Kapiteln 2 und 3 zunächst als Basis der Netzfrequenzmessung mit Hilfe eines Mikrocontrollers verschiede Methoden konzeptionell für die Netzfrequenzmessung vorgestellt. Dabei werden nicht nur die programmiertechnischen Verfahren, sondern auch elektrotechnische Verfahren mit dem Ziel der Implementierung in Betracht gezogen. In Kapitel 2 wird dabei zunächst auf die direkte Frequenzmessung durch Signalumformung eingegangen.

In Kapitel 3 werden Methoden der Signalverarbeitung vorgestellt, welche zur Frequenzanalyse dienen, jedoch auch zur Implementierung einer Frequenzmessung genutzt werden können. Hier spielen vor allem effektive Algorithmen eine vorrangige Rolle.

Mit Kapitel 4 erfolgt die Simulation der elektrotechnischen sowie programmiertechnischen Verfahren die ein Ganzes bilden. Dabei werden die vorgestellten Algorithmen eingehend untersucht.

Hieran schließt sich in Kapitel 5 die Implementierung der simulierten und getesteten Verfahren an. Zum einen erfolgt dies im eigentlichen Schaltungsaufbau und durch die programmiertechnische Implementierung.

Die Arbeit schließt in Kapitel 6 mit einem Vergleich der ausgewählten Verfahren und einem Fait ab.

# 2 Verfahren der digitalen Frequenzmessung

## 2.1 Allgemeine Vorgehensweise

Im Allgemeinen gibt es zwei Verfahren der herkömmlichen digitalen Frequenzmessung zur Bestimmung der Frequenz von Wechselspannungen. Entweder über die direkte Frequenzmessung oder aber reziprok über die Periodendauermessung. Beider Messungen liegt eine Zählerschaltung zugrunde, die die Nulldurchgänge des gegebenen Signals detektiert und zählt [RHL16]. Dazu muss das Eingangssignal zunächst in ein Recktecksignal umgewandelt werden. Diese Methode dient vor allem der Vereinfachung eines analogen Signals, welches nicht nur aus einem Frequenzanteil besteht, da es von harmonischen Schwingungen anderer, meist hoher Frequenzen, oder einem Rauschen (s. Abb. 2-1) überlagert ist.


Abbildung 2-1:Sinus mit Rauschen

Um von diesem verrauschten Signal die Nulldurchgänge zu detektieren wird nach Spannungstransformation und Tiefpassfilterung eine elektrische Komparator-Schaltung der sogenannte Schmitt-Trigger (s. Kap. 2.2) eingesetzt, dessen Ausgangssignal ein Rechtecksignal ist und die Nullstellen des Eingangssignals somit die positiven bzw. negativen Flanken des Rechtecks sind. Mit diesem Verfahren kann man Verfälschungen durch mehrfach Durchgänge an den Null-

stellen oder durch generelles Rauschen verhindern. Dabei ist zu beachten, dass sich das Signal-Rausch-Verhältnis des Signals, welcher den Wiedererkennungswert eines Signals angibt, im Zulassungsbereich befindet. Ist dies der Fall kann aus dem digitalen Signal die gesuchte Frequenz $f_x$ bestimmt werden, welcher nach Umwandlung in Form eines Rechtecksignals vorliegt.

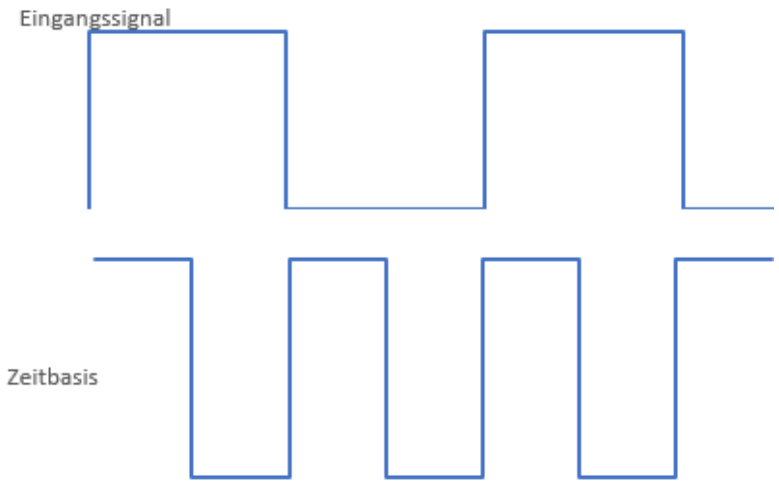



Abbildung 2-2:Digitalsignal im Bezug zur Zeitbasis

Abb. 2-2 verdeutlicht, dass das Digitalsignal zur Weiterverarbeitung benutzt wird um mit Hilfe einer Zeitbasis, ein Rechtecksignal mit bekannter Periodendauer, die gesuchte Frequenz $f_x$ zu bestimmen. Dabei werden die übereinstimmenden positiven oder negativen Flanken innerhalb einer definierten Zeit $T_{ref}$ gezählt [NIC17]. Benutzt man in diesem Zusammenhang einen Mikrocontroller, kann die Anzahl der steigenden Flanken $N_x$ am digitalen Eingang direkt ermittelt werden, wobei die Zeit $T_{ref}$ die resultierende Messdauer darstellt. Aus dieser Überlegung ergibt sich dann die Formel für die gesuchte Frequenz [RHL16]:

$$f_x = \frac{N_x}{T_{ref}}$$

Nimmt man an, dass die Referenzzeit in der gemessen wird $T_{ref}$ = 40ms beträgt und die Zeitbasis eine Periodendauer von $T_z$ = 4ms hat, würde die Anzahl der gezählten positiven Flanken, bei einem Signal mit der Periodendauer von $T_s$ =

20ms, $N_x$ = 2 betragen. Benutzt man die obige Formel ergibt sich daraus die gesuchte Frequenz von $f_x$ = 50 Hz. Zu beachten ist, dass die erste positive Flanke nicht mitgezählt wird, sie löst ausschließlich einen Timer-Start (s. Kap. 2.3) aus.

## 2.2 Analoge Vorschaltung

Da das Prinzip der Nulldurchgangsdetektion auf dem Schmitt-Trigger, einer elektronischen Schaltung, beruht ist es von Nöten eine Anpassung der gemessenen Netzspannung vorzunehmen. Dies erreicht man indem man die Netzspannung von 230V auf eine für die analoge Schaltung passende Spannung runter transformiert. Anschließend kann eine Tiefpassfilterung von hochfrequenten Anteilen vorgenommen werden. Dabei wird von der 50Hz Netzfrequenz ausgegangen. Um die Aufgabe des Schmitt-Triggers weiter zu vereinfachen und den Zulassungsbereich des Mikrocontrollers beizubehalten ist es vorgesehen die negativen Halbwellen der Netzspannung mit einer Diode zu entfernen und eine Spannungslimitierung mit einer zusätzlichen Zener-Diode vorzunehmen. Durch galvanische Trennung[1] der Vorschaltung mittels eines Optokopplers kann der Schmitt-Trigger schließlich zum Einsatz kommen. [INA15]

### 2.2.1 Tiefpassfilterung

Wurde die Netzspannung auf eine für die elektronischen Bauteile konforme Spannung runtertransformiert kann eine Tiefpassfilterung zur Glättung der Wechselspannung und Rauschunterdrückung von hochfrequentem Rauschen erfolgen. Ein solcher Filter besteht aus einem in Reihe geschalteten Winderstand und einem Kondensator (s. Abb. 2-3), welches auch als RC Glied be-

[1] Die galvanische Trennung dient als Schutzmaßnahme bei einer Signalübertragung zwischen zwei Schaltungen die elektrisch nicht verbunden sind, hierbei wird das Signal optoelektronisch übertragen

zeichnet wird. Über das Verhältnis der Ein- und Ausgangsspannung und der Impedanzen kann die Grenzfrequenz berechnet werden [PSD10], welcher für die Netzfrequenzmessung ungefähr bei 50 Hz eingestellt wird. Demnach werden alle Frequenzen oberhalb der 50 Hz Grenze gefiltert.

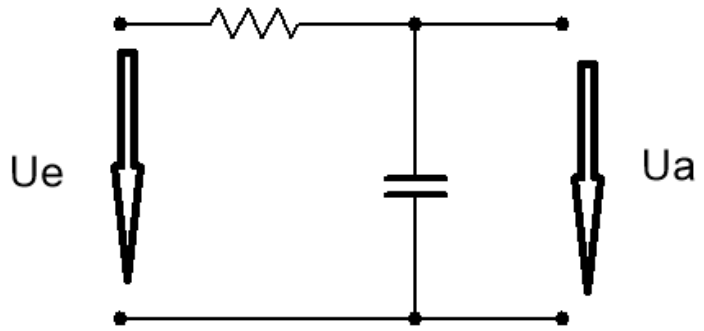



Abbildung 2-3: RC-Glied 1. Ordnung

Über die Formel der Grenzfrequenz $f_g$ kann schließlich die Kapazität des Kondensators bestimmt werden, hierbei wird der Wert des Widerstands durch Auswahl z.B. 1kΩ vorgegeben:

$$f_g = \frac{1}{2\,\pi\,R\,C}$$

### 2.2.2 Einweggleichrichter und Spannungslimitierung

Nach der Tiefpassfilterung werden die negativen Halbwellen der Wechselspannung welcher am Ausgang der Tiefpassschaltung gemessen werden kann entfernt. Dies kann man mit einem Einweggleichrichter erreichen (s. Abb. 2-4). Zusätzlich dazu wird eine Zener-Diode in entgegengesetzter Durchlassrichtung mit der Einbruchsspannung von ca. 5V geschalten, damit die Ausgangsspannung die 5V Grenze, welcher vor allem für die Weiterverarbeitung des Signals dienlich ist, nicht überschreitet. In Abb. 2-4 erkennt man den Einbruch des Span-

nungsverlaufs, welcher am Ausgang der dargestellten Schaltung zu messen ist. Dieser Effekt wird als Zener-Effekt[2] bezeichnet welcher in Sperrrichtung auftritt, wohingegen in Durchlassrichtung das Verhalten der Diode sich nicht von einer einfachen Diode unterscheidet [ELK17].

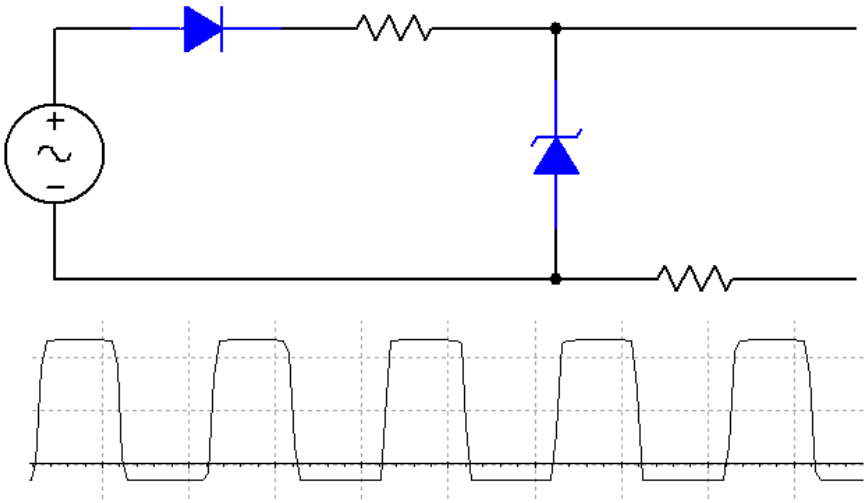


Abbildung 2-4:Einweggleichrichter und Z-Diode

Außerdem kommen zwei Widerstände zum Einsatz welche den Zweck haben die Flanken der Ausgangsspannung gerader und steiler zu machen. Der erste Widerstand ist in diesem Zusammenhang wichtig da der Einweggleichrichtiger einen Spannungseinbruch verursacht, welcher durch den in Reihe geschalteten Widerstand vermieden werden kann.

### 2.2.3 Schmitt-Trigger[3]

Bevor das Signal digital weiterverarbeitet und z.B. mit einem Rechtecksignal als Zeitbasis (s. Kap. 2.1) verglichen werden kann ist es von Nöten über dem Schmitt-Trigger ein präzises Rechtecksignal mit steilen Flanken zu formen. Da die

[2] Bei dem Zener-Effekt kommt es bei einer bestimmten Sperrspannung zu einer schlagartigen Zunahme des Stroms und zum Durchbruch der Spannung

[3] Der hier verwendete Schmitt-Trigger ist ein invertierender Schmitt-Trigger, da das Eingangssignal am invertierenden Eingang anliegt

erste Schaltungshälfte direkt am Netz verbunden ist empfiehlt es sich zunächst eine galvanische Trennung mittels eines Optokopplers[4] vorzunehmen [PMS96]. Der Optokoppler besteht aus einer Photodiode als Signalsender und einem Phototransistor als Signalempfänger. Das empfangene Signal kann dann direkt am Eingang des Schmitt-Triggers angelegt werden (s. Abb. 2-5).

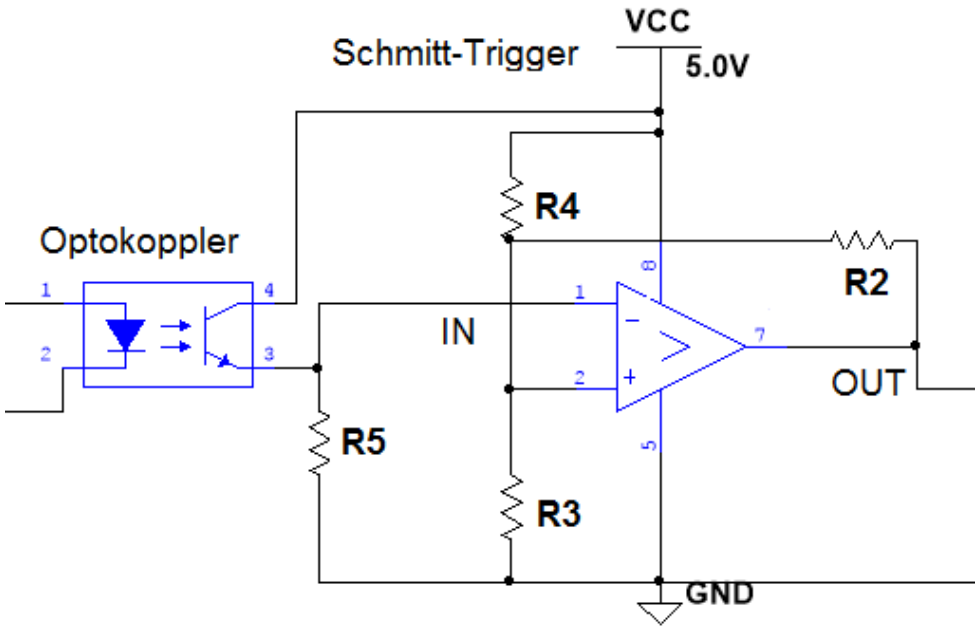



Abbildung 2-5:Optokoppler und Schmitt-Trigger

Der in Abb. 2-5 dargestellte Schmitt-Trigger besteht aus einem Komparator, welcher die Spannungen am nicht-invertierenden und invertierenden Eingang vergleicht. Ist die Spannung am invertierenden Eingang, an dem das Signal anliegt, kleiner als die Referenzspannung[5] $U_{ref}$ am nicht-invertierenden Eingang erhält man am Ausgang der Schaltung einen „High“, andernfalls einen

[4] Der hier verwendete Optokoppler kann durch die einfache Diode im Eingang nur die positive Halbwelle durchlassen und ist somit für die Verwendung im AC Betrieb nicht geeignet

[5] Die Referenzspannung liegt an R3 an und kommt durch den Spannungsteiler R3 und R4 zustande. In Bezug zum Widerstand R2 sind die Widerstände R3 und R4 parallel.

„Low“ [STE17]. Durch den mitgekoppelten Widerstand $R_2$, auch positive Feedback genannt, wird die Referenzspannung zustandsabhängig verändert und somit erhält man zwei Schwellenwerte die die Hysterese[6] bilden. Ist der Zustand demnach „High“ bewirkt die Mitkopplung einen höheren Schwellenwert. Erreicht das Signal diesen höheren Schwellenwert schaltet der Komparator in den Zustand „Low“, wodurch der Schwellenwert auch niedriger wird. Nun muss das Signal den niedrigeren Schwellenwert erreichen damit der Zustand wieder geändert wird. So erhält man am Ausgang der Schaltung ein präzises Rechtecksignal unabhängig davon ob das Signal am Eingang rauschbehaftet ist.

Bei der Dimensionierung der Widerstände müssen Hysterese $U_H - U_L$, die „High/Low“- Werte, die zwischen $V_{cc}$ und GND (s. Abb. 2-5) pendeln und mindestens ein Widerstand z.B. $R_4$ durch Auswahl bekannt sein. Die restlichen Widerstände lassen sich über die Referenzspannung berechnen [GKK02]:

$$U_{ref} = \frac{U_L V_{cc}}{V_{cc} - U_H + U_L}$$

Mit dem Spannungsteiler kann der Widerstand $R_3$ berechnet werden:

$$R_3 = R_4 \frac{U_{ref}}{V_{cc} - U_{ref}}$$

Zur Berechnung des Widerstands $R_2$ werden folgende Formeln benutzt:

$$R_1 = \frac{R_3 R_4}{R_3 + R_4}$$

$$R_2 = R_1 \frac{U_H - V_{cc} - U_L}{U_L - U_H}$$

[6] Die Hysterese ist die Spannung zwischen den beiden Schwellenwerten

## 2.3 Mikrocontroller-basierter Zählalgorithmus

Das am Ausgang der analogen Schaltung gewonnen Signal enthält die Frequenzinformation des Eingangssignals und liegt in digitalisierter Form vor. Somit ist es für die Weiterverarbeitung durch einen Mikrocontroller gut geeignet. Der Mikrocontroller kann das Signal über einen digitalen Eingang einlesen und erkennt an welchen Stellen das Signal steigende Flanken aufweist. Wird die erste steigende Flanke erkannt, löst dies einen Timer-Start aus. Um ein genaueres Ergebnis zu erhalten werden die Flanken über mehrere Perioden gezählt. Ist eine definierte Periodenanzahl erreicht wird ein Timer-Stop ausgeführt und die Frequenz kann mit der in Kapital 2.1 angegeben Formel bestimmt werden. Im weiteren Verlauf müssen die Zählvariablen zurückgesetzt werden, damit die nächste Routine für eine weitere Messung durchgeführt werden kann, dabei ist zu beachten, dass diese Messung/Zählung in einer Endlosschleife durchgeführt wird (s. Abb.2-6).

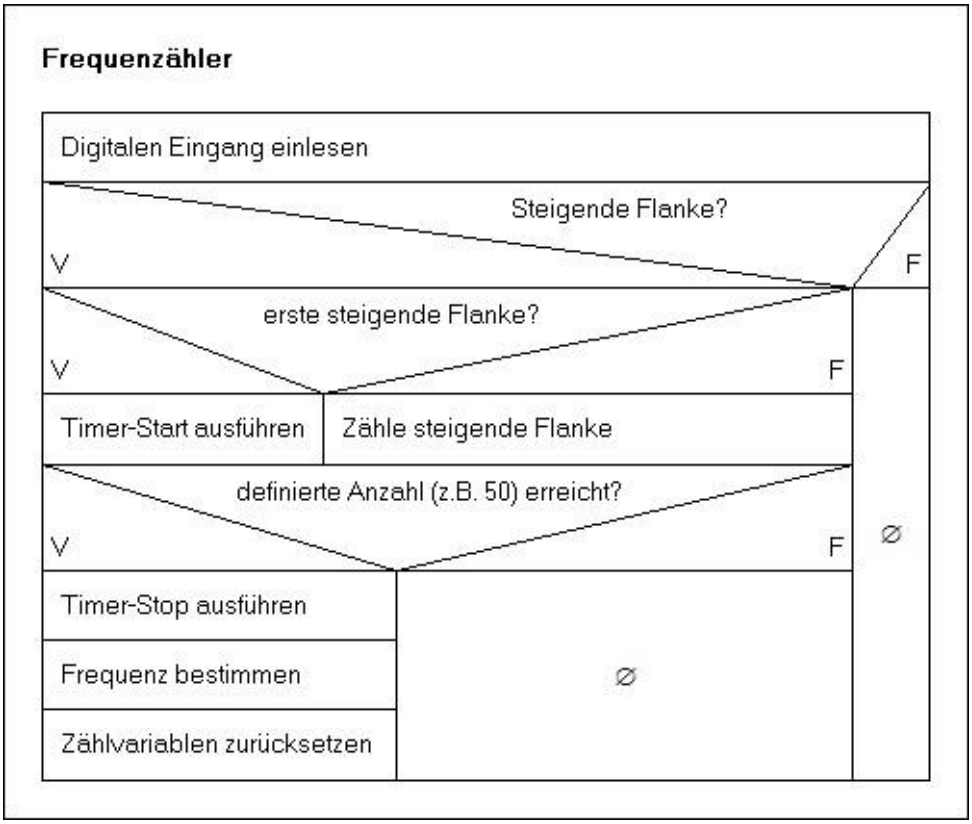



Abbildung 2-6: Struktogramm für den Zählalgorithmus

Das Struktogramm in Abb.2-6 stellt den Programmablauf für die Frequenzzählung dar, während *V=True* und *F=False* bedeutet [DMU15]. Diese Zählung wird in einer Endlosschleife, welcher auch als Loop Routine (hier Frequenzzähler) bezeichnet wird, durchgeführt. Für eine Zählung/Messung wird die Schleife, wenn man von einer Zählung über 50 Perioden ausgeht, 51mal durchlaufen.

## 2.4 Phasenregelkreis (PLL)

Mit der in Kapitel 2.1 bis 2.3 vorgestellten Methode ist es möglich die Frequenzinformation eines rauschbehafteten Signals zu extrahieren und anschließenden über einen Zählalgorithmus die gesuchte Frequenz zu bestimmen. Liegen jedoch Schwankungen hinsichtlich einer Phasenverschiebung vor kann ein Phasenregelkreis, auch Phase Locked Loop genannt, optional zwischen der analogen Vorschaltung und dem Mikroprozessor geschaltet werden. Dieser Regelkreis vergleicht und regelt die Phasen und die Frequenzen vom Eingangssignal und eines Referenzsignals, welches über eine Rückführung am Ausgang des Regelkreises gewonnen wird (s. Abb. 2-7).

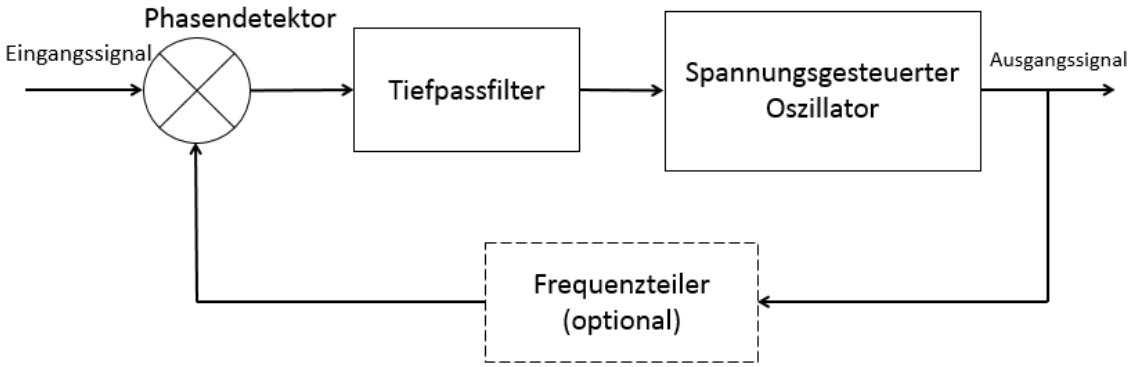



Abbildung 2-7:Blockschaltbild eines Phasenregelkreises

Die Komponenten des Regelkreises bestehen aus einen Phasendetektor, einem Tiefpass, und einem spannungsgesteuerten Oszillator [PMS96]. Der Phasendetektor ermittelt die Phasenunterschiede und Frequenzen der am Eingang anliegenden Signale, dabei ist die Ausgabe je nach Art des Detektors nicht immer der gesuchte Unterschied bzw. es können weitere Frequenzanteile enthal-

ten sein, die das Ergebnis verfälschen. Aus diesem Grund wird im nächsten Schritt ein Filter als Tiefpassfilter aufgestellt, welcher hohe Frequenzen filtern soll. Stimmen die Frequenzen bei unterschiedlicher Phasenlage vom Eingang und Ausgang überein erhält man am Ausgang des Filters eine Konstante die den Phasenunterschied der beiden Signale enthält [HBE09]. Mit dieser Konstanten kann der Oszillator gesteuert werden. Im Freilaufzustand weist dieser eine Freilauffrequenz auf welcher durch die Konstante im Eingang so verändert wird, dass das Ausgangssignal und Eingangssignal nach und nach gleiche Frequenzen aufweisen und in den Selben Phasengang einrasten. Kommt es also zu einer Phasenverschiebung am Eingang regelt der Phasenregelkreis diese abrupte Veränderung, wobei es zu einer Einrastzeit kommt. Durch weitere Kontrollmaßnahmen kann diese Einrastzeit überbrückt werden und der entstandene Fehler, welcher durch die Phasenverschiebung entstanden ist, aufgehoben werden.

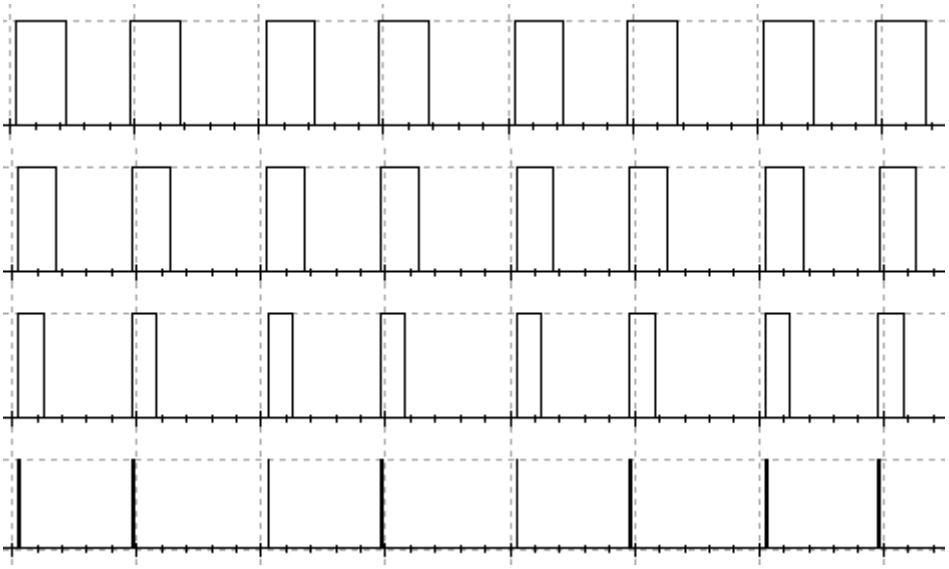


Abbildung 2-8:Fehlersignal des Phasendetektors

In Abb. 2-8 ist das Fehlersignal während der Einrastzeit zusehen. Dieses Signal ist das Ausgangssignal des Phasendetektors, welches an den Oszillator weitergegeben wird. Dieser schwingt in die neu entstandene Phasenlage ein und minimiert somit den Fehler bis es beseitigt wurde.

# 3 Verfahren der digitalen Frequenzanalyse

## 3.1 Allgemeine Vorgehensweise

Im Gegensatz zur in Kapitel 2 dargestellten Methode, können mit den Methoden der Frequenzanalyse die Frequenzanteile von periodischen Signalen bestimmt werden. Liegt ein Signal mit Ober- und Unterwellen vor besteht es nicht nur aus einer Frequenz, sondern aus mehreren Signalen mit verschiedenen Frequenzen. Um diese Frequenzanteile filtern zu können muss das vorliegende Signal zunächst vom Zeitbereich in den Frequenzbereich transformiert werden. Diese Transformation ist eine mathematische Methode und erfolgt über die sogenannte Fourier-Transformation. Da das Signal für die Verarbeitung mittels eines Mikrocontrollers, welcher die Fourier-Transformation ausführt, in digitaler Form vorliegen muss, muss das Signal entsprechend vorher umgewandelt werden. Diese Umwandlung erfolgt mit einem Analog/Digital-Wandler. Das analoge Signal wird abgetastet und als Datensatz abgespeichert, d.h. es wird von einer stetigen Funktion ohne Unterbrechung in eine Abfolge von Zahlenwerte umgewandelt.

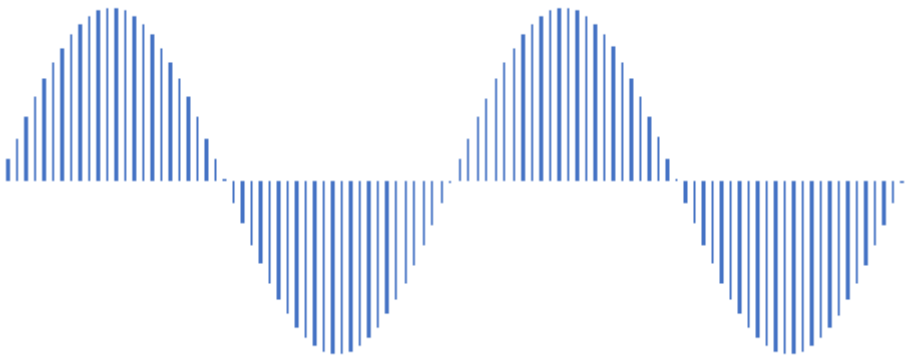


Abbildung 3-1: Abgetasteter Sinus

Abb. 3-1 verdeutlicht das Prinzip der Abtastung grafisch, anhand einer Sinus-Funktion. Damit eine erfolgreiche Abtastung vollzogen werden kann, ist es also zunächst notwendige das periodische Signal während des Messvorgangs bei-

zubehalten. Da Mikrocontroller jedoch keine negativen Werte Verarbeiten können muss nach einer Spannungstransformation dem Signal eine Offsetspannung[7] hinzugefügt werden. Somit ist es bereit für die Abtastung durch einen A/D-Wandler, welcher auch auf dem Mikrocontroller integriert sein kann. Bei diesem Vorgang ist die Abtastrate, welche sich aus den Samples[8] pro Sekunde berechnen lässt, ausschlaggebend. Ist nur das Intervall zwischen zwei Abtastwerten bekannt, kann die Abtastfrequenz berechnet werden [ITW17]:

$$Abtastfrequenz = \frac{1}{Abtastintervall}$$

Dabei ist die Einhaltung des Abtasttheorems[9] zu beachten, wonach die Abtastfrequenz mindestens das Doppelte der Signalfrequenz sein muss, d.h. das Signal muss mindestens zweimal pro Periode abgetastet werden.

Liegt das Signal in abgetasteter Form vor, kann nach Abspeicherung der Abtastwerte die Fourier-Transformation durchgeführt werden. Da diese Methode eine Integraltransformation darstellt, muss es für die diskrete Verarbeitung als Summe umgeschrieben werden. Daraus ergibt sich die Diskrete Fourier-Transformation (DTF) [HSB17]:

$$X[f] = \sum_{n=0}^{N-1} x[n] e^{-i\frac{2\pi}{N}fn}$$

Aus der Formel wird ersichtlich, dass eine Funktion *x[n]*, während *n* die Abtastastwerte darstellen, vom Zeitbereich in den Frequenzbereich transformiert wird. Das Ergebnis ist *X[f]* eine Funktion anhängig von der Frequenz. In der

---

[7] Die Offsetspannung überlagert der Wechselspannung einen Gleichanteil und verschiebt das Signal somit vertikal

[8] Als Sample werden die Abtastwerte bezeichnet, gekennzeichnet durch eine vertikale Linie (s. Abb. 3-1)

[9] In der Fachliteratur auch Nyquist-Shannon-Abtasttheorem genannt

Praxis wird die DFT als Fast Fourier Transform (FFT) umgesetzt und gilt als Standardverfahren der digitalen Signalverarbeitung. Weitere Algorithmen der Signalverarbeitung, wie der Fit Algorithmus[10], kommen auch in Frage und werden in den folgenden Kapiteln, mit dem Ziel der Netzfrequenzmessung, vorgestellt.

## 3.2 Analoge Vorschaltung

Für die Frequenzanalyse muss die Qualität des Signals beibehalten werden. Aus diesem Grund ist es notwendig die Schaltung nach Kapitel 2.2 zu modifizieren. Nach der Spannungstransformation der Netzspannung, kann ein Bandpassfilter benutzt werden, da dadurch eine Eingrenzung der Grundschwingungsfrequenz, die es zu messen gilt, vorgenommen wird. In diesen Fall würde eine Tiefpassfilterung nicht ausreichen (vgl. Kap. 2.2), da es im Gegensatz zum Bandpassfilter zu ungenau wäre. Um eine Anpassung der Spannung vorzunehmen empfiehlt es sich des Weiteren einen Spannungsteiler dazu zu schalten. Weiterhin ist die galvanische Trennung der beiden Schaltungshälften einzuhalten. Dies erfolgt z.B. mit einem optoelektronischen Element wie es in der vorherigen Schaltung vorgenommen wurde. Nach der Trennung kann das Signal skaliert und verschoben werden, weil der Mikrocontroller im analogen Eingang nur Werte zwischen 0V – 5V einlesen kann. Die Verschiebung findet über einen Operationsverstärker statt, welcher dem Eingangssignal eine Offsetspannung addiert. Ein weiterer Operationsverstärker muss eingesetzt werden um das Signal zu skalieren, damit kann schließlich erreicht werden, dass das Signal sich im Zulassungsbereich befindet. Für die letzte Verarbeitung des Signals, auch Signal Processing genannt, muss das Signal letztlich mit dem AD Wandler welcher auf dem Mikrocontroller integriert ist abgetastet werden. Somit kann der ausgesuchte Algorithmus zu Einsatz kommen.

---

[10] Der Fit Algorithmus basiert auf die Methode der Kleinsten Quadrate (s. Kap. 3.4)

### 3.2.1 Bandpassfilter

Ein Tiefpassfilter dient zur Filterung von hochfrequenten Signalen und weist eine Verstärkungsabnahme für Frequenzen oberhalb der Grenzfrequenz auf. Bei einem Hochpassfilter[11] findet diese Verstärkungsabnahme für Frequenzen unterhalb der Grenzfrequenz statt. Diese Eigenschaften können benutzt werden um einen Bandpassfilter zu designen. Dieser filtert Frequenzen außerhalb eines bestimmten Bereichs, während die untere Grenze $f_L$ durch die Grenzfrequenz des Hochpassfilters und die obere Grenze $f_H$ durch die Grenzfrequenz des Tiefpassfilters vorgegeben wird. In der Schaltung in Abb. 3-2 spiegelt sich das in der Hintereinanderschaltung beider Filter wider [ELT17]. Dabei findet die Dimensionierung der Schaltungen einzeln statt. Da beide Filter aus einen RC Glied bestehen kann die Formel für die Grenzfrequenz aus Kap. 2.3 benutzt werden, dabei muss der Bereich, auch Bandwidth genannt, $BW = f_H - f_L$ und die Widerstände durch Auswahl bekannt sein. So können die Kapazitäten der Kondensatoren durch Umformen der Formel bestimmt werden und die Schaltung entsprechend implementiert werden.

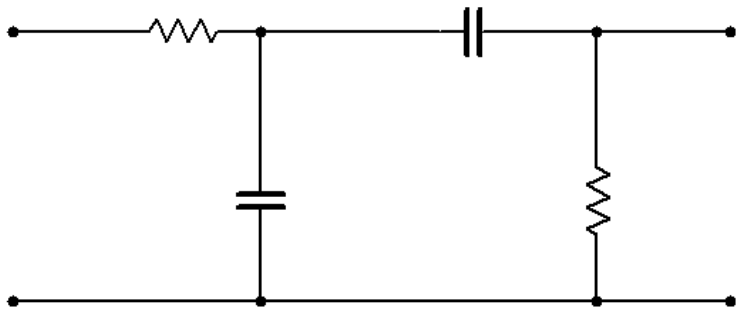


Abbildung 3-2:Bandpassfilter

[11] Der Hochpassfilter 1. Ordnung besteht aus der Reheinschaltung aus einen Kondensator und einem Widerstand wie bei dem Tiefpassfilter 1. Ordnung, während die Ausgangsspannung am Widerstand anliegt.

### 3.2.2 Nicht-invertierender Verstärker

Liegt das Signal gefiltert vor, kann weiterhin über einen einfachen Spannungsteiler das Signal weiter angepasster werden. Anschließend um die beiden Schaltungshälften zu trennen, kann ein Optokoppler oder alternativ ein Transformator zur galvanischen Trennung zwischen Verstärker und Filterung hinzugeschaltet werden. Die Verstärkerschaltung dient zur Verstärkung eines Signals kann aber auch im Zusammenhang mit einer Addition benutzt werden. Somit verstärkt der Operationsverstärker das Signal, welcher aus dem Mittelwert überlagerter Spannungen besteht, um den Faktor *V*. Da diese Schaltung den Zweck der Addition von einer Offsetspannung hat, muss der Verstärkungsfaktor auf die Anzahl der überlagerten Signale eingestellt werden. Diese Signale sind zum einen das Eingangssignal, eine Wechselspannung, und eine Gleichspannungsquelle mit dem Offsetbetrag um die das Eingangssignal verschoben werden soll. Die Überlagerung findet in einem Bezugspunkt, auch als virtuelle Masse bezeichnet, statt, dabei werden die Signale über die Widerstände $R_3$ und $R_4$, die am Knotenpunkt verbunden sind, eingespeist (s. Abb. 3-3).

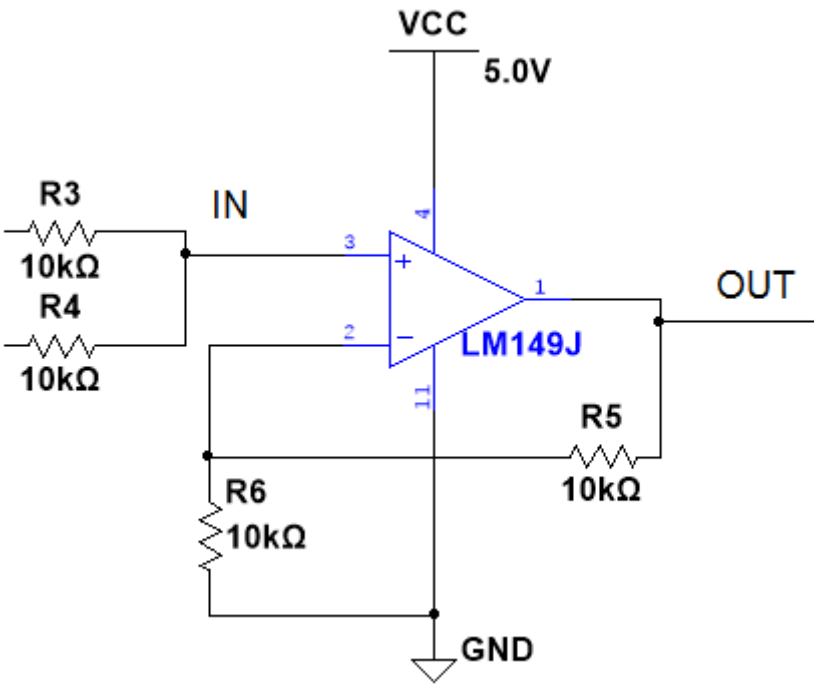



Abbildung 3-3:Verstärkerschaltung

Aus der Schaltung in Abb. 3-3 wird ersichtlich, dass die virtuelle Masse am nicht-invertierenden Eingang des Verstärkers anliegt. Über die Rückkopplung[12] am invertierenden Eingang kann die Verstärkung des Signals eingestellt werden, welcher auf 2 eingestellt werden muss, da am Eingang zwei Signale anliegen. Diese Einstellung erfolgt über die zwei Widerstände $R_5$ und $R_6$ [GAU17]:

$$V = 1 + \frac{R_5}{R_6}$$

Werden also die beiden Widerstände aus der obigen Formel gleich gewählt, kann eine Verstärkung um 2 erreicht werden. Dabei kann die Ausgangsspannung, welcher in diesem Fall eine Wechselspannung mit Gleichanteil ist, über diese Verstärkung berechnet werden:

$$U_a = U_e * V$$

### 3.2.3 A/D-Wandler

Der A/D-Wandler wandelt ein analoges Signal in ein digitales Signal um, d.h. dass ein dezimaler Messwert in einem bestimmten Zeitpunkt durch den A/D-Wandler als Binärzahl ausgedrückt wird. Um diese Umwandlung zu vollziehen muss zum einen der Spanungsbereich der Eingangsspannung bekannt sein und zum anderen die binäre Auflösung des A/D-Wandlers. Ist der A/D-Wandler auf dem Mikrocontroller integriert liegt dessen Eingang am anlogen Eingang des Mikrocontrollers und der Spannungsbereich ist durch den Mikrocontroller gegeben, welcher zu meist zwischen 0-5V liegt. Aus diesem Grund wurde das Signal vorher durch die Addition einer Offsetspannung (s. Kap.3.2.2) in den Spannungsbereich eingestellt, während die Qualität des Signals beibehalten wurde [BBD17]. Ist zudem die binäre Auflösung des A/D-Wandlers bekannt

[12] Es gibt zwei Arten der Rückkopplung, zum einen die Mittkopplung (positive Feedback) und die Gegenkopplung (negative Feedback), die in der Schaltung durch den Spannungsteiler zum Einsatz kommt.

kann daraus das Genauigkeitsintervall, auch analoge Auflösung genannt, berechnet werden. Dazu wird der Spannungsbereich in Schritten entsprechend der binären Auflösung, welcher durch die Bit-Länge *n* gegeben ist, unterteilt:

$$binäre\ Auflösung = 2^n$$

Somit kann daraus die analoge Auflösung berechnet werden, welcher für die korrekte Ermittlung von Messwerten von Bedeutung ist [WHE10]:

$$ana.Aufösung = \frac{Spanungsobergrenze - Spannungsuntergrenze}{binäre\ Auflösung}$$

Liegt der Spannungsbereich also zwischen 0-5V und die Bit-Länge der darzustellenden analogen Messwerte ist 10 Bit, ergibt dies nach obiger Formal eine analoge Auflösung oder Schrittweite von 4.88 mV. So kann der digitale Binärwert welcher am Ausgang des A/D-Wandlers ausgegeben wird in den entsprechenden Spannungswert durch Multiplikation umgewandelt werden.

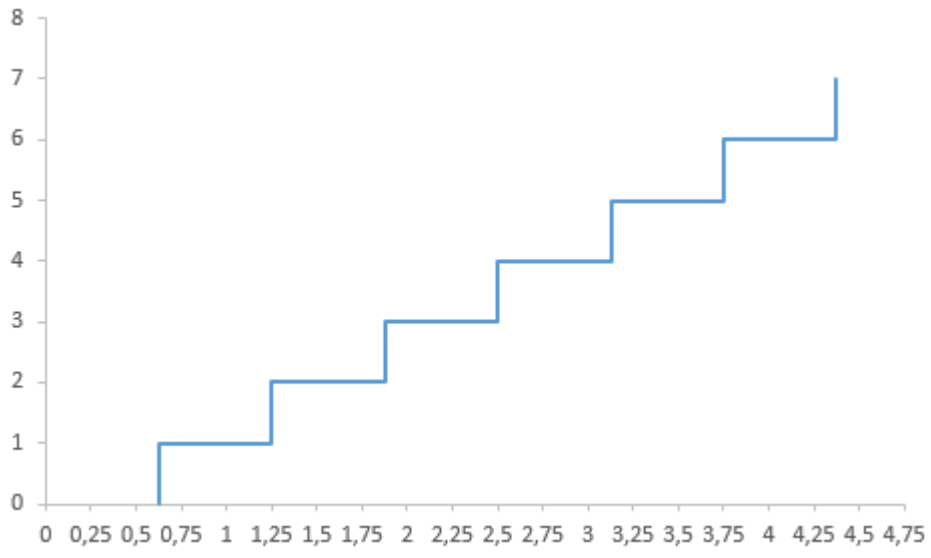



Abbildung 3-4:3-Bit A/D-Wandlung

In Abb. 3-4 wird das Prinzip eines Analog-Digital Wandlers anhand eines 3-Bit A/D-Wandlers verdeutlicht. Dabei stellt die x-Achse die möglichen Eingangswerte, bei einem Spannungsbereich von 0-5V, in Volt dar, währende auf der y-Achse der entsprechende Binärwert angezeigt wird. Liegt am Eingang z.B. ein

Spannungswert von 2.25 V an, kann der Binärwert aus dem Diagramm entnommen werden, dieser entspricht 3 oder in binärer Schreibweise $(011)_2$. Daraus kann dann im nächsten Schritt der Ausgabewert berechnet werden, indem die analoge Auflösung berechnet wird. Nach obiger Formel beträgt die analoge Auflösung 625 mV und somit kann der Ausgabewert berechnet werden:

$$625mV * 3 = 1.875\ V$$

Somit ist es auch möglich durch einfache Multiplikation die Binärwerte am Ausgang des A/D-Wandlers wieder in Spannungswerte zu konvertieren und im weiteren Verlauf einen Algorithmus zu implementieren welcher diese Werte abspeichert und weiterverarbeitet. Wobei eine höhere binäre Auflösung für eine genauere Abtastung entscheidend ist, da dadurch die Abstände in x-Richtung kleiner werden und die Eingangswerte von den Ausgangwerten sich nur noch minimal unterscheiden.

## 3.3 Fast Fourier Transformation (FFT)

Die Fast Fourier Transformation lässt sich aus der DTF herleiten (s. Kap. 3.1) und als Anwendung für die Frequenzanalyse programmiertechnisch implementieren. Dabei wird ein Vektor $x_n$, welcher einen Datensatz an Messwerten enthält, entsprechend der DFT in einen Vektor $F_k$ mit Frequenzinformationen umgewandelt:

$$F_k = \sum_{n=0}^{N-1} x_n e^{-i\frac{2\pi}{N}kn},$$

mit *n: 0, 1, 2, …… N-1* und *k: 0, 1, 2, …… N-1*, während *N* die Anzahl der Samples angibt. Aus dieser mathematischen Formel der DFT ist zu erkennen, dass für die Berechnung von $F_k$ die N-fachen Operationen der Sampleanzahl benötigt werden, da für jedes *k* eine Summe aus den Samples $x_n$ multipliziert mit der Exponentialzahl gebildet wird. Beträgt die Sampleanzahl z.B. *N=4* würden $4^2=16$ Operationen benötigt werden, da die Summenformel für jedes *k* 4-mal durchlaufen wird. Aus diesem Grund muss diese Formel vereinfacht wer-

den, um programmiertechnisch einen effizienten Algorithmus mit weniger Operationen zu implementieren. Eine mögliche Vereinfachung nach Cooley und Tukey[13] ist die Aufteilung der Samples durch Bit Revision[14] in einen geraden und ungeraden Anteil, daraus ergeben sich zwei Summenformeln [PMA06]:

$$F_k = \sum_{m=0}^{\frac{N}{2}-1} x_{2m} e^{-i\frac{2\pi}{N}k(2m)} + \sum_{m=0}^{\frac{N}{2}-1} x_{2m+1} e^{-i\frac{2\pi}{N}k(2m+1)}$$

Über eine einfache Umformung kann der komplexe Koeffizient aus der zweiten Summe rausgezogen werden:

$$F_k = \sum_{m=0}^{\frac{N}{2}-1} x_{2m} e^{-i\frac{2\pi}{\frac{N}{2}}km} + e^{-i\frac{2\pi}{\frac{N}{2}}k} \sum_{m=0}^{\frac{N}{2}-1} x_{2m+1} e^{-i\frac{2\pi}{\frac{N}{2}}km}$$

Aus dieser Umformung wird ersichtlich, dass sich die Berechnungen nur noch durch den komplexen Koeffizienten unterscheiden. Ist N=4 würde die obere Aufteilung die Berechnung der Summenformel für jeweils zwei Samples bedeuten:

$$F_0^{even} = x_0 + x_2$$

$$F_1^{even} = x_0 - x_2$$

$$F_0^{odd} = x_1 + x_3$$

$$F_1^{odd} = x_1 - x_3$$

---

[13] Die Vereinfachung basiert auf der Annahme, dass die Sampleanzahl als Potenz zur Basis 2 dargestellt werden kann also $N=2^m$. Für N=1024 entspricht es bei einem 10 Bit A/D Wandler $2^{10}$.

[14] Durch Bitreversal wird die Reihenfolge der Bits geändert somit kommt es zu einer Aufteilung in gerade und ungerade Zahlen. Z.B. wird 011=3 in 110=6 umgewandelt.

Somit kann über die Kombination der Ergebnisse der zwei DTFs das Endresultat berechnet werden, d.h. die Berechnung der FFT erfolgt in zwei Schritten, was aus der zweier Potenz $N=2^2$ hervorgeht.

$$F_0 = F_0^{even} + C^0 F_0^{odd}$$

$$F_1 = F_1^{even} + C^1 F_1^{odd}$$

$$F_2 = F_0^{even} + C^3 F_0^{odd}$$

$$F_3 = F_1^{even} + C^4 F_1^{odd}$$

, während $C^k$ den komplexen Koeffizienten darstellt. Bei dieser Kombination fällt auf, dass für $F_0$ alle Samples aufsummiert werden da $C^0=1$ wird und für $k>N/2-1$ aufgrund der Periodizität[15] der komplexen Zahlen die Ergebnisse $F_2$ und $F_3$ aus den Kombinationen der vorher berechneten Werten bestehen (s. Abb. 3-5).

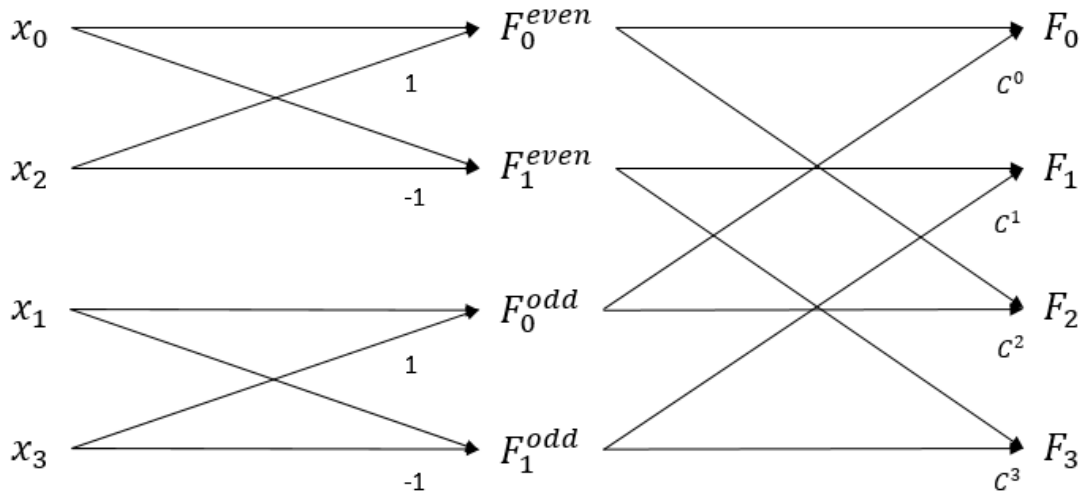



Abbildung 3-5:FFT Schmetterlingsgraph

[15] Die Periodizität kann über die Eulersche Formel nachgewiesen werden:

$e^{-i\frac{2\pi}{N}k2m} = \cos\left(\frac{2\pi}{N}k2m\right) + isin(\frac{2\pi}{N}k2m) = \cos\left(\frac{2\pi}{N}(\frac{N}{2}+k)2m\right) + isin(\frac{2\pi}{N}(\frac{N}{2}+k)2m) = \cos\left(2m\pi + \frac{2\pi}{N}k2m\right) + isin(2m\pi + \frac{2\pi}{N}k2m)$

In Abb. 2-5 wird das Prinzip der FFT anhand der einfachen Aufteilung verdeutlicht, dabei ist das Ziel die Samples in zweier Paare zu unterteilen. Bei einer Sampleanzahl N=8 müssen deshalb zwei Aufteilungen gemacht werden wobei der komplexe Koeffizient für den zweiten Schritt $C^{2k}$ und für den dritten Schritt $C^{k}$ ist[16]. Durch diesen Vorgang kann der Arbeitsaufwand also minimiert werden indem eine Aufteilung entsprechend der Zweierpotenz vorgenommen wird. Somit kann die Änderung der Ordnung für die Berechnung der DFT von $N^2$ auf $N \log_2 (N)$, aufgrund der N-fachen Berechnung je Schritt, erreicht werden (s. Tab.3-1) [PMA06].

| Samples N | $N^2$ | $N \log_2 N$ |
|---|---|---|
| 1000 | $10^6$ | $10^4$ |
| $10^6$ | $10^{12}$ | $20\times10^6$ |
| $10^9$ | $10^{18}$ | $30\times10^9$ |

Tabelle 3-1: Effizienz der FFT

Auch weitere Vereinfachungen hinsichtlich der programmiertechnischen Implementierung kommen in Frage wie z.B. die Berechnung über die Matrixschreibweise und zur erfolgreichen in-place Berechnung. Aufgrund der industriell hohen Anwendung der FFT z.B. bei Kompression für Bild-, Audio- oder Videoformate werden Bibliotheken für verschiedene Programmiersprachen zur Verfügung gestellt und weisen hohe Effizienz bei der Berechnung der FFT nicht nur für den vorgestellten Fall der Radix-2 Berechnung auf, sondern auch für N ohne entsprechender Basis. Dabei wird die Ordnung wie bei der Radix-2 Berechnung entsprechend runtergesetzt (s. Tab. 3-1).

---

[16] Durch erneutes Aufteilen der Summenformeln in jeweils einem geraden und ungeraden Anteil kann das Prinzip der Aufteilung für $N=8$ und allgemein für $N=2^m$ verwendet werden, dabei stellt $m-1$ die benötigten Schritte für die Aufteilung dar.

## 3.4 Sinus Fit Algorithmus

Der Sinus Fit Algorithmus basiert auf der Methode der kleinsten Quadrate, dabei ist das Ziel einem gegebenen Datensatz $x_n$ die möglichst beste Sinus Kurve anzupassen, wobei entsprechend dem Additionstheorem[17] jede Sinus Funktion als Addition von Sinus und Kosinus dargestellt werden kann:

$$\varepsilon = \sum_{n=0}^{N} (x_n - A_i \cos(\omega_i t_n) - B_i \sin(\omega_i t_n) - C_i)^2$$

Aus der Formel wird ersichtlich, dass der Fehler $\varepsilon$ über die Summe der Quadrate gebildet wird, während die vier Parameter $A_i$, $B_i$, $C_i$ und $\omega_i$ gesucht sind. Über die partiellen Ableitungen kann das Minimum des Summenfehlers und dementsprechend vier Formeln für die vier Parameter, mit $\omega_i = \omega_{i-1} + \Delta\omega_{i-1}$[18], berechnet werden. Dazu wird zunächst eine Schätzung der Frequenz, z.B. mit der DFT oder über eine Nullstellendetektion, vorgenommen. Ist eine erste Frequenz mit $\omega_0$ gegeben können so zunächst die Anfangswerte für $A_0$, $B_0$ und $C_0$ berechnet werden, indem die partielle Ableitung zunächst für drei Parameter gemacht wird, d.h. das Minimum wird bei gegebener Frequenz gesucht. In Matrixschreibweise lässt sich die Berechnung der Anfangswerte $y_0$ folgendermaßen ausdrücken [WAC94]:

$$y_0 = (D_0^T D_0)^{-1} (D_0^T x)$$

, wobei $D_0$ die Koeffizienten aus der partiellen Ableitung für drei Variablen enthält.

$$D_0 = \begin{pmatrix} \cos(\omega_0 t_1) & \sin(\omega_0 t_1) & 1 \\ \cos(\omega_0 t_2) & \sin(\omega_0 t_2) & 1 \\ \vdots & \vdots & \vdots \end{pmatrix}$$

---

[17] $\sin(\omega t + \theta) = \sin(\omega t)\cos(\theta) + \cos(\omega t)\sin(\theta)$

[18] Die Frequenz $\omega$ kann nicht direkt berechnet werden, es wird rekursiv aus $\Delta\omega$ berechnet

Für die weitere Berechnung für *i>0* wird die partielle Ableitung[19] für vier Variablen benutz. Deshalb muss die obige Formel und die Matrix angepasst werden:

$$y_i = (D_i^T D_i)^{-1}(D_i^T x)$$

Während $D_i$ aus der partiellen Ableitung für vier Variablen hervorgeht:

$$D_i = \begin{pmatrix} \cos(\omega_i t_1) & \sin(\omega_i t_1) & 1 & -A_{i-1} t_1 \sin(\omega_i t_1) + B_{i-1} \cos(\omega_i t_1) \\ \cos(\omega_i t_2) & \sin(\omega_i t_2) & 1 & -A_{i-1} t_2 \sin(\omega_i t_2) + B_{i-1} \cos(\omega_i t_2) \\ \vdots & \vdots & \vdots & \vdots \end{pmatrix}$$

Somit enthält *y* zunächst die Werte $A_0$, $B_0$ und $C_0$. Für die weitere Berechnung jedoch wird *y* um den Wert $\Delta\omega$ erweitert, weshalb die Berechnung von $\omega$ rekursiv über den alten $\omega$ Wert ausgeführt werden muss (s. Abb.3-6).

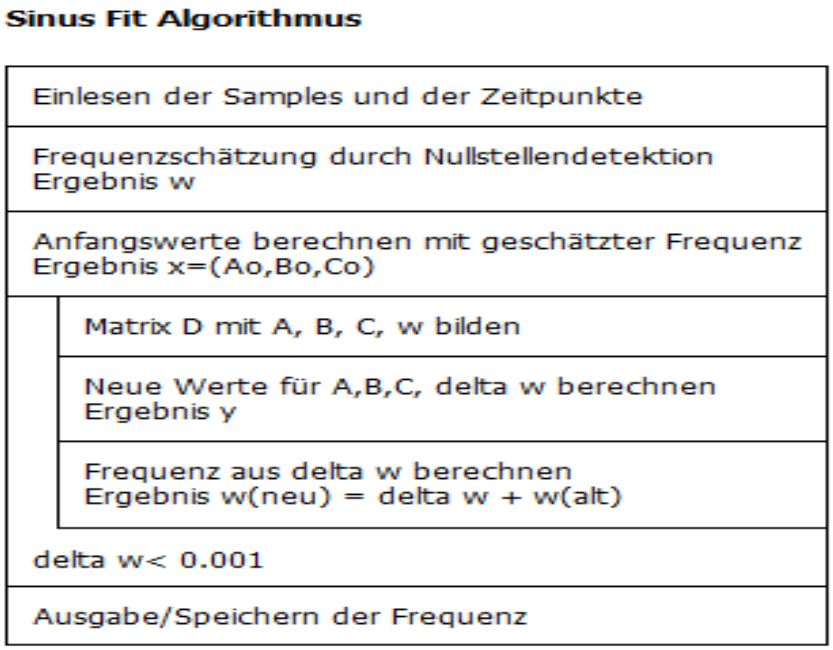



Abbildung 3-6:Struktogramm für den Sinus Fit Algorithmus

[19] Ist eine Funktion abhängig von mehreren Variablen, kann die partielle Ableitung nach einer Variablen berechnet werden, dabei werden die anderen Variablen als konstanten betrachtet

## 3.5 Autokorrelation

Auch die Korrelation des Signals mit sich selber kann herangezogen werden um die Periodizität eines Rauschsignals besser darstellen zu können. Dabei wird das Signal zeitversetzt mich sich selber gefaltet. Die entstehende Funktion ist dabei anhängig von der zeitlichen Verschiebung, die die beiden Korrelationsfunktionen aufweisen. Ist die zeitliche Verschiebung sehr groß und es gibt keine Überlappung der Funktionen kann es auch zu keiner Korrelation kommen. Kommt es aber zu einer Überlappung beider Funktionen entsteht eine Korrelation. Diese Korrelation ist maximal bei einer Verschiebung t=0, da das ganze Signal entsprechend folgender Formel gefaltet wird [AAU18]:

$$R[k] = \sum_{m=0}^{N} x[m] \cdot x[m+k]$$

Somit kann die Periodizität eines Signals mit Rauschen einfacher mit der Autokorrelationsfunktion abgelesen werden. Die Abb. 3-7 stellt ein Sinussignal mit seiner entsprechenden Autokorrelationsfunktion dar:

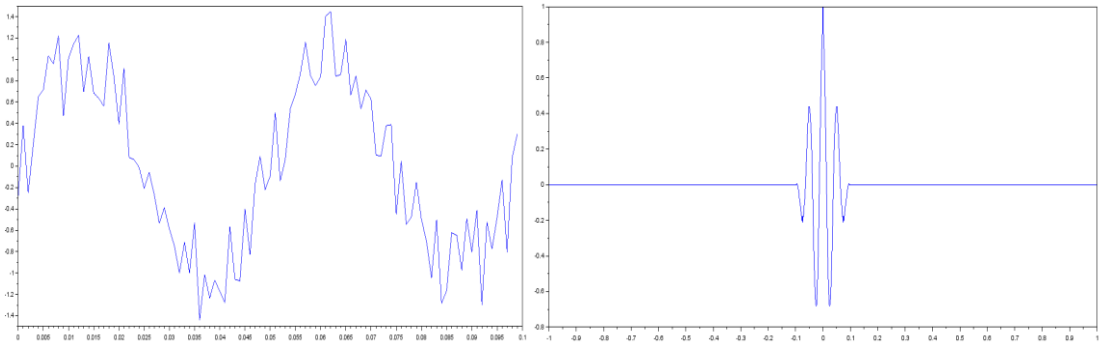


Abbildung 3-7:Autokorrelation eines Sinus mit Rauschen

Während das eigentliche Signal ein Rauschen aufweist und somit schwer durch Nullstellendetektion oder Detektion der Maxima die Periode berechnet werden kann, weist die Autokorrelation kein Rauschen auf, hat aber die selbe Periodizität des eigentlichen Signals. Somit kann anhand der Autokorrelation die Frequenz des Signals berechnet werde, indem zunächst die obige Formel implementiert wird. Dazu stehen Funktionen z.B. in Scilab zu Verfügung.

# 4 Simulation ausgewählter Verfahren

Bevor die in Kap. 2 und 3 vorgestellten Konzepte erfolgreich mit einem Mikrocontroller implementiert werden können ist es notwendig verschiedene Simulationen durchzuführen. Dabei werden zum einen die vorgestellten Vorschaltungen simuliert und zum anderen werden die vorgestellten Algorithmen mit Hilfe von mathematisch numerischer Software eingehend getestet. Somit wird mit der Simulation eine Grundlage für die folgende Implementierung gelegt. Dabei können die Verfahren in die drei grundlegenden Schritte Signaltransformation, Signalumwandlung und Signalverarbeitung unterteilt werden. In Abb. 4-1 wird der Ablauf entsprechend der drei Schritte der Signalvorbereitung dargestellt:

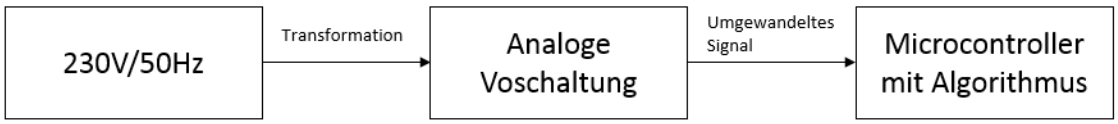



Abbildung 4-1: Vorbereitung der Signalverarbeitung

Der Unterschied zwischen Vorschaltung 1 und 2 liegt in der unterschiedlichen Umwandlung der Signale. Bei *Schaltung 1* (s. Kap. 2.2) wird das Signal in ein Rechtecksignal umgewandelt, welcher direkt digital weiterverarbeitet werden kann, d.h. das Signal kann an den digitalen Eingang des Mikrocontrollers gelegt werden. Bei der *Schaltung 2* (s. Kap. 3.2) wird das Signal verschoben und skaliert, damit es durch den auf den Mikrocontroller integrierten A/D-Wandler am analogen Eingang erfolgreich abgetastet werden kann. Bei der Signalverarbeitung kommen drei Algorithmen zum Einsatz. Entsprechend der *Schaltung 1* wird ein *Zählalgorithmus* programmiert (s. Kap. 2.3) und für *Schaltung 2* können *FFT* (s. Kap. 3.3) und *Sinus Fit Algorithmus* (s. Kap. 3.4) in Betracht gezogen werden. Die *FFT* ist ein Algorithmus zur Transformation von Zeitsignalen in den Frequenzbereich, weshalb es noch keine vollständige Aussagekraft über die gesuchte Frequenz der Grundschwingung gibt. Aus diesem Grund muss nach eingehender Untersuchung eine Methode für die Filterung der Grundfrequenz gefunden werden, da das Ziel die Netzfrequenzmessung darstellt.

## 4.1 Dimensionierung der Vorschaltungen

### 4.1.1 Schaltung 1

Die erste analoge Vorschaltung, hier als Schaltung 1 (s. Anhang 1) bezeichnet, besteht aus zwei Schaltung, dem Tiefpassfilter und dem Schmitt-Trigger, die entsprechend dimensioniert werden müssen. Dazu wurden in Kap. 2.2 Formeln angegeben aus denen die Werte für die Bauteile berechnet werden können.

Für den Tiefpassfilter muss die Grenzfrequenz und ein Widerstand vorgegeben werden. Wird die Grenzfrequenz $f_g=50Hz$ und $R_1=10k\Omega$ gesetzt kann daraus $C$, die Kapazität des Kondensators, berechnet werden:

$$C_1 = \frac{1}{2\,\pi\,R_1\,f_g} = \frac{1}{2\pi \cdot 10k\Omega \cdot 50Hz} \approx 318nF$$

Um passend zu diesem Wert einen Bauteil auszuwählen muss der entsprechende Wert aus der E-Reihe[20] entnommen werden. Somit kann der nächst größte Wert mit *330nF* ausgewählt werden.

Für die Dimensionierung des Schmitt-Triggers müssen zum einen Hysterese, $V_{cc}$ und einen Widerstandswert bekannt sein (s. Kap. 2.2.3). Werden diese Werte durch $V_{cc}=5V$, $U_L=2V$, $U_H=3V$ und $R_6=10k\Omega$ vorgegeben, können die weiteren Widerstände berechnet werden dazu werden die Formeln aus dem Kapitel 2.2.3 benutzt. Bei der Vorgabe dieser Werte muss darauf geachtet werden, dass der Zulassungsbereich für den Mikrocontroller eingehalten wird:

$$U_{ref} = \frac{U_L V_{cc}}{V_{cc} - U_H + U_L} = \frac{2V \cdot 5V}{5V - 3V + 2V} = 2{,}5V$$

$$R_5 = R_6 \frac{U_{ref}}{V_{cc} - U_{ref}} = 10k\Omega \frac{2{,}5V}{5V - 2{,}5V} = 10k\Omega$$

[20] Die E-Reihe stellt eine genormte Reihe für Wert von elektrischen Bauelementen dar.

$$R_7 = R_8 \frac{U_H - V_{cc} - U_L}{U_L - U_H} \; mit \; R_8 = \frac{R_5 R_6}{R_5 + R_6}$$

$$R_7 = \frac{10k\Omega \cdot 10k\Omega}{10k\Omega + 10k\Omega} \cdot \frac{3V - 5V - 2V}{2V - 3V} = 20k\Omega$$

Mit diesen Werten kann also ein Rechtecksignal mit dem Höchstwert von 5V entsprechend des $V_{cc}$ Werts erzeugt werden, was für die Dimensionierung der Schaltung 1 genügt (s. Tab. 4-1).

| Bauteil | Wert |
|---|---|
| R1 | 10kΩ |
| C1 | 330nF |
| R2 | 1kΩ |
| R3 | 1kΩ |
| R4 | 10kΩ |
| R5 | 10kΩ |
| R6 | 10kΩ |
| R7 | 20kΩ |

Tabelle 4-1: Dimensionierung von Schaltung 1

In Tab. 4-1 wurden die Bauteile dessen Werte aus der obigen Berechnung hervorgehen blau markiert. Die übrigen Werte wurden frei gewählt. Außerdem wurden für den Aufbau der Schaltung 1, welcher für die Simulation in NI Multisim genutzt wird, weitere Bauteile eingesetzt. Aus der Abbildung in Anhang 1 wird ersichtlich, dass V1 die runtertransformierte Spannung darstellt. Außerdem wurde ein Einweggleichrichter vom Modell 1BH62 und eine Zener-Diode, mit einer Sperrspannung von ca. 5V, vom Modell 1N750A genutzt. Der verwendete

Optokoppler ist ein DC-Typ Optokoppler mit der Bezeichnung VO615A-3 und als Komparator wurde ein LT1720CDD verwendet.

### 4.1.2 Schaltung 2

Zur Dimensionierung der zweiten analogen Vorschaltung, hier als Schaltung 2 (s. Anhang 2) bezeichnet, müssen der Bandpassfilter und der Verstärker dimensioniert werden. Dazu werden die angegebenen Formeln aus Kap. 3.2 benutzt.

Der Bandpassfilter besteht aus einen Tiefpass- und einen Hochpassfilter, d.h. aus zwei separaten RC Gliedern deren Berechnung somit separat erfolgt. Dazu kann die Formel für einen einfaches RC Glied (s. Kap. 4.1.1) herangezogen werden. Durch Vorgabe gibt der Hochpassfilter die untere Grenze mit $f_L$=5Hz und der der Tiefpassfilter die obere Grenze mit $f_H$=500Hz an. Außerdem können die beiden Widerstände mit $R_1$=$R_2$=10kΩ vorgeben werden. Somit können die Kapazitäten der Kondensatoren berechnet werden:

$$C_1 = \frac{1}{2\,\pi\,R_1\,f_H} = \frac{1}{2\pi \cdot 10k\Omega \cdot 500Hz} \approx 31{,}8nF$$

$$C_2 = \frac{1}{2\,\pi\,R_2\,f_L} = \frac{1}{2\pi \cdot 10k\Omega \cdot 5Hz} \approx 3{,}18\mu F$$

Die entsprechenden Werte können aus der E-Reihe entnommen werden. Für $C_1$ ist der nächste größte Wert aus der E-Reihe *32,4nF* während $C_2$ mit *3,24µF* dimensioniert werden kann.

Bei der Verstärkung des Signals wird zunächst eine Addition vollzogen, dabei werden zwei Signale mit einander addiert. Dies für zur Mittelwertbildung im Additionspunkt, weshalb der Verstärker eine Verstärkung um *V=2* ausweisen muss. Dies wird erreich indem die beiden Widerstände Ausgang des Verstärkers den Selben Wert aufweisen, d.h. $R_1$=$R_2$=10kΩ:

$$V = 1 + \frac{R_5}{R_6} = 1 + \frac{10k\Omega}{10k\Omega} = 2$$

Somit wird am Ausgang der Schaltung 2 eine Wechselspannung $U_a=U_e \cdot V$ mit $V=2$ generiert, welcher um einen festen Gleichanteil von 2.5V nach oben verschoben ist, damit es sich Zulassungsbereich des Mikrocontrollers befindet. Die Werte für alle Bauteile sind in Tab. 4-2 zusammengefast:

| Bauteil | Wert |
|---|---|
| R1 | 10kΩ |
| C1 | 32,4nF |
| R2 | 10kΩ |
| C2 | 3,24µF |
| R3 | 10kΩ |
| R4 | 10kΩ |
| R5 | 10kΩ |
| R6 | 10kΩ |
| R7 | 10kΩ |
| R8 | 10kΩ |

Tabelle 4-2:Dimensionierung von Schaltung 2

Die Spannungsquelle V1 in der Abbildung in Anhang 2 stellt eine skalierte und transformierte Spannung dar. Nach erfolgreicher Filterung durch den Bandpassfilter, wurde ein Transformator T1 zur elektrischen Isolation beider Schaltungshälften eingebaut. Damit im weiteren Verlauf eine Addition stattfinden kann müssen die zu addierenden Spannungsquellen parallelgeschaltet werden, dazu werden die Wechselspannung und die Gleichspannung, welche durch einen Spannungsteiler aus der Versorgungsspannung $V_{cc}$ gewonnen wird, am Eingang des Verstärkers miteinander verbunden. Am verbundenen Punkt stellen die Spannungsquellen zunächst einen Mittelwert dar, weshalb der Verstärker

zum Einsatz kommt. Der Spannungsteiler für die Eingangsgleichspannung wird über $R_7$ und $R_8$ gebildet, während $R_5$ und $R_6$ zur Einstellung des Verstärkungsfaktors dienen. Über die Widerstände $R_3$ und $R_4$ werden die Spannungsquellen verbunden.

## 4.2 Simulation der Vorschaltungen

Nachdem die Vorschaltungen dimensioniert wurden, können entsprechend der Schaltungen in Anhang 1 und 2 Simulationen gestartet werden um zu prüfen ob die gewünschten Ergebnisse erzielt werden. Dabei werden die verschiedenen Schaltungsteile getestet. NI Multisim stellt dazu eine Umgebung zur Verfügung mit verschiedenen Tools um Messung durchzuzuführen. So können mit einem Oszilloskop Spannungsverläufe dargestellt werden und mit dem Bode-plotter die Eigenschaft von Filtern analysiert werden. Zudem kann auch ein Spektrums-analysator eingesetzt werden um den Frequenzspektrum entsprechend der FFT darzustellen.

### 4.2.1 Schaltung 1 in Multisim

Nach Spannungstransformation (s. Anhang 1) wird eine Tiefpassfilterung vorgenommen welcher mit den eingestellten Werten eine Grenzfrequenz von ca. 50 Hz aufweist. Um dies zu überprüfen kann ein Bode-Plotter eingesetzt werden. Dieser stellt den typischen Verlauf des Betrags des Spannungsverhältnisses $U_a/U_e$ in Dezibel[21] über der Frequenz dar, während die Grenzfrequenz bei -3db zu messen ist, da ab dieser Frequenz eine Abnahme erfolgt. Über die Simulation kann mit dem Bode-plotter diese Grenzfrequenz ermittelt werden, dabei wird der Amplitudengang des Tiefpassfilters geplottet und der Cursor entsprechend der Definition bei ca. -3db eingestellt. In Abb. 4-2 wird die Simulation

[21] Die logarithmische Darstellung des Amplitudengangs vereinfacht das Überlagern komplexere Bode-diagrammen höherer Ordnung

des Bode-diagramms dargestellt. Die rote Linie stellt dabei den Cursor dar, welcher entsprechend den unter dem Diagramm angegeben Werten bei 50.5Hz und -3.22db eingestellt wurde:


Abbildung 4-2:Bodediagramm des Tiefpassfilters

Somit werden größere Frequenzen gefiltert und die Eingangsspannung kann dadurch geglättet werden. Im weiteren Verlauf werden die Halbwellen der Wechselspannung entfernt, damit es mit dem DC-Optokoppler kompatibel ist, und durch eine Zener-Diode eine Spannungslimitierung durchgeführt. Dies führt zu einem rechteckähnlichen Spannungsverlauf (s. Kap. 2.2.2) mit einer Amplitude von ca. 5V. Nach der Signalübertragung durch den Optokoppler wird der Schmitt-Trigger eingesetzt, welcher einen Rechteckt generiert. Wird also $U_H$= 3V von dem Eingangssignal des Schmitt-Triggers überstiegen schaltet der Komparator auf „Low“ mit der Ausgangspannung $U_a$=0V. Fällt das Signal jedoch unter $U_L$=2V erhält man am Ausgang einen „High“ mit der Ausgangsspannung $U_a$=5V (s. Abb. 4-3).

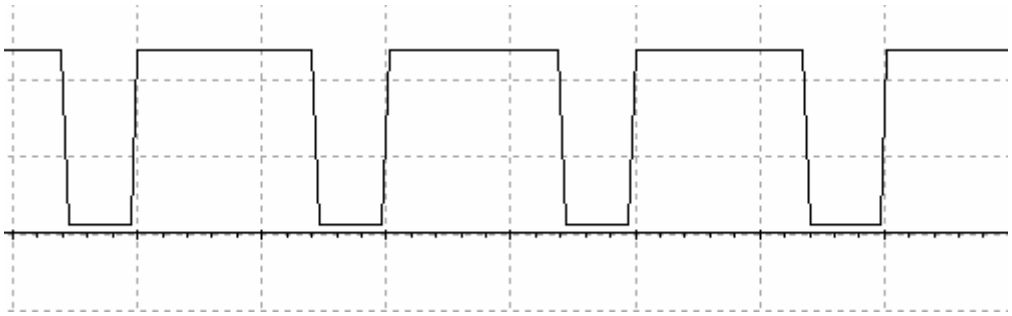


Abbildung 4-3:Ausgangssignal der Schaltung 1

Das Signal in Abb. 4-3 wurde mit einem Oszilloskop simuliert und hat eine Auflösung von 2V/Div. in y-Richtung und 10ms/Div. in x-Richtung. Somit kann über die Analyse der steigenden Flanken mit dem auf einen Mikrocontroller implementierten Programm die Frequenz des Signals ermittelt werden. Graphisch kann die Frequenz aus der Periodendauer berechnet werden. Da eine Periode zwei Divisionen dauert lässt sich die Frequenz aus dem Kehrwert von der abgelesenen Periodendauer berechnen:

$$f_x = \frac{1}{2Dev. \cdot \frac{10ms}{Dev.}} = 50Hz$$

### 4.2.2 Schaltung 2 in Multisim

Um eine Spektrums-analyse durchzuführen darf das Signal nur in y-Richtung skaliert und verschoben werden, dabei werden die für eine Frequenzmessung benötigten Frequenzinformationen der Grundfrequenz gefiltert. In der Schaltung (s. Anhang 2) spiegelt sich die Filterung im Bandpassfilter wieder. Anschließend wird über die Mittelwertbildung dem Eingangssignal eine Offsetspannung addiert. Dadurch entsteht eine Stauchung des Signals was mit dem Verstärker wieder kompensiert werden kann. Für die Analyse des Bandpassfilters kann erneut der Bode-plotter eingesetzt werden, um die eingestellte obere und untere Grenzfrequenz darzustellen (s. Abb. 4-4).

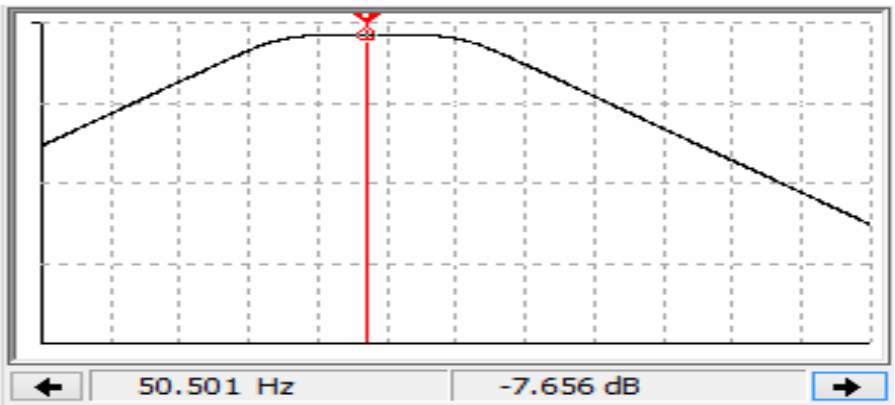



Abbildung 4-4:Bodediagramm des Bandpassfilters

In Abb. 4-4 wird der Amplitudengang des Bandpassfilters dargestellt, dabei wird die gewünschte Frequenz von 50 Hz nur mit ca. -7db übertragen, was durch die Bandbreite zustande kommt. Somit sind die Grenzfrequenzen bei ca. -10db zu messen. Durch Vergrößerung der Bandbreite könnte ein genaueres Ergebnis erzielt werden mit einer Übertragung von ca. 0db. Da dadurch jedoch hohe Frequenzen nicht gefiltert werden könnten müsste der Band nach links verschoben werden, was den Badpassfilter zu einem Tiefpassfilter machen würde. Solange also keine Lösung für einen Bandpassfilter mit relativ geringer Bandbreite gefunden werden kann, muss der oben simulierte Tiefpassfilter bei der Implementierung genutzt werden. Für die Simulation genügt der hier dimensionierte Bandpassfilter. Für die Trennung der beiden Schaltungshälften wurde ein Transformator in Reihe zu dem Filter geschaltet. Dieser hat somit eine 1:1 Übertragung, weshalb an dessen Ausgang die Eingangsspannung gemessen werden kann. Durch die Parallelschaltung des Eingangssignals mit einer Gleichspannung gebildet durch den Spannungsteiler, kann eine Verschiebung des Signals erfolgen, wobei es zu einer Mittelwertbildung also Stauchung um 0.5 kommt (s. Abb. 4-5).

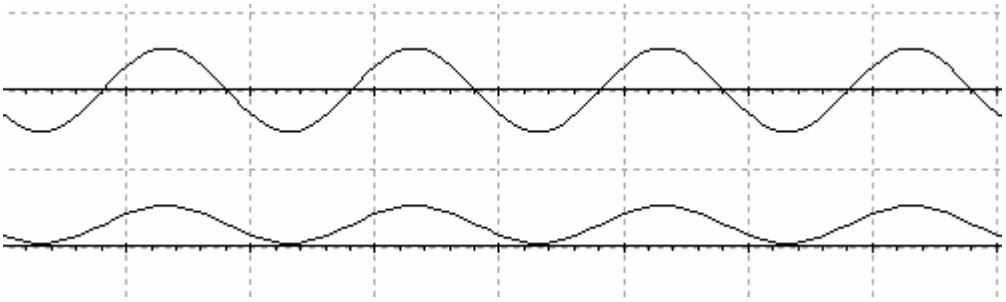


Abbildung 4-5:Mittelwertbildung zweier Signale

Die hier gewählten Einstellungen am Oszillator sind 5V/Div. In y-Richtung und 10ms/Div. In x-Richtung. Rechnerisch betrachtet kann die in der Abb. 4-5 dargestellte Umformung folgendermaßen dargestellt werden:

$$U_{neu} = \frac{\mathrm{A} \cdot \sin(\omega t + \varphi) + B}{2}$$

Während *B=2.5V* die Verschiebung durch die Gleichspannungsquelle darstellt. Um nun die Stauchung zu kompensieren muss der Verstärker zum Einsatz kommen welcher eine Verstärkung von *2* aufweist. Das Ausgangssignal des Verstärkers stellt somit das verschobene Signal ohne Skalierung dar (s. Abb. 4-6).

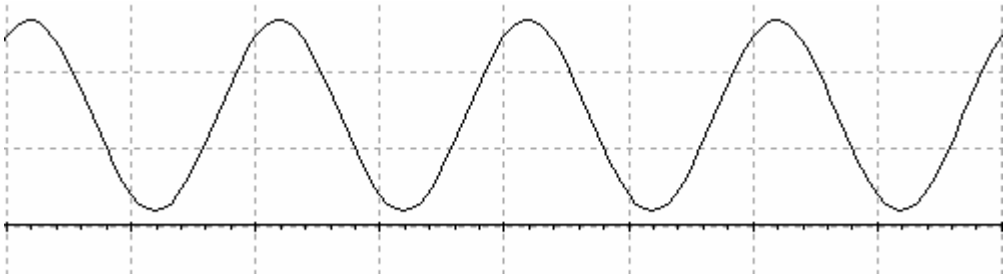


Abbildung 4-6:Ausgangssignal der Schaltung 2

In Abb. 4-6 wird das Ausgangssignal der Schaltung 2 dargestellt mit 2V/Div. in y-Richtung und 10ms/Div. in x-Richtung. Dieses Ausgangssignal kann somit im Folgenden durch einen A/D Wandler abgetastet werden. So können die Algorithmen der Signalverarbeitung zum Einsatz kommen. Diese werden in den nächsten Kapiteln mit einer mathematisch-nummerischen Umgebung getestet.

## 4.3 Testen der Algorithmen

Im Zusammenhang mit Schaltung 1 wird einen Zählalgorithmus implementiert, um die steigenden Flanken des in der Simulation abgebildeten Rechtecksignals über mehrere Perioden zu zählen und mit der in Kap. 2.1 angegeben Formel die resultierende Frequenz zu bestimmen. Aus diesem Grund ist diese Methode nur vom Ausgangssignal der Vorschaltung abhängig und kann nicht weiter programmiertechnisch beeinflusst werden. Im Gegensatz dazu ist die Effektivität des FFT Algorithmus und Sinus Fit Algorithmus, einschließlich von Modifikationen, zu prüfen. Dabei wird die FFT als vorgegebene Funktion implementiert, sodass der Quellecode für weitere Modifikationen herangezogen werden muss bzw. das Ergebnis der FFT eingehend analysiert werden muss um die gewünschten Information zu filtern. Der Sinus Fit Algorithmus hingegen wird als

rekursiver Algorithmus implementiert und gibt direkt die gesuchte Frequenz aus. Beide Algorithmen werden somit im Folgenden in Scilab einer mathematisch nummerischen Software mit Programmierumgebung programmiert und getestet.

### 4.3.1 Sinus Fit Algorithmus in Scilab

Um den Sinus Fit Algorithmus zu implementieren muss zunächst eine initiale Frequenzschätzung gemacht werden. Liegt ein Sinussignal ohne Rauschen also tiefpassgefiltert als *Data* vor kann diese Schätzung über das zählen der Nullstellenübergänge gemacht. Wird also ein Übergang detektiert wird diese als Periodenanfangszeit gespeichert. Somit wird die Periodenendzeit beim dritte Übergang der Nullstelle detektiert und die resultierende Kreisfrequenz kann berechnet werden. In den Abbildung 4-7 und 4-8 wird das Programm zur Frequenzschätzung, welcher als Einheit zu sehen ist, vorgestellt:

```
//Abschätzen von der Frequenz durch Lage der Nullstellen
//Startsample speichern
data1=data(1)
//Zählvariable zählt die Nullstellen-Übergänge
j=0
//Nullstellen-Übergänge zweiter aufeinanderfolgender Samples detektieren
//und Periode ermitteln
for i=2:len(2)
//Übergang von Plus zu Minus
if data1>0 & data(i)<0 then
//Inkrement der Zählvariablen
j=j+1
//Überprüft ersten Übergang (Anfang der Periode)
if j==1 then
//Speichern der Start-Zeit als Mittelwert
startT=(t(i)+t(i-1))/2
end
//Überprüft dritten Übergang (Ende der Periode)
if j==3 then
//Speichern der End-Zeit als Mittelwert
endT=(t(i)+t(i-1))/2
//Nach Erfassen der End-Zeit wird Zählung gestoppt
break
end
```

Abbildung 4-7:Frequenzschätzung Teil 1

```
//Übergang von Minus zu Plus
elseif data1<0 & data(i)>0 then
//Inkrement der Zählvariablen
j=j+1
//Überprüft ersten Übergang (Anfang der Periode)
if j==1 then
//Speichern der Start-Zeit als Mittelwert
startT=(t(i)+t(i-1))/2
end
//Überprüft dritten Übergang (Ende der Periode)
if j==3 then
//Speichern der End-Zeit als Mittelwert
endT=(t(i)+t(i-1))/2
//Nach Erfassen der End-Zeit wird Zählung gestoppt
break
end
end
//Übergang für nächste Samples durch Variablenwechsel vorbereiten
data1=data(i)
//Ende der Schleife
end
//Kreisfrequenz aus ermittelten Werten berechnen
w=(2*pi)/abs(endT-startT)
```

Abbildung 4-8:Frequenzschätzung Teil 2

Doch bevor der Schätzalgorithmus, welcher nur die Initialisierung des Sinus Fit Algorithmus darstellt und somit nur eine mögliche Vorgehensweise darstellt, muss für die Simulation ein Sinussignal generiert werden und als obige Variable *Data* gespeichert werden. Über die Eingabe des Folgenden Befehls in Scilab kann dies erreicht werden:

$$data = \sin(2 * 3.141 * f * t)$$

Wobei f die Frequenz des Signals darstellt und t das Zeitintervall in dem das Signal dargestellt werden soll. Dabei kann aus den aufeinanderfolgenden Werten in t die Abtastfrequenzbestimmt werden. Wählt man f=50Hz und t= [0;99] in Millisekunden können damit fünf Perioden abgebildet werden:

$$Periodeanzahl = f * Anzahl(t) * Abtastzeit = 50Hz * 100 * ms = 5$$

Somit kann über das Folgende Befehl ein Sinussignal mit einer Frequenz von 50Hz mit 5 Perioden dargestellt werden (s. Abb. 4-9):

$$data = \sin(2 * 3.141 * 50Hz * \frac{(0:99)}{1000})$$

Wobei Data einen Vektor mit 100 Werten, die mit einer Abtastfrequenz von 1kHz berechnet wurde, darstellt.

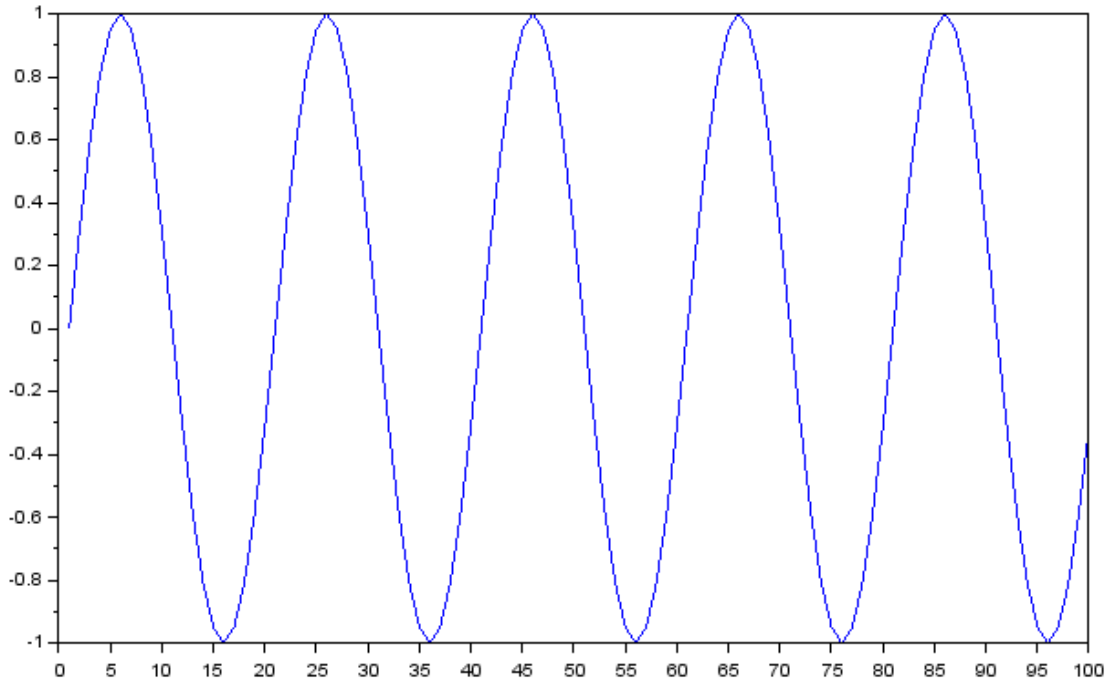



Abbildung 4-9: Sinussignal in Scilab

Das in Abb. 4-9 dargestellte Signal wurde mit der *plot(data)* Funktion geplottet und verdeutlicht die obige Berechnung graphisch. Geht man davon aus, dass die Frequenz unbekannt ist und das Signal nur in Form von Samples vorliegt kann somit die oben vorgestellte Frequenzschätzung gemacht werden in dem die Nullstellenübergänge detektiert werden. Wurde eine erste Schätzung gemacht kann zur genaueren Frequenzbestimmung der Sinus Fit Algorithmus benutzt werden (s. Anhang 3). Somit ist man auf die initiale Schätzung im generellen Fall angewiesen. Wird der Sinus Fit Algorithmus jedoch für die Netzfrequenzmessung herangezogen kann diese Schätzung vermieden werden da die

Netzfrequenz sich immer im Rahmen von 50Hz aufhält. Dies stellt sich als Vorteil heraus, da die Frequenzschätzung anfällig für Fehler ist, wenn die Anzahl von Data relativ gering gegenüber der Frequenz ist. Somit ist man nicht auf eine initiale Schätzung angewiesen, wenn man die Frequenz als Anfangswert initialisiert. Die folgende Tabelle 4-3 stellt die Ergebnisse von verschiedenen Tests zunächst mit dem obigen Schätzalgorithmus dar:

| Eingabe data=sin(2*3.141*f*t) | Ausgabe der Frequenz von sinus-Fit(data,t) |
|---|---|
| data=sin(2*3.141*10*(0:99)/1000) | Error |
| data=sin(2*3.141*20*(0:99)/1000) | 20.000001 |
| data=sin(2*3.141*45*(0:99)/1000) | 45.000023 |
| data=sin(2*3.141*50*(0:99)/1000) | 50.000002 |
| data=sin(2*3.141*1000*(0:99)/1000) | Error |

Tabelle 4-3:Sinus Fit mit Frequenzschätzung

Aus der Tabelle 4-3 erkennt man, dass der Schätzalgorithmus für eine relativ hohe und niedrige Anzahl an Perioden funktioniert. Das Problem stellt sich nur ein, wenn das Signal nur über eine Periode gemessen wurde oder wenn die Perioden Anzahl zu hoch ist. Draus kann gefolgert werden, dass für den Algorithmus bei einer Frequenz von 50 Hz somit eine Abtastfrequenz von 1kHz vorliegen muss und die Anzahl der Messungen 100 betragen muss. Die Abtastrate kann mit einem Oszilloskop gemessen werden welcher sich in der Loop Routine des Mikrocontrollers einstellt. Geht man jedoch von einer initialen Frequenz ohne die obige Schätzung aus könnte auch über einer Periode also mit geringere Abtastrate oder weniger Messwerte gemessen werden. Dies muss bei der Implementierung geprüft werden. Weitere Aussage über die Genauigkeit des Algorithmus gibt das Quadratische Mittel des Fehlers und die Graphen der Fit

Funktion an. Dies kann auf die obigen Eingaben angewendet werden. Dabei lässt sich die das Quadratische Mittel des Fehlers folgendermaßen berechnen:

$$\varepsilon_{rms} = \sqrt{\frac{1}{M}\sum_{n=1}^{M} r_n^2}$$

Dabei stellt M die Anzahl der Messwerte dar und r die Differenz von Messwerten und der Fit Funktion, die wiederum eine Menge von diskreten Werten darstellt [WAC94]. In folgender Tabelle 4-4 wurde der Fehler nach obiger Formel für die obigen Eingaben in Tabelle 4-3 berechnet:

| Eingabe data=sin(2*3.141*f*t) | Ausgabe des mittleren Fehlerquadrats |
|---|---|
| data=sin(2*3.141*50*(0:99)/1000) | 0.0000020 |
| data=sin(2*3.141*45*(0:99)/1000) | 0.0000058 |
| data=sin(2*3.141*20*(0:99)/1000) | 0.0000003 |

Tabelle 4-4:Mittlere Fehlerquadrat für Messwerte ohne Rauschen

Für ideale Messwerte funktioniert der Sinus Fit Algorithmus also ohne starke Abweichungen. Nur die initiale Frequenzschätzung stellt für Funktionen außerhalb eines bestimmten Frequenzbereiches was in Tab. 4-3 dargestellt wurde. Doch diese Fehler können programmiertechnisch bei der Implementierung angepasst werden. In Tab. 4-5 wird der Sinus Fit für reale Messungen simuliert:

| Eingabe mit r=rand(0:99)-0.5 | Frequenz | Fehler |
|---|---|---|
| data=sin(2*3.141*50*(0:99)/1000)+r | 49.844273 | 0.2961893 |
| data=sin(2*3.141*45*(0:99)/1000)+r | 44.688263 | 0.2937556 |
| data=sin(2*3.141*20*(0:99)/1000)+r | 19.455587 | 0.2886665 |

Tabelle 4-5:Sinus Fit für Funktionen mit Rauschen

Aufgrund des hohen Rauschens bei den Nulldurchgängen konnten für die in Tab. 4-5 dargestellten Berechnungen keine Frequenzschätzung basierend auf der Nulldurchgangsdetektion gemacht werden und mussten somit vorgegeben werden. Für die Implementierung muss die Stärke des Rauschens an den Nulldurchgängen untersucht werden. In Anhang 4 sind die Fitgraphen zu finden.

### 4.3.2 Fast Fourier Transformation (FFT) in Scilab

Die FFT ist eine Funktion die standardmäßig in Scilab integriert ist und somit mit den Samples als Eingabeparameter aufgerufen werden kann. Das Ergebnis ist eine vom Zeitbereich ins Frequenzbereich transformierte diskrete Funktion. Das Ergebnis enthält somit Frequenzinformationen des Eingangssignals die einfacher lesbar sind als des reinen Signals im Zeitbereich. Somit korrespondiert zu jedem Signalgraphen mit einer Zeitachse ein Diagramm mit einer Frequenzachse. Wählt man also als Simulation ein Signal mit einer Frequenz von 50Hz wobei das Zeitintervall und die Abtastung groß genug sein muss, kann die FFT berechnet werden. Das Signal im Zeitbereich wird in Abb. 4-10 angezeigt:

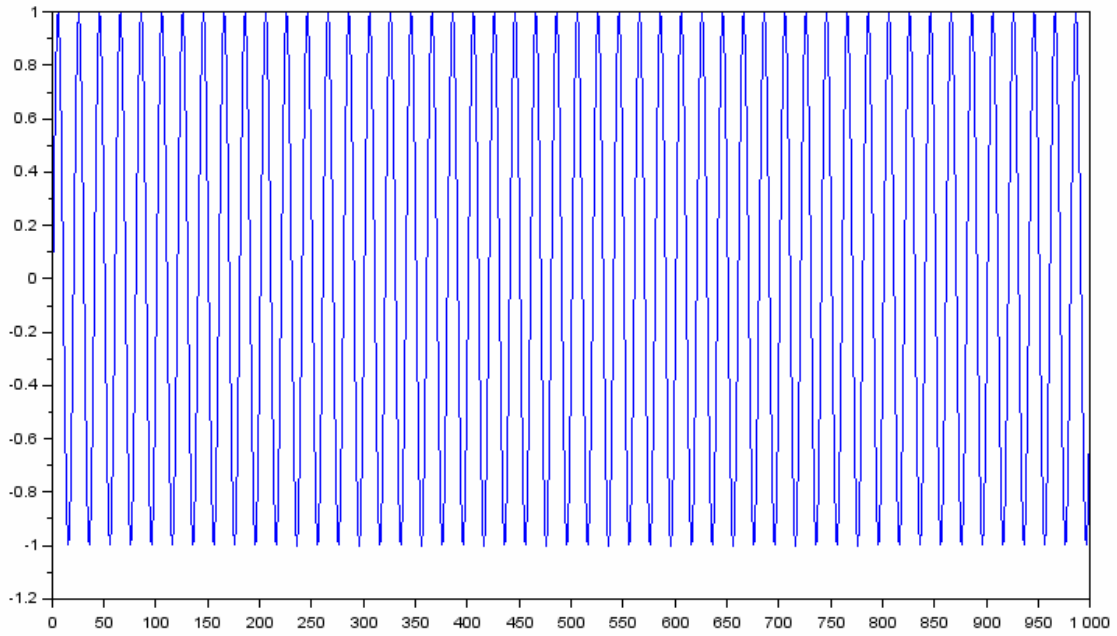



Abbildung 4-10:Sinussignal mit 50Hz und 50 Perioden

Dabei wurde der oben angewendete Befehl *plot(data)* verwendet, während *data=sin(2*pi*50*(0:999)/1000)* ist. Will man nun die FFT für dieses gegebene Signal berechnen kann f=*fft(data)* ausgeführt werden, wobei das Resultat der Berechnung einen Vektor mit komplexen Zahlen darstellt, deren Länge entscheidend für das Filtern der Frequenzinformation ist. Somit muss also zunächst die Länge der komplexen über abs(f) berechnet werden. Wird dieser Vektor mit den Längen geplottet erkannt man die Lage der Grundfrequenz (s. Abb. 4-11).

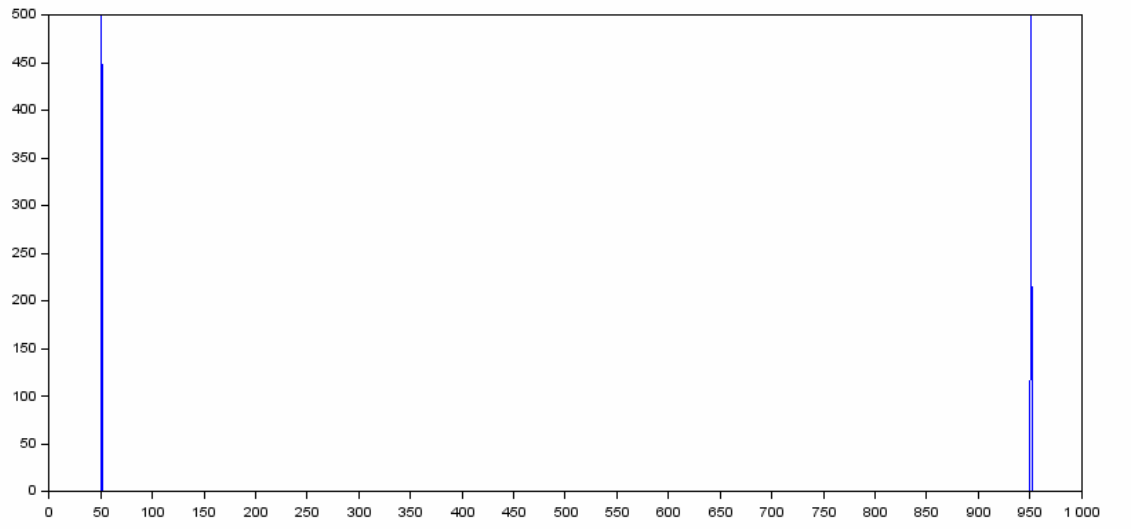



Abbildung 4-11:FFT Diagramm für das Sinussignal mit 50Hz

Um weitere Fehler durch dezimale Frequenzinformationen mit Nachkommastellen zu vermeiden kann eine sogenannte Fensterung ausgeführt werden. Dabei werden die Samples data mit einer Fensterfunktion multipliziert damit die Ecken einen geraden Übergang haben und somit eine Periodizität aufweisen. Trotz dessen kann die Frequenz nicht auf Nachkommastellen genau angezeigt werden, da die Frequenz zunächst nur durch die Lage des Betragsmaximums gegeben ist. Durch weitere Maßnahmen wie z.B. Zeropadding jedoch lässt sich der FFT Algorithmus modifizieren und für eine genaue Frequenzmessung implementieren. Zunächst ein Algorithmus zum Filtern einer ungenauen Frequenzinformation mittels FFT programmiert, welcher auch für eine initiale Frequenzschätzung bei starken Rauschsignalen für den Sinus Fit Algorithmus ge-

nutzt werden kann. In Tabelle 4-6 werden die Eingaben aus Tabelle 4-5 als Grundlage genommen und die Berechnungen mit dem modifizierten Sinus Fit Algorithmus vorgenommen:

| Eingabe mit r=rand(0:999)-0.5 | Frequenz | Fehler |
|---|---|---|
| data=sin(2*3.141*50*(0:999)/1000)+r | 49.997289 | 0.2817492 |
| data=sin(2*3.141*45*(0:999)/1000)+r | 45.003512 | 0.2819162 |
| data=sin(2*3.141*20*(0:999)/1000)+r | 19.998865 | 0.2813946 |

Tabelle 4-6:Sinus Fit mit FFT für Funktionen mit Rauschen

Im Unterschied zu den vorherigen Berechnungen musste hier eine höhere Datenmenge gewählt werden, damit die FFT bessere Ergebnisse erzielen konnte. Dabei wurde die Berechnung in Abb. 4-10 und Abb. 4-11 als Grundlage genommen. In der Tab. 4-6 weist die Zeitachse eine Abtastung von 0.001s auf und die Anzahl der Daten sind 1000, weshalb es hier zu einem genauen Ergebnis kommt. Dies geht vor allem aus der Formel für das Frequenzintervall hervor:

$$\Delta f = \frac{1}{n \cdot \Delta t}$$

Somit kann mit n als Sampleanzahl und Delta t als Abtastzeit der Abstand der diskreten Frequenzachse bestimmt werden [NIC18] und das obige Problem für Eingaben die eine zu niedrige bzw. Abtastung bzw. Sampleanzahl aufweisen behoben werden, indem die Frequenzachse mit Delta f neu skaliert wird. In Tab. 4-7 wurde der FFT Algorithmus nach Anhang 5 mit entsprechenden Delta f Angaben durchgeführt. Dabei wurde Delta f über die zeitliche Abtastung berechnet welcher der programmierten Funktion fftfilter(data,t) als Eingabeparameter übergeben wurde, während Data ein Standard Sinus Funktion darstellt mit gleichbleibender Frequenz von 50 Hz. In der ersten Spalte sind somit die verschiedenen Eingaben der Zeitvariablen zu sehen mit entsprechend Abgeänderter Abtastung, welche durch Division durchgeführt wird:

| Eingabe von T | Frequenz-Bin | Delta f |
|---|---|---|
| T=(0:999)/1000 | 50 | 1 |
| T=(0:999)/500 | 100 | 0.5 |
| T=(0:999)/200 | 250 | 0.2 |

Tabelle 4-7:FFT mit Ausgabe des Frequenz-Bins und Delta f

Durch Multiplikation vom Frequenz-Bin welcher die Frequenzinformation der Grundfrequenz enthält und von Delta f kann somit die ursprüngliche Lage des Betragsmaximums bestimmt werden. Doch auch diese Berechnung kann keine genaue Frequenzbestimmung z.B. auf eine Nachkommastelle genaue liefern. Eine Methode ist die Vergrößerung der Auflösung indem die Anzahl der Samples mit „Zeros“ erhöht wird. Somit kann eine FFT mit dem neuen Signal gemacht werden, welcher als Ergebnis eine genauere Auflösung liefert, was an den kleiner werdenden Werten von Delta f erkannt werden kann. In Tab. 4-8 wurde der FFT Algorithmus mit Erhöhung der Auflösung bei Sinussignalen mit einer Abtastung von 1kHz und der Anzahl von 1000 Samples ausgeführt:

| Genaue Frequenz | Frequenz-Bin | Delta f | Frequenz |
|---|---|---|---|
| 50.2Hz | 452 | 0.111111 | 50.22 |
| 50.4Hz | 454 | 0.111111 | 50.44 |
| 50.7Hz | 456 | 0.111111 | 50.66 |

Tabelle 4-8:FFT mit Erhöhung der Auflösung

Somit kann mit diesem Verfahren eine auf eine Nachkommastelle genaue Bestimmung der Frequenz gemacht werden indem die zweite Nachkommastelle auf- oder abgerundet wird. So kann auch die dritte Lösung auf eine Nachkommastelle genau angegeben werden. Die obigen Messungen basieren auf den Algorithmus in Anhang 5.

### 4.3.3 Autokorrelation in Scilab

Wie aus Kap. 3.5 hervorgeht, kann die Autokorrelation in Zusammenhang mit einer Detektion der Maxima die Frequenz von periodischen Signalen berechnet werden. Dazu wurde ein Algorithmus, welcher über die Maxima-Detektion und deren zeitlich die Frequenz des Signals berechnet, implementiert. Für Simulations- und Testzwecke können die obigen Eingaben für Signale ohne Rauschen und mit Rauschen als Eingabe gewählt werden (s. Tab. 4-9).

| Eingabe | Ergebnis ohne Rauschen in Hz | Ergebnis mit Rauschen in Hz |
|---|---|---|
| data=sin(2*3.141*50*(0:99)/1000) | 50 | 50 |
| data=sin(2*3.141*45*(0:99)/1000) | 45.454545 | 45.454545 |
| data=sin(2*3.141*20*(0:99)/1000) | 20 | 19.607843 |
| data=sin(2*3.141*50.3*(0:99)/1000) | 50 | 50 |

Tabelle 4-9:Autokorrelation bei Signalen mit und ohne Rauschen

Somit ist zu erkennen das die diskrete Autokorrelation nach Anhang 6 schon bei Signalen ohne Rauschen keine genauen Ergebnisse liefert. Dies kann an der zu geringen Anzahl der Messwerte bzw. niedrigen Abtastung liegen und muss deshalb geprüft werden (s Tab. 4-10).

| Eingabe | Ergebnis der Frequenz |
|---|---|
| data=sin(2*3.141*51*(0:9999)/100000) | 51.09862 |
| data=sin(2*3.141*50.5*(0:9999)/100000) | 50.530571 |
| data=sin(2*3.141*51.5*(0:9999)/100000) | 51.706308 |

Tabelle 4-10:Autokorrelation mit höherer Abtastung

Auch hier kommt es zu Ungenauigkeiten trotz genauer Abtastung.

# 5 Implementierung ausgewählter Verfahren

## 5.1 Frequenzzählung

Im Rahmen eines Masterprojekts wurde ein Frequenzzähler welcher sich der Schaltung 1(s. Kap. 4.2.1) bedient entwickelt. Dieser wurde hier nochmal getestet, dabei wurde die Schaltung zur Umformung des Eingangssignals vorgeschaltet und der in Kapitel 2.3 vorgestellte Algorithmus auf den angeschlossenen Mikrocontroller implementiert.

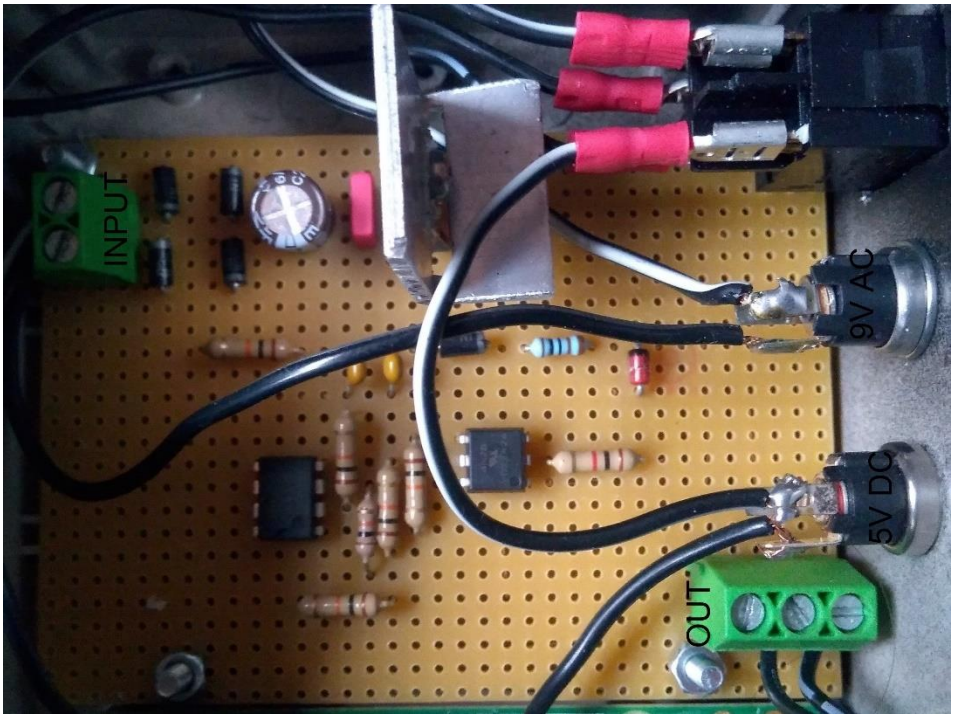



Abbildung 5-1:Vorschaltung für die Rechteckumformung

In Abbildung 5-1 ist die Schaltung für die Rechteckumformung zu sehen, dabei wird eine 9V Wechselspannung zu einem Rechteck mit gleicher Frequenz umgewandelt. Das umgewandelte Signal wird am Output der Schaltung an den digitalen Eingang des Mikrocontrollers angelegt. Da dieser sich im Zulassungsbereich von 0-5V befinden muss, wurde dies vorher mit Hilfe eines Oszilloskops festgestellt. Dabei wurde das Signal welcher sich nach der Umformung am

Output der Schaltung befindet an einen Oszilloskopen angeschlossen. Das Ergebnis dieser Messung ist in der folgenden Abbildung 5-2 zu sehen:

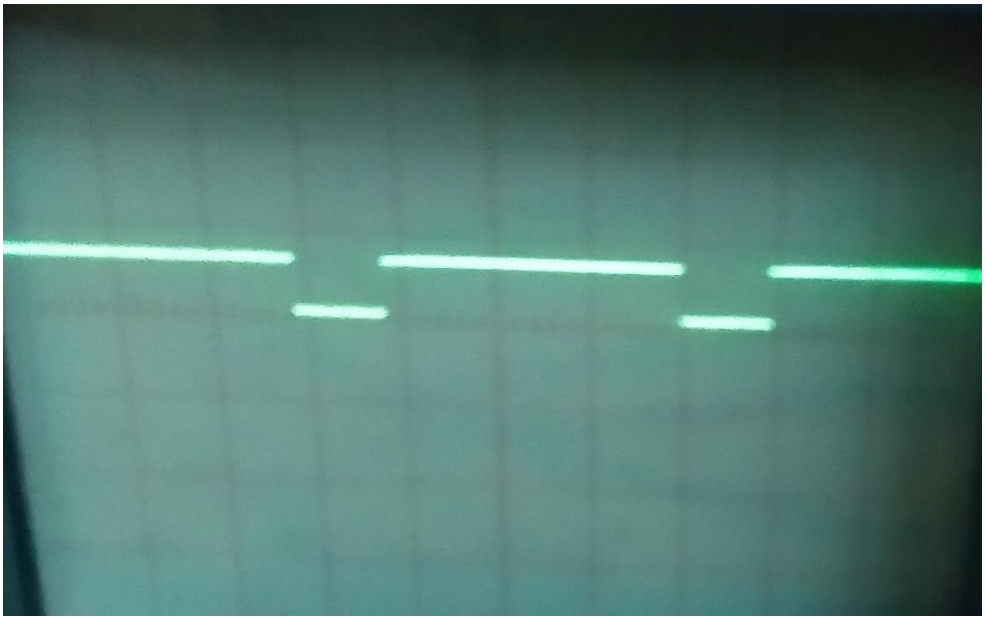


Abbildung 5-2:Ausgangssignal von Schaltung 1

Bei dieser Messung wurde die Zeitbasis auf 5ms pro Division und die Spannungsbasis auf 5V pro Division eingestellt, woraus das in der Abb. 5-2 zu sehende Signal entstanden ist. Dies zeigt nämlich ein Rechtecksignal mit einer Frequenz von ca. 50 Hz. Im nächsten Schritt ist aus diesem Rechtecksignal die entsprechende Frequenzmessung zu machen, welcher nach der in Kap. 2.3 vorgestellten Methode aus der Flankenzählung zu machen ist. Somit wird der vorgestellte Algorithmus demensprechend implementiert und das Ergebnis kann mit dem seriellen Monitor, welcher die Arduino Programmierumgebung zur Verfügung stellt, nachvollzogen werden. Die Programmierung des Arduinos findet mit der C Programmiersprache statt während die Loop Routine für den wiederholten Ablauf des Codes verantwortlich ist. So kann bei ausreichender Flankenzählung, welcher für eine höhere Präzision ausschalgebend ist, die resultierende Frequenz berechnet werden. Das geschriebene Programm kann dann auf den Chip des Mikrocontrollers geladen werden und als Sketch abgespeichert werden (s. Anhang 7). Das Ergebnis der Rechnung kann in Abbildung 5-3 zu Testzwecken nachvollzogen werden.

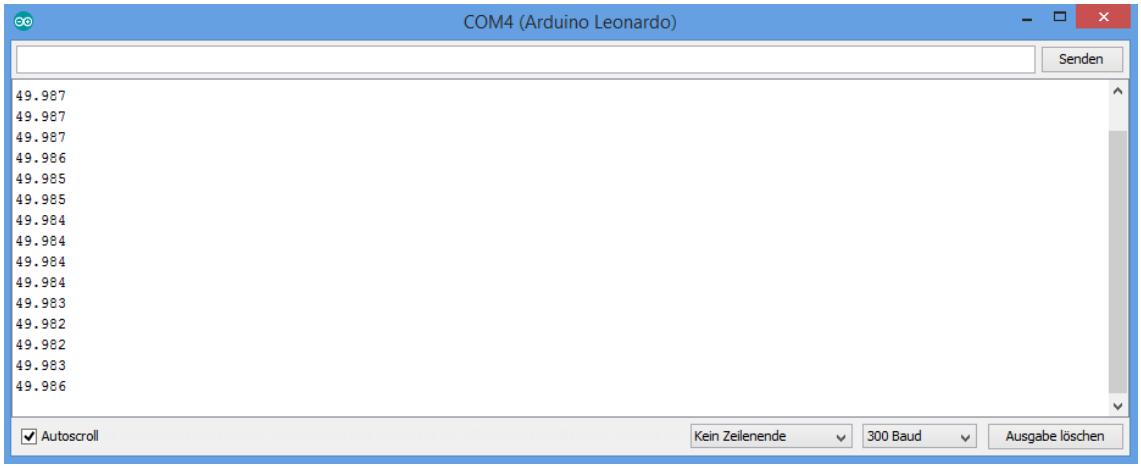



Abbildung 5-3:Ausgabe am seriellen Monitor

Um den Mikrocontroller autonomer zu gestalten empfiehlt es sich als Ausgabe ein Display anzuschließen. Doch da im Rahmen dieser die Verfahren im Vordergrund stehen reicht die dargestellte Ausgabe in Abb. 5-3. Diese generiert die berechneten Werte in einem Intervall von 50 steigenden Flanken. Somit ist diese Methode eine einfache Methode zur Berechnung der Frequenz von Signalen, wobei hier das Augenmerk auf die Ermittlung der Netzfrequenz gelegt wurde.

## 5.2 Messdatenerfassung mit dem Mikrocontroller

Für die Flankenzählung ist die Qualität des Signals nicht entscheidend. Doch für das reale Austesten der in Kap. 4.3 simulierten Algorithmen muss die Qualität des Signals weitgehend beibehalten werden. Aus diesem Grund wurde ein Prototyp für eine Schaltung nach Kap. 4.1.2 entwickelt welcher die Qualität des Signals nicht verändert. Die Aufgabe dieser Schaltung besteht lediglich darin das Signal für den Zulassungsbereich des Mikrocontrollers zu verschieben und zu skalieren. So kann das repräsentative Signal am analogen Eingang des Mikrocontrollers angelegt werden. Dieser analoge Eingang stellt zudem ein Analog-Digital-Wandler dar welcher nach einer bestimmen Abtastrate welche durch die Loop-Routine gegeben ist das Eingangssignal abtastet. Da es sich bei dem verwendeten Model um einen 10 Bit A/D-Wandler handelt muss der entspre-

chende Umwandlungswert nach Kap. 3.2.3 berechnet werden. Die Vorschaltung wurde als Prototyp auf einem Steckboard zusammengeschaltet (s. Abb. 5-4):

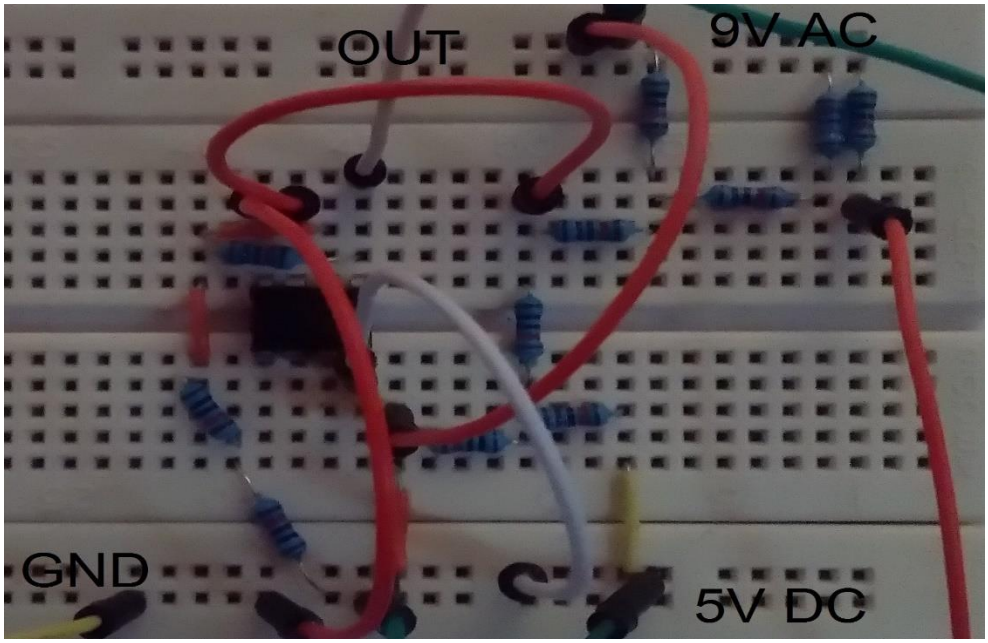



Abbildung 5-4:Vorschaltung für die Messdatenerfassung

Somit kann GND und OUT an die entsprechenden Stellen am Mikrocontroller verbunden werden. Doch bevor die Schaltung angeschlossen wird, kann erneut der Verlauf und der Bereich vorher mit dem Oszilloskop getestet werden (s. Abb. 5-5):

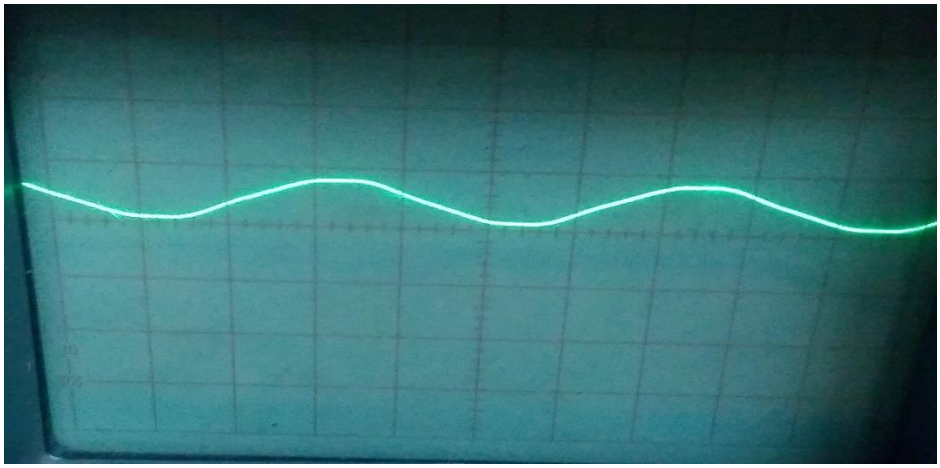


Abbildung 5-5:Ausgangssignal von Schaltung 2

Das Oszilloskop wurde erneut auf 5ms/Div. für die Zeitbasis und 5V/Div. für den Spannungsbereich eingestellt. Somit liegt das Ausgangssignal im Zulassungsbereich des Mikrocontrollers und kann an den analogen Eingang angeschlossen werden. Als Mikrocontroller wird ein Arduino Uno verwendet welcher über die USB Schnittstelle mit dem Computer verbunden werden kann. Somit können auch die Ergebnisse der Messdatenerfassung über den seriellen Port angezeigt werden. Diese können über bestimmte Programme wie z.B. Putty in eine Textdatei geloggt werden und die so erhaltenen Messwerte können entweder über Scilab oder mittels anderer Programmiersprache ausgewertet werden. Doch bevor die Signalverarbeitung stattfinden kann muss ein Skript geschrieben werden welcher die analogen Messwerte einliest und digitalisiert [ELL18]. Das Skript muss auf dem Mikrocontroller laufen (s. Abb. 5-6) und die Algorithmen zur Frequenzermittlung können separat auf einen weiteren Computer ausgeführt werden. So können verschiedene Algorithmen getestet werden.

```
//Variable der eingelesenen Messerte
int sensorValue=0;
//Variable der umgewandelten Messwerte
float voltage=0;


void setup() {
 //Initialisieren des seriellen Monitors
  Serial.begin(9600);
}

void loop() {
  //Analogen Messwert einlesen
  sensorValue = analogRead(A0);
 //Digitalen Messwert in Spannungswert umwandlen
  voltage = sensorValue*(5.0/1023.0);
 //Messwert seriell ausgeben
  Serial.print(voltage);
  Serial.print("\t");

}
```

Abbildung 5-6:Skript zum Einlesen der Messwerte

Die so gewonnenen Messwerte können dann entweder über dem seriellen Monitor der Arduino IDE oder wie oben beschrieben über eine weitere Software wie Putty angezeigt werden. Dabei ist das Abspeichern dieser Messwerte endscheidend, da diese weitere verarbeitet werden müssen. Diese Funktion bietet Putty, worüber eine Kommunikation zum seriellen Port aufgebaut werden kann und die Ausgabe in eine Textdatei eingeloggt werden kann. So kann eine erste Darstellung der Messung gemacht werden in dem die Daten mit der plot(data) Funktion in Scilab als Diagramm geplottet werden (s. Abb. 5-7):

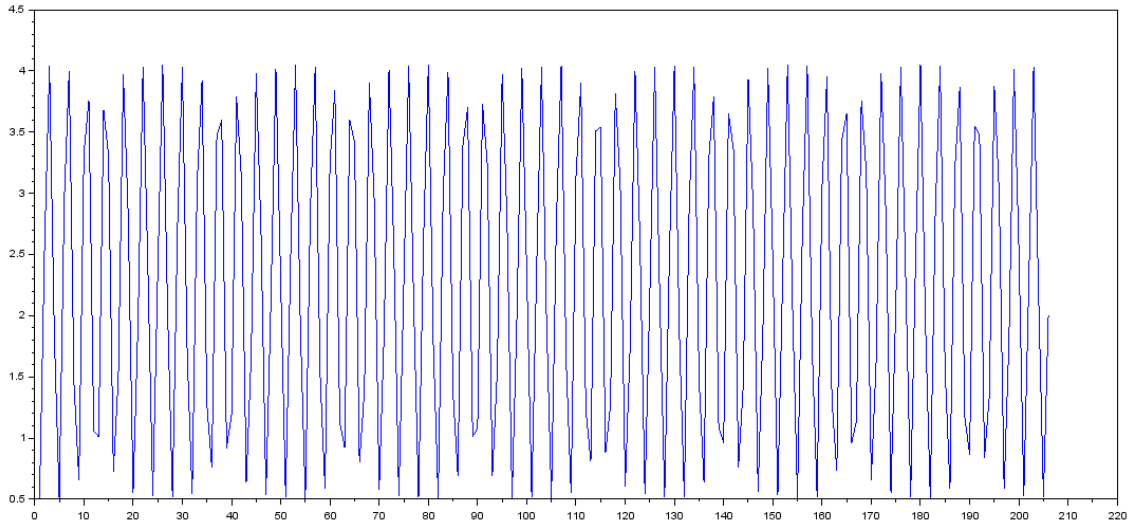


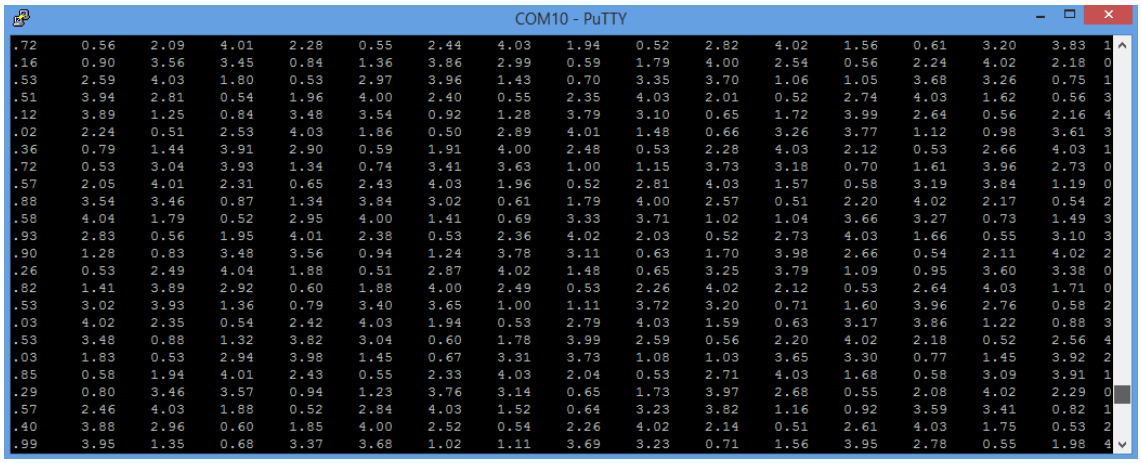

Abbildung 5-7:Dargestellte Messung des Mikrocontrollers

An der y-Achse kann der Spannungsbereich erkannt werden welcher sich ca. zwischen 0.5 bis 4 Volt befindet. Dieser kann an der Schaltung durch Änderung der Widerstände angepasst werden. Die x-Achse muss mit der Abtastfrequenz angepasst werden und stellt hier noch keine Zeitachse dar. Ob diese für die Algorithmen von Bedeutung ist muss noch geprüft werden. An der Darstellung ist eine Regelmäßigkeit zu erkennen welche somit ein periodisches Verhalten erkennen lässt. Somit können an diesen Messdaten die verschiedenen Algorithmen zur Frequenzermittlung ausgetestet werden. Da der Sinus Fit Algorithmus (s. Kap. 4.3.1) eine erste Frequenzabschätzung erfordert kann eine initiale Abschätzung gemacht werden. Auch die FFT kann mit dem plotten der Be-

tragsmaxima wie in Kap. 4.3.2 getestet werden, dabei können die in dem Signal enthaltenen Frequenzanteile angezeigt werden.

## 5.3 Sinus Fit Algorithmus

Da der Sinus Fit Algorithmus eine initiale Schätzung der zugrundeliegenden Daten voraussetzt, welcher in Kap. 4.3.1 auf der Basis von Sinusfunktionen die über der Nullachse oszillieren vorgestellt wurde, muss der Schätzalgorithmus angepasst werden. Die zu analysierenden Daten haben nämlich eine von der Nullachse verschiedene Bezugsachse. Außerdem ist es festzustellen wie groß der Abtastintervall ist um der sinusFit(data,t) Funktion die entsprechenden Zeitwerte zu übergeben. Da es mit einem Timer nicht bestimmbar war musste das Abtastintervall durch Ablesen der Ausgabewerte bestimmt werden (s. Abb. 5-8):




| | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| .72 | 0.56 | 2.09 | 4.01 | 2.28 | 0.55 | 2.44 | 4.03 | 1.94 | 0.52 | 2.82 | 4.02 | 1.56 | 0.61 | 3.20 | 3.83 | 1 |
| .16 | 0.90 | 3.56 | 3.45 | 0.84 | 1.36 | 3.86 | 2.99 | 0.59 | 1.79 | 4.00 | 2.54 | 0.56 | 2.24 | 4.02 | 2.18 | 0 |
| .53 | 2.59 | 4.03 | 1.80 | 0.53 | 2.97 | 3.96 | 1.43 | 0.70 | 3.35 | 3.70 | 1.06 | 1.05 | 3.68 | 3.26 | 0.75 | 1 |
| .51 | 3.94 | 2.81 | 0.54 | 1.96 | 4.00 | 2.40 | 0.55 | 2.35 | 4.03 | 2.01 | 0.52 | 2.74 | 4.03 | 1.62 | 0.56 | 3 |
| .12 | 3.89 | 1.25 | 0.84 | 3.48 | 3.54 | 0.92 | 1.28 | 3.79 | 3.10 | 0.65 | 1.72 | 3.99 | 2.64 | 0.56 | 2.16 | 4 |
| .02 | 2.24 | 0.51 | 2.53 | 4.03 | 1.86 | 0.50 | 2.89 | 4.01 | 1.48 | 0.66 | 3.26 | 3.77 | 1.12 | 0.98 | 3.61 | 3 |
| .36 | 0.79 | 1.44 | 3.91 | 2.90 | 0.59 | 1.91 | 4.00 | 2.48 | 0.53 | 2.28 | 4.03 | 2.12 | 0.53 | 2.66 | 4.03 | 1 |
| .72 | 0.53 | 3.04 | 3.93 | 1.34 | 0.74 | 3.41 | 3.63 | 1.00 | 1.15 | 3.73 | 3.18 | 0.70 | 1.61 | 3.96 | 2.73 | 0 |
| .57 | 2.05 | 4.01 | 2.31 | 0.65 | 2.43 | 4.03 | 1.96 | 0.52 | 2.81 | 4.03 | 1.57 | 0.58 | 3.19 | 3.84 | 1.19 | 0 |
| .88 | 3.54 | 3.46 | 0.87 | 1.34 | 3.84 | 3.02 | 0.61 | 1.79 | 4.00 | 2.57 | 0.51 | 2.20 | 4.02 | 2.17 | 0.54 | 2 |
| .58 | 4.04 | 1.79 | 0.52 | 2.95 | 4.00 | 1.41 | 0.69 | 3.33 | 3.71 | 1.02 | 1.04 | 3.66 | 3.27 | 0.73 | 1.49 | 3 |
| .93 | 2.83 | 0.56 | 1.95 | 4.01 | 2.38 | 0.53 | 2.36 | 4.02 | 2.03 | 0.52 | 2.73 | 4.03 | 1.66 | 0.55 | 3.10 | 3 |
| .90 | 1.28 | 0.83 | 3.48 | 3.56 | 0.94 | 1.24 | 3.78 | 3.11 | 0.63 | 1.70 | 3.98 | 2.66 | 0.54 | 2.11 | 4.02 | 2 |
| .26 | 0.53 | 2.49 | 4.04 | 1.88 | 0.51 | 2.87 | 4.02 | 1.48 | 0.65 | 3.25 | 3.79 | 1.09 | 0.95 | 3.60 | 3.38 | 0 |
| .82 | 1.41 | 3.89 | 2.92 | 0.60 | 1.88 | 4.00 | 2.49 | 0.53 | 2.26 | 4.02 | 2.12 | 0.53 | 2.64 | 4.03 | 1.71 | 0 |
| .53 | 3.02 | 3.93 | 1.36 | 0.79 | 3.40 | 3.65 | 1.00 | 1.11 | 3.72 | 3.20 | 0.71 | 1.60 | 3.96 | 2.76 | 0.58 | 2 |
| .03 | 4.02 | 2.35 | 0.54 | 2.42 | 4.03 | 1.94 | 0.53 | 2.79 | 4.03 | 1.59 | 0.63 | 3.17 | 3.86 | 1.22 | 0.88 | 3 |
| .53 | 3.48 | 0.88 | 1.32 | 3.82 | 3.04 | 0.60 | 1.78 | 3.99 | 2.59 | 0.56 | 2.20 | 4.02 | 2.18 | 0.52 | 2.56 | 4 |
| .03 | 1.83 | 0.53 | 2.94 | 3.98 | 1.45 | 0.67 | 3.31 | 3.73 | 1.08 | 1.03 | 3.65 | 3.30 | 0.77 | 1.45 | 3.92 | 2 |
| .85 | 0.58 | 1.94 | 4.01 | 2.43 | 0.55 | 2.33 | 4.03 | 2.04 | 0.53 | 2.71 | 4.03 | 1.68 | 0.58 | 3.09 | 3.91 | 1 |
| .29 | 0.80 | 3.46 | 3.57 | 0.94 | 1.23 | 3.76 | 3.14 | 0.65 | 1.73 | 3.97 | 2.68 | 0.55 | 2.08 | 4.02 | 2.29 | 0 |
| .57 | 2.46 | 4.03 | 1.88 | 0.52 | 2.84 | 4.03 | 1.52 | 0.64 | 3.23 | 3.82 | 1.16 | 0.92 | 3.59 | 3.41 | 0.82 | 1 |
| .40 | 3.88 | 2.96 | 0.60 | 1.85 | 4.00 | 2.52 | 0.54 | 2.26 | 4.02 | 2.14 | 0.51 | 2.61 | 4.03 | 1.75 | 0.53 | 2 |
| .99 | 3.95 | 1.35 | 0.68 | 3.37 | 3.68 | 1.02 | 1.11 | 3.69 | 3.23 | 0.71 | 1.56 | 3.95 | 2.78 | 0.55 | 1.98 | 4 |

Abbildung 5-8:Horizontale Ausgabe der Messwerte in Putty

Die Periodendauer kann am Spannungsverlauf in Abb. 5-5 abgelesen werden, welcher 20ms aufweist. Dadurch ist der Abstand zwischen zwei Maxima bekannt. So kann das Abtastintervall abgelesen werden. An der Ausgabe in Abb. 5-8 kann horizontal abgelesen werden, dass zwischen zwei Maxima drei Werte liegen, somit weist diese Messung eine Abtastung jede 5ms auf. Diese Ausgabe kann über die entsprechenden Einstellungen in Putty in eine Textdatei umgeleitet werden und somit als Eingabewerte für die simulierten Algorithmen in Scilab übergeben werden. Vergleicht man die Kurve von Abb. 5-7, die die

Messwerte aus der Messung mit dem Mikrocontroller enthält, mit den Kurven aus Anhang 4 erkannt man, dass die Messungen nicht das gleiche Rauschen aufweisen und somit für die Nulldurchgangsdetektion als initiale Frequenzschätzung geeignet sind. Der Algorithmus muss nur noch für eine andere Basis angepasst werden, weil die Messungen einen Gleichanteil, welcher in der Schaltung dazu addiert wurde, aufweisen. Nach der Anpassung kann der Algorithmus auf die in Abb. 5-8 dargestellten Messwerte angewendet werden. Dabei wurde in der folgenden Tabelle 5-1 der Sinus Fit auf die Datenmenge mit verschiedener Anzahl angewendet:

| Anzahl der Messwerte | Frequenz | Fehler |
|---|---|---|
| 47 | 51.990836 | 0.0603612 |
| 70 | 52.943833 | 2.3239439 |
| 206 | 35.489725 | 1.407744 |
| 479 | 50.2701 | 1.280763 |

Tabelle 5-1:Sinus Fit für reale Messwerte

An den Ergebnissen in Tab. 5-1 sieht man das der Sinus Fit Algorithmus für eine niedrige oder hohe Anzahl an Messwerten am effektivsten ist. Weitere Verbesserungen können durch eine höhere Abtastfrequenz bzw. bessere Auflösung der Spannungswerte erreicht werden, was jedoch nicht mit der Effektivität des Algorithmus zusammenhängt. Auch die Simulationen haben gezeigt das der Sinus Fit Algorithmus sehr genau arbeitet und als Algorithmus für die Auswertung von Messreihen in Frage kommt. Die Ergebnisse der Berechnungen in Tab. 5-1 wurden in Anhang 8 als Diagramme dargestellt. Dabei wurde mit der plot() Funktion die Fit Kurve und die Datenmenge in das selbe Diagramm gezeichnet, um die Ähnlichkeit beider Kurven zu verdeutlichen. Wird die Anzahl der zu messenden Zahlen zu groß verkleinert sich der die Fit Kurve wobei die Frequenz nicht stark von der idealen Frequenz abweicht. Da auch hieran erkannt wird das der Sinus Fit Algorithmus sehr effektiv arbeitet muss für die Ab-

tastung der Messwerte eine Lösung gefunden werden, d.h. die Implementierung der Messdatenerfassung muss verbessert werden.

## 5.4 Fast Fourier Transformation

Der Sinus Fit Algorithmus kann bei einer geringen Sampleanzahl die Datenmenge sehr gut fitten, wobei die Frequenz von der eigentlichen Netzfrequenz abweicht, was durch eine geringe Abtastung zustande kommt. Nichtsdestotrotz kann das Ergebnis des Sinus Fit Algorithmus als Referenz genutzt werden um die Effektivität der FFT zu untersuchen. Dabei wird die FFT entsprechend der Simulation in Tab. 4-8 mit Erhöhung der Auflösung implementiert (s. Tab. 5-2).

| Sampleanzahl | Frequenz-Bin | Delta f | Frequenz |
|---|---|---|---|
| 47 | 110 | 0.4728132 | 52.009452 |
| 70 | 164 | 0.3174603 | 52.0634892 |
| 206 | 469 | 0.1107420 | 51.937998 |
| 479 | 540 | 0.0962001 | 51.948054 |

Tabelle 5-2:FFT für reale Messungen

In Anhang 9 sind die entsprechenden Frequenzspektren zu sehen die alle 4 Peaks aufweisen. Der erste Peak enthält dabei die Verschiebung des Signals und muss bei der Frequenzberechnung missachtet werden. Somit enthält der zweite Peak die gesuchte Frequenzinformation. Die beiden weiteren Peaks kommen durch den negativen Anteil der komplexen Zahlen in der DFT vor. An den Frequenzspektren kann vor allem erkannt werden, dass desto größer die Anzahl der Samples ist umso genauer die Frequenz dargestellt werden kann, weil Delta f kleiner wird. Aber ungeachtet der Sampleanzahl können sehr gute Ergebnisse ohne starke Abweichung erzielt werden. Deshalb ist die FFT stabiler als der Sinus Fit, welcher eine höher Abweichung aufweist und kommt in Frage für die Auswertung von Messreihen.

## 5.5 Autokorrelation

Auch wenn die Autokorrelation keine zufriedenstellenden Ergebnisse bei der Simulation liefern konnte, wird der Algorithmus aus Anhang 6 zum Zweck des Vergleichs auch auf die Messungen die mit der Schaltung aus Kap. 5.2 und einem Mikrocontroller gemacht wurden angewendet.

| Sampleanzahl | Ergebnis der Frequenz in Hz |
|---|---|
| 47 | 50 |
| 70 | 50 |
| 206 | 50 |
| 479 | 50 |

Tabelle 5-3:Autokorrelation und Frequenzbestimmung der Messungen

Auch hier wird erkannt, dass die Frequenzbestimmung ungenau ist aber für eine Frequenzschätzung geeignet ist. Als Sampling-Rate wurde auch hier 200Hz gewählt, welche zuvor aus den Messwerten ermittelt wurde. In Anhang 10 sind die Autokorrelationen der obigen Berechnungen zu finden. An den Diagrammen erkennt man, dass die Autokorrelation bei genauere Auflösung einen hohen Stellenwert hat, da ihr Graph pyramidenförmig angeordnet ist und somit für die Detektion der Maxima gut geeignet ist. Bei zu niedriger Abtastung jedoch liefert die Berechnung kein zufriedenstellendes Ergebnis, denn die FFT und der Sinus Fit liefern für diese Messung die Frequenz von 52,9 Hz bedingt durch die niedrige Abtastung und niedrige Auflösung des A/D Wandlers welcher auf dem Mikrocontroller integriert ist. Nimmt man diesen Referenzwert aus den Kap. 5.3 und 5.4 weist die Berechnung über die Autokorrelation einen anderen Wert auf welcher zwar die genaue Frequenz vom Netz darstellt aber für diese Messung auf der Basis des Referenzwertes zu ungenau ist.

# 6 Fazit

## 6.1 Zusammenfassung und Vergleich

In dieser Arbeit wurden zwei unterschiedliche Methoden vorgestellt mit der man die Netzfrequenz bestimmen kann. Während in der ersten Methode, welche im Rahmen eines Masterprojekts implementiert wurde, die Frequenz über eine Flankenzählung bestimmt wird, müssen in der zweiten Methode effektive Algorithmen der Signalverarbeitung implementiert werden. Beide Methoden wurden mit einem Mikrocontroller implementiert. Dazu wurden zwei Schaltungen (s. Anhang 1 und 2) angefertigt die dem Mikrocontroller das Signal mit der gewünschten Frequenzinformation übergeben. Da die erste Methode sich der Flankenzählung bedient, konnte über die elektronische Schaltung ein Rechtecksignal mit steilen Flanken geformt werden, welcher die gleiche Frequenz wie das Ausgangssignal aufweist. So musste nur noch ein Programm auf dem Mikrocontroller geschrieben werden welcher die Zählung und Ausgabe der Frequenz ausführt (s. Anhang 7). Somit ist diese Vorgehensweise eine einfache und direkte Lösung für eine Netzfrequenzmessung. Dabei muss noch weiterhin geprüft werden wie die erhaltenen Messungen abgespeichert werden bzw. Weiterverarbeitet werden sollen. Diese wurde hier nicht in Betracht gezogen. Im Gegensatz zu dieser Methode welche auch herkömmliche Messgeräte anwenden, wurde über die zweite elektronische Schaltung das Signal mit Frequenzinformation sowie auch die Signalform dem Mikrokontroller übergeben. Da es in der digitalen Signalverarbeitung keine analogen Methoden gibt musste das Signal zunächst abgetastet werden und somit ein bestmöglichstes Abbild des Signals erstellt werden. Dabei spielte nicht nur die Abtastung des Signals eine Rolle, sondern auch die Digitalisierung welche mit einem Analog-Digital-Wandler umgesetzt wurde. So konnten die resultierenden Werte erfasst und seriell Ausgeben werden. Die serielle Ausgabe ist entscheidend um die Messwerte weiterverarbeiten zu können, denn im Gegensatz zur ersten Methode wurden Algorithmen getestet, die das Signal tiefergehender analysieren. Mit dem Sinus Fit

Algorithmus (s. Anhang 3) z.B. wurde das Signal bestmöglich rekonstruiert indem die entsprechenden Sinus Fit Funktionen anhand der vorliegenden Messwerte gesucht wurde. Das Ergebnis dieses Algorithmus stellt somit eine Fitfunktion dar welcher die gleiche Frequenz aufweist wie die Ausgangsfunktion. Diese wurde in einer Simulation geprüft, indem der Sinus Fit auf reine Sinus Funktionen angewendet wurde. Dabei konnte aus dem mittleren Fehler die Übereinstimmung von Ausgangsfunktion und Fitfunktion erkennt werden. Weiterhin wurde das Selbe Algorithmus auf Funktionen die einen Rauschen aufweisen angewendet, wobei die Graphen in Anhang 4 zur Darstellung generiert wurden. Auch hier kann mit dem Algorithmus die Charakteristik der Funktion wiederhergestellt werden, wohingegen der Fehler für diese Berechnung größer wird. Außerdem wurden verschiedene Methoden ausprobiert um eine Frequenzschätzung zu machen, welcher für den Sinus Fit Algorithmus Vorrausetzung ist. Eine Nullstellendetektion konnte zwar für Funktionen ohne Rauschen gemacht werden, jedoch bei Funktionen mit Rauschen wurde diese fehlerhaft. Somit konnte über eine FFT eine weitere Frequenzschätzung gemacht werden. Die FFT liefert einen sogenannten Frequenz-Bin der die Signalinformation der Grundfrequenz enthält. Dies ist für eine Frequenzschätzung zwar ausreichend aber für eine genaue Frequenzbestimmung nicht geeignet. Deshalb musste die die Auflösung des Signals erhöht werden damit eine entsprechend genaue FFT für eine Frequenzbestimmung gemacht werden konnte. Die Auflösung wurde einfach durch anhängen von Nullen an das Ende des Ausgangssignals auch Zeropadding genannt erhöht. Das genaue und korrekte Ergebnis konnte dann über die Berechnung von Delta f gemacht werden (s. Anhang 5), welcher den Abstand der Frequenz-Bins im transformierten Bereich darstellt. Somit ist das konnte eine relativ genaue Frequenzmessung mit dieser Methode gemacht werden. Eine weitere Methode der Signalverarbeitung stellte die Autokorrelation dar, welche das Signal mit sich Selber korreliert. Das dabei entstehende Ergebnis wird dabei nicht wie mit der FFT in den Frequenzbereich transformiert, sondern ist eine Funktion mit derselben Periodizität. Doch da es auch keine Fitfunktion darstellt wie es der Sinus Fit Algorithmus liefert, konnte die genaue Fre-

quenz aus dieser letzten getesteten Methode (s. Anhang 6) nicht bestimmt werden. Nachdem alle Algorithmen in mit simulierten Messwerten getestet wurden konnte diese auf die eigentlichen Messungen die das Resultat der Messdatenerfassung aus der zweiten Methode war angewendet werden. Auch hier konnte der Sinus Fit Algorithmus und die FFT überzeugen. Während der Sinus Fit die Messung bei geringen Messwerten gut fitten konnte (s. Anhang 8), war die FFT bei einer höheren Anzahl an Messwerten akkurater (s Anhang 9). Die Ergebnisse der Autokorrelation können zwar nicht aus den entsprechenden Graphen in Anhang 10 erkannt werden aber aus denen ist zu erkennen, dass die Periodizität des Signals nicht verloren geht. Die zugehörigen Berechnungen liefern jedoch auch hier kein genaues Ergebnis was zuvor aus den Tests hervorging. Zum Vergleich wurden drei Skripts in Scilab geschrieben welche einige Eingabewerte für die Algorithmen zusammenfasst, dabei wurde in drei Gruppen unterteilt. Die erste Eingabewerte sind Daten ohne Rauschen (s. Anhang 11), die zweiten welche ein Rauschen aufweisen (s. Anhang 12) und als drittes (s. Anhang 13) die Messwerte aus der Messdatenfassung (s. Tab. 6-1):

| Eingabe Frequenz | Autocorr() | Fftfilter() | SinusFit() | SinusFitFFT() |
|---|---|---|---|---|
| 50 | 50 | 50 | 50.000001 | 50.000001 |
| 50.5 | 50 | 50.444444 | 50.492253 | 50.492253 |
| 50.7 | 50 | 50.666667 | 50.699845 | 50.695277 |
| 50.2 | 50 | 50.222222 | 52.708124 | 50.200502 |

Tabelle 6-1:Test 1 Eingabesignal weist kein Rauschen auf

An den Ergebnissen der Tabelle erkannt man, dass der Sinus Fit und die FFT genau arbeiten wie zuvor beschrieben. Bei der FFT ist nur der zweite Wert zu prüfen, da es durch Aufrundung der zweiten Nachkommastelle nicht genau wird, hier könnte als Ansatz eine höhere Auflösung gewählt werden beim Zeropadding des Signals. Bei Sinus Fit welcher auf die Nullstellendetektion als

Frequenzschätzung beruht tritt bei der letzten Eingabe ein Fehler auf, welcher möglichweise durch horizontale Verschiebung des Signals oder Verschiebung der Nullstellendetektion beseitigt werden kann. Zieht man als Frequenzschätzung die FFT vor kann die Schätzung fehlerfrei gemacht werden und der Sinus Fit liefert wieder den gesuchten Wert.

| Eingabe Frequenz | Autocorr() | Fftfilter() | SinusFit() | SinusFitFFT() |
|---|---|---|---|---|
| 50 | 50 | 50 | 111.19865 | 50.00316 |
| 50.5 | 50 | 50.444444 | 500.09434 | 50.495482 |
| 50.7 | 50 | 50.666667 | 50.685022 | 50.679385 |
| 50.2 | 50 | 50.222222 | 52.683511 | 50.204675 |

Tabelle 6-2:Test 2 Eingabesignal weist Rauschen auf

In Tab. 6-2 sind die Ergebnisse für den zweiten Test aufgelistet (s. Anhang 12). Es gibt keine großen Unterschiede zum ersten Testlauf was die Effektivität der Algorithmen unterstreicht. Die auffälligen Werte in der vierten Spalte zeigen erneut, dass die Nullstellendetektion Unsicherheiten aufweist, was für ein Signal mit Rauschen vor allem an den Nullübergängen eine logische Konsequenz darstellt.

| Sampleanzahl | Autocorr() | Fftfilter() | SinusFit() | SinusFitFFT() |
|---|---|---|---|---|
| 47 | 50 | 52.009456 | 51.990836 | Singulär |
| 70 | 50 | 52.063492 | 52.943834 | Singulär |
| 206 | 50 | 51.937984 | 35.489724 | Singulär |
| 479 | 50 | 51.948052 | 50.270101 | Singulär |

Tabelle 6-3:Test 3 Eingabe stammen aus Messdatenerfassung

In Tab. 6-3 wurden noch einmal die Ergebnisse aus der Implementierung zusammengefasst (s. Anhang 13). Aus diesem Grund kann hier erneut erkannt werden, dass der Sinus Fit für eine geringere Anzahl an Samples besser funktioniert als die FFT welche für eine höhere Anzahl präziser wird. Dieser Vergleich konnte gezogen werden, da 51,9 Hz als Referenzwert für die Frequenz des Datensatzes gewählt wurde. Aufgrund der genauen Frequenzbestimmung der FFT konnte in diesem Fall der Sinus Fit, welcher als Frequenzschätzung sich der FFT bedient, nicht herangezogen werden, da die Gleichungen wegen einer Singularität nicht lösbar waren. Dieses Problem könnte durch Addition einer Ungenauigkeit auf die Schätzfrequenz umgangen werden.

## 6.2 Fazit und Ausblick

Die Methode 1 der Frequenzbestimmung die hier behandelt wurde hat gezeigt, dass es durch elektrotechnische Methoden einfach ist eine Frequenzbestimmung der Netzfrequenz vorzunehmen ist. Die Effektivität und Präzision der Bestimmung kann ist in diesem Fall durch die innere Schaltung und dem Mikrokontroller gegeben, welcher eine Zählung durchführt. Da eine vorhandene Schaltung hier als Grundlage gewählt wurde musste diese getestet werden. Vergleicht man die Ausgabe am seriellen Monitor und die Oszilloskop Messung des Ausgangssignals, stimmen die Werte überein, wobei eine Ungenauigkeit von 0,02 Hz nicht ins Gewicht fallen.

Mit der Methode 2 hingegen konnte keine direkte Frequenzbestimmung gemacht werden, weshalb viele Faktoren für eine präzise Frequenzbestimmung des Eingangssignals eine Rolle spielen. Zum einen muss das Signal genauer abgetastet werden und zum anderen bedarf es einer genaueren Auflösung des A/D-Wandlers. Nichtsdestotrotz konnten die Algorithmen vor allem der Sinus Fit und die FFT überzeugen, da diese Algorithmen unabhängig von der Messdatenerfassung wirken, indem sie auf jeden Datensatz angewendet werden können. Die Autokorrelation wurde unter anderem auch als Algorithmus vorgeschlagen, da es von rauschbehafteten Signalen das Rauschen filtern kann,

aber die Periodizität nicht verloren geht, welche jedoch nicht genau genug war aufgrund der Verarbeitung.

Da Mikrokontroller meistens als Embedded Systems Anwendung finden und somit autonom fungieren müsste die obige Implementierung weiter ausgebaut werden. Damit einhergehen müsste eine Methode für die autonome Speicherung von Messdaten gefunden werden. Außerdem müsste weiterhin geprüft werden welche der getesteten Algorithmen auch zur Implementierung auf einen Microkontroller geeignet sind. Da die FFT in der modernen Signalverarbeitung sehr starke Verbreitung hat, kommt vor allem diese Methode für eine autonome Implementierung in Frage. Die Flankenzählung wurde als Projekt weitgehendst autonom gestaltet in, wobei auch hier die Optimierung für eine Speicherung von Messdaten vorgenommen werden muss. Doch unabhängig von der autonomen Gestaltung, konnte der Mikrocontroller effektiv zur Messdatenerfassung genutzt werden, wobei wie an anderen Stellen beschrieben die Auflösung die Abtastung des Signals erhöht werden muss damit eine präzisere Lösung gefunden werden kann. Die Ungenauigkeiten bei der Abtastung könnten auch durch die elektronische Schaltung entstanden sein, welcher einen Operationsverstärker zur Verstärkung des Signals benutzt. Auch hier kommen weitere Operationsverstärker in Frage als Alternative.

Ungeachtet der Verbesserungsmöglichkeiten für eine Implementierung wurde die das Ziel des Vergleichs und Auswertens von Verfahren der Frequenzbestimmung am Beispiel der Netzfrequenzmessung erfüllt, da die Verfahren und somit das Testen dieser Methoden im Vordergrund standen. Vor allem die Simulationen haben gezeigt das die Algorithmen auch für rauschbehafteten Signale gut funktionieren. Somit können diese Ergebnisse auch für weitere zukünftige Tests in diesem Bereich als Grundlage genommen werden. Diese gilt auch für die weitere Implementierung der Algorithmen in anderen Programmiersprachen wie z.B. in C einer maschinennahen Sprache oder in VHDL für die Implementierung auf einem FPGA.

# Literaturverzeichnis


| | |
|---|---|
| [RHL16] | Reinhard Lerch,<br>Elektrische Messtechnik: Analoge, digitale und computergestützte Verfahren,<br>Springer, 7. Auflage, Berlin, Heidelberg, 2016, Seiten 411-412 |
| [NIC17] | National Instruments Co.,<br>Messungen von Frequenzen,<br>URL: http://www.ni.com/tutorial/7111/de/#toc2 [Stand: 25.10.2017] |
| [INA15] | Ibrahim Nasser,<br>Power Frequency Website: Beschreibung der Hardware-Schaltung zur Netzfrequenzmessung,<br>Forschungsseminar im Master Elektrotechnik, Fachhochschule Köln, 15.März 2015 |
| [PDS10] | Prof. Dr. Jürgen Schneider,<br>Skript zur Vorlesung Elektronik 1,<br>Fachhochschule Köln, Köln, 01.09.2010, Seite 19 |
| [ELK17] | Elektronik Kompendium,<br>Zener-Dioden,<br>URL:https://www.elektronik-kompendium.de/sites/bau/0201211.htm<br>[Stand:27.10.2017] |
| [STE17] | Lernvideos und Vorträge,<br>Der Schmitt-Trigger (Elektronik-Kurs),<br>URL:https://www.youtube.com/watch?v=d4VwGOXaFqI<br>[Stand:30.10.2017] |
| [PMS96] | Prof. Dr. Manfred Seifart,<br>Analoge Schaltungen,<br>Verlag Technik Berlin, 5. Auflage, Berlin, 1996, Seite 504, 496 |
| [GKK02] | Dip. Ing. Gerhard Krucker,<br>Versuche zum Schmitt-Trigger,<br>Uni Bern, 28.5.2002 |

| | |
|---|---|
| [DMU15] | Dervis Mujagic,<br>Power Frequency Website: Beschreibung des Microcomputers Arduino zur Netzfrequenzmessung,<br>Forschungsseminar im Master Elektrotechnik, Fachhochschule Köln, 15. März 2015 |
| [HBE09] | Helmut Berka,<br>Phase Locked Loop,<br>DARC Verlag GmbH, CQ DL deutsches Amateurfunkmagazin, 2009 |
| [ITW17] | IT Wissen,<br>Abtastrate,<br>URL: http://www.itwissen.info/Abtastrate-sampling-rate.html<br>[Stand: 03.11.2017] |
| [TUW17] | TU Wien,<br>Die Diskrete Fourier Transformation,<br>https://ti.tuwien.ac.at/cps/teaching/courses/dspv/files/DFT-FFT.pdf,<br>Seite 4 [Stand: 05.11.2017] |
| [ELT17] | Electronics Tutorials,<br>Passive Band Pass Filter,<br>URL: http://www.electronics-tutorials.ws/filter/filter_4.html<br>[Stand: 10.11.2017] |
| [GAU17] | Georg-August-Universität Göttingen,<br>Der Operationsverstärker,<br>URL: https://lp.uni-goettingen.de/get/text/4456#label162<br>[Stand:13.11.2017] |
| [BBD17] | Breadbording.de,<br>FFT (Fast Fourier Transform),<br>URL: http://www.breadboarding.de/arduino-fft-spektrumanalysator/<br>[Stand:21.11.2017] |

| | |
|---|---|
| [WHE10] | Dr.-Ing. Wolfgang Heenes,<br>Grundlagen der Rechnertechnologie: 11. Vorlesung,<br>TU Darmstadt, 29.06.2010, Seite 15 |
| [PMA06] | J. G. Proakis, D. G. Manolakis,<br>Digital Signal Processing,<br>Pearson, 4th Ed., New Jersey, 2006, Page 519 |
| [WAC94] | Waveform Measurements and Analysis Committee,<br>IEEE Standard for Digitizing Waveform Recorders,<br>The Institute of Electrical and Electronics Engineers, Inc., USA, NY,<br>1994, Page 17 f. |
| [AAU18] | Aalborg University,<br>Pitch Detection,<br>URL: http://kom.aau.dk/group/04gr742/pdf/pitch_worksheet.pdf,<br>Seite 11 [Stand: 08.01.2018] |
| [NIC18] | National Instruments Co.,<br>Using Fast Fourier Transforms and Power Spectra in LabVIEW,<br>URL: http://www.ni.com/white-paper/4541/en/#toc7<br>[Stand: 08.01.2018] |
| [ELL18] | Elektronik-Labor,<br>Arduino Duo ein erster Test,<br>URL: http://www.elektronik-labor.de/Arduino/Due.html<br>[Stand: 11.01.2018] |

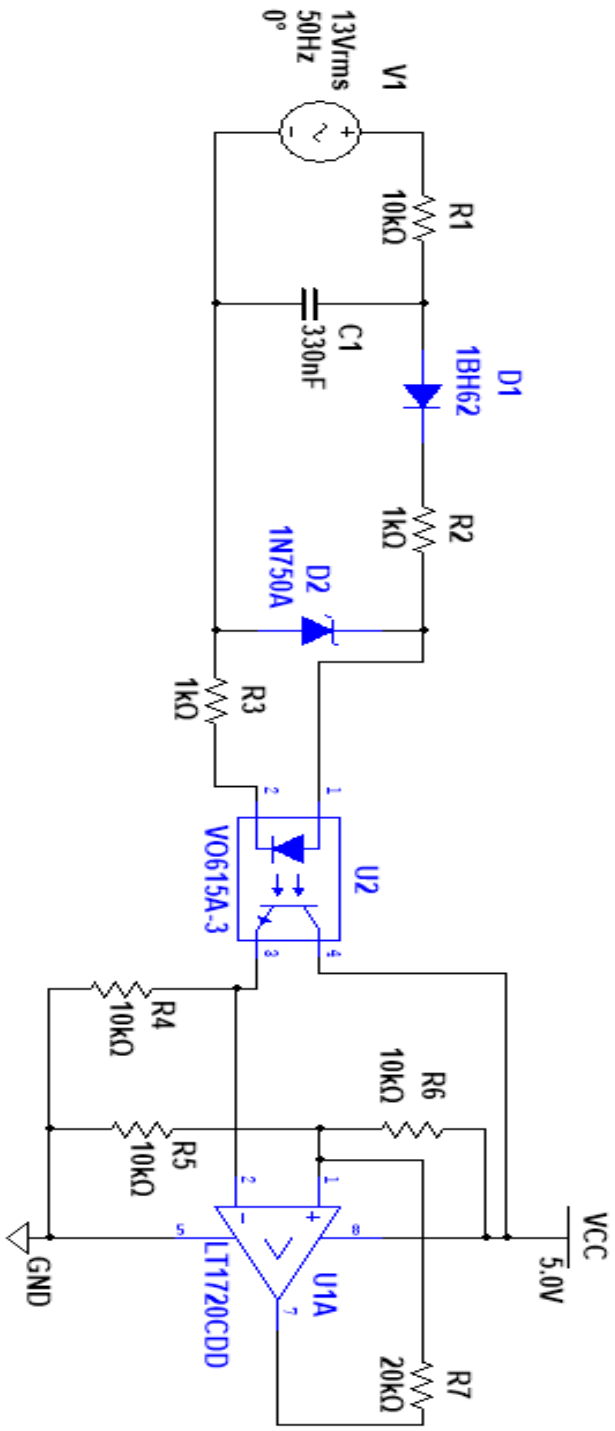
# Abkürzungsverzeichnis

| | |
|---|---|
| Abb. | Abblidung |
| A/D | Analog-Digital |
| AC | Alternating Current |
| ana. | analog |
| BW | Bandwidth |
| bzw. | beziehungsweise |
| C | Formelzeichen für einen Kondensator |
| ca. | circa |
| dB | Einheit Dezibel |
| DC | Direct Current |
| $\Delta f$ | Abtastfrequenz der Frequenz-Bins |
| $\Delta \omega$ | Kreisfrequenzintervall |
| $\Delta t$ | Abtastintervall |
| d.h. | das heisst |
| DFT | Diskrete Fourier Transformation |
| Div. | Division |
| $\varepsilon$ | mittlerer Fehler Epsilon |
| $\varepsilon_{rms}$ | Root Mean Square Epsilon |
| E-Reihe | genormte Folge von Eigenschaftswerten elektrischer Bauelemente |
| F | Einheit Farad |
| f. | Folgende |
| FFT | Fast Fourier Transformation |
| $f_g$ | Grenzfrequenz |

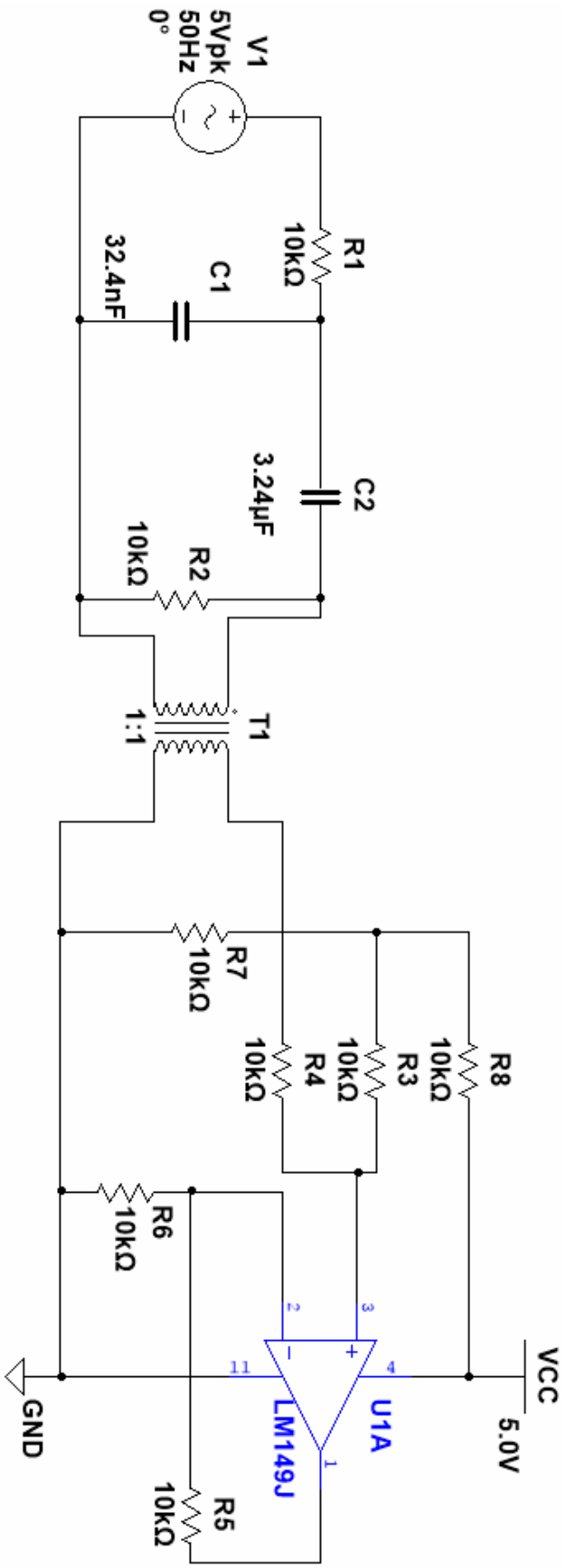
| | |
|---|---|
| $f_H$ | Obere Grenzfrequenz |
| $f_L$ | Untere Grenzfrequenz |
| $f_x$ | Gesuchte Frequenz |
| $F_k$ | Frequenz-Bin-Vektor |
| FPGA | Field Programmable Gate Array |
| GND | Ground |
| Kap. | Kapitel |
| kHz | Kilohertz |
| kΩ | Kiloohm |
| µF | Mikrofarad |
| mV | Millivolt |
| ms | Millisekunden |
| N | Anzahl der Messwerte |
| nF | Nanofarad |
| $N_x$ | Anzahl der steigenden Flanken |
| NI | National Instruments |
| Ω | Ohm |
| Opam | Operational Amplifier |
| PLL | Phase Locked Loop |
| Φ | Phasenverschiebung Phi |
| Π | Kreiszahl Pi |
| R | Formelzeichen für einen Widerstand |
| R[k] | Autokorrelation |
| s. | siehe |

| | |
|---|---|
| s | Sekunde |
| Tab. | Tabelle |
| T | Periodendauer |
| $T_{ref}$ | Messdauer |
| t | Formelzeich für die Zeit/zeitliche Verschiebung |
| $U_a$ | Ausgangsspannung |
| $U_e$ | Eingangsspannung |
| $U_H$ | Obere Grenzspannung |
| $U_L$ | Untere Grenzspannung |
| $U_{ref}$ | Referenzspannug |
| URL | Uniform Resource Locator |
| V | Volt/Verstärkungsfaktor |
| $V_{cc}$ | Versorgungsspannung |
| vgl. | Vergleich |
| VHDL | Very High Speed Integrated Circuit Hardware Description Language |
| ω | Kreisfrequenz |
| $x_n$ | Datensatz mit Messwerten |

# Anhang



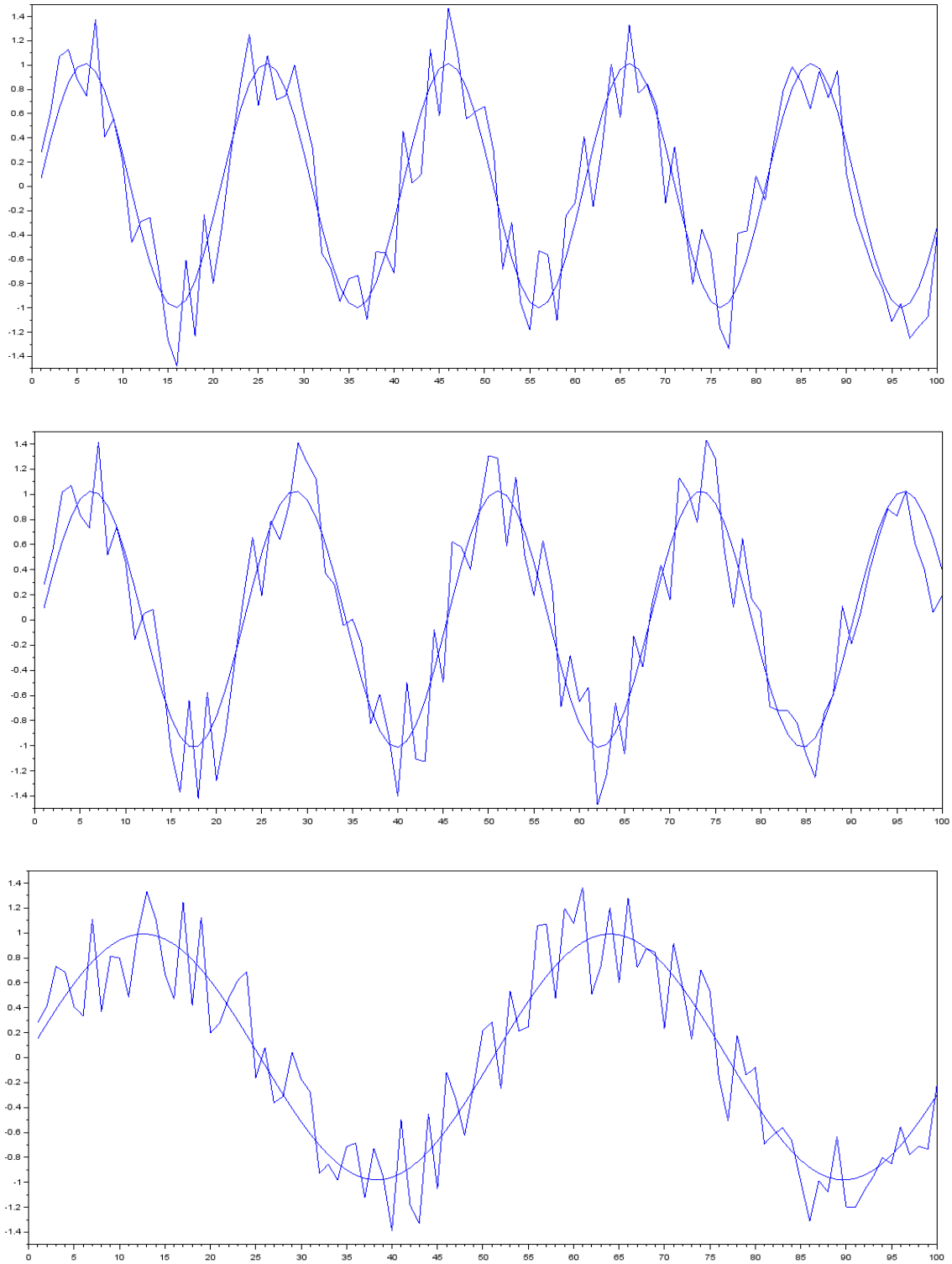

Anhang 1:Schaltung 1 für die Simulation in Multisim




Anhang 2: Schaltung 2 für die Simulation in Multisim

```
//Die Funktion sinusFit erstellt die beste Fit Funktion fur eine
//Sinusförmige Funktion als Ausgabe wird die resultierende Kreis-
//Frequenz berechnet
function [Ausgabe]=sinusFit(data,t)
//Kreiszahl PI
pi=3.14159265359
//Zeit Data
//t=(0:99)/1000
//Data generieren
//data=sin(2*pi*f*t)
//Anzahl der Samples in len(2)
len=size(data)
//Sinus Fit mit geschätzter Frequenz
//Matrix für Anfangswerte anlegen
//Erste Schleife um Zeilen anzulegen
for i=1:len(2)
//Zweite Schleife um Spalten anzulegen
for j=1:3
//Erste Spalte für aktuelle Zeile anlegen
if j==1 then
A(i,j)=cos(w*t(i))
//Zweite Spalte für aktuelle Zeile anlegen
elseif j==2 then
A(i,j)=sin(w*t(i))
//Dritte Spalte für aktuelle Zeile anlegen
else
A(i,j)=1
end
end
end
//Anfangswerte berechnen nach SinusFit Formel
M=(A'*A)^(-1)
N=(A'*data')
//Ergebnisvektor x mit Anfangswerten A,B,C
x=M*N
//Initialisierung der 4. Spalte des Ergebnisvektors für Delta w
x(4)=1
//Initialisierung von Delta x (Fehler aufeinanderfolgender Ergebnisse)
dx=1
//Alten Ergebnisvektor speichern
y=x
//////////////////////////////////////////////////////////////////////////////
//Hier ist die Frequenzschätung einzufügen
//Sinus Fit mit unbekannter Frequenz
//Schleife ausführen solange der Fehler nicht minimal g
while dx>0.001 do
//Matrix für nächsten Ergebisvektor anlegen
//Erste Schleife um Zeilen anzulegen
for i=1:len(2)
//Zweite Schleife um Spalten anzulegen
for j=1:4
//Erste Spalte für aktuelle Zeile anlegen
if j==1 then
D(i,j)=cos(w*t(i))
//Zweite Spalte für aktuelle Zeile anlegen
elseif j==2 then
D(i,j)=sin(w*t(i))
//Dritte Spalte für aktuelle Zeile anlegen
elseif j==3 then
D(i,j)=1
//Vierte Spalte für aktuelle Zeile anlegen
else
D(i,j)=-x(1)*t(i)*sin(w*t(i))+x(2)*t(i)*cos(w*t(i))
end
end
end
//Neuen Ergebnisvektor berechnen nach SinusFit Formel
M=(D'*D)^(-1)
N=(D'*data')
//Neuer Ergebnisvektor x mit A,B,C,delta w
x=M*N
//Neue Kreisfrequenz w aus delta w + alte Kreisfrequenz
w=x(4)+w
//Ergebnisfehler berechnen
dx=abs(y-x)
//Alten Ergebnisvektor speichern
y=x
//Ende der Schleife
end
//Plotten der Sinus Fit Funktion
res=x(1)*cos(w*t)+x(2)*sin(w*t)+x(3)
//plot(res)
//Fehler berechnen
fehler=data-res
//Fehlerquadrate addieren
for i=1:len(2)
e=fehler(i)*fehler(i)+e
end
//REsultierenden Fehler berechnen
e=sqrt(e/len(2))
//Kreisfrequenz und Fehler ausgeben
Ausgabe=[w e]
//Ende der Funktion SinusFit
endfunction
```

Anhang 3:Sinus Fit Algorithmus für die Simulation in Scilab




Anhang 4:Simulation von Sinus Fit Funktionen für f=50, f=45, f=20 bei Rauschen

```
function [Ausgabe]=fftfilter(data,t)
//Kreiszahl PI
pi=3.14159265359
//Anzahl der Samples in len(2)
len=size(data)
//Berechen von Delta f
df=1/(len(2)*t(2))
//Zeropadding
if df<1 then
data=[data zeros(1,(1/t(2))*8)]
elseif df==1 then
data=[data zeros(1,8/t(2))]
else
data=[data zeros(1,len(2)*8)]
end
//Neue Anzahl der Samples in len(2)
len=size(data)
//Berechen von neuem Delta f
df=1/(len(2)*t(2))
//Erzeugen der Hann Fenster Funktion
fen=window('hn',len(2))
//Multiplikation von Data und Fensterfunktion
res=data.*fen
//FFT berechnen
res=fft(res)
//Länge der komplexen Zahlen berechen
res=abs(res)
//Länge der Samples berechnen
le=size(res)
//Länge der Samples berechnen
le=size(res)
//Maximum berechen
m=max(res(60:le(2)/2))
//Frequenz initialisieren
f=0
//Schleife zur Bestimmung der Frequenz
for i=2:le(2)
//Lage des Maximums abfragen
if m==res(i) then
//Grundfrequenz liegt beim Maximun
f=i-1
//Schleife unterbrechnen
break
end
end
//Gibt das Frequenzspektrum aus
plot2d(res)
//Ausgabe des Frequenz-Bins und von Delta f
Ausgabe=[f df]
endfunction
```

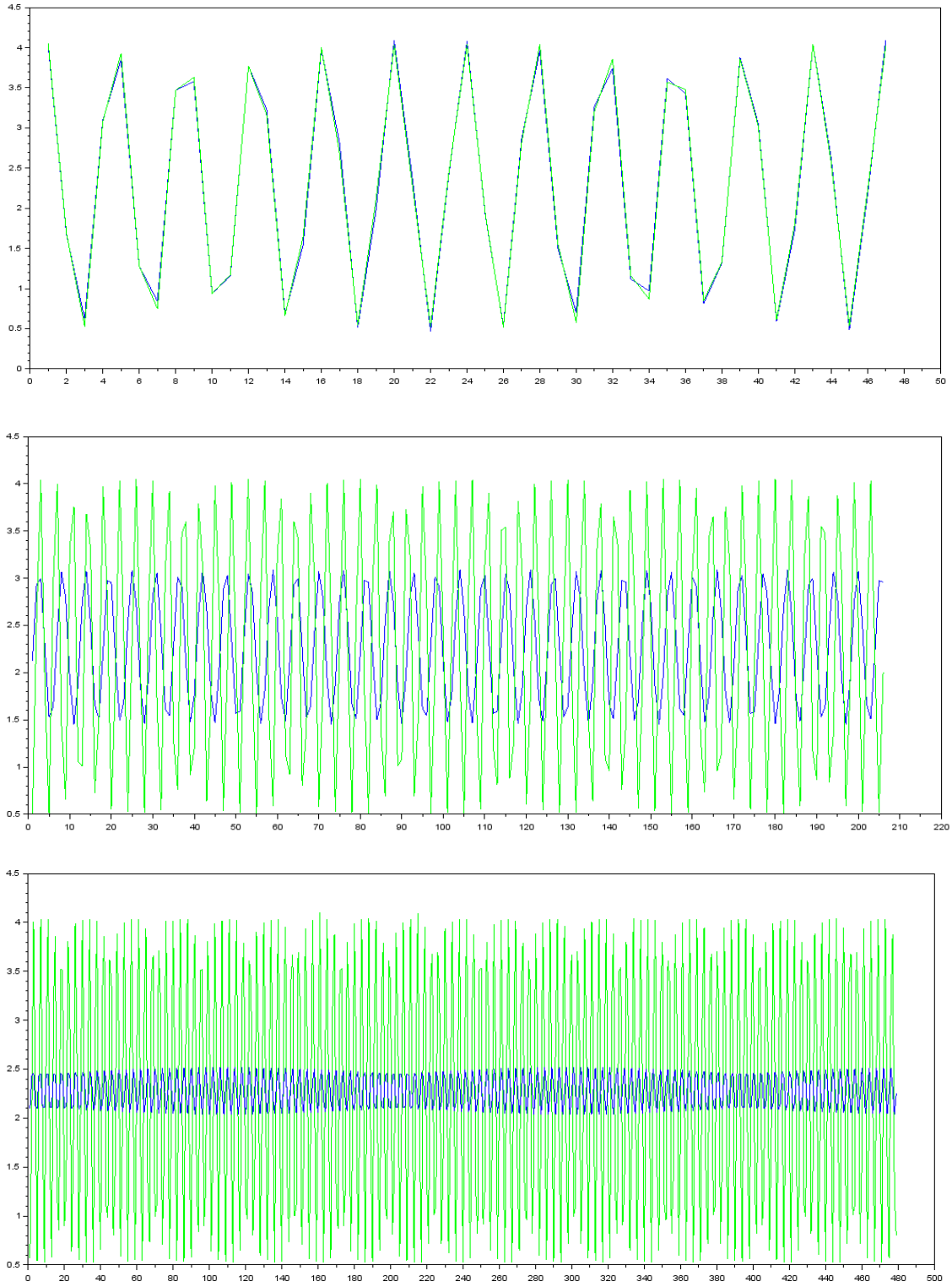
Anhang 5:FFT zur Filterung der Frequenzinformation in Scilab

```
function [f]=autofreq(data,fs)
//Berechnung der Autokorrelation von data
[autocor lags]=xcorr(data,fs,'coeff')
//Berechung der Anzahl der Daten
len=length(autocor)
//Bestimmung des Maximum
m=max(autocor)
//Initialisierung von j
j=0
//Schleife zur Bestimmung von Zwei Maxima
for i=len/2:len
//Abfrage ob  dieAutokorrealtion maximal ist
if autocor(i)==m then
//Inkrement von j
j=j+1
//Gefundenen Maximum Speichern
maxima(j)=i
//Wenn zwei Maxima gefunden
if j==2 then
//Schleife unterbrechen
break
end
end
//Wenn Autokorrealtion Null wird
if autocor(i)~=0 & autocor(i+1)==0 then
//Schleife unterbrechen
break
end
//Wenn nächste Steigung positiv ist
if autocor(i)<autocor(i+1) then
//Nächstes Maximum detektieren
m=max(autocor(i:len))
end
end
//Ausgabe der berechneten Frequenz an hand der Lage der Maxima
f=[1/((lags(maxima(2))/fs)-(lags(maxima(1))/fs)) maxima']
//Ausgabe der Autokorrelaion als Graph
plot(lags/fs,autocor)
endfunction
```

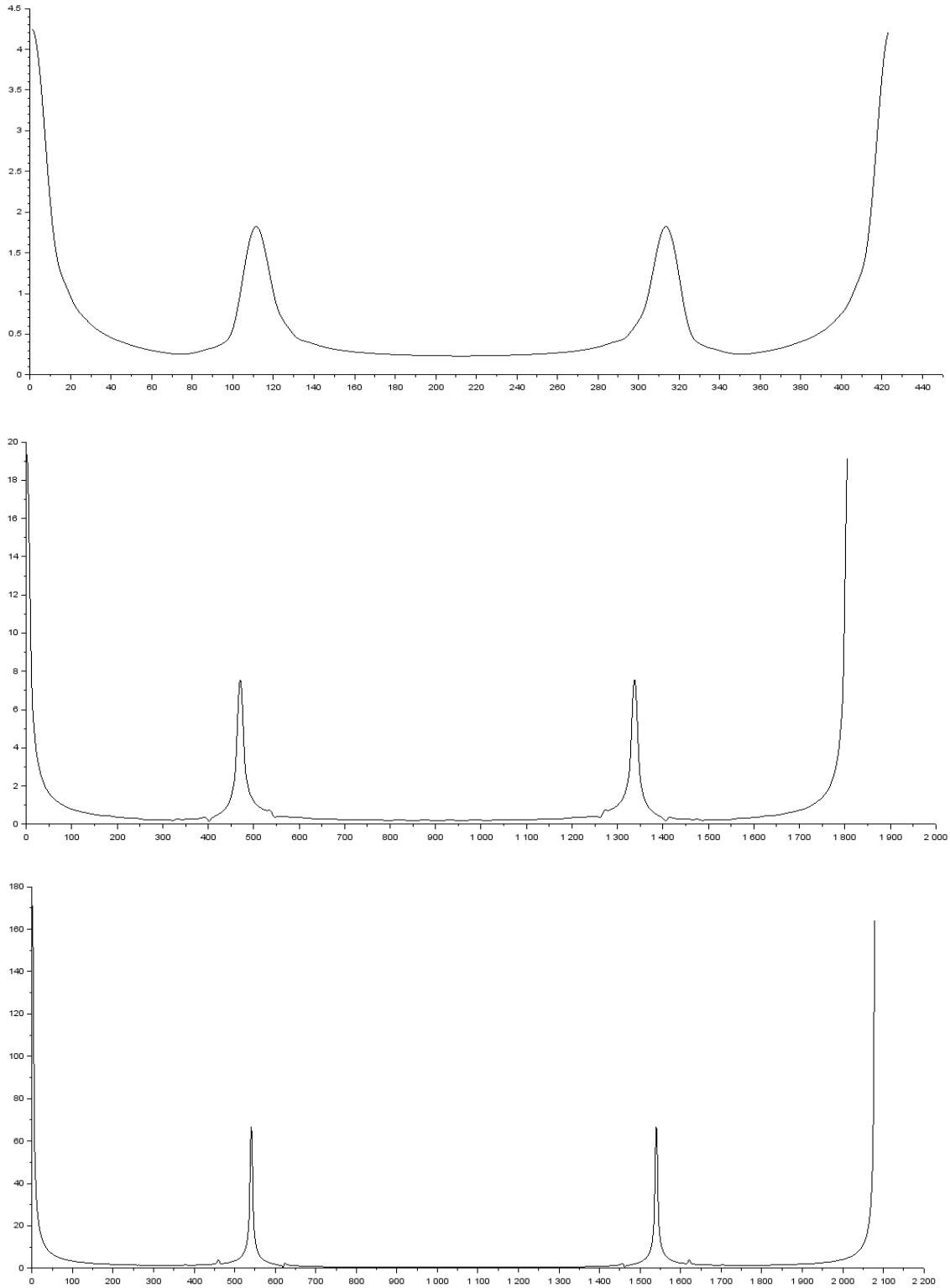
Anhang 6:Autokorrelation und Frequenzbestimmung in Scilab

```
//Zählalgorithmus
//Variable für den Zustand der aktuellen Flanke
boolean newState=LOW;
//Variable für den Zustand der alten Flanke
boolean oldState=LOW;
//Anzahl der steigenden Flanken
int risingEdge=0;
//Startzeit der Flankenzählung
double oldTime=0;
//Endzeit der Flankenzählung
double newTime=0;
//Resultierende Frequenz
double frequency=0;
void setup() {
  //Serielle Kommunikation einstellen
  Serial.begin(9600);
  //Pin 8 als Inputpin deklarieren
  pinMode(8,INPUT);
  //Alten Zustand einlesen über den digitalen Pin 8
  oldState=digitalRead(8);
}
void loop() {
//Neuen Zustand einlesen über den digitalen Pin 8
newState=digitalRead(8);
//Prüfen ob steigende Flanke detektiert wurde
if(oldState==LOW and newState==HIGH)
{
//Prüfen ob die initiale steigende Flanke detektiert wurde
    if(risingEdge==0)
//Startzeit der Flankenzählung initialisieren
    oldTime=micros();
//Steigende Flanke zählen
    risingEdge++;
}
//Bei 51 gezählten Flanken Frequenz berechnen
if(risingEdge==51)
  {
//Endzeit der Flakenzählung einlesen
    newTime=micros();
//Frequenz berechnen
    frequency=50.000E6/(newTime-oldTime);
// Ausgabe der Frequenz über seriellen Monitor
    Serial.println(frequency,3);
//Flankenzählung für nächste Zählung initialisieren
    risingEdge=0;
  }
//Alten Zustand der Flanke einlesen
oldState=newState;
}
```

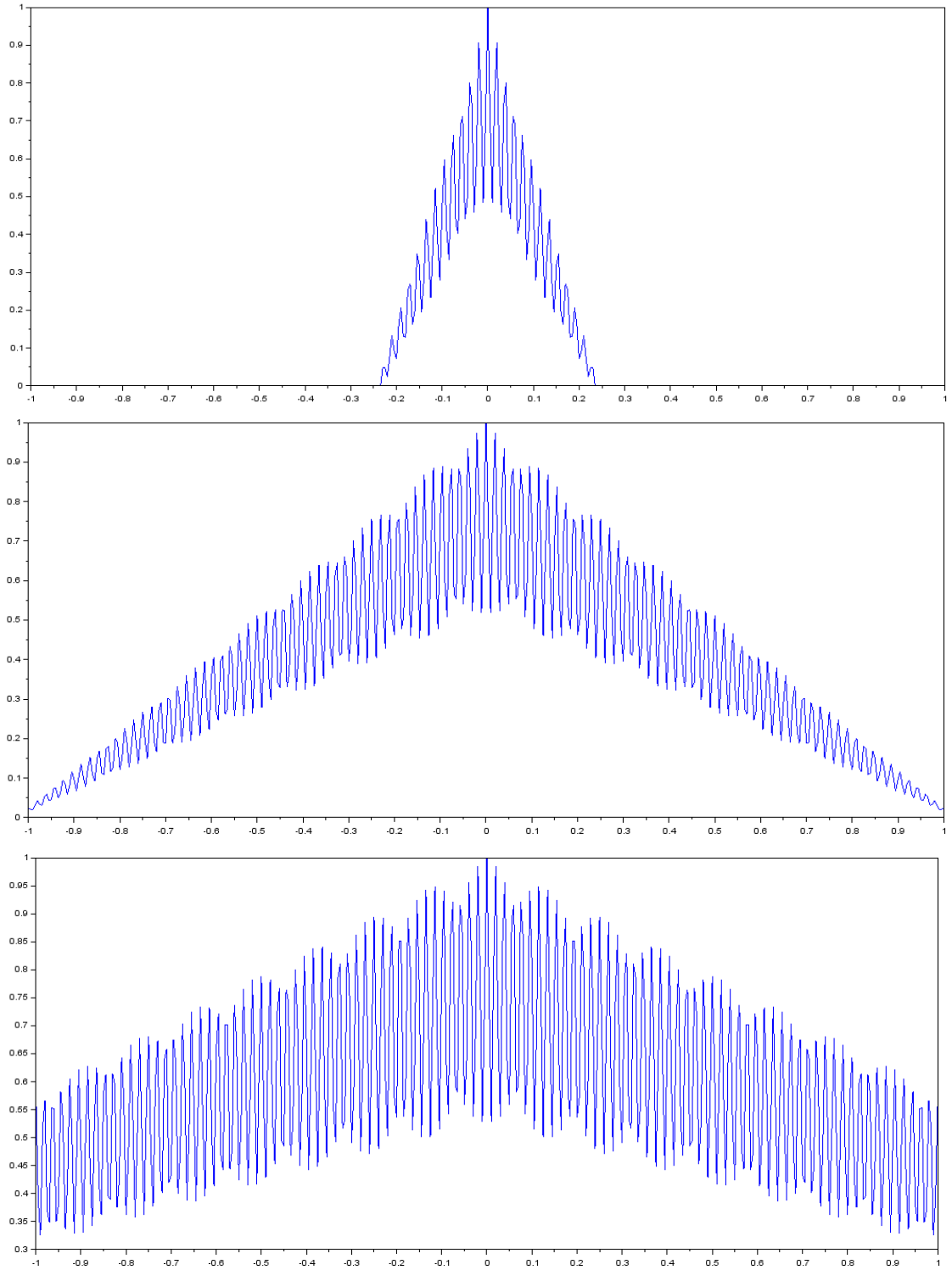
Anhang 7:Zählalgorithmus auf dem Mikrocontroller




Anhang 8:Sinus Fit Funktionen (blau) für die realen Messungen (grün) mit Anzahl 47, 206, 479




Anhang 9:Frequenzspektren der Messungen mit Anzahl 47, 206, 479




Anhang 10:Autokorrelationen der Messung mit Anzahl 47,206,479

```
//Testet die Algorithmen für ideale Messungen
function [Ausgabe]=test1()
//Schleife um den Funktionen die verschiedenen Daten zu über
for i=1:4
//Testlauf 1 für die Frequenz f=50
if i==1 then
data=sin(2*3.141*50*(0:999)/1000)
end
//Testlauf 2 für die Frequenz f=50.5
if i==2 then
data=sin(2*3.141*50.5*(0:999)/1000)
end
//Testlauf 3 für die Frequenz f=50.7
if i==3 then
data=sin(2*3.141*50.7*(0:999)/1000)
end
//Testlauf 4 für die Frequenz f=50.2
if i==4 then
data=sin(2*3.141*50.2*(0:999)/1000)
end
//Zeit Variable bestimmen
t=(0:length(data)-1)/1000
//Abtastfrequenz bestimmen
fs=1/t(2)
//Autokorrelation aufrufen
//Ausgabe enthält die Frequenz
f1=autofreq(data,fs)
//FFT aufrufen
//Ausgabe enthält Frequenz-Bin und Delta f
f2=fftfilter(data,t)
//Sinus Fit aufrufen
//Ausgabe enthält die Kreisfrequenz und den Fehler
w1=sinusFit(data,t)
//Sinus Fit mit FFT aufrufen
////Ausgabe enthält die Kreisfrequenz und den Fehler
w2=sinusFitFFT(data,t)
//Für den aktuellen Testlauf die Frequenzen bestimmen
//res stellt eine Matrix dar und legt pro Testlauf eine Zeile an
res(i,:)=[f1(1) f2(1)*f2(2)  w1(1)/(2*3.141) w2(1)/(2*3.141)]
//End der Schleife
end
//Ausgabe der Matrix mit den Vergleichswerten
Ausgabe=res
//Ende der Funktion
endfunction
```

Anhang 11: Skript für den ersten Testlauf

```
//Überlagert dem Signal s einen Rauschen
function [ns]=rauschen(s,rf)
//Bestimmen der Länge von s
len=length(s)
//Rauschen mit der Zufallsfunktion generieren
rauschen=rand(0:len-1)-rf
//Rauschen auf das Ausgangssignal addieren
//Neues Signal ausgeben
ns=s+rauschen
//Ende der Funktion
endfunction
//Testet die Algorithmen für generierte reale Messungen
function [Ausgabe]=test2()
//Schleife um den Funktionen die verschiedenen Daten zu übergeben
for i=1:4
//Testlauf 1 für die Frequenz f=50
if i==1 then
data=sin(2*3.141*50*(0:999)/1000)
end
//Testlauf 2 für die Frequenz f=50.5
if i==2 then
data=sin(2*3.141*50.5*(0:999)/1000)
end
//Testlauf 3 für die Frequenz f=50.7
if i==3 then
data=sin(2*3.141*50.7*(0:999)/1000)
end
//Testlauf 4 für die Frequenz f=50.2
if i==4 then
data=sin(2*3.141*50.2*(0:999)/1000)
end
//Zeit Variable bestimmen
t=(0:length(data)-1)/1000
//Den Daten einen Rauschen überlagern
data=rauschen(data,0.5)
//Abtastfrequenz bestimmen
fs=1/t(2)
//Autokorrelation aufrufen
//Ausgabe enthält die Frequenz
f1=autofreq(data,fs)
//FFT aufrufen
//Ausgabe enthält Frequenz-Bin und Delta f
f2=fftfilter(data,t)
//Sinus Fit aufrufen
//Ausgabe enthält die Kreisfrequenz und den Fehler
w1=sinusFit(data,t)
//Sinus Fit mit FFT aufrufen
////Ausgabe enthält die Kreisfrequenz und den Fehler
w2=sinusFitFFT(data,t)
//Für den aktuellen Testlauf die Frequenzen bestimmen
//res stellt eine Matrix dar und legt pro Testlauf eine Zeile an
res(i,:)=[f1(1) f2(1)*f2(2)  w1(1)/(2*3.141) w2(1)/(2*3.141)]
//End der Schleife
end
//Ausgabe der Matrix mit den Vergleichswerten
Ausgabe=res
//Ende der Funktion
endfunction
```

Anhang 12: Skript für den zweiten Testlauf

```
//Testet die Algorithmen für reale Messungen aus der Messdateneerfassung
function [Ausgabe]=test3()
//Messwerte aus der Messdatenerfassung, Anzahl der Werte beträgt 47
x47=[4.05   1.69    0.53    3.08    3.93    1.28    0.75    3.47    3.63
     0.94   1.17    3.77    3.16    0.66    1.65    4.00    2.69    0.55
    2.10    4.03    2.29    0.53    2.44    4.04    1.94    0.52    2.82
    4.04    1.55    0.58    3.21    3.86    1.16    0.87    3.57    3.48
    0.85    1.33    3.86    3.02    0.61    1.80    4.04    2.57    0.54
    2.22    4.03]
//Messwerte aus der Messdatenerfassung, Anzahl der Werte beträgt 70
x70=[2.74   4.03    1.63    0.55    3.14    3.89    1.29    0.81    3.50
    3.55    0.91    1.26    3.81    3.08    0.64    1.73    3.99    2.62
    0.55    2.17    4.02    2.23    0.53    2.52    4.04    1.87    0.52
    2.90    4.02    1.47    0.65    3.29    3.79    1.09    0.98    3.63
    3.36    0.78    1.45    3.91    2.88    0.57    1.91    4.01    2.45
    0.54    2.31    4.03    2.09    0.52    2.67    4.03    1.71    0.53
    3.06    3.94    1.31    0.74    3.44    3.63    1.03    1.16    3.75
    3.16    0.67    1.64    3.98    2.71    0.55    2.08]
//Messwerte aus der Messdatenerfassung, Anzahl der Werte beträgt 206
x206=[.51   2.56    4.04    1.82    0.51    2.94    4.00    1.43    0.66
    3.33    3.76    1.06    1.01    3.68    3.31    0.73    1.50    3.97
    2.82    0.55    1.95    4.03    2.39    0.53    2.35    4.05    2.05
    0.52    2.71    4.03    1.68    0.54    3.08    3.92    1.27    0.76
    3.47    3.60    0.92    1.19    3.79    3.13    0.64    1.68    3.98
    2.68    0.54    2.12    4.02    2.26    0.52    2.47    4.05    1.90
    0.51    2.84    4.03    1.52    0.59    3.25    3.84    1.13    0.92
    3.60    3.42    0.80    1.39    3.90    2.95    0.58    1.85    4.01
    2.51    0.53    2.26    4.04    2.14    0.52    2.61    4.05    1.77
    0.51    2.99    3.99    1.37    0.69    3.39    3.70    1.01    1.08
    3.73    3.23    0.69    1.55    3.97    2.77    0.51    2.01    4.02
    2.34    0.52    2.40    4.03    1.99    0.51    2.75    4.04    1.63
    0.55    3.14    3.90    1.23    0.81    3.51    3.54    0.88    1.27
    3.82    3.07    0.61    1.75    4.00    2.61    0.54    2.17    4.03
    2.20    0.52    2.53    4.04    1.86    0.52    2.89    4.03    1.48
    0.64    3.29    3.79    1.09    0.96    3.65    3.36    0.76    1.44
    3.93    2.89    0.56    1.91    4.02    2.45    0.53    2.32    4.05
    2.08    0.51    2.66    4.04    1.72    0.52    3.04    3.95    1.31
    0.73    3.43    3.65    0.96    1.14    3.76    3.17    0.66    1.63
    3.98    2.81    0.55    2.08    4.03    2.31    0.52    2.44    4.05
    1.95    0.51    2.80    4.04    1.56    0.59    3.20    3.87    1.17
    0.87    3.55    3.48    0.84    1.34    3.88    3.00    0.59    1.81
    4.01    2.55    0.53    2.23    4.03    2.17    0.52    2]
//Messwerte aus der Messdatenerfassung, Anzahl der Werte beträgt 479
x479=[0.57  2.04    4.01    2.34    0.54    2.41    4.03    1.98    0.53
    2.79    4.03    1.59    0.58    3.16    3.86    1.21    0.86    3.52
    3.51    0.90    1.30    3.81    3.05    0.64    1.75    3.99    2.61
    0.55    2.17    4.02    2.23    0.53    2.53    4.03    1.85    0.53
    2.90    4.01    1.47    0.66    3.29    3.86    1.10    0.98    3.62
    3.35    0.80    1.44    3.89    2.90    0.59    1.90    4.00    2.47
    0.54    2.29    4.03    2.11    0.55    2.64    4.03    1.73    0.54
    3.03    3.94    1.34    0.73    3.41    3.66    0.99    1.12    3.71
    3.20    0.71    1.59    4.01    2.75    0.56    2.03    4.01    2.34
    0.54    2.39    4.03    1.99    0.53    2.77    4.03    1.61    0.58
    3.15    3.87    1.22    0.84    3.51    3.52    0.90    1.28    3.81
    3.07    0.64    1.74    3.99    2.62    0.55    2.17    4.02    2.23
    0.53    2.53    4.03    1.86    0.53    2.89    4.02    1.48    0.65
    3.27    3.79    1.10    0.97    3.61    3.37    0.80    1.43    3.90
    2.91    0.59    1.88    4.00    2.48    0.55    2.29    4.02    2.11
    0.53    2.64    4.03    1.74    0.54    3.02    3.95    1.36    0.73
    3.40    3.67    1.00    1.11    3.70    3.22    0.71    1.57    3.94
    2.77    0.56    2.02    4.02    2.35    0.54    2.39    4.10    2.00
    0.53    2.77    4.03    1.61    0.57    3.14    3.88    1.23    0.87
    3.50    3.53    0.90    1.27    3.80    3.08    0.64    1.74    3.99
    2.63    0.55    2.16    4.03    2.23    0.53    2.51    4.04    1.87
    0.53    2.89    4.02    1.49    0.65    3.27    3.79    1.11    0.97
    3.61    3.38    0.81    1.42    3.89    2.92    0.59    1.88    4.00
    2.49    0.54    2.28    4.03    2.12    0.53    2.63    4.09    1.74
    0.53    3.01    3.95    1.36    0.72    3.39    3.68    1.01    1.11
    3.70    3.23    0.72    1.56    3.94    2.78    0.62    2.01    4.01
    2.36    0.54    2.39    4.03    2.00    0.53    2.76    4.03    1.62
    0.57    3.14    3.88    1.23    0.82    3.51    3.53    0.91    1.26
    3.80    3.08    0.65    1.72    3.98    2.63    0.56    2.15    4.02
    2.24    0.53    2.51    4.03    1.88    0.53    2.88    4.02    1.49
    0.65    3.27    3.79    1.11    0.96    3.61    3.38    0.82    1.41
    3.88    2.93    0.60    1.87    4.00    2.49    0.55    2.27    4.03
    2.13    0.53    2.63    4.03    1.75    0.53    3.01    3.96    1.36
    0.72    3.39    3.67    1.00    1.10    3.70    3.23    0.72    1.56
    3.94    2.78    0.57    2.00    4.01    2.36    0.54    2.39    4.03
    2.01    0.53    2.76    4.03    1.63    0.57    3.13    3.88    1.23
    0.82    3.50    3.54    0.91    1.26    3.80    3.09    0.65    1.72
    4.04    2.63    0.55    2.15    4.02    2.24    0.54    2.50    4.03
    1.88    0.53    2.88    4.02    1.50    0.65    3.26    3.80    1.12
    0.96    3.61    3.39    0.82    1.41    3.88    2.93    0.60    1.87
    4.00    2.49    0.55    2.27    4.03    2.13    0.53    2.63    4.03
    1.75    0.54    3.01    3.95    1.37    0.72    3.39    3.68    1.01
    1.10    3.70    3.23    0.72    1.56    3.94    2.82    0.56    2.01
    4.02    2.36    0.54    2.38    4.03    2.00    0.53    2.76    4.03
    1.62    0.56    3.13    3.88    1.24    0.82    3.50    3.54    0.91
    1.25    3.80    3.09    0.65    1.72    3.99    2.64    0.55    2.15
    4.02    2.24    0.53    2.50    4.03    1.88    0.53    2.88    4.02
    1.50    0.65    3.26    3.80    1.12    0.99    3.61    3.39    0.82
    1.42    3.89    2.93    0.60    1.87    4.00    2.49    0.54    2.27
    4.03    2.13    0.53    2.63    4.04    1.75    0.53    3.01    3.95
    1.36    0.72    3.39    3.68    1.01    1.10    3.70    3.23    0.72
    1.56    3.94    2.77    0.57    2.01    4.01    2.37    0.54    2.39
    4.03    2.00    0.53    2.76    4.03    1.62    0.57    3.14    3.88
    1.23    0.8]
```

```
//Schleife um den Funktionen die verschiedenen Daten zu übergebe
for i=1:4
//Übergabe des ersten Datensatzes
if i==1 then
data=x47
end
//Übergabe des zweiten Datensatzes
if i==2 then
data=x70
end
//Übergabe des dritten Datensatzes
if i==3 then
data=x206
end
//Übergabe des vierten Datensatzes
if i==4 then
data=x479
end
//Zeit Variable bestimmen
t=(0:length(data)-1)/200
//Abtastfrequenz bestimmen
fs=1/t(2)
//Autokorrelation aufrufen
//Ausgabe enthält die Frequenz
f1=autofreq(data,fs)
//FFT aufrufen
//Ausgabe enthält Frequenz-Bin und Delta f
f2=fftfilter(data,t)
//Sinus Fit aufrufen
//Ausgabe enthält die Kreisfrequenz und den Fehler
w1=sinusFit(data,t)
//Sinus Fit mit FFT kann nicht aufgerufen werden
//Aufgrund von Singularitäts problemen
//w2=sinusFitFFT(data,t)
//Für den aktuellen Testlauf die Frequenzen bestimmen
//res stellt eine Matrix dar und legt pro Testlauf eine Zeile an
res(i,:)=[f1(1) f2(1)*f2(2)  w1(1)/(2*3.141)]
//End der Schleife
end
//Ausgabe der Matrix mit den Vergleichswerten
Ausgabe=res
//Ende der Funktion
endfunction
```

Anhang 13:Skript für den dritten Testlauf mit realen Messwerten