Entwurf und Umsetzung einer Software zur Echtzeit-Zeit- und Echtzeit-Spektralanalyse von Signalen (Oszilloskop)

Christopher Müller, Fabian Hanitzsch, Adnan Yahya

July 2021

Contents

1 Einleitung

Im täglichen Leben, werden wir Menschen mit unzähligen Signalen konfrontiert, diese können verschiedenste Formen annehmen, z.B. Herzmuskel Bewegungen, elektronische Spannungen beim Fernseher, usw. Um genau solche Signale analysieren zu können, widmen wir uns in dieser Arbeit dem Entwurf und der Umsetzung einer Software zur Echtzeit-Zeit und Echtzeit-Spektralanalyse von Signalen. Das Ziel stellt hierbei die Umsetzung des erlangten Wissens in Form eines Oszilloskopes dar.[2]

2 Grundlagen

2.1 Was ist ein Oszilloskop?

Ein Oszilloskop bezeichnet ein Gerät, welches sich zur Darstellung von Graphen eignet. Die Graphen sind hierbei die Darstellung eines elektrischen Signals. Meistens werden zwei Achsen auf dem Graphen benutzt. Hierbei stellt die Y-Achse meist die Voltage dar, wohingegen die X-Achse die Zeit repräsentiert. Es kann auch vorkommen, dass die Helligkeit und Intensität des Displays, als Z-Achse benutzt wird. Aus dem Graph eines Oszilloskop lassen sich einige Werte ablesen. So zeigt es die Zeit und Voltage Werte eines Signals, aus diesen kann auch die Frequenz und Amplitude eines Signals errechnet werden. Wenn ein Schaltkreis durch ein Signal repräsentiert wird, so sieht man anhand des Signals die "bewegenden" Teile. Es kann auch erkannt werden, ob ein fehlerhaftes Bauteil das Signal stört.[3]

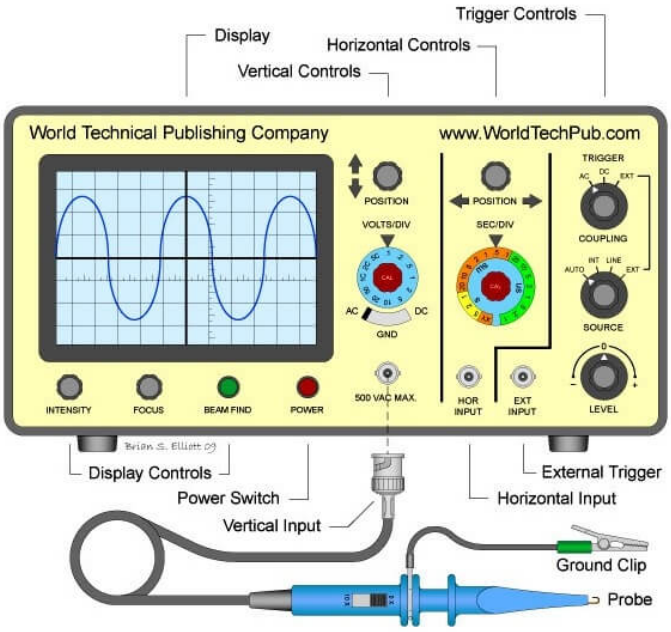

Figure 1: Ein Oszilloskop

2.2 Geschichte des Oszilloskops

Das erste Oszilloskop wurde von dem französischen Physiker André Blondel erfunden. 1893 baute und stellte er das erste elektromechanische Oszilloskop vor. Es war in der Lage Werte von elektrischen Größen, wie zum Beispiel die Wechselstromstärke, zu registrieren. Eine Spule, mit einem daran befestigtem Tintenpendel zeichnete diese Informationen, auf einem sich bewegenden Papierband auf. Diese frühen Oszilloskope verwendeten im Laufe der Arbeit auch mehrere mechanische Geräte, wodurch ihre Messungen sehr ungenau, und ihre Bandbreite gering war. Diese lag zwischen 10 und 19 kHz. Ein großer Entwicklungsschritt, wurde im Jahre 1897 erreicht, als der deutsche Physiker Karl Ferdinand Braun eine Kathodenstrahlröhre (CRT) erfand.[4]

Im Jahr 1932 war die britische Firma A. C. Cossor die erste Firma, die die CRT adaptierte und ihr erstes Oszilloskop präsentierte. Nach dem Ende

des 2. Weltkrieges nahm neben der Entwicklung anderer Messgeräte auch der Fortschritt an der Entwicklung des Oszillokops zu. Das war vorallem in den USA und in Europa spürbar. 1946 gründeten Howard Vollum und Melvin Jack Murdock das Unternehmen Tektronix, welches zu einem führenden Unternehmen in der Oszillographie wurde. Im selben Jahr erfanden sie ihr erstes Oszilloskop mit getriggertem Sweep, das Modell 511, welches eine Bandbreite von 10 MHz besaß. Der getriggerte Sweep ermöglichte zum ersten Mal eine stationäre Anzeige einer sich wiederholenden Wellenform.[4]

In den 1950er Jahren begannen alle technologisch fortgeschrittenen Länder mit der Herstellung von Oszilloskopen, was diese zu einem universellen Messwerkzeug machte. Als zunächst mit der Entwicklung von industriellen analogen Oszilloskope, deren Genauigkeit und Bandbreite zunahmen, folgte 1985 die Erfindung der digitalen Oszilloskope. Walter LeCroy, Gründer der Firma LeCroy, erfand das erste digitale Oszilloskop, für die Forschungseinrichtung CERN. Von da an nahm die Entwicklung digitaler Oszilloskope rasant zu und machte sie bis heute unersetzlich.[4]

3 Terminology

Einige Fachbegriffe müssen bekannt sein, damit die einzelnen Funktionen eines Oszilloskops verständlich sind. Daher werden diese kurz erklärt.

3.1 Signale

Die Hauptaufgabe eines Oszilloskops besteht darin, Signale visuell für den Nutzer darzustellen. Ein Signal ist im Allgemeinen ein Nachrichtenträger mit einer Nachricht. Nachrichten sind dabei physikalische und zeitlich veränderliche Größen. Im Falle des Oszilloskops besteht das Signal aus einer zeitabhängigen Spannung.

3.2 Frequenz und Periode

Die Frequenz gibt an, wie oft sich ein Signal in einer Sekunde wiederholt bzw. in wie vielen Sekunden sich das Signal wiederholt. Dafür wird die Einheit Hertz benutzt, welche durch das Symbol Hz dargestellt wird. Ein Hz entspricht hierbei einer Wiederholung pro Sekunde. Mit der Frequenz verbunden, ist die Periode. Diese wird in Sekunden angegeben und entspricht dem Zeitraum, welcher zwischen den Wiederholungen des Signals ist. Es gilt hierbei: $Periode = \frac{1}{Frequenz}$ und $Frequenz = \frac{1}{Periode}$

3.3 Amplitude

Die Amplitude gibt die Stärke des Signals an. Da unser Signal die Spannung ist, wird diese in Volt(V) angegeben. Die Amplitude ist dabei der Unterschied, zwischen der Ruhelage, welche meist bei 0V liegt, und dem größten Ausschlag, wobei dieser sowohl positiv, als auch negativ sein kann.

3.4 Signalarten

Auch wenn in der Theorie, nach der Definition Signale in allen Formen auftreten können, so nehmen Signale in den meisten Fällen nur wenige gut unterscheidbare Formen an (s. Kapitel 5 Arten von Signalen).

3.5 Samplingmethoden / Messmethoden

Damit ein Signal visuell angezeigt werden kann, müssen zuerst dessen Werte erfasst werden. Um dies zu erreichen existieren verschiedene Methoden. Wir werden hier kurz zwei davon vorstellen und erklären, wie diese funktionieren.

3.5.1 Echtzeit

Die Daten werden in Echtzeit erfasst und gespeichert. Dafür wird in festen Intervallen die Spannung des Signals gemessen. Die Werte zwischen zwei gemessenen Punkten, werden durch Interpolation geschätzt. Um ein Signal

möglichst genau rekonstruieren zu können, sollte das Nyquist - Theorem erfüllt werden. Dieses besagt, dass die Abtastrate mindestens doppelt so groß sein muss, wie die höchste Frequenz des abzutastenden Signals. [7]

3.5.2 Zufällige verschachtelte Messungen

Diese Messmethode verlangt, dass das Signal periodisch wiederkehrt. Es nutzt das Wiederholen des Signals aus, um an leicht zufälligen, noch nicht gemessenen Punkten, Messungen durchzuführen. Dafür werden die Abtastpunkte, bei erneutem Durchlauf des Signals leicht verrückt. Durch die Überlagerungen der Messungen, wird eine hohe effektive Abtastrate erreicht. [7]

3.6 Interpolation

Mit der Hilfe von Interpolation, können fehlende Punkte zwischen zwei Messwerten angezeigt werden, indem die zwei Punkte verbunden werden. Dadurch ist eine Schätzung der fehlenden Punkte vorhanden, welche nicht der Realität entsprechen müssen, da zu diesen Zeitpunkten keiner Abtastung durchgeführt wurde. Eine oft genutzte Form der Interpolation, ist die lineare Interpolation, wobei die Punkte durch gerade Linien miteinander verbunden werden. Dies ist bei einem Sägezahn oder Rechtecksignals vorteilhaft. Bei Wellenförmigen Signalen, wie dem Sinussignal führt dies allerdings dazu, dass der Sinus eckig, statt abgerundet aussieht. Für solche Wellen empfiehlt es sich, die Punkte mit $\frac{sin(x)}{x}$ zu verbinden. Durch diese Methode können Wellensignale besser dargestellt werden, allerdings würde für Rechtecksignale eine höhere Abtastrate benötigt werden, um diese ordentlich darstellen zu können.[5]

4 Funktionsweise

Um zu verstehen wie ein Oszilloskop funktioniert, müssen wir uns die verschiedenen Arten anschauen. Denn es ist zu bedenken, dass Analoge Oszilloskope anders funktionieren wie digitale Oszilloskope.

4.1 Analoge Oszilloskop

Wenn ein analoges Oszilloskop mit einem Schaltkreis oder Signalgeber verbunden wird, so wandert das Signal durch die Steuerelektrode zum Verticalen System des Oszilloskop. Je nachdem wie der Benutzer das Gerät einstellt, wird die Signalstärke entweder verstärkt oder abgeschwächt. Danach nimmt das Signal den direkten weg zu den vertikalen Reflektionsplatten. Dort bewirkt die angelegte Spannung, eine Bewegung eines kleinen Punktes. Eine Positive Spannung bewegt den Punkt nach oben, wogegen eine negative Spannung den Punkt fallen lässt. Gleichzeitig, bewegt dich das Signal auch zu einem Triggersystem, welches eine Horizontale Bewegung auslöst. Das trigern des horizontalen Sytems lässt den Punkt von links nach rechts über das Display bewegen, wobei dies in einer festgelegten Zeit geschieht und somit die Zeitachse repräsentiert. Viele Bewegungen in einer sehr schnellen Sequence lässt die Bewegung auf dem Display als durchgehende Linie auftauchen. Es sind bis zu 500.000 Bewegungen über den Bildschirm jede Sekunde möglich. Gemeinsam bilden die horizontale und verticale Bewegung das Graph eines Signals ab.[3]

Ein Analoges Oziloskop muss meistens über 3 Einstellungsmöglichkeiten verfügen. Eine Voltage Kontrolle um die Amplitude des Signals auf die Messreichweite anzupassen. Die Zeit Angabe, um einzustellen, wie viel Zeit pro angezeigtem Bild vergeht. Dies bestimmt damit auch, wie viel Zeit insgesamt auf dem Bildschirm angezeigt wird. Die letzte notwendige Einstellungsmöglichkeit ist das Triggern des Oszilloskop. Das Trigger Level wird benutzt, um das Oszilloskop für ein wiederholendes Signal zu stabilisieren oder auf ein einzelnes Signal zu reagieren.[3]

Figure 2: Ein analoges Oszilloskop

4.2 Digitale Oszilloskope - Digitale Speicher Oszilloskope

Manche der Digitalen Oszilloskope funktionieren genau wie analoge. Digitale Oszilloskope zeichnen sich durch zusätzliche Daten Processing Systems aus. Diese sammeln Daten über das gesamte Graph und zeigen diese auf einem Display an. Ebenso wie bei den analogen Systemen ist auch hier ein Amplifier die erste Stufe, welche ein Input überwinden muss. Diese Stufe wird auch weiterhin zur Kontrolle der Messreichweite der Amplituden benutzt. Die darauf folgende Stufe dient der Wandelung eines analogen Signals in ein digitales Signal. Diese Aufgabe übernehmen analog-to-digital Converter (ADC). Dieses Bauteil nimmt die konkreten Punkte im Zusammenhang zur Zeit auf und wandelt diese zu Abtastpunkte um. Die Abtastclock des horizontalen Systems gibt die Zeit an, nach welcher der ADC ein neues Sample aufnimmt. Diese Rate wird als Abtastrate bezeichnet. Diese Abtastpunkte werden innerhalb des Speichers, als Datenpunkte gespeichert. Hierbei werden mehrere Abtastpunkte zu einem

Datenpunkte kombiniert. Gemeinsam bilden alle Datenpunkte, eine Wellenaufnahme. Als Aufnahmelänge bezeichnet man die Anzahl an Datenpunkte, welche benutzt werden, um eine Wellenaufnahme zu erstellen. Das Trigger System steuert hierbei den Start- und Stoppunkt der Aufnahme. Sobald die Punkte im Speicher vorhanden sind, hat das Display Zugriff auf diese und kann diese darstellen. Ja nach Modell ermöglicht weiteres Verarbeiten der Datenpunkte, eine weitere Verbesserung der Ausgabe des Signals. Damit würde auch die Möglichkeit der Existenz von Vortriggern entstehen, um somit Ereignisse schon vor dem eigentlichen Trigger sichtbar zumachen. Ein digitales Oszilloskop folgt einer Serial Processing Architektur, damit der Microprocessor, neben der Verarbeitung des Signals auch Displayaktivitäten, als auch die Frontpanel Kontroll Möglichkeiten steuert.[3]

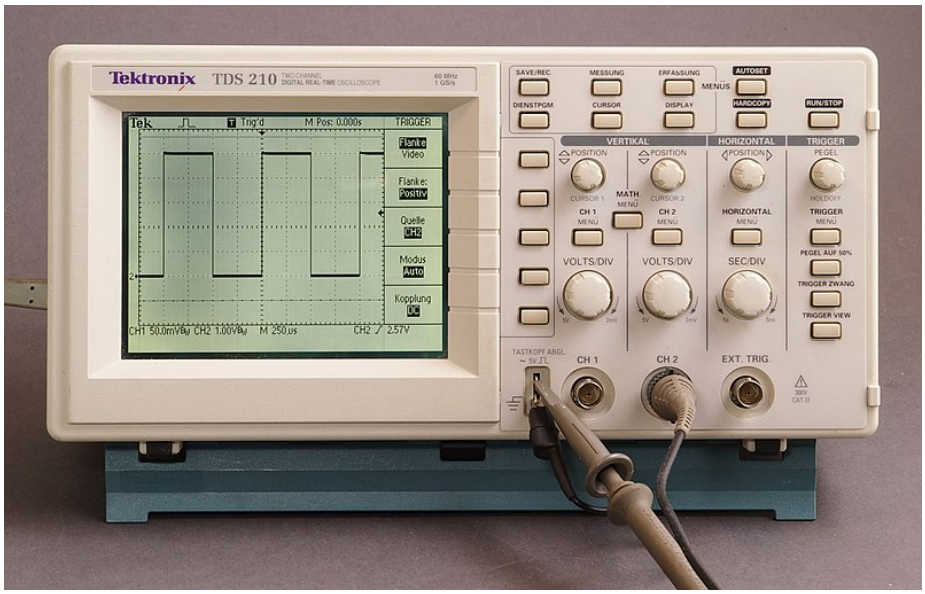

Figure 3: Ein digital Oszilloskop

4.3 Digitale Oszilloskope - Digitale Phosphor Oszilloskope

Digitale Phosphor Oszilloskope(DPO) bringen einen neuen Ansatzpunkt in die Architektur ein. Ähnlich wie bei den analogen Oszilloskopen, ist die er-

ste Verarbeitungsebene ein Verticalverstärker. Die zweite Ebene ist wie bei den Digitalen Oszilloskopen üblich ein ADC. Aber genau nach diesem Bauteil ändert sich der Aufbau. Im Gegensatz zu analogen Geräten, setzt dieses nicht auf chemisches Phospor, sondern auf das sogenannte digitale Phosphor. Dieses stellt eine ständig aktualisierte Datenbank dar. Diese Datenbank enthält eine Zelle an Informationen, für jedes Pixel auf dem Zielbildschirm. Jedes Mal, wenn eine Waveform gecaptured wird, also der Trigger auslöst, wird diese in die Datenbank für das zugehörige Pixel gemappt. Durch mehrfaches Mapping in der selben Zelle, wird eine IntensitätsInformation aufgebaut. Dadurch ist es dem Display möglich auch die Intensität darzustellen. Der Vorteil zu einem analogen Oszilloskop ist die Möglichkeit, variierende Intensitäten durch Kontraste darzustellen. Dadurch lässt sich sehr gut erkennen, ob ein Signal z.B.: bei jeden Trigger auftaucht, oder nur jeden hundertsten. Das DPO benutzt für diesen Zweck eine Parallel-processing-architecture, um den gesamten Darstellungsprozess nicht zu verlangsamen. Der Mikroprozessor, welcher auch hier für Anzeigeverwaltung und Messautomatisierung genutzt wird, befindet sich außerhalb des Anzeigesignalweges, wo er dessen Darstellungsgeschwindigkeit nicht beeinflusst.[3]

Figure 4: ceyear-4456d

5 Arten von Signalen

Signale können verschiedenste Formen annehmen, die geläufigsten und am häufigsten vorkommenden werden in den folgenden Kapiteln behandelt.

5.1 Sinuswellen

Die Sinuswelle ist aus mehren Gründen die Grundwellenform. So variiert z.B die Spannung in der Steckdose in Form einer Sinuswelle. Ein Sonderfall der Sinuswelle, ist die gedämpfte Sinuswelle. Dieser tritt z.B in einer schwingenden Schaltung auf und schwächt mit der Zeit ab. Bei Signalgeneratoren, welche in Oszilloskopen eingebaut sein können, ist es möglich verschiedene Signalarten zu erzeugen, wobei der Sinus in der Regel immer dabei ist.[6]

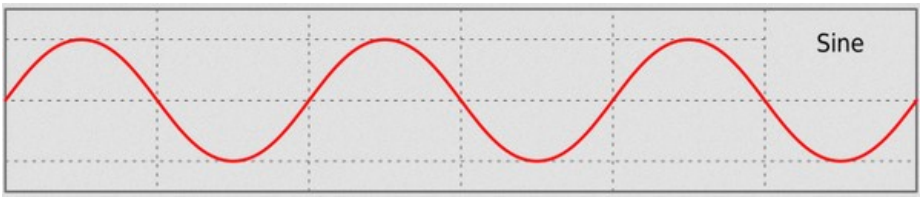

Figure 5: Sinuswellen

5.2 Quadratische oder Rechteckwellen

Die Rechteckwelle ist eine weitere gängige Wellenform. Grundsätzlich ist eine Rechteckwelle, eine Spannung, die sich in regelmäßigen Abständen ein- und ausschaltet (oder hoch und niedrig geht). Es ist eine Standardwelle zum Testen von Verstärkern. Gute Verstärker erhöhen die Amplitude einer Rechteckwelle mit minimaler Verzerrung. Fernseh-, Radio- und Computerschaltkreise verwenden häufig Rechteckwellen für Zeitsteuerungssignale. Die Rechteckwelle ist wie die quadratische Welle, außer dass die hohen und niedrigen Zeitintervalle nicht gleich lang sind. Dies ist besonders wichtig bei der Analyse digitaler Schaltungen.[6]

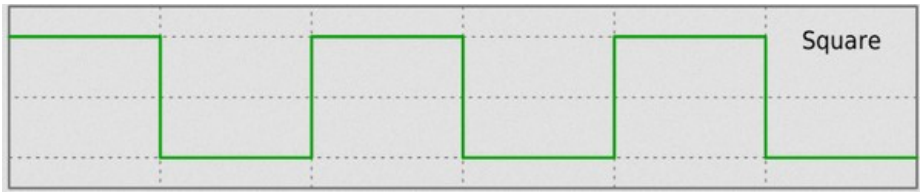

Figure 6: Rechteckwellen

5.3 Sägezahn- oder Dreieckswellen

Sägezahn- und Dreieckswellen resultieren aus Schaltungen, die Spannungen linear steuern, wie zum Beispiel die horizontale Abtastung eines analogen Oszilloskops oder die Rasterabtastung eines Fernsehers. Die Übergänge zwischen den Spannungspegeln dieser Wellen ändert sich mit konstanter Geschwindigkeit. Diese Übergänge werden Rampen genannt.[6]

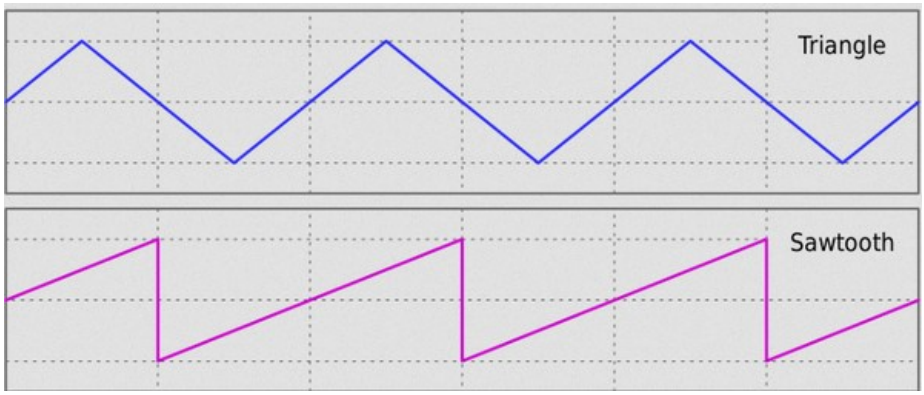

Figure 7: Dreickwellen

5.4 Schritt- und Pulsformen

Signale wie Schritte und Pulse, die selten oder nicht periodisch auftreten, werden als Single-Shot- oder Transient-Signale bezeichnet.

Ein Schritt zeigt eine plötzliche Spannungsänderung an, ähnlich der Spannungsänderung, die Sie sehen, wenn Sie einen Netzschalter einschalten.

Der Impuls hingegen zeigt zwei plötzliche Spannungsänderungen an. Dies ist z.B. der Fall, wenn sie nach dem Einschalten des Netzschalters, diesen wieder ausschalten. Ein Impuls kann ein Informationsbit darstellen, das durch eine Computerschaltung läuft, oder es kann ein Störimpuls oder ein Defekt in einer Schaltung sein. Digitale Komponenten in einem Computer kommunizieren über Impulse miteinander.

Diese Impulse können in Form eines seriellen Datenstroms vorliegen, oder es können mehrere Signalleitungen verwendet werden, um einen Wert in einem parallelen Datenbus darzustellen. Pulse sind auch in Röntgen-, Radar- und Kommunikationsgeräten üblich.[6]

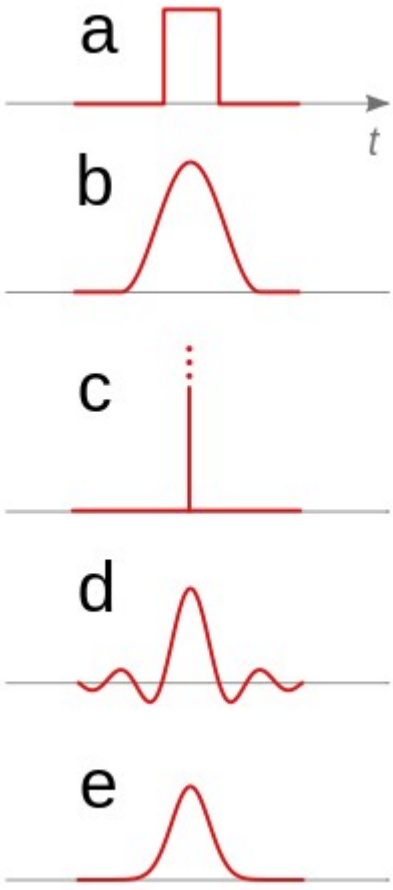

Figure 8: verschiedene Wellenarten

5.5 Periodische und nicht periodische

Sich wiederholende Signale werden als periodische Signale bezeichnet, während sich ständig ändernde Signale als nicht periodische Signale bezeichnet werden. Ein Standbild entspricht einem periodischen Signal, während ein Film einem nicht periodischen Signal entspricht.[6]

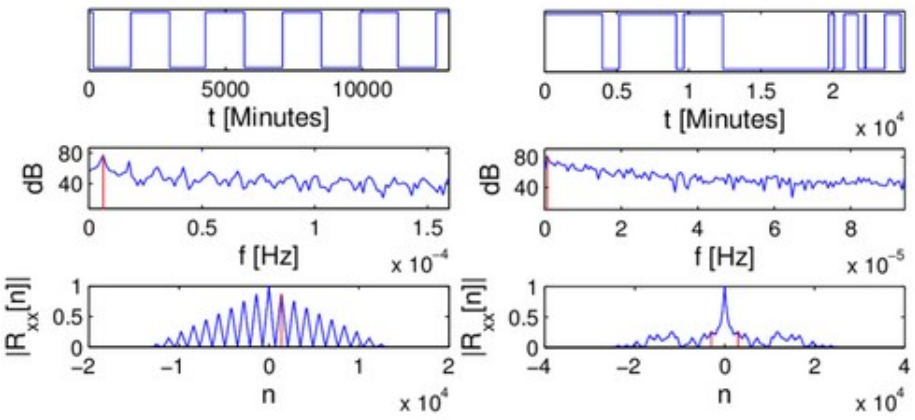

Figure 9: Periodische und nicht periodische Wellen

5.6 Synchrone und asynchrone

Wenn zwischen zwei Signalen eine zeitliche Beziehung besteht, werden diese Signale als synchron bezeichnet. Takt-, Daten- und Adresssignale innerhalb eines Computers sind Beispiele für synchrone Signale.

Asynchrone Signale sind Signale, zwischen denen keine zeitliche Beziehung besteht. Da keine zeitliche Korrelation zwischen dem Berühren einer Taste auf einer Computertastatur und der Uhr im Computer besteht, werden diese Signale als asynchron betrachtet.[6]

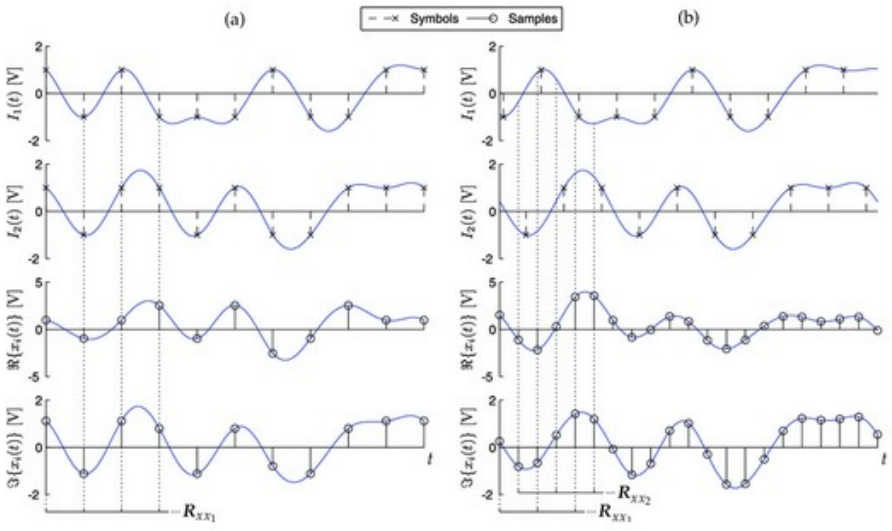

Figure 10: Synchrone und asynchrone Wellen

5.7 Komplexe Wellen

Einige Wellenformen kombinieren die Eigenschaften von Sinus, Rechtecken, Schritten und Impulsen, um komplexe Wellenformen zu erzeugen. Die Signalinformationen können in Form von Amplituden-, Phasen- und/oder Frequenzvariationen eingebettet sein.[6]

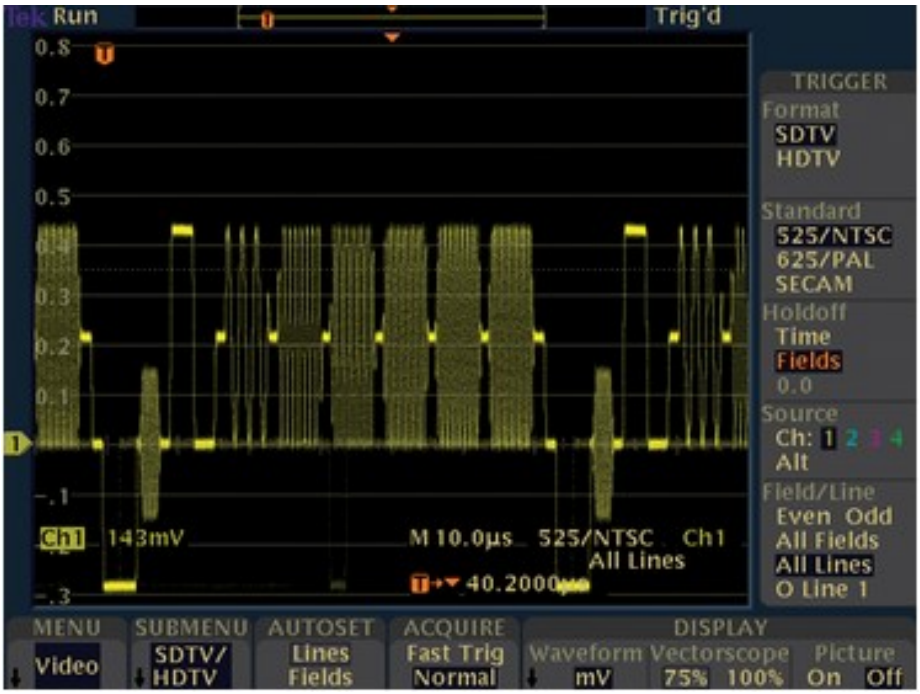

Figure 11: Komplexe Wellen

6 Kalibration

Um sicherzustellen, dass das Oszilloskop genaue Messwerte erhält, sollte es vor der Verwendung kalibriert werden. Die genaue Kalibrierungsmethode variiert von verschiedenen Oszilloskopen und kann in der jeweiligen Bedienungsanleitung nachgeschaut werden. Ein Rechteckwellenmuster wird im Allgemeinen auf dem Bildschirm platziert, um die Amplitude und die Zeitdauer zu kalibrieren. Die Rechteckwelle kann auch verwendet werden, um Verzerrungen zu korrigieren, die durch die Tastköpfe des Oszilloskops verursacht werden.

Spannung, Zeitraum und Wellenform werden vom Oszilloskop als perfekte Referenz für den Kalibrierungsvorgang bereitgestellt. Auf dem Bildschirm erscheint eine Rechteckwelle. Ab diesem Punkt können die entsprechenden Einstellungen vorgenommen werden, damit die Welle auf dem Bildschirm der er-

warteten Referenzwelle entspricht. Die meisten Oszilloskope verwenden eine Rechteckwelle mit 0,5 Volt bei 1000 Hz als Referenz.

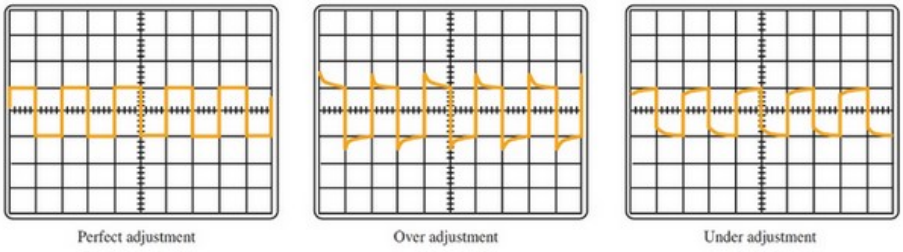

Figure 12: Rechteckwellen adjustments

Oben abgebildet sind drei Beispiele für die Rechteckwelle. Die abgerundeten oder scharfen Ecken des Rechteckwellenmusters weisen darauf hin, dass die Sonde justiert werden muss. [1]

6.1 Intensität und Fokus

Fokus und Intensität steuern das Erscheinungsbild der Welle auf dem CRT-Bildschirm. Der Fokus wird verwendet, um die Welle schärfer erscheinen zu lassen. Es beseitigt jedes unscharfe Aussehen, das die Welle haben kann. Die Intensität steuert die Helligkeit des Lichtstrahls, der auf die Vorderseite des Bildschirms trifft. Die Intensität sollte niemals höher eingestellt werden, als es für eine bequeme Beobachtung des Wellenmusters erforderlich ist. Wenn die Intensität zu hoch eingestellt ist, kann der Bildschirm dauerhaft beschädigt werden. [1]

6.2 Zusatzfunktionen

Heutzutage sind viele Oszilloskope vollständig digital und enthalten einen Computer als Teil des Oszilloskopsystems. Das Wellenmuster kann beobachtet und in einem batteriebetriebenen Computerspeicher gespeichert werden. Viele Zielfernrohre sind auch zum Drucken einer Papierkopie des Wellenmusters aus-

gestattet.

Einige heute verwendete Oszilloskope sind eine Kombination aus VOM(Volt-Ohm-Milliammeter) und Oszilloskop mit einem Bildschirm. Der Bildschirm zeigt nicht nur das Wellenmuster an, sondern zeigt auch digital die Frequenz- und Spannungswerte.

Oszilloskope können auch direkt an einen PC angeschlossen werden. Dies wird als Datenerfassung und -übertragung bezeichnet. Diese Anwendung erfordert normalerweise eine Erweiterungskarte für den Computer, Software zum Ausführen des Programms und ein Schnittstellenkabel zum Anschließen des Oszilloskops an den Computer. Diese Peripheriegeräte werden oft mit dem Umfang mitgeliefert.

Ein auf diese Weise verwendeter PC zeigt die Wellenmuster, Spannung und Frequenz der Wellenform an. Die Informationen können direkt in Dokumente zur Verwendung in Berichten oder Schulungshandbüchern heruntergeladen werden. [1]

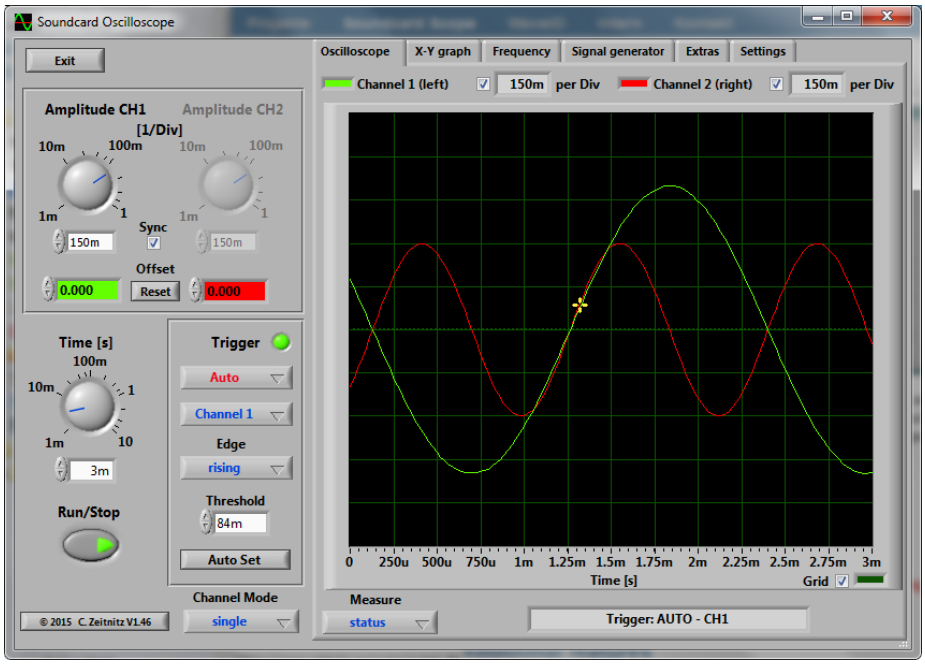

Figure 13: Soundcard Oscilloscope

7 Anforderungsanalyse

7.1 Sinn und Zweck

Unser Oszilloskop soll zu der Analyse von hörbaren Signalen nutzbar sein. Dazu sollte es Signale von 20-20.000 Hz bearbeiten und darstellen können. Durch Nyquist-Theorem ist bekannt, dass mindestens 40.000 Abtastpunkte pro Sekunde benötigt werden. Wir orientieren uns dabei an die üblich 44.100, die häufig von Audiosystem bereits genutzt werden.

7.2 funktionale Anforderungen

funktionale Anforderungen bezeichnen in unseren Fall Anforderungen, welche wir an das Projekt stellen, um dieses als abgeschlossen bezeichnen zu können. Bei den Einstellungen und Verhalten wurde sich an das Programm Soundcard

Scope [8] orientiert. Eine Anforderung besteht darin, dass der Nutzer die Amplitude und Frequenz des Signals einstellen kann. Unser Ziel besteht dabei Frequenzen bis zu 20.000 Hz darzustellen. Dieses Oberlimit wurde festgelegt, da wir somit die hörbaren Frequenzen darstellen können. Zudem sollen 44.100 Abtastpunkte pro Sekunde berechnet werden, um Standard Aufnahme Qualität vorweisen zu können. Als untere Grenze wurde 1Hz gewählt, sodass auch längere Aufnahmen mit ablesbaren Signalen möglich sind. Das Ziel hierbei war bis zu zehn Sekunden Aufnahmen darzustellen. Dazu muss die zeitliche Darstellung des Geräts anpassbar sein, sodass eingestellt werden kann, wieviele Sekunden ein einzelnes Feld ist. Der Plan liegt dabei, dass in der X-Richtung zehn Felder und in der Y-Richtung elf Felder dargestellt werden.

Wie groß ein Signal auf dem Bildschirm dargestellt wird, sollte ebenfalls durch den User einstellbar sein, hierfür wird die Einstellung der Voltage verwendet. Zur besseren Analyse, soll die Möglichkeit des Verschiebens des Graphens in Y-Richtung sorgen. Ebenso sollte die Position des Triggers sowohl in x als auch y Richtung einstellbar sein. Wie bei einem echten Oszilloskop, besteht auch bei unserer Umsetzung die Möglichkeit das Signal zu invertieren. Die Dartsellung der Frequenzanalyse, soll Einstellungsmöglichkeiten für Start und Endpunkt der angezeigten Frequenzen bieten.

7.3 nicht funktionale Anforderungen

nicht funktionale Anforderungen, bezeichnen in diesem Projekt Eigenschaften, welche nicht direkt, etwa mit den Funktionalitäten, zu tun haben und eher der Usability dienen. Sollten diese nicht erfüllt werden, so kann das Projekt weiterhin als Erfolg angesehen werden. Unsere Umsetzung soll vom Interface an ein echtes Oszilloskop erinnern, und dem User eine einfache Möglichkeit geben, verschiedenste Einstellungen zu treffen. Für diesen Zweck sollen Drehregler(Dials) eingesetzt werden, hier sollte der User ohne große Anstrengungen die Möglichkeit haben, den von ihm gewünschten Wert einzustellen, solange dieser in dem von uns festgelegten Bereich liegt. Eine große Bereicherung würde die

gleichzeitige Darstellung der Frequenzanalyse und des originalen Signals bieten. Am besten würde sich hier eine Darstellung nebeneinander anbieten. Natürlich sollte jeder User unser Programm benutzen können, unabhängig des von ihm verwendeten Betriebssystems. Ein großer und auch wichtiger Punkt, wäre es die Signale so darzustellen, dass Daten gut ablesbar sind.

8 Technische Umsetzung

Um unsere Version eines Oszilloskops umzusetzen, haben wir die folgenden Vorbetrachtungen auszuführen: Woher soll unser Signal bezogen werden?, Welche Einstellungen soll der User treffen und welche wiederum sind fest im Code verankert? Welcher Algorithmus wird gewählt und wieso?, Wie sehr achten wir auf die Performance?

8.1 Verwendete Bibliotheken

Zur Darstellung und Berechnungen wurden Bibliotheken zur Hilfe genommen. Diese werden später genauer erklärt, fürs erste werden sie nur kurz in ihre Kategorien eingeordnet. Für die Darstellung der Daten wurde PyQt5, sys und matplotlib genutzt (mehr dazu in Kapitel 8.6). Für die Berechnungen wurde numpy, math und scipy genutzt. Für das zeitgleichen sammeln mehrerer Daten wurde threading, time und pyaudio genutzt.

8.2 Wie werden die Daten dargestellt

Zur Darstellung der Daten wurde das Animationsmodul von matplotlib genutzt. Mit diesem wurde erreicht, dass unser Graph mehrmals die Sekunde geupdatet wird. Er wurde so eingestellt, dass er 33ms nach dem er fertig ist, den Graphen zu updaten, die Funktion erneut aufruft. Dies bedeutet nicht, dass die Funktion alle 33ms aufgerufen wird, sondern dass die Zeit zwischen den Aufrufen den 33ms plus die Zeit für die Berechnung beträgt. Dies führt zu einer weitaus geringeren effektiven fps, als die theoretischen 30. Die Zeit zwis-

chen Aufrufen befand sich beim Testen meistens bei ca. 40 ms, was einer fps von 25 entspricht. Selbst in den Fällen, wo diese Zahl weiter sinkt ist es weniger problematisch, da zum Ablesen von Werten in der Regel mehr Zeit benötigt wird, weswegen der Nutzer keine Interessante Daten übersehen wird, weil das Oszilloskop zu langsam geupdatet hat. Die Zeit wurde auf 33ms statt 16ms oder geringer eingestellt, um dem Problem vom unruhigem Signal, auch wenn nur geringfügig, entgegenzuwirken (s. Kapitel 8.8.2 wackelndes Signal)

Die wiederholende Funktion wurde neben der Darstellung des Graphen dazu genutzt, die aufgenommenen Daten zu verarbeiten und zu ordnen. Dazu werden die aufgenommenen Signale vom Mikrofon und Signalgenerator entgegengenommen und die jeweiligen Listen zur Speicherung entleert (s. Kapitel 8.3 Woher soll unser Signal bezogen werden?). Danach wird für jeden Signalpunkt vom kleineren Datensatz jeder Punkt je nach Einstellung des Nutzers der neuste Datenpunkt berechnet. So ist der neueste Punkt die Summe vom Signal der Signalgenerator, Mikrofons und dem eingestellten Offset in Y-Richtung (s. Kaptiel 8.6.5 Verschiebung in y-Richtung), wenn der Nutzer Mikrofon und Signalgenerator aktiviert hat. Nachdem der neue Punkt berechnet wurde, wird dessen Vorzeichen evtl. umgedreht, falls der Nutzer Signal invertieren aktiviert hat. Dieser Datenpunkt wird dann in einem kleinen, temporären Array gespeichert, welches später in das Array für gespeicherte Daten integriert wird (s. Kapitel 8.7 Wie sehr achten wir auf die Performance?).

Nachdem das Signal abgespeichert wurde, wird gecheckt, ob durch das neue Signal der Trigger aktiviert wird. Da der Trigger auf Kantenerkennung basiert, werden immer mindestens zwei Punkte pro Anzeige benötigt, um diesen zu triggern. Der Trigger wurde so implementiert, dass zuerst gecheckt wird, ob der Wert unter dem Schwellenwert liegt. Falls dies der Fall ist, wird sich gemerkt, dass ein Punkt unter dem Schwellenwert existiert. Beim nächsten Punkt wird dann geschaut, ob der über dem Schwellenwert liegt, falls ein Punkt unter dem Schwellenwert bekannt ist. Da nun bekannt ist, dass ein Punkt unter

dem Schwellenwert existiert und der neueste Punkt über dem Schwellenwert liegt, ist bekannt, dass zwischen den beiden neuesten Punkten der Schwellenwert überschritten wurde (positive/steigende Kante). Falls das Signal invertiert wurde, wird zuerst nach einem größeren Punkt und danach nach einem kleineren Punkt gesucht (negative/fallende Kante). Nachdem der Trigger sich aktiviert hat, wird für jeden neuen Datenpunkt (startend mit dem Trigger Auslöser) ein Integer, welcher in Abhängigkeit zur X-Position des Triggers steht, um eins erhöht. Sobald der Integer größer, oder gleich der Länge der angezeigten Daten entspricht, bedeutet dies, dass der angezeigte Graph geupdatet werden muss. Dafür werden die Daten aus dem temporären Array mit den neuen Daten in das Array gespeichert, wo ein Überblick über die zuletzt aufgenommenen Daten sich befinden. Dieses Array wird sowohl für die Darstellung der Daten verwendet, sowie als Vorlage für das Graph zur Frequenzanalyse genutzt. Es wird daraufhin der vorhin erwähnte Integer auf die Triggerposition gesetzt. Diese berechnet sich folgendermaßen: int(TriggerXOffset * totalTime * Abtastrate).

Es folgt ein Update von Variablen, um festzustellen, dass der Graph in diesem Durchlauf neue Zahlen darzustellen hat, sowie um festzustellen, ob der Bildschirm bereits einmal mit Daten gefüllt wurde. Nachdem das temporäre Array mit den neuen Daten nämlich ins Array mit gespeicherten Daten integriert wurde, wird je nachdem ob der Bildschirm schon einmal gefüllt wurde andere Daten angezeigt. Wenn der Bildschirm noch nicht gefüllt wurde, dann werden alle derzeitig gespeicherten Daten angezeigt. Das zeigt sich Visuell darin, dass sich der Graph von rechts nach links mit Daten füllt. Erst wenn der Graph das erste Mal voll gefüllt wurde, werden die Daten nur noch abhängig von der Triggerposition angezeigt und auch nur jedes mal, wenn genügend Datenpunkte vorhanden sind, um vom Triggerpunkt bis zum Ende des Graphens zu füllen. Diese Methode wurde bevorzugt im Vergleich zum sofortigem Anzeigen von Daten nach dem triggern des Triggers. Die Anzeige der Frequenzanalyse ist im Vergleich weitaus einfacher. Sie hat wie der Graph zur Signal-Wiedergabe auch eine Funktion, die sich 33ms nach Abschluss ihrer Aufgabe erneut ausgeführt.

Diese Funktion ist allerdings weit aus kürzer, da alle benötigten Daten vom anderen Graphen abgerufen werden können. Es werden die derzeit angezeigte Daten genommen, je nachdem ob der Graph bereits einmal gefüllt wurde, ist dies entweder das Array mit allen Daten, oder das Array mit den angezeigten Daten. Diese Daten werden mittels scipys fft verarbeitet und auf den Graphen dargestellt. Daraus folgt, dass die Frequenzanalyse immer nur die derzeit angezeigten Werte anzeigt und daher bei sehr kleinen Zeiteinstellungen eventuell nicht die richtige Frequenz herausbekommen kann. Für die Anzeige der Daten wird dabei die X-Axis auf den durchs Dial angegebene Startwert und Stoppwert beschränkt. Die Y-Achse wird dabei durch die Angabe der Voltage pro Feld vom Voltage-Zeit Graphen beeinflusst.

8.3 Woher soll unser Signal bezogen werden?

Als Signale für unser Oszilloskop, war die Möglichkeit der Liveaufnahme, das Auslesen aus einem Soundfile, sowie ein Soundgenerator geplant. Von diesen wurde das lesen aus einer Soundfile aus zeitlichen Gründen nicht geschafft. Der Soundgenerator, soll aus durch den Nutzer eingestellten Werten für die Amplitude und Frequenz, sowie der Signalart, ein Signal erzeugen und direkt an das Oszilloskop übergeben. Als Signalarten wurde der Sinus, die Sägezahnwelle und Rechteckwelle gewählt. Die Liveaufnahme des Mikrofons wurde mittels pyaudio erreicht.

Es kann die Addition der beiden Signale angezeigt werden. Dies machte bei der Implementierung allerdings einige Probleme, da auch bei gleicher Abtastrate es beim Multithreading nicht garantiert ist, dass bei beiden Signalen immer die selbe Anzahl an Zahlenwerten vorhanden sind. Bei einer Messung der Durschnittlichen Anzahl an Werten pro Sekunde, kam beim Signalgenerator ca. 44110 heraus, während bei der Live Aufnahme es ca. 44090 waren.

Dieses Problem wurde gelöst, indem bei dem Signalgenerator die überflüssigen Zahlen verworfen wurden, und sein Abtastpunkt neu eingestellt wurde. Bei der Live Aufnahme hingegen wurden die überflüssigen Daten behalten, und verar-

beitet, sobald der Signalgenerator hinterhergekommen ist. Dies führt zwar zu einer Verzögerung von Daten, aber da beide Signale die chunks der Daten 60 mal die Sekunde geben, beträgt das entstandene Delay ebenfalls nur bei 1/60 Sekunden, bzw. bei der Update-Geschwindigkeit des Graphens, da diese langsamer ist.

8.4 Der Signalgenerator

Signalgenerator.py enthält die Klasse SignalGenerator, diese dient der Generation von verschiedensten Signalen. Hierfür wurden die Biblitotheken scipy, und numpy importiert. Um die verschiedensten Signale zu erzeugen, bedienen wir uns der Bibliothek scipy. Diese enthält zum Beispiel die Funktion square() welche genutzt wird, um eine Rechteckwelle zu erzeugen. Diese hat dann eine Periode von 2*pi, mit einer Amplitude von 1. Zusätzlich wird nicht nur eine bestimmte Anzahl an Werten erzeugt, sondern eine unendliche Menge an harmonischen Schwingungen. Gleiches gilt auch für die sawtooth() Funktion, lediglich der Sinus wird mithilfe von Numpy erzeugt. Um somit eine beliebige Welle zu erzeugen, also auf User eingaben zu reagieren, ergibt sich der folgende Algorithmus:

```
def generate_time_vector(self, start, end, amount):
    self.time_vector = np.linspace(start=start, stop=end, num=amount)

def generate_sinus(self):
    Sinus = self.amplitude*np.sin(2*np.pi* self.frequency * self.time_vector)
    self.Signal = Sinus
    return(Sinus)
```

Figure 14: Zur Generierung des Sinus mittels numpy

Wobei start, stop und amount vom Thread zur Signalgenerierung beim Oszilloskop übergeben wird. Somit wird sowohl die Amplitude, als auch die Frequenz mit in die Berechnung eingebunden. Zusätzlich enthält scipy.signal auch die Möglichkeit der Erzeugung eines Chirps, welches ein Signal darstellt, das über die Zeit abnimmt, Erzeugung eines Gausspulses sowie die Möglichkeit der

Erzeugung eines Einheitsimpulses. Da hierfür die zu übergebenden Variablen, auch der Start und Stopwerte, also zu welchem Zeitpunkt soll das Signal beginnen, und wann soll es enden enthält, haben wir diese Signale nicht als für den User erzeugbar eingebaut.

8.5 Welche Einstellungen soll der User treffen und welche wiederum sind fest im Code verankert?

Die Einstellungen, welche der User treffen kann, sollten auch bei unserer Umsetzung sehr nah an einem echten Oszilloskop gehalten sein. Hierzu benötigen wir Einstellmöglichkeiten für das Zeitintervall, die Voltage als auch den Trigger. Um Signale erzeugen zu können, müssen zusätzlich Amplitude und Frequenz einstellbar sein. Da wir ebenfalls planen die Frequenzanalyse einzubauen, benötigen wir dort die Start und Stop Position der Frequenz. Um den Graphen besser analysieren zu können, lassen sich auch noch X und Y Position prozentual im Fenster bewegen. Hierzu sollten die, Werte für Frequenz, Zeit, FrequenzStart und FrequenzStop exponential übergeben werden. Fürs bessere einstellen wurde festgelegt, dass nur die Zahlen die mit 1,2 oder 5 anfangen von Interesse sind. dementsprechend wurde auch die Einteilung der möglichen Einstellungen der Dials vorgenommen.

8.6 Welcher Algorithmus wird gewählt und wieso?

8.6.1 Zeit pro Feld

Durch die Einstellung der Zeit, wird eine unterschiedliche Anzahl an Werten die angezeigt werden benötigt. So werden bei insgesamt 1s von Anzeige 44100 Werte benötigt, während bei z.B. 0.1s Anzeige nur noch ein Zehntel davon benötigt wird. Hierfür gibt es zwei Optionen diese Werte ordentlich anzuzeigen. Bei der ersten wird am Anfang die Länge der anzugebenden Werte, womit die maximale Anzeige Länge auch abgedeckt werden kann. Bei dieser Methode wird dann bei kleineren Zeitachsen nur ein bestimmter Bereich der gespe-

icherten Daten angezeigt. Bei der zweiten Methode wird beim Ändern der Zeitachse die Arrays mit gespeicherten Daten neu erstellt, ohne auf die bisherigen aufgenommenen Werte zu achten. Dies hat als Resultat, dass die alten Werte gelöscht werden und neue aufgenommen werden müssen. Auch wenn die erste Variante die Daten beibehält, so wurde sich für die zweite Variante entschieden, da diese einfacher zu implementieren schien. Bei der Änderung der Zeiteinstellung wird die X-Position des Triggers auf die neue Zeiteinteilung mit angepasst (s. Kapitel 8.6.4 Verschiebung in x-Richtung)

8.6.2 Voltage pro Feld

Mittels der Voltage pro Feld Einstellung kann die Anzeige angepasst werden. Wenn z.B. 0.1V/Feld angegeben wird, dann ist die Y-Achse zwischen -0.5V und 0.5V. Durch Änderung dieser Angabe, kann der dargestellte Graph gestreckt, oder gekürzt werden, womit der Nutzer die Werte besser ablesen kann. Es werden hierbei Einstellungen von 0.1V/Feld bis 1V/Feld ermöglicht.

8.6.3 Triggerwert

Mit dem Triggerwert wird der Wert angegeben, welcher durchquert werden muss, damit der Graph geupdatet wird. Dieser Wert wird in Prozenten angegeben, womit man die Höhe einstellt, abhängig von der derzeitigen Y-Achse Einstellung. Der Triggerwert ist dann auch visuell bei der selben Höhe, wenn sich die Y-Achse durch die Voltage pro Feld Einstellung ändert. Diese Einstellung erlaubt es zwar gut den Triggerwert visuell anzugeben, kann allerdings zu Verwirrung führen, wenn die Voltage Anzeige geändert wird und keine Werte mehr getriggert werden. Es wurde trotzdem diese Methode verwendet, da eine Vielzahl von verschiedenen Trigger Einstellungen leicht einzustellen sind.

8.6.4 Verschiebung in x-Richtung

Die x-Position des Graphen wird durch eine Verschiebung der Trigger Position erreicht. Dies führt bei periodischen Signalen dazu, dass diese Visuell

verschoben werden.

Das Verschieben der Trigger Position führt zu dem, dass je nach Position mehr oder weniger pre Trigger Daten angezeigt werden. Da wir den Trigger nur dann immer resetten, wenn der Graph mit neuen Daten gefüllt wurden, entstehen hauptsächlich zwei Methoden die post Trigger Daten anzuzeigen. Die erste ist sofortiges Anzeigen, auch wenn noch nicht genügend Daten vorhanden sind, um den Graphen zu füllen. Diese Anzeige führt dazu, dass die Pre-Trigger Daten jedes Mal sofort vorhanden sind, während die Post Trigger Daten noch "geladen" werden müssen. Dies kann zudem dazu führen, dass entweder falsche/veraltete Daten angezeigt werden, oder diese im vorhinein gelöscht wurden, weswegen Daten nah am Trigger länger zu sehen sind als Daten am Ende der Anzeige, da diese schnell wieder entfernt werden.

Aus diesen Gründen wurde sich für die zweite Variante entschieden. Bei dieser wird der Graph nur geupdatet, wenn genügend Daten vorliegen. Bei längeren Aufnahmen, Zeiteinstellungen und frühen Trigger Positionen, führt dies dazu, dass das Updaten des Bildschirms etwas länger dauert.

8.6.5 Verschiebung in y-Richtung

Die Verschiebung Die Verschiebung des Graphen in Y-Richtung funktioniert, indem das Signal abhängig von der Verschiebung verstärkt, bzw. abgeschwächt wird. Damit kann der Nutzer sich das Signal zu Recht schieben, sodass er die Werte besser ablesen kann. Der Schwellwert für den Trigger bewegt sich dabei nicht mit, weswegen ein Signal nach der Verschiebung evtl. nicht mehr geupdatet wird. Dies kann bei standard Signale die durch den Nullpunkt gehen nerven, erlaubt aber auch eine bessere Analyse von Signalen, die als durchschnittswert z.B. 0,2V haben, statt die üblichen 0V.

8.7 Wie sehr achten wir auf die Performance?

Auch wenn die Performance anfangs von geringer Interesse war, so mussten an verschiedenen Stellen die Algorithmen verbessert werden, sodass das Programm

eine ordentliche Bedienbarkeit vorweisen kann und nicht bei der Änderungen einiger Einstellungen sehr langsam reagiert.

Eine der Änderungen ist, dass die Signale vom Signalgenerator ursprünglich für jeden Punkt einzeln berechnet wurden. Dies funktionierte zwar bei numpy.sin() sehr gut, führte allerdings bei scipy.sawtooth() dazu, dass der Thread zur Generierung des Signals zu langsam war, um den Load stand zu halten.

Eine weitere Änderung war, dass ursprünglich jeder neue Wert sofort bei dem Array den gespeicherten Werten angehangen wurde, und der älteste Wert gelöscht wurde. Da das löschen aber mittels pop(0) durchgeführt wurde, war dies eine Aktion, die linear mit der Länge der Liste skalierte. Dies führte dazu, dass bei der Anzeige von längeren Zeiten, wie z.B. 1 Sekunde/div, dass Programm nicht mehr reagierte, da für jeden neuen Wert ca. 441.000 Werte verschoben werden mussten.

Stattdessen wird bei jeder Aktualisierung dem Bildschirm die Werte in einem kleinen Array gespeichert und dieses je nach Situation ins Array mit allen Daten gespeichert (s. Kapitel 8.2 Wie werden die Daten dargestellt).

8.8 Ungelöste Probleme

Bei der Erstellung des Oszilloskop sind neben den gelösten Probleme auch ungelöste Probleme entstanden. Diese Probleme reichen von nicht implementierten Zusatzfunktionen, bis zur Einschränkung der Benutzbarkeit des Oszilloskops. Es wurde versucht die größeren Probleme zu lösen, allerdings war dies Aufgrund mangelnder Zeit und fehlgeschlagenen Lösungsansätze nicht gelungen.

8.8.1 Abtasten von Signalwerten in Real-Time

Es wurde versucht, das Signal mittels eines Threads und der time Bibliothek in Python in Echtzeit abzutasten. Damit sollten Signale nicht zu bestimmten Intervalle erstellt, sondern abgelesen werden mit minimalen Unterschiede der Zeitabstände. Dies sollte das Abtasten von pyhsischen Oszilloskopen simulieren und das erzeugte Signal mit der angegebenen Abtastrate authentischer machen.

Dies war allerdings letzten Endes nicht möglich. Es war zwar möglich den Thread so vorzubereiten, dass er in Theorie immer einen Wert in bestimmten Abständen misst, so ist dies in der Praxis aber nicht umsetzbar. Der Grund dafür liegt darin, dass wir einen Thread nur für ein Minimum von 2ms schlafen legen konnten. in dieser Zeit entstanden dann keine Messungen mehr und die gewünschte Abtastrate konnte nicht erreicht werden. Diese Funktion wurde daraufhin nicht weiter verfolgt, da es einerseits nicht zu den Anforderungen gehörte und andererseits uns keine Methode bekannt war, das Abtasten in Microsekunden einzuplanen.

8.8.2 wackelndes Signal

Bei hohen Frequenzen wird das Signal unruhig und bewegt sich wie verrückt, womit das Ablesen von Werten erschwert wird, bzw. nicht mehr möglich ist. Dieses Problem entsteht dadurch, dass unsere Triggerfunktion nur schaut, ob die Schwelle mit dem letzten Datenpunkt überschritten wurde. Bei vielen Datenpunkten ist diese herangehensweise auch in Ordnung, nur führt das bei großen visuellen Abstände dazu, dass die Triggerschwelle nicht genau an dem Triggerpunkt überschritten wurde. Eine Lösung zum Problem wäre, zu berechnen an welcher Stelle zwischen den beiden Punkte die Schwelle überschritten wird. Sobald dieser Punkt berechnet wurde, wird der Graph um den entsprechenden Wert verschoben. Es werden zudem die Markierungen auf der X-Achse angepasst, sodass es für den Nutzer aussieht, als ob keine Verschiebung stattfand.

Auch wenn eine Lösungsmöglichkeit somit bekannt war, konnte diese nicht umgesetzt werden. Dies ist eine Mischung aus Zeitmangel, sowie dass bei der Suche nach der richtigen Triggerposition die erwartete Formel

$$\text{new}_x = old_x - \frac{(F(oldX) - triggerPoint) * delta(X)}{(F(oldX) - F(oldX - 1)}$$

$$= \text{start-position - } \frac{distance}{anstieg}$$

nicht funktioniert hatte. Sie führte bei nicht extrem Werten zu Ungenauigkeiten bei ca. 1/3 der Bildschirmupdates. Bei der Untersuchung dieses Problem wurde aber festgestellt, dass bei geringen Zeit-Einstellungen die dargestellte Samplerate nicht den angegebenen 44100 Hz entsprachen, sondern teilweise bis auf 30.000 Hz herunterging, was daraufhin behoben wurde.

8.8.3 Daten werden verworfen, wenn Zeit-Achse verändert wird

Auch wenn dies bei Soundcard Scope [8] während einer Live-Aufnahme ebenfalls so implementiert wurde, so bestand wenigstens die Möglichkeit die Aufnahme zu unterbrechen und im Nachhinein die Signale analysieren zu können. Das Problem war frühzeitig bekannt, allerdings wurde es von geringer Interesse eingestuft. Dies ist auch der Grund weswegen wir zu diesem Problem keine Lösungsansätze haben, obwohl das Problem bekannt ist.

8.8.4 Programm kann bei manchen Einstellungen sehr langsam werden

Es existieren einige Einstellungskombinationen, welche dazu führen, dass das Programm nur noch ca.10fps anzeigen kann. Diese Einstellungen sind aber Einstellungen, womit ordentliche Messungen sowieso nicht unternommen werden können. So wird viel Zeit für die Darstellung benötigt, wenn ein langes Zeitfenster bei einer hohen Frequenz eingestellt wird. Dies führt dazu, dass hundert tausende von Datenpunkte zeitgleich von Matplotlib angezeigt werden. Dies scheint es zudem noch weiter zu verlangsamen, wenn die Signalstärke so ausgewählt wird, dass es den gesamten Graphen visuell einnimmt.

Eine Lösung zu dem Problem wäre, bei längeren Aufnahmen den Durchschnitt von mehreren Datenpunkten zu nehmen. So geht zwar die Genauigkeit der Signale herunter, aber da wir die Signale sowieso nicht speichern, wäre der Verlust dieser Genauigkeit von geringer Interesse.

8.9 Wie soll das Userinterface entstehen?

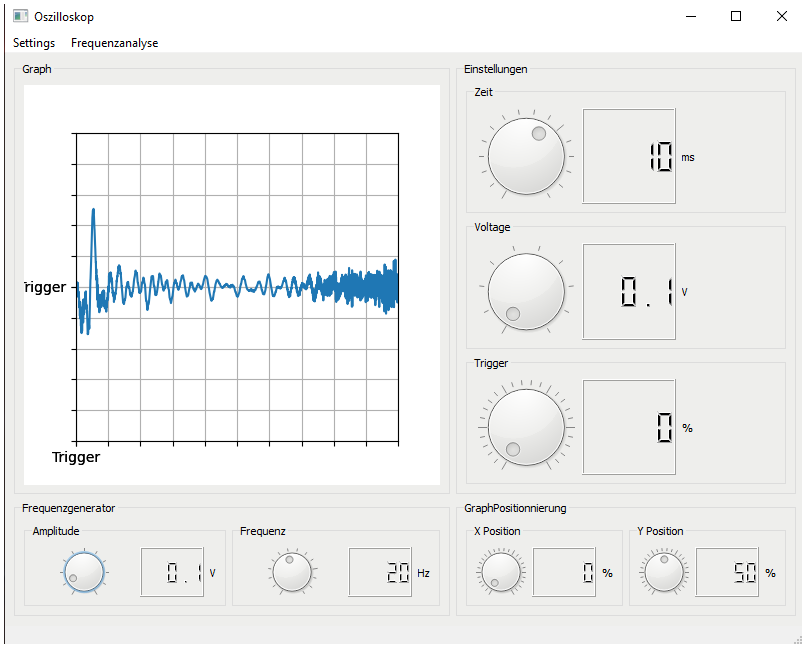

Figure 15: Ein Standbild zu der Live Aufnahme mittels des Mikrofons

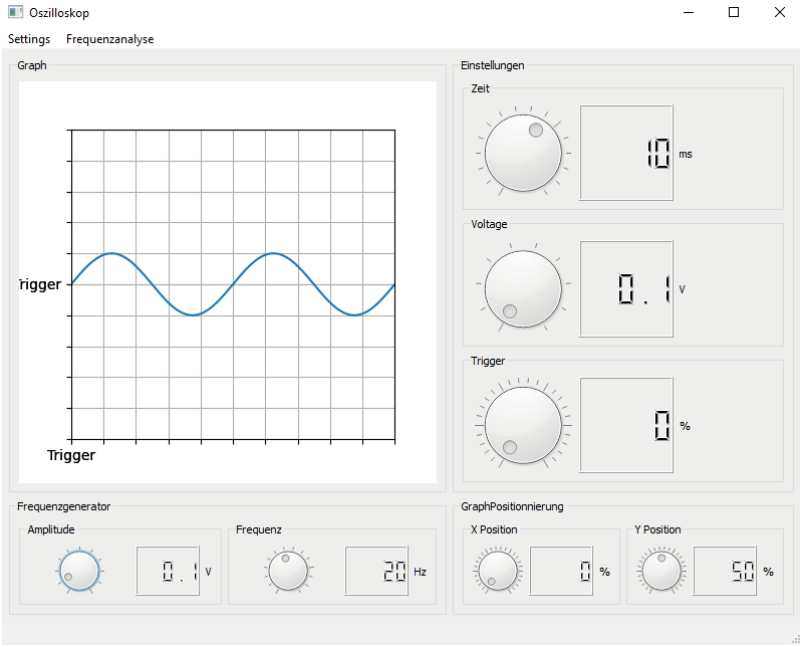

Figure 16: Aussehen des Programms, nachdem es geöffnet wurde mit Standard Einstellungen

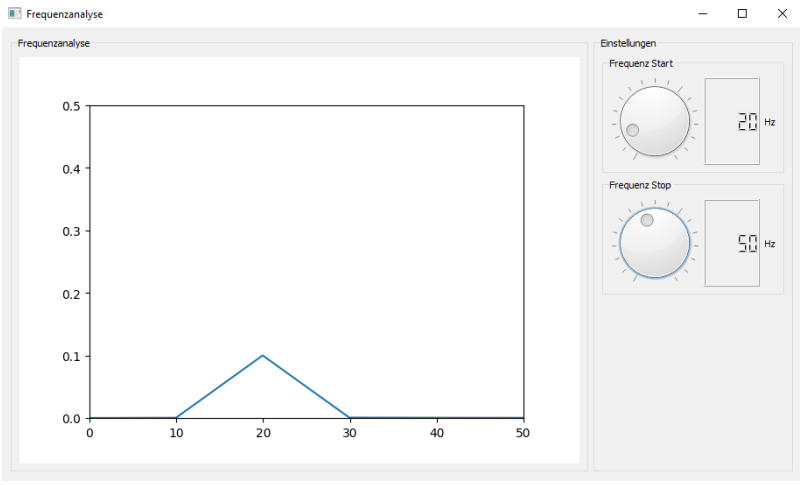

Figure 17: Darstellung der Frequenzanalyse in seinem eigenem Bildschirm

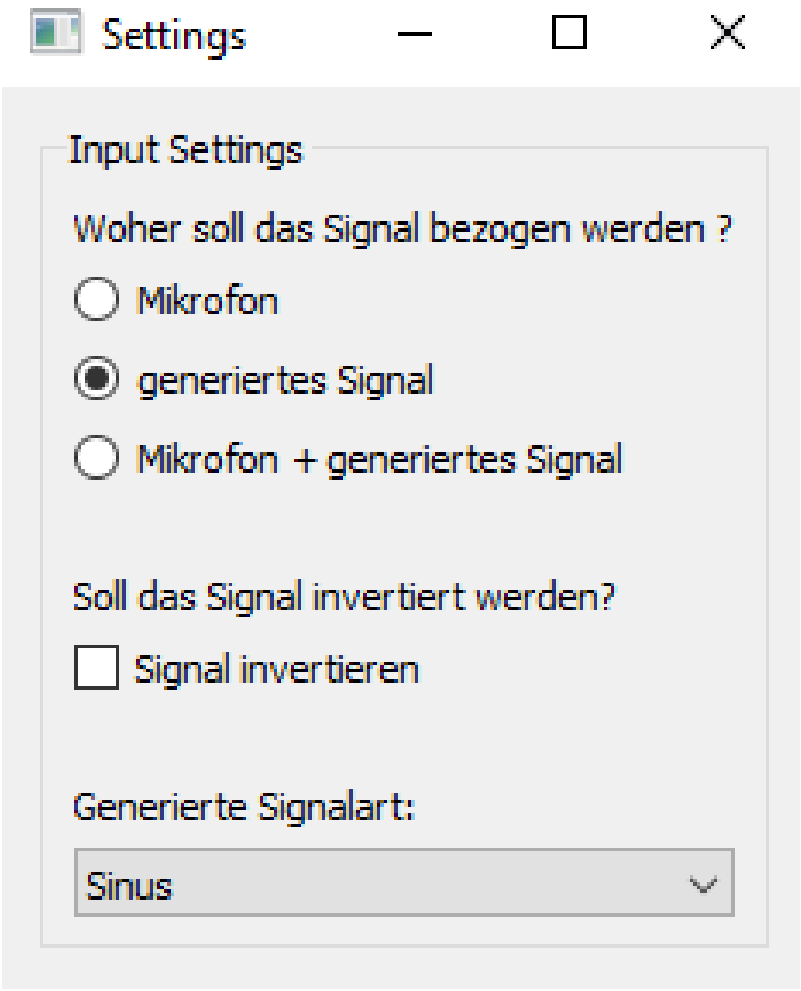

Figure 18: Erweiterte Settings im extra Menü

Das UserInterface ist die Verbindung des Users zu den Funktionalitäten unseres Programms. Die ursprüngliche Idee sah eine Verwendung der Bibliothek Kivy, bzw KivyMD vor, um so eine modern anmutende, möglicherweise sogar Mobil laufende Version eines Oszilloskops zu schaffen. Da wir hierbei immer wieder auf Probleme mit der Darstellung von Graphen, getroffen sind, mussten wir uns nach Alternativen umschauen. Die bekanntesten Alternativen für UI Erstellung in Python sind Tkinter oder PyQT. Da wir in PyQT bessere Ken-

ntnisse besitzen, haben wir uns für dieses Modul entschieden. Zudem bietet matplotlib, die Bibliothek, welche zum Erzeugen des Graphen dienen soll, in ihrem Backend die Möglichkeit der nahtlosen Integration in QT. Für unser erdachtes Design benötigen wir zwei Graphen, wovon einer das Signal darstellen soll, während der andere, währenddessen die Frequenzanalyse darstellt. Wir benötigen insgesamt zwölf Möglichkeiten um Werte und Einstellungen einstellen zu können, hier haben wir uns bei neun dieser Elemente für Dials entschieden, da diese sich den Drehreglern eines richtigem Oszilloskop ähneln. Die anderen drei Einstellungsmöglichkeiten bestehen aus einer Radiobox, einem Drop-Down-Menu und einer Checkbox. Es sollte immer bewusst sein, welcher Wert, an diesem Regler eingestellt werden kann, welcher Wert gerade eingestellt ist und welche Einheit dieser Wert hat. Um eine gewisse Konsistenz innerhalb unserer Anwendung zu bieten, und dem User, mit gleichem Aussehen zu zeigen, dass alle Einstellungen etwas bewirken, sind alle gleich aufgebaut. Zuerst erstellen wir eine Groupbox, diese stellt eine Art Box dar, wo in der oberen Kante ein Text steht. Dieser Text soll in unserem Fall, dann jeweils den einstellbaren Wert anzeigen. In dieser Groupbox wird ein QHBoxLayout platziert, dieses ordnet alle untergeordneten Elemente, in einer horizontalen Linie an. Innerhalb dieses Layouts platzieren wir dann zuerst das QDial, welches unseren Drehregler darstellt. Ein Dial bietet die Möglichkeit, dass der User Werte zwischen einem Minimum und einem Maximum einstellen kann. Dieses verändern des Wertes triggert dann eine Funktion, in welcher wir die Werte abrufen können, und je nachdem entweder noch verändern in eine expontielle Richtung oder direkt an die zugehörige Funktion übergeben. Das nächste Element nach dem Dial, ist dann ein QLCDNumber, welches ein LCD Display ergibt, welches nur zur Anzeige von Nummern verwendet werden kann. Hier sollen dann dem User jeweils, die eingestellten Werte angezeigt werden. Das letzte Element innerhalb des Layouts ist dann ein einfaches QLabel, welches nur die Einheit anzeigen soll. Wir benötigen neben den Dials aber noch weitere Einstellungsmöglichkeiten: einmal die Möglichkeit den Ursprung des Signals einzustellen, also soll das Signal Live, aus Datei oder durch Generation enstehen. Hierfür wurden Radiobut-

tons gewählt, aus dem Grund, dass es dem User so nur möglich ist, genau eine Eingabemöglichkeit auszuwählen. Die Möglichkeit das Signal zu invertieren ist gegeben durch ein Checkbox, welche an oder abgewählt werden kann. Als letzte Einstellung benötigen wir noch für den Signalgenerator, eine Einstellung, welche Wellenart erzeugt werden soll. Hierfür wurde eine QComboBox gewählt, in welche die drei Signalarten geaddet werden. Nun müssen wir alle Elemente, noch dem Hauptwindow adden. Dieses hat ein GridLayout erhalten um die Elemente gut anordnen zu können. Nach dem erstmaligen Hinzufügen aller Elemente, zeigt sich eine ziemliche Überladung des Userinterfaces. Daher mussten die weniger oft benutzen Elemente ausgelagert werden. Hierfür bietet PyQT die Möglichkeit ein QWidget zu erstellen. Sobald ein QWidget keinem Elternteil zugeordnet ist, taucht es als eigenständiges Fenster auf. Somit wurden die zwei Teile unseres Oszilloskops, welche für die allgemeinen Settings und die Frequenzanalyse zuständig sind, in eigenständige Fenster zu packen. Damit diese nur auftauchen, wenn sie auch benötigt werden, wurde eine menuBar hinzugefügt, welche jeweils einen Button für die Settings und die Frequenzanalyse enthält. Werden diese gedrückt, so öffnen sich die zugehörigen Fenster.

9 Ergebnisse

9.1 Was haben wir gemacht?

Wir haben im Laufe der Erarbeitung dieser Arbeit, mithilfe der Programmiersprache Python, ein funktionales Oszilloskop mit dazugehörigem User Interface erstellt. Hierzu wurde sich verschiedenster Bibliotheken bedient, darunter numpy, scipy, matplotlib, pyqt, time, threading, pyaudio und weiterer. Mit deren Hilfe haben wir die Erzeugung und Aufnahme verschiedenster Signale umgesetzt, welche dann analysiert und angezeigt werden können.

9.2 Resultate

Uns ist es gelungen verschiedenste Signale, welcher der User einstellen kann anzuzeigen. Diese können auch analysiert werden. Wir mussten aber aufgrund von Zeitmangel, ein paar Dinge streichen und wir sind auch mit ein paar Umsetzungen unsererseits unzufrieden. So wird das Signal unruhig, wenn hohe Frequenzen eingstellt wird. Dies führt dazu, dass der Nutzer die Werte nicht ordentlich ablesen kann. Wir mussten die Möglichkeit aus einer Sounddatei zu lesen und das daraus resultierende Signal zu analysisieren, als Funktion streichen, da uns die Zeit fehlte. Zudem ist unser Oszilloskop kein "echtes" Oszilloskop, da es nicht in der Lage ist ein Signal selbstständig abzutasten. Diese Funktionalität war zwar zwischenzeitlich vorhanden, wurde aber aufgrund von Problemen mit der Synchronisation der einzelnen Threads wieder entfernt. Wir haben somit zwar ein Oszilloskop geschaffen, es existieren aber weit aus bessere Alternativen.

9.3 Ausblick

Mit mehr Zeit wäre es möglich, die oben genannten Fehler zu beheben und somit ein "echtes" Oszilloskop zu schaffen. Zusätzlich besteht die Möglichkeit einer Erweiterung um mehr Trigger Funktionen, wie z.B. A-B Trigger. Da wir aber keinen wirklichen Anwendungszweck für dieses Projekt haben, werden wir dieses nicht weiter verfolgen. Der Code steht aber öffentlich auf Github zur Verfügung und kann gerne geforkt und erweitert werden.

Literaturverzeichnis

[1] Electrical Academia. *How does an Oscilloscope Work*. URL: https://electricalacademia.com/electronics/oscilloscope-work-oscilloscope-parts-functions/. (accessed: 04.09.2021).

[2] Chris Donner. *Dazu brauchst du ein Oszilloskop*. URL: https://oszilloskop-kaufen.com/ratgeber/dazu-brauchst-du-ein-oszilloskop/#. (accessed: 05.09.2021).

[3] mcclurel. *The XYZ of an Oscilloscope*. URL: http://ecee.colorado.edu/~mcclurel/txyzscopes.pdf. (accessed: 01.08.2021 (not longer online)).

[4] Toolboom Team. *Oscilloscopes: History and Classification*. URL: https://toolboom.com/en/articles-and-video/oscilloscopes-history-and-classification/. (accessed: 04.09.2021).

[5] Tektronix. *ABC der Oszilloskope*. URL: https://download.tek.com/document/03G_8605_6_MR_A4.pdf_14_12_2012.pdf. (accessed: 05.09.2021).

[6] Tektronix. *Oscilloscope Basics: Waveforms, Graph, Measurement Reading*. URL: https://www.tek.com/document/online/primer/xyzs-scopes/ch1/oscilloscope-basics. (accessed: 04.09.2021).

[7] Tektronix. *The Three Systems*. URL: https://de.tek.com/document/online/primer/xyzs-scopes/ch4/oscilloscope-systems-and-controls. (accessed: 04.09.2021).

[8] Christian Zeitnitz. *Soundcard Oscilloscope*. URL: https://www.zeitnitz.eu/scms/scope_en. (accessed: 04.09.2021).

List of Figures