Protokoll zu

Einführung in rechnergestütztes Experimentieren

Jannik Tim Zarnitz (E-Mail: j_zarn02@wwu.de)
Leonhard Segger (E-Mail: l_segg03@wwu.de)

in der Woche 03.09.2018 bis 06.09.2018
betreut von
Dr. Jürgen Berkemeier

16. September 2018

Inhaltsverzeichnis

1 Tag 1

1.1 Aufbau einer Sinus- bzw. Bessel-Funktion

Zunächst soll unter Verwendung des graphischen Programmiersystems „LabVIEW“ eine Sinus-Funktion realisiert werden. Mithilfe der von LabVIEW zur Verfügung gestellten numerischen Operationen, Konstanten und Schleifen lässt sich ein Programm schreiben, welches eine Sinusschwingung u der Form

$$u(A, f, t, \varphi) = A \cdot \sin(2\pi f t + \varphi) \ , \tag{1}$$

umsetzt. Wobei A die Schwingungsamplitude, f die Frequenz und φ die Phasenverschiebung bilden. Außerdem ist jedes t durch

$$t = \frac{i}{N} \tag{2}$$

gegeben. N ist dabei die Zahl der Stützstellen und zugleich die Anzahl der zu berechnenden Wertepaare. Der Index i kann Werte im Bereich zwischen 0 und N annehmen. Der besagte Programmcode befindet sich im Blockdiagramm von LabVIEW und ist in Abb. 1 zu sehen.

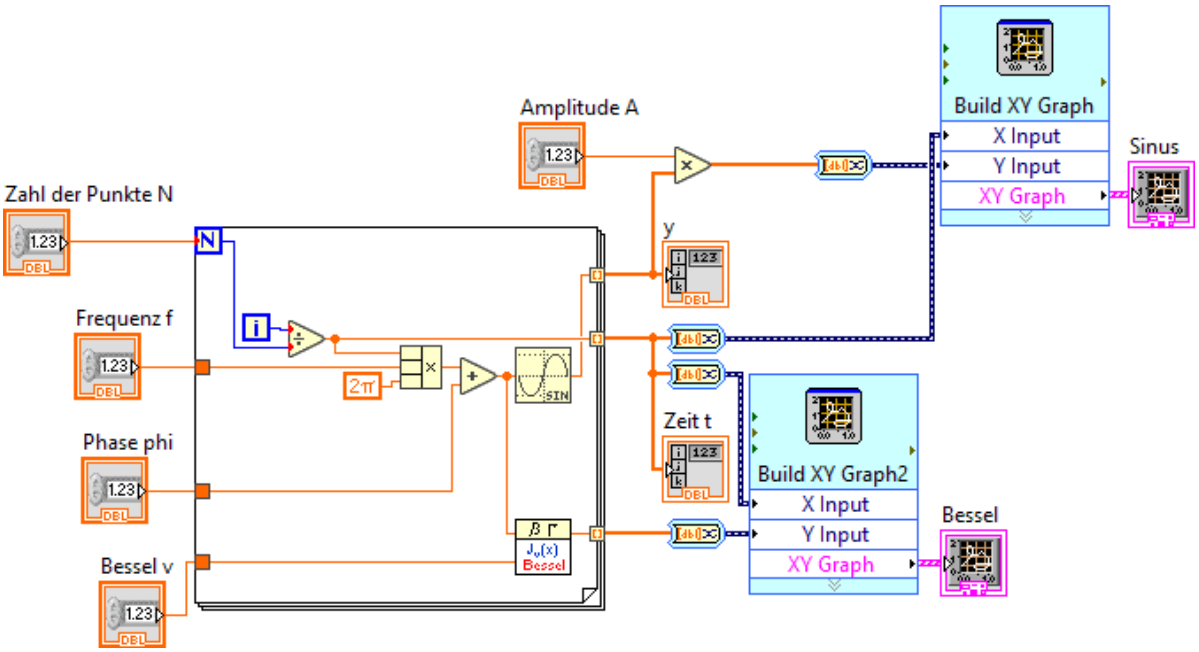


Abbildung 1: Die Abbildung zeigt ein LabVIEW-Blockdiagramm sowie den Aufbau des Programms zur Umsetzung einer Sinus- bzw. Bessel-Funktion. Die eigentliche Werte-Berechnung findet innerhalb der For-Schleife statt. Dazu werden feste, aber beliebige Fließkommazahlen für die Amplitude A, die Zahl der Punkte N, die Frequenz f, die Phase φ und die Ordnung ν der Bessel-Funktion herangezogen. Die Ausgabe der Wertepaare erfolgt in Form zweier X-Y-Graphen sowie Arrays.

Das dazugehörige ausgegebene Frontpanel ist in Abb. 2 erkennbar. Die Gleitkommazahlen-Werte für A, f, N und φ sollen frei wählbar sein, zumal u insbesondere von diesen abhängt. Das Programm ist daher so eingerichtet, dass auf der linken Seite des Frontpanels die entsprechenden Bedien- bzw. Eingabeelemente erscheinen. Unter gleichem Namen sind die jeweiligen Blockdiagrammobjekte im Programmcode zu finden. Im Zentrum des Programmcodes in Abb. 1 lässt sich eine For-Schleife erkennen. Gemäß Gleichung (1) sowie Gleichung (2) findet in dieser unter Verwendung des Laufindexes i die tatsächliche Berechnung der t- und u-Werte statt. Die Blockdiagrammobjekte auf der rechten Seite des Programmcodes in Abb. 1 bilden die Anzeige- bzw. Ausgabeelemente des Frontpanels in Abb. 2. Zum einen werden dabei zwei $N+1$ Einträge umfassende, eindimensionale Arrays angelegt, die jeweils alle t- sowie sämtliche u-Werte beinhalten und zum anderen wird ein X-Y-Diagramm erzeugt, in dem alle u-Werte gegen die entsprechenden t-Werte aufgetragen sind. Demnach befindet sich t auf der x-Achse und u auf der y-Achse. Um nun die Informationsübertragung zu bewerkstelligen, ist es notwendig im Programmcode vor den Anschlüssen der „Build XY Graph“-Kästen weitere Blockdiagrammobjekte hinzuzufügen, welche das Leitungssignal in ein für das Anzeigeelement (X-Y-Diagramm) passendes Eingangssignal umwandeln. Zumeist geschieht dies in LabVIEW automatisch und wird dann an der entsprechenden Stelle auf der Leitung mit einem kleinen, roten Punkt markiert.

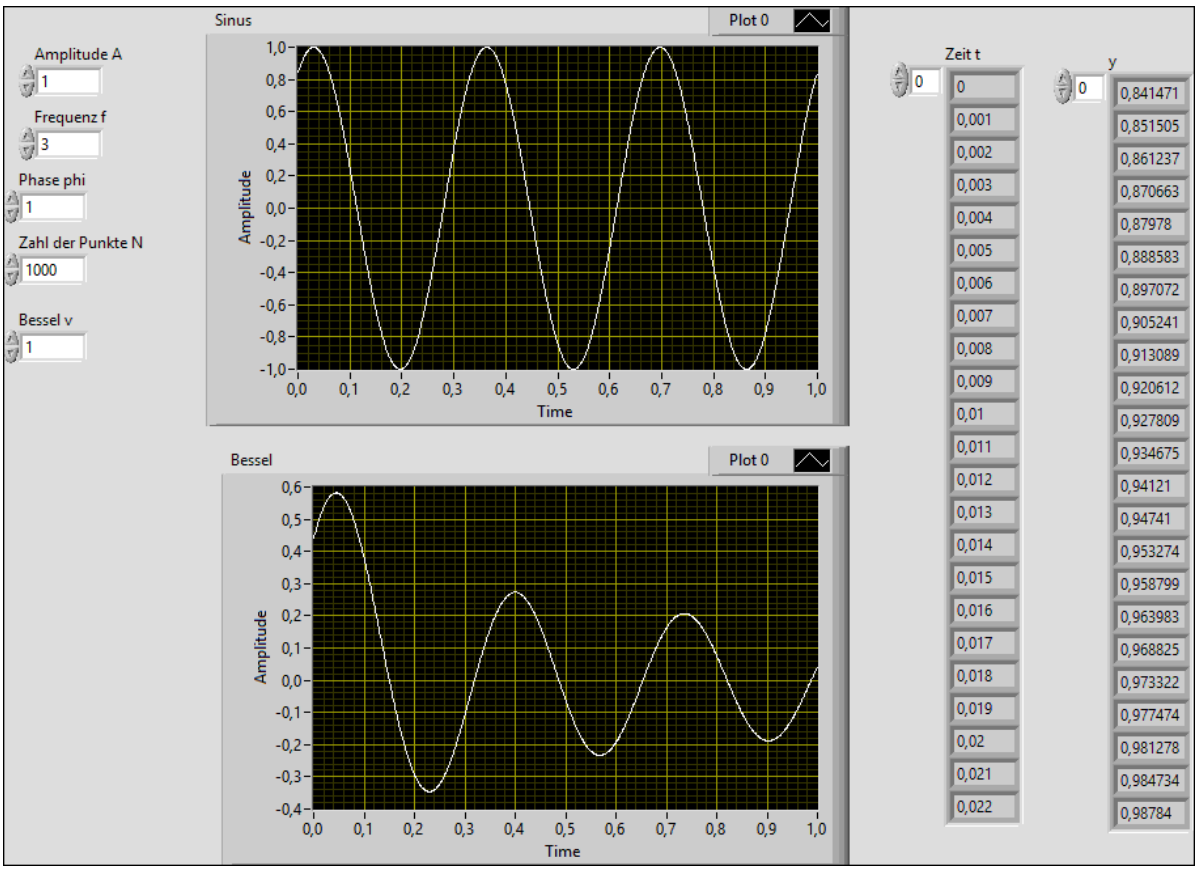


Abbildung 2: In dieser Abbildung ist das Frontpanel als Ausgabe und Benutzeroberfläche dargestellt.

Zusätzlich zur Sinus-Funktion wird eine Bessel-Funktion erster Gattung J_ν mit LabVIEW realisiert und ebenfalls in das Programm in Abb. 1 mit aufgenommen. Das entsprechende Blockdiagrammobjekt ist mit „$J_\nu(x)$ Bessel“ gekennzeichnet. Die Bessel-Funktion besitzt die allgemeine Form

$$J_\nu(x) = \sum_{j=0}^{\infty} \frac{(-1)^j \cdot \left(\frac{x}{2}\right)^{2j+\nu}}{j! \cdot \Gamma(\nu + j + 1)} \ . \tag{3}$$

Wobei ν die Ordnung der Bessel-Funktion bildet, $\Gamma(\cdot)$ die Gamma-Funktion darstellt und x auf Basis des Programmcodes in Abb. 1 als

$$x := 2\pi f t + \varphi = \frac{2\pi f \cdot i}{N} + \varphi \tag{4}$$

definiert ist. Der Fließkommazahlen-Wert für ν ist, ähnlich wie beim Sinus-Programm, über das entsprechende Bedienelement auf der linken Seite des Frontpanels einstellbar, was in Abb. 2 zu erkennen ist. Genauso wie beim Sinus-Programm findet die Berechnung der x- und $J_\nu(x)$-Werte innerhalb der For-Schleife statt. Die Werteausgabe erfolgt in Form eines X-Y-Diagramms, dessen Anzeigeelement im unteren Bereich der Abb. 2

ersichtlich ist und in dem $J_\nu(x)$ als Funktion von x dargestellt ist. Zudem muss das Eingangssignal für das dazugehörige Blockdiagrammobjekt in Abb. 1 passend umgewandelt werden, so wie zuvor beim Sinus-Programm.

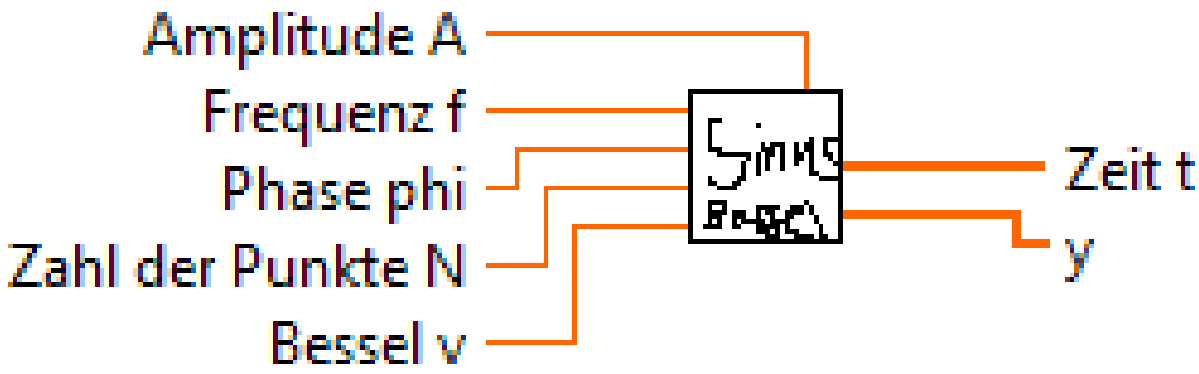


Abbildung 3: Die Abbildung zeigt das gestaltete Programm-Icon sowie die gesetzten Anschlussmöglichkeiten.

LabVIEW bietet die Möglichkeit ein bereits vorhandenes, selbst geschriebenes, funktionsfähiges Programm so zu bearbeiten, dass es für andere LabVIEW-Programme bzw. -Projekte als Blockdiagrammobjekt nutzbar wird. Dazu müssen zunächst die Anschlüsse festgelegt werden, sowohl die Eingänge als auch die Ausgänge. Allgemein kann es sich dabei um verschiedenste Arten von Ein-und Ausgangssignalen handeln. Bei der programmierten Sinus-Funktion verhält es sich demnach so, dass fünf Eingangs- und zwei Ausgangsanschlüsse generiert werden müssen, welche jeweils nur Gleitkommazahlen annehmen bzw. übergeben. Die Amplitude A, die Frequenz f, die Phasenverschiebung φ, die Anzahl der Punkte N und die Ordnung ν der Bessel-Funktion bilden hierbei die Eingänge, die t-Werte sowie die u- bzw. y-Werte erscheinen über die beiden Ausgänge. Des Weiteren ist es in LabVIEW möglich das entsprechende Blockdiagrammobjekt-Icon des Programms zu gestalten. Das komplette Resultat ist in Abb. 3 dargestellt.

Hinweis: Aus der Betrachtung des Programmcodes in Abb. 1 geht hervor, dass das Blockdiagrammobjekt, welches das Array-Ausgabeelement des Frontpanels bildet und alle u-Werte abgreifen soll, inkorrekt eingefügt ist. Denn anstelle einer Implementierung nach der Multiplikation mit der Amplitude A, befindet sich das besagte Blockdiagrammobjekt davor. Sodass das Array lediglich die Werte des Ausdrucks

$$\sin(2\pi f t + \varphi) \tag{5}$$

angibt, welche zwischen -1 und 1 liegen. Da die Ausgabefunktion als Array einen erheblichen Einfluss auf das folgende Vorgehen hat, kann mit dem Sinus-Programm in dieser Form nicht weitergearbeitet werden. Im weiteren Verlauf der Projektbearbeitung ist der beschriebene Fehler allerdings behoben worden. Eine berichtigte, aktualisierte Version des Programms liegt an dieser Stelle jedoch nicht vor.

1.2 Lissajous-Figuren

Im Folgenden sollen je nach Werteeinstellungen auf dem Anzeigeelement der Benutzeroberfläche verschiedene Lissajous-Figuren entstehen. Dies ist in Abb. 5 zu sehen. Die Basis für eine solche Lissajous-Figur bilden zwei verschiedene Sinusschwingungen, welche beide dieselbe Gestalt wie die Gleichung (1) besitzen:

$$u_1(A_1, f_1, t, \varphi_1) = A_1 \cdot \sin(2\pi f_1 t + \varphi_1) \quad \text{und} \quad u_2(A_2, f_2, t, \varphi_2) = A_2 \cdot \sin(2\pi f_2 t + \varphi_2)\ , \quad (6)$$

mit den unterschiedlichen Amplituden A_1 und A_2, den Frequenzen f_1 und f_2 sowie den Phasenverschiebungen φ_1 und φ_2. Zudem bedarf es eines zweidimensionalen kartesischen Koordinatensystems mit Rechts- und Hochachse. Eine Lissajous-Figur entsteht nun dadurch, dass man der x-Achse die Funktion u_1 und der y-Achse die Funktion u_2 zuweist. Je nachdem wie man die Werte für A_1, A_2, f_1, f_2, φ_1, φ_2 und N wählt, ergeben sich dann unterschiedliche Lissajous-Figuren.

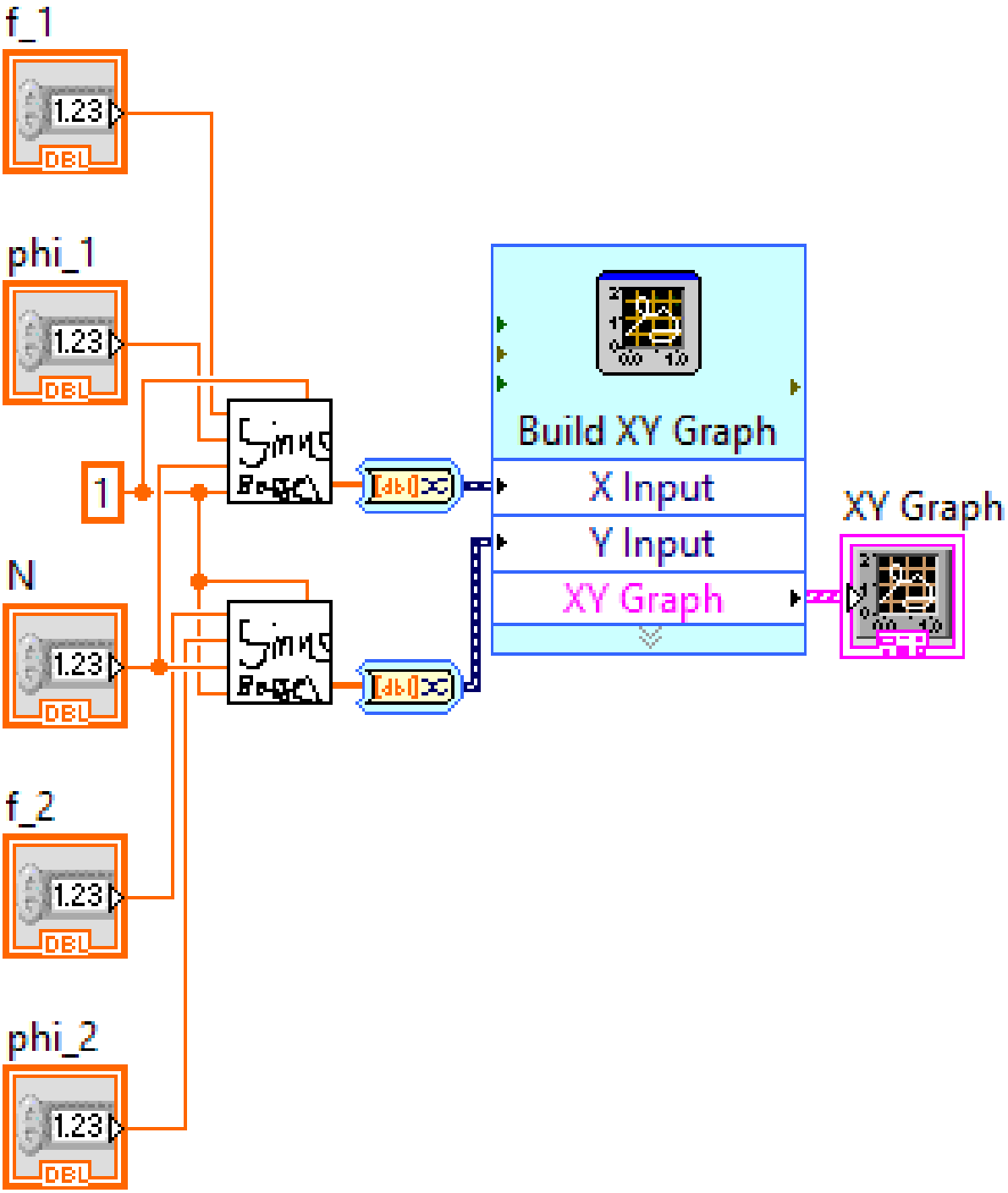


Abbildung 4: Dargestellt ist das Blockdiagramm und der LabVIEW-Programmcode, mit dem sich Lissajous-Figuren realisieren lassen.

Das dazugehörige LabVIEW-Programm sowie dessen Blockdiagramm und Programmcode sind in Abb. 4 zu erkennen. Die in Abschnitt 1.1 erstellte Sinus-Funktion bildet die Grundlage, denn mit dieser lassen sich die beiden separaten Schwingungen u_1 und

u_2 aus Gleichung (6) realisieren. Dabei ist es möglich das Sinus-Programm als Blockdiagrammobjekt in den Programmcode einfügen, jeweils einmal für u_1 und u_2, was in Abb. 4 deutlich wird. An die offenen Eingangsanschlüsse müssen entsprechende, sogenannte „Controler“ für die Frequenzen f_1 und f_2, für die Phasenverschiebungen φ_1 und φ_2 sowie für die Zahl der Stützstellen N angeschlossen werden. Die dazugehörigen Bedien- bzw. Eingabeelemente befinden sich links auf dem Frontpanel des Programms, so wie es in Abb. 5 dargestellt ist. Für die Amplituden soll einfachheitshalber $A_1 = A_2 = 1$ gelten.

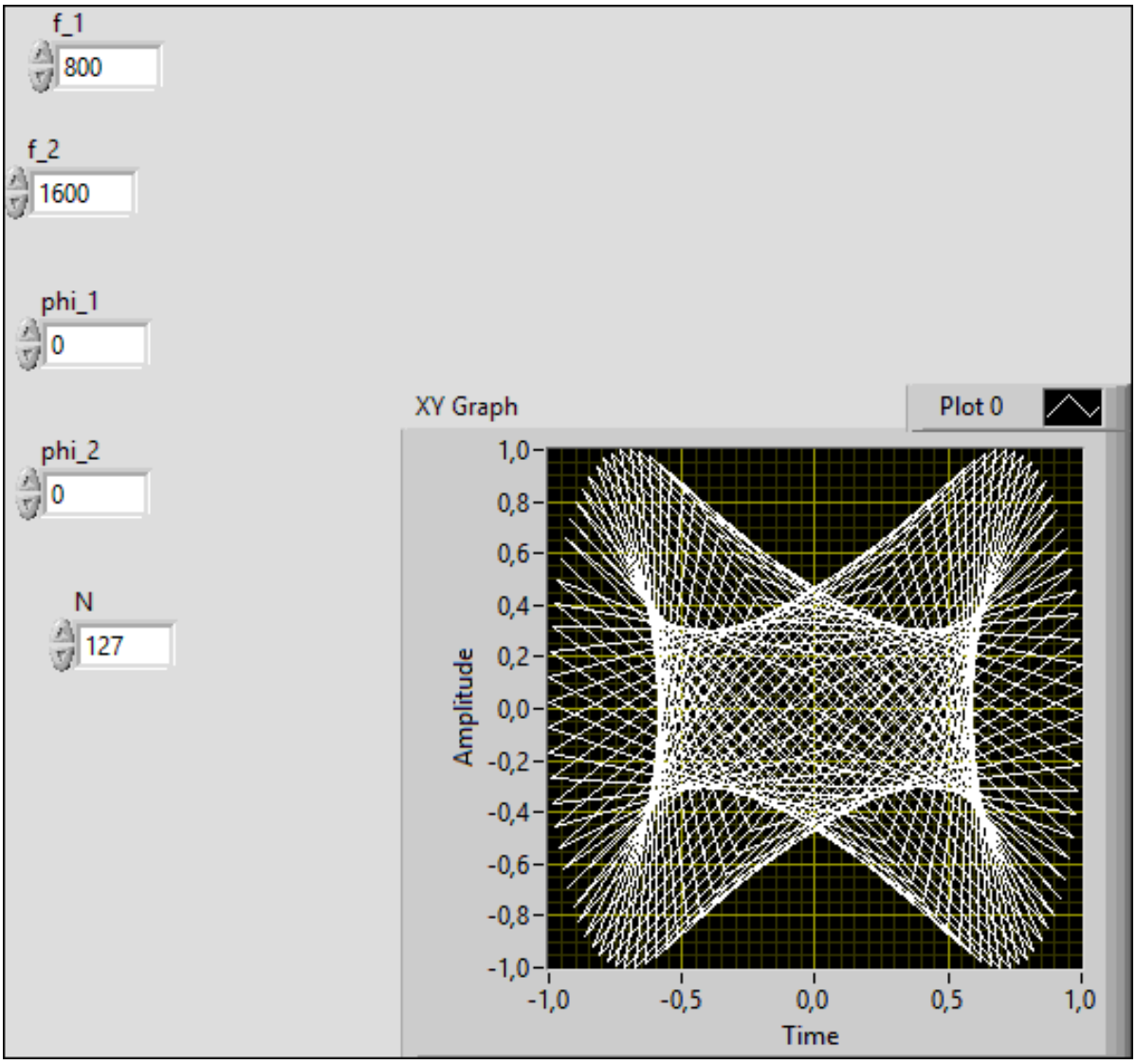


Abbildung 5: Die Abbildung zeigt das Frontpanel bzw. die Benutzeroberfläche, in dem sich im Anzeigeelement eine Lissajous-Figur befindet. Auf der linken Seite können die entsprechenden f_1-, f_2-, φ_1-, φ_2- und N-Werte abgelesen werden.

An den Ausgangsanschlüssen wird ein X-Y-Diagramm angebracht, wobei die u_1-Werte

auf der x-Achse und die u_2-Werte auf der y-Achse liegen. Wie beim Programm in Abschnitt 1.1, ist dem Ausgabeelement an der entsprechenden Stelle auf der Leitung ein passendes Blockdiagrammobjekt zur Signalumwandlung vorgeschaltet. Je nach Wahl der f_1-, f_2-, φ_1-, φ_2- und N-Werte können sich somit die unterschiedlichsten Lissajous-Figuren formen.

2 Tag 2

2.1 Digitales Oszilloskop mit ExpressVI

Es wird ein Funktionsgenerator verwendet. Dessen Signal wird über einen Analog-Digital-Wandler durch den Computer erfasst. Zunächst wird das Signal in LabView mit dem entsprechenden ExpressVI verarbeitet. Das zugehörige Programm ist in Abb. 6 und die Frontplatte in Abb. 7 dargestellt.

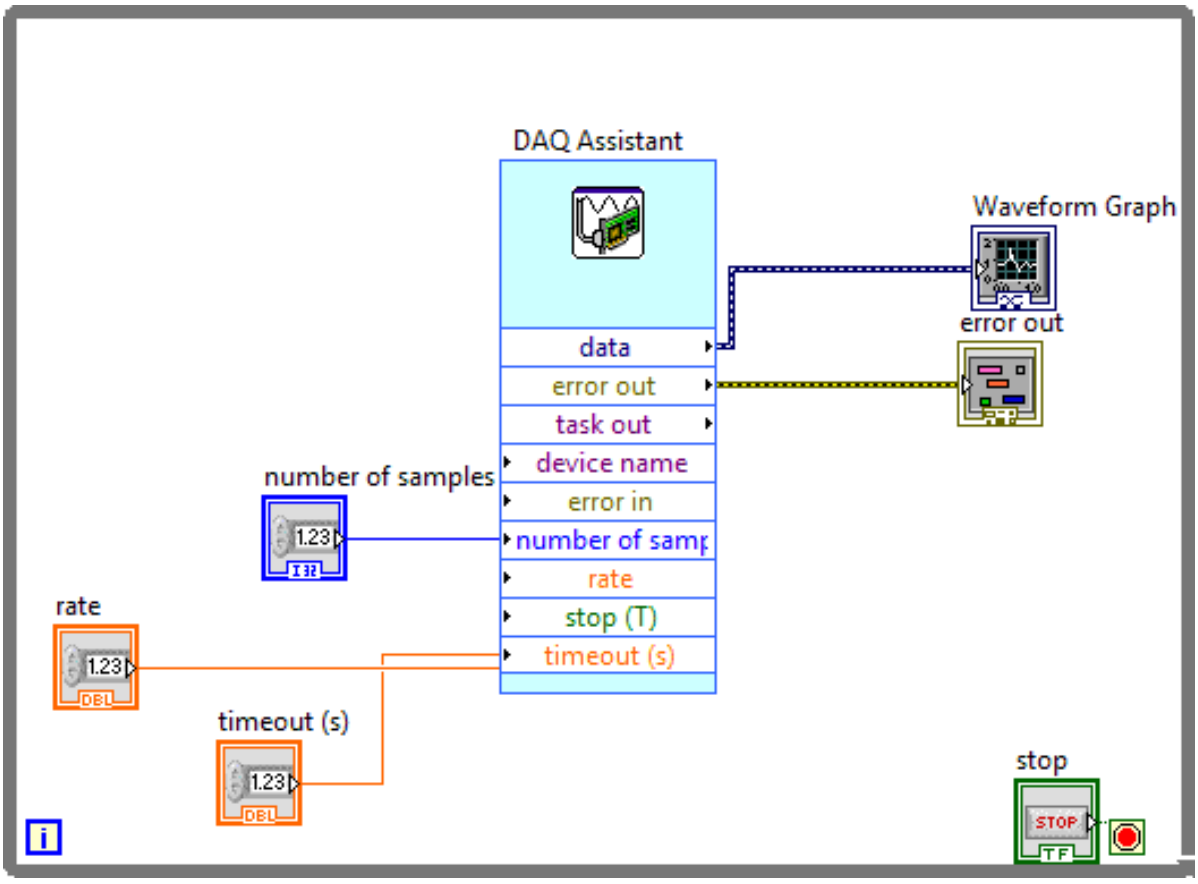


Abbildung 6: Einfaches Oszilloskop mithilfe des ExpressVIs zur Verarbeitung von Daten von Messgeräten.

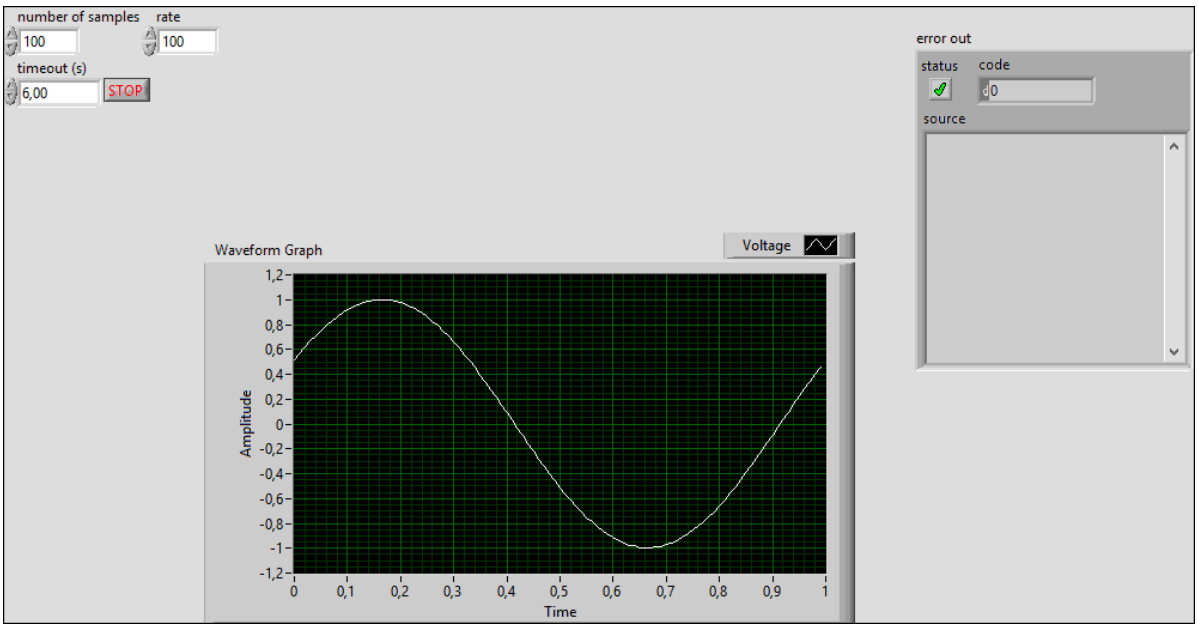


Abbildung 7: Frontplatte des einfachen Oszilloskop mithilfe des ExpressVIs zur Verarbeitung von Daten von Messgeräten. Dabei wurde durch den Funktionsgenerator ein sinusförmiges Signal ausgegeben.

2.2 Fouriertheorem

Gemäß des Fouriertheorems kann jede periodische Funktion als Fourierreihe bzw. Fourierintegral ausgedrückt werden. Dies ist nützlich bei der Zerlegung eines möglicherweise verrauschten Signals in seine Bestandteile. Das grundsätzliche Problem hierbei ist, dass Messprozesse immer zeitlich begrenzt ist, weshalb das Signal nicht bis in die positive und negative Unendlichkeit periodisch sein kann. Dies verursacht den sogenannten „Leakage-Effekt“, auf den in Abschnitt 3.1 näher eingegangen wird.

2.3 Abtasttheorem

Um ein Signal abzutasten, werden im Analog-Digital-Wandler mithilfe einer Sample-and-Hold-Schaltung zeitlich diskrete Messungen durchgeführt. Dies lässt sich als Multiplikation des Signals mit einem Delta-Kamm ausdrücken. Dabei treten Summen- und Differenzfrequenzen der Abtastfrequenz (und deren Oberfrequenzen) und den Frequenzen im Signal auf. Im Frequenzraum ergibt sich hierdurch eine periodische Fortsetzung des Spektrums des ursprünglichen Signals, wobei die Periode der Abtastfrequenz entspricht, wobei auch die Differenzfrequenzen auftauchen. Wenn die Abtastfrequenz hinreichend groß ist, kann man nun mithilfe eines Tiefpasses das Signal herausfiltern. Dazu muss sie allerdings größer als das Doppelte der höchsten im Signal auftretenden Frequenz sein, da sich ansonsten das Spektrum des Signals mit den Differenzfrequenzen der nächsten Periode überlagern. Dies bezeichnet man als Aliasing.

3 Tag 3

3.1 Leakage-Effekt und Fensterfunktion

Wenn die Signalfrequenz kein Vielfaches des Produkts aus Abtastfrequenz und Zahl an Messpunkten pro Messung ist, tritt aufgrund der Endlichkeit des Messprozesses der sogenannte „Leakage-Effekt“ auf. Dieser verursacht eine Verbreiterung des Peaks der Signalfrequenz im Frequenzbild. Um diesen Effekt zu minimieren können Fensterfunktionen angewendet werden. Im Folgenden wird das „Von-Hann-Fenster“ (auch „Hanning-Fenster“) verwendet.

3.2 Digitales Oszilloskop ohne ExpressVI

Das Oszilloskopprogramm von zuvor wird ersetzt durch eines, dass anstelle des ExpressVIs Konfiguration, Messung und Cleanup getrennt enthält. Dieses ist in Abb. 8 dargestellt, während in Abb. 9 die Frontplatte bei einem Eingangssignal von 600 Hz.

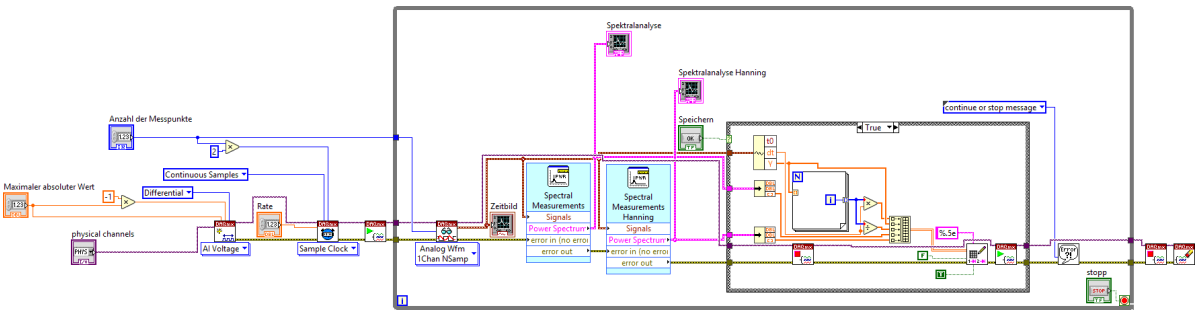


Abbildung 8: Oszilloskopprogramm, das eine Abtastung vornimmt und das Ergebnis im Zeitbild sowie im Frequenzbild mit und ohne Hann-Fenster darstellt. Außerdem ist die Speicherung der Daten in einem Textdokument ermöglicht.

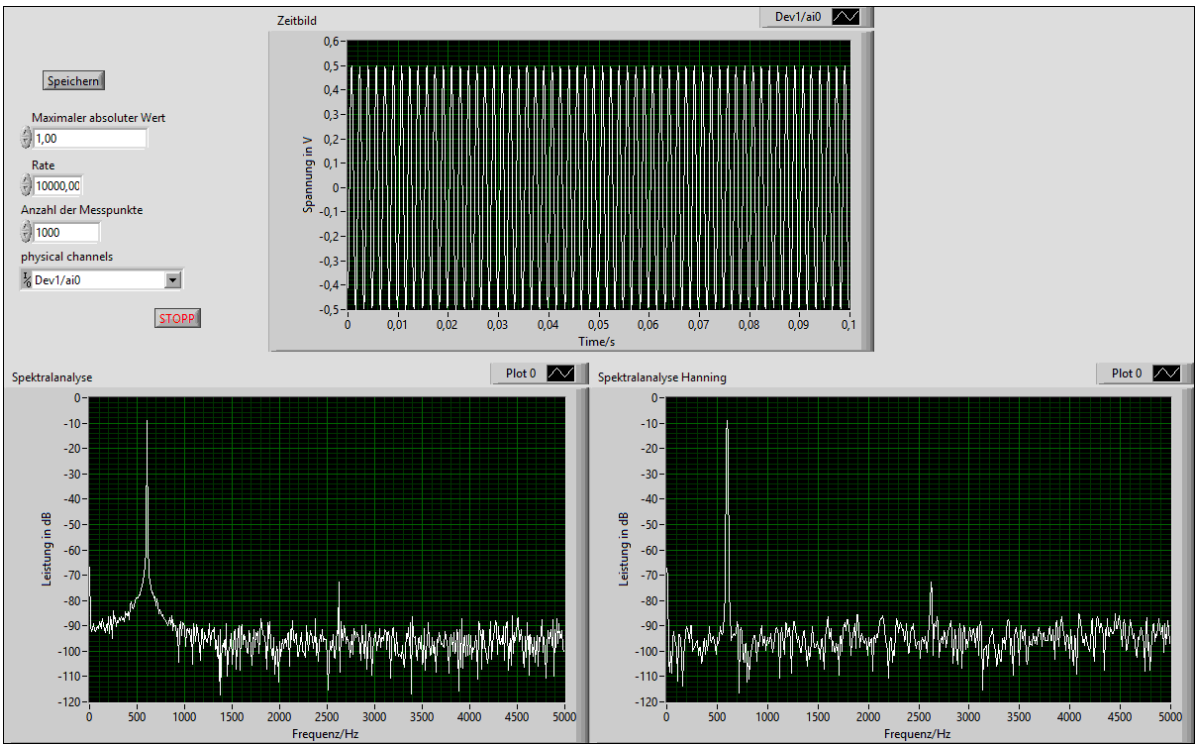


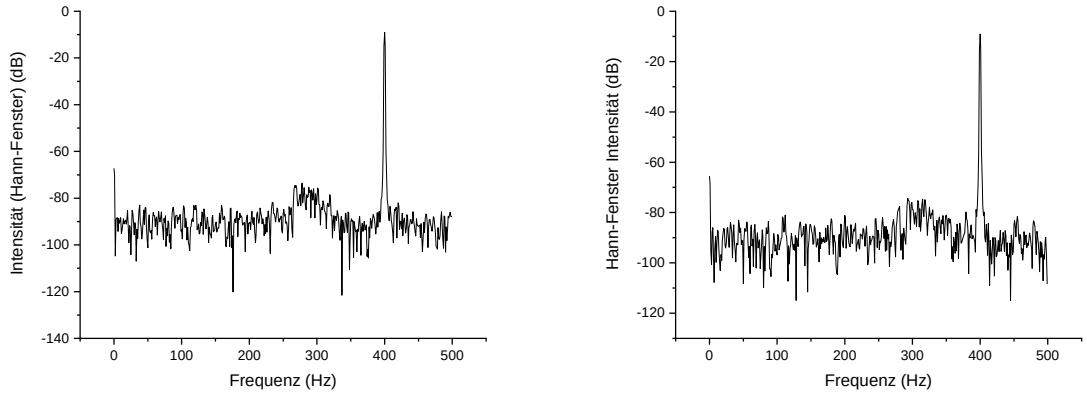
Abbildung 9: Oszilloskopfrontplatte. Dargestellt wird das gemessene Signal im Zeitbild sowie im Frequenzbild mit und ohne Von-Hann-Fenster. Einstellbar ist die Abtastfrequenz, die Anzahl der Messpunkte, der Eingangskanal des Messgerätes und der maximale messbare Wert, wobei der minimale auf das Negative des maximalen gesetzt wird. Mithilfe des Stopp-Knopfes kann das Programm gestoppt werden und mit dem Speichern-Knopf werden die aktuellen Messwerte aus allen drei Diagrammen in eine Textdatei exportiert.

Hierbei wurde ebenfalls die Möglichkeit die aktuellen Messwerte zu speichern eingeführt. Dazu wurden die Strukturen, die in den Diagrammen dargestellt werden in ihre Bestandteile zerlegt und die Arrays der Y-Werte mit zwei Arrays für fortlaufende Zeit- und Frequenzwerte kombiniert und der Speicherfunktion zugeführt. Der Speichervorgang befindet sich innerhalb eines case-Konstrukts, das ignoriert wird, solange der Speichern-Knopf nicht gedrückt wurde. Außerdem werden Fehlermeldungen einer Fehlerdialogfunktion zugeführt, die dem Nutzer erlaubt bei Fehlern wahlweise das Programm anzuhalten oder weiterlaufen zu lassen. Um ein Volllaufen des Mess-Buffers zu vermeiden, wird dieser doppelt so groß wie die Anzahl der Messpunkte pro Zyklus gewählt.

3.3 Aliasing

Um den Effekt des Aliasings absichtlich herbeizuführen, werden bei einer Abtastfrequenz von 1000 Hz zwei verschiedene Signale abgetastet. Da bei dieser Abtastfrequenz die höchste Frequenz im Signal kleiner als 500 Hz sein muss, ist hierbei zu erwarten, dass

ein Signal von 400 Hz korrekt abgetastet wird, während eines mit 600 Hz falsch abgetastet wird. Das Spektrum des abgetasteten Signals bei diesen beiden Signalfrequenzen ist in Abschnitt 3.3 dargestellt.



(a) 400 Hz (b) 600 Hz

Abbildung 10: Mit einer Abtastfrequenz von 1000 Hz abgetastete Signale, deren Frequenz einmal 400 Hz und einmal 600 Hz beträgt, sodass einmal korrekt und einmal falsch abgetastet wird.

Es fällt auf, dass sich zwischen diesen beiden Signalen kein Unterschied erkennen lässt. Dies liegt daran, dass bei einer Signalfrequenz von 600 Hz die eigentliche Signalfrequenz nicht mehr vom idealen Tiefpass übertragen wird, während die Differenzfrequenz von Abtastfrequenz und Signalfrequenz 400 Hz beträgt. Es ist an den beiden Signalen zu erkennen, dass sich ein falsch abgetastetes Signal nicht mehr von einem korrekt abgetasteten Signal bei der entsprechenden Differenzfrequenz unterscheiden lässt. Man stellt fest, dass man vor der Abtastung bereits wissen muss, welche Frequenzen im Signal vorkommen.

3.4 Amplitudenmodulierte Signale

3.4.1 Erzeugung

Es soll nun ein amplitudenmoduliertes Signal erzeugt werden und über die Soundkarte des verwendeten PCs ausgegeben werden. Dabei wird das Signal der Soundkarte wieder auf den Analog-Digital-Wandler gegeben und mit dem zuvor erstellten Oszilloskopprogramm verarbeitet. In Abb. 11 ist das dazu verwendete LabView-Programm dargestellt. Moduliert werden soll die Überlagerung zweier Signale mit einstellbarer Frequenz und Amplitude. Auch Trägerfrequenz und Modulationsgrad soll einstellbar sein. Die Amplitudenmodulation lässt sich durch die Gleichung

$$S_{AM}(t) = S_T(1 + m \cdot s_s(t)) \cdot \cos(2\pi f_T t) \tag{7}$$

beschreiben. Da die Überlagerung von zwei sinusförmigen Signalen beliebiger aber fester Frequenz moduliert werden soll gilt:

$$s_s(t) = A_1 \cos(2\pi f_1 t) + A_2 \cos(2\pi f_2 t) \tag{8}$$

Diese beiden Gleichungen sollen im Folgenden durch Operationen auf Arrays realisiert und als Audiosignal ausgegeben werden. Dazu wird das in Abschnitt 1.1 erstellte Programm verwendet, um Arrays von Sinusfunktionen zu erstellen. Diesem wird in jedem Fall die Phase 0 zugeführt und 44100 Stützstellen gewählt, weil dies der Abtastrate bei CDs entspricht.Dem Programm wird zunächst eine Amplitude von 1 gegeben und dann werden alle Array-Elemente mit der gewählten Phase multipliziert. Dies hängt lediglich damit zusammen, dass zuvor im Sinus-erstellenden Programm ein Fehler vorhanden war, der dies nötig machte. In Abb. 11 ist zu erkennen, dass zunächst zwei Sinus-Arrays erstellt werden, für die jeweils eine Signalfrequenz und -amplitude gewählt werden. Die beiden resultierenden Arrays werden addiert, mit dem Modulationsgrad multipliziert und dann mit dem Array der Trägerfrequenz multipliziert. Dieses wurde zuvor in gleicher Art und Weise mit der Trägerfrequenz erstellt. Für die Amplitude des Trägersignals wird der Kehrwert der maximalen Signalamplitude gewählt, damit die resultierende Amplitude immer auf 1 liegt, da die Soundkarte höhere Werte nicht verarbeitet. Das resultierende Array wird im Zeit- und Frequenzbild dargestellt und dann mithilfe der entsprechenden VIs über die Soundkarte ausgegeben.

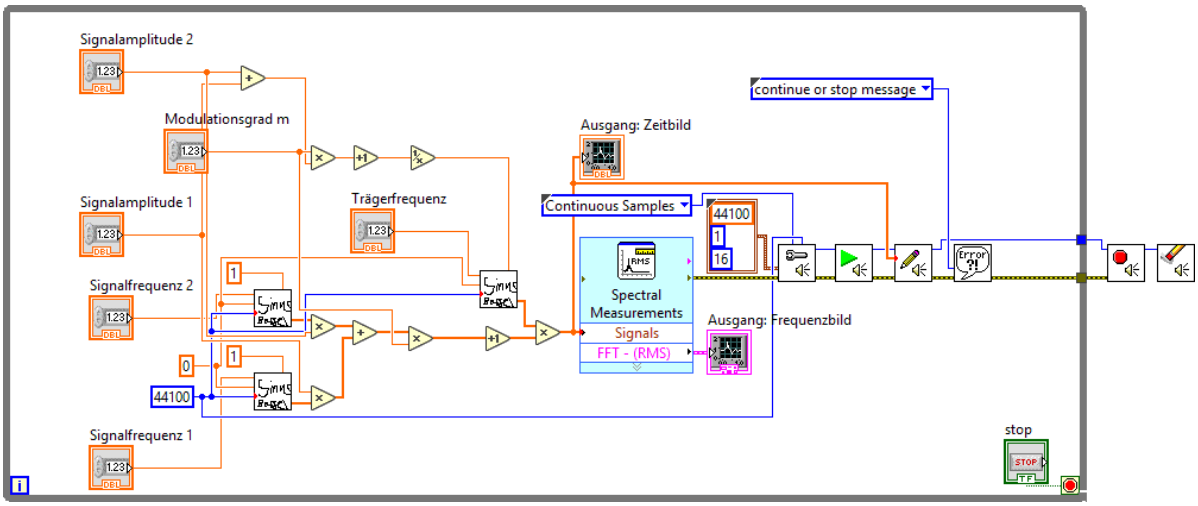


Abbildung 11: Blockdiagramm des Programms zur Erzeugung eines amplitudenmodulierten Signals und Ausgabe dessen über die Soundkarte.

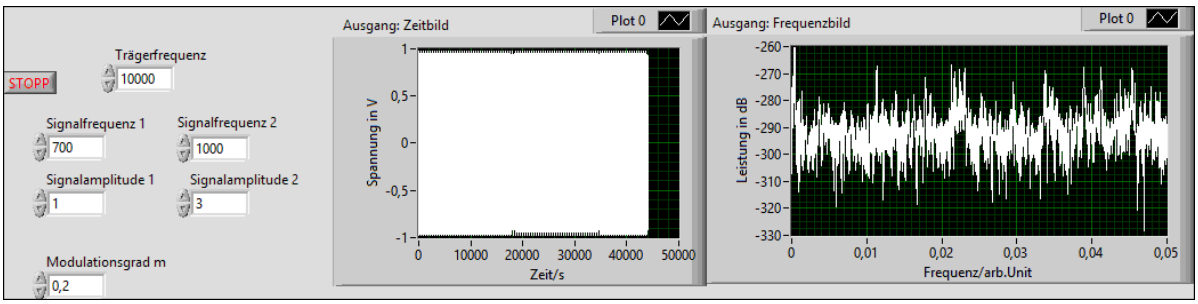


Abbildung 12: Frontplatte des Programms zur Erzeugung eines amplitudenmodulierten Signals und Ausgabe dessen über die Soundkarte.

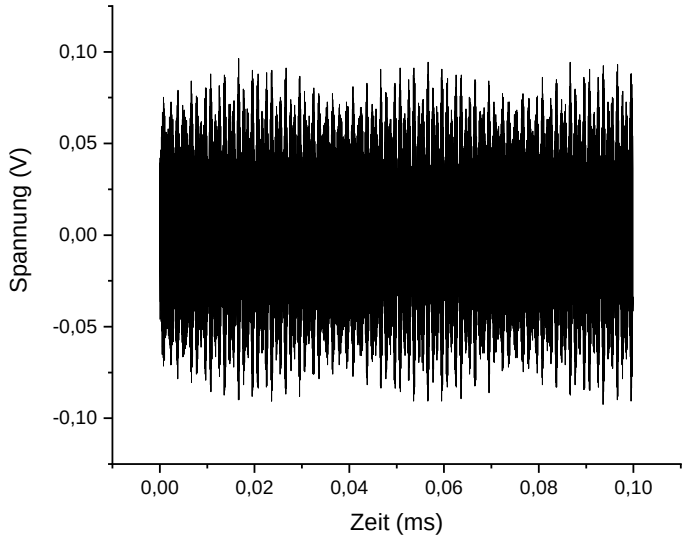


Abbildung 13: Ausgabe des Oszilloskopprogramms im Zeitraum bei Erfassung des amplitudenmodulierten Signals, das mittels der Soundkarte ausgegeben wird.

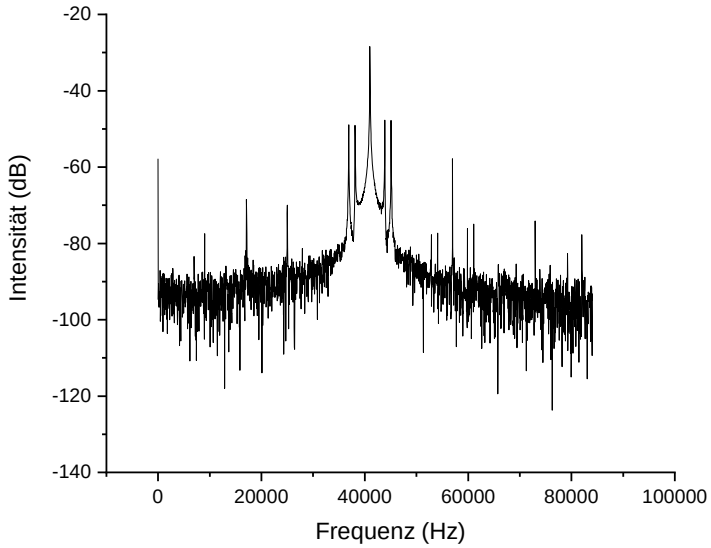


Abbildung 14: Ausgabe des Oszilloskopprogramms im Frequenzraum bei Erfassung des amplitudenmodulierten Signals, das mittels der Soundkarte ausgegeben wird.

3.4.2 Demodulation

4 Tag 4

4.1 Demodulation eines AM-Signals mittels Trägerfrequenzmultiplikation

Das allgemeine Ziel dieses Abschnittes ist

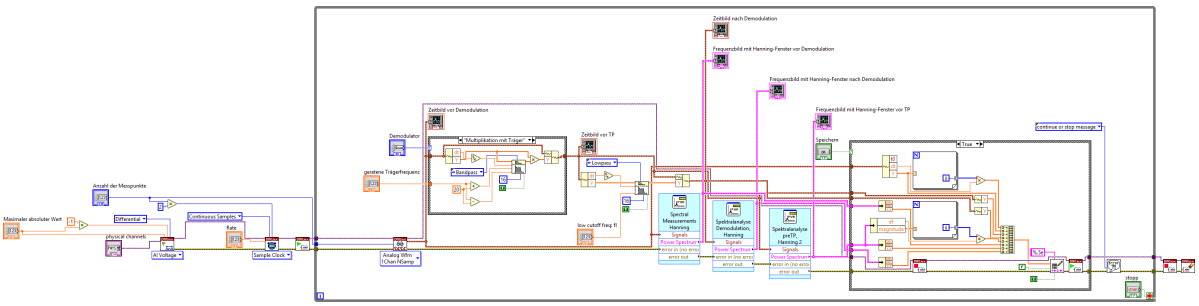


Abbildung 15: .

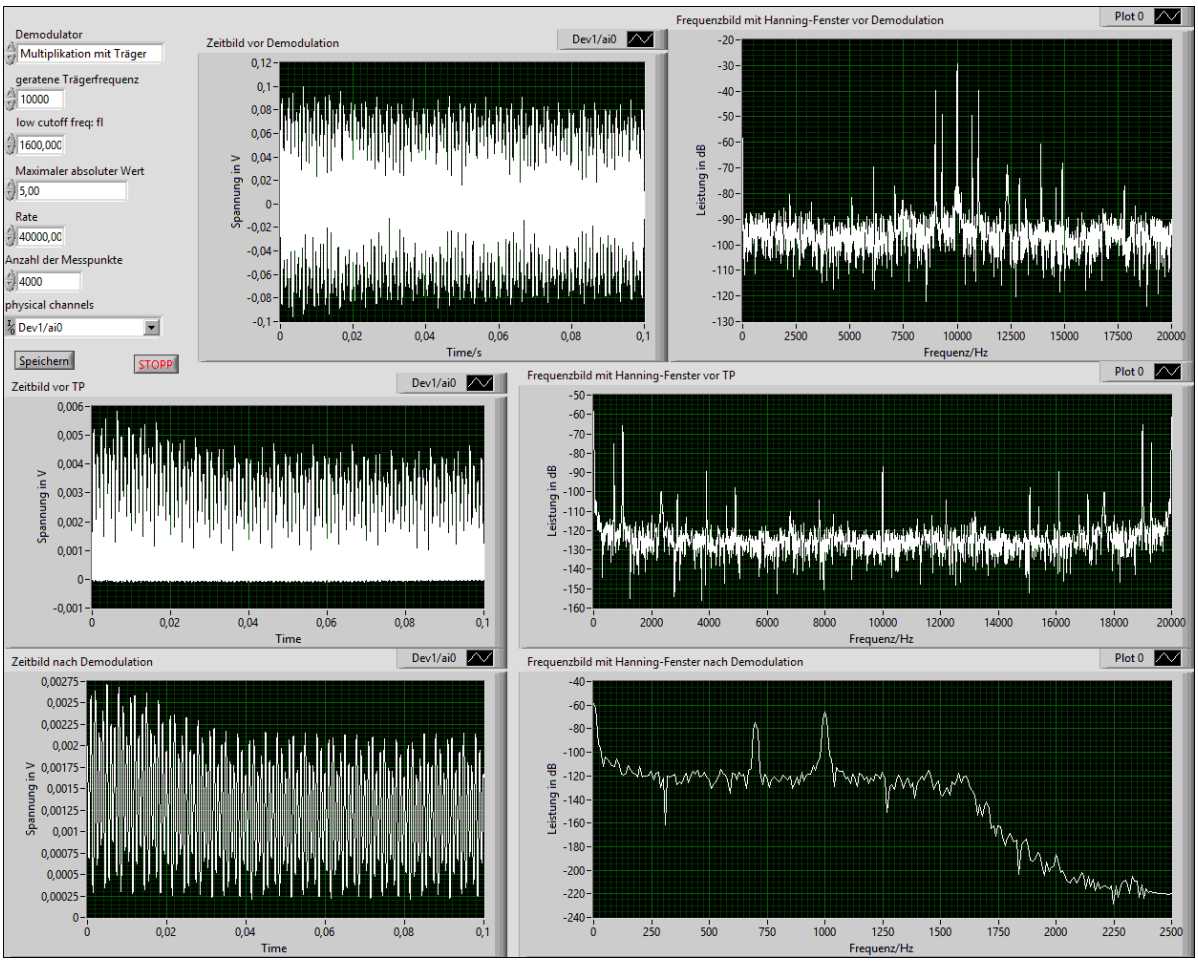


Abbildung 16: .

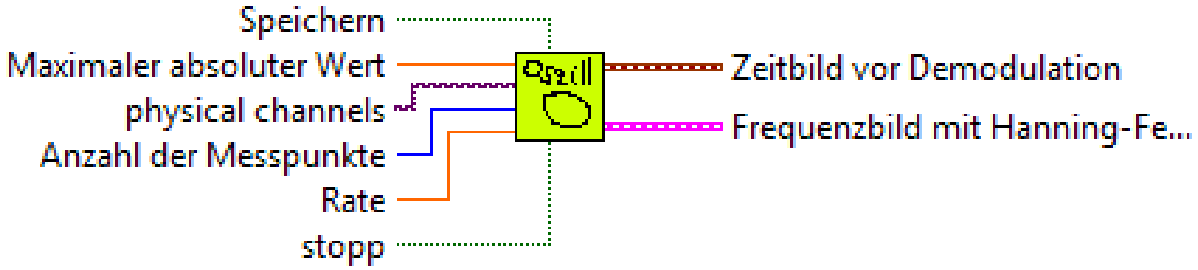


Abbildung 17: .

4.2 Erzeugung eines phasen- bzw. frequenzmodulierten Signals

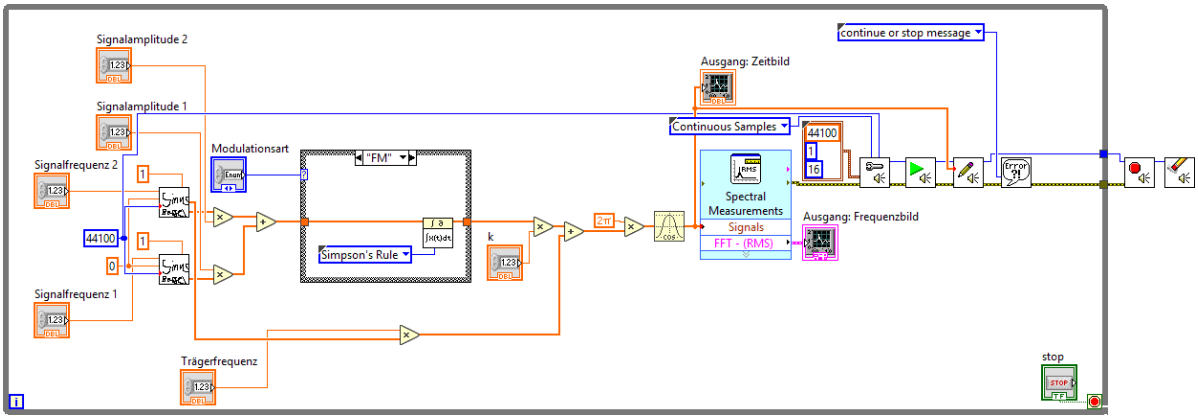


Abbildung 18: .

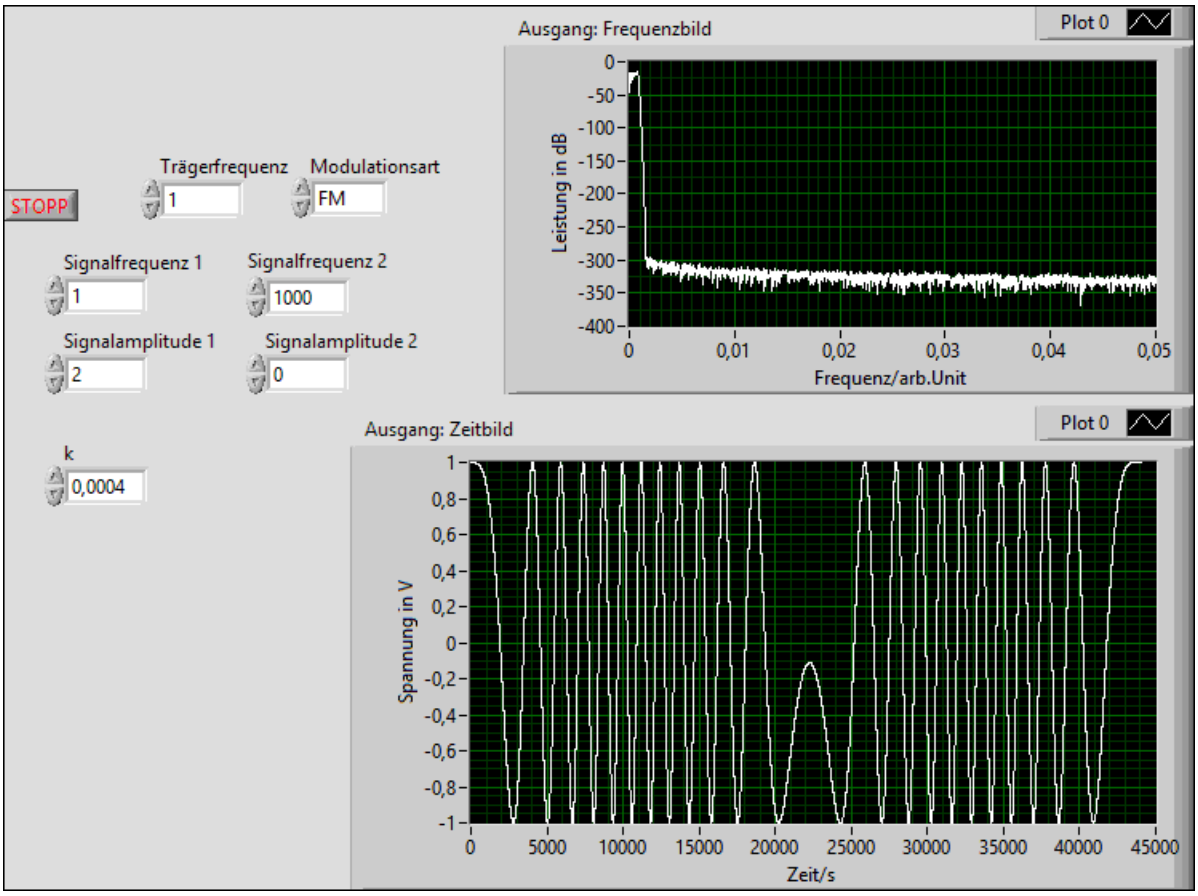


Abbildung 19: .

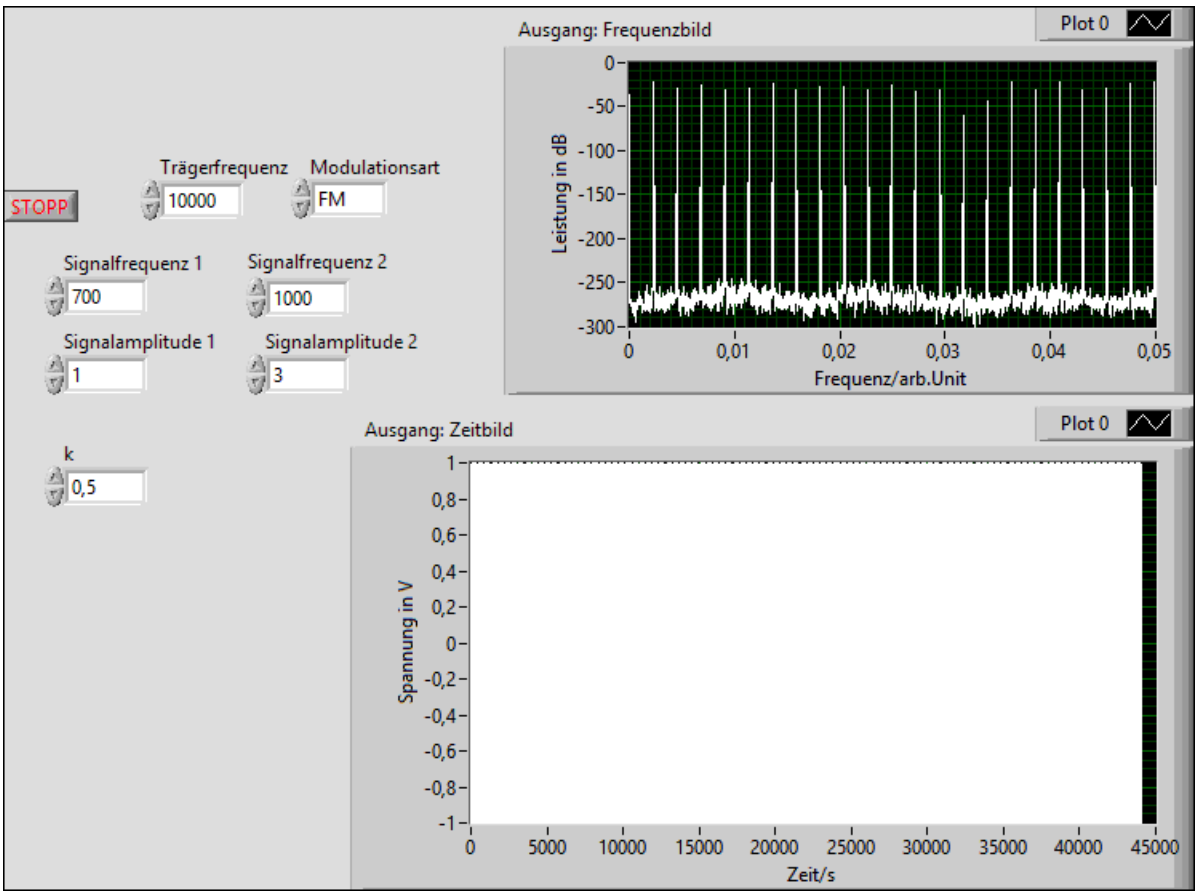


Abbildung 20: .

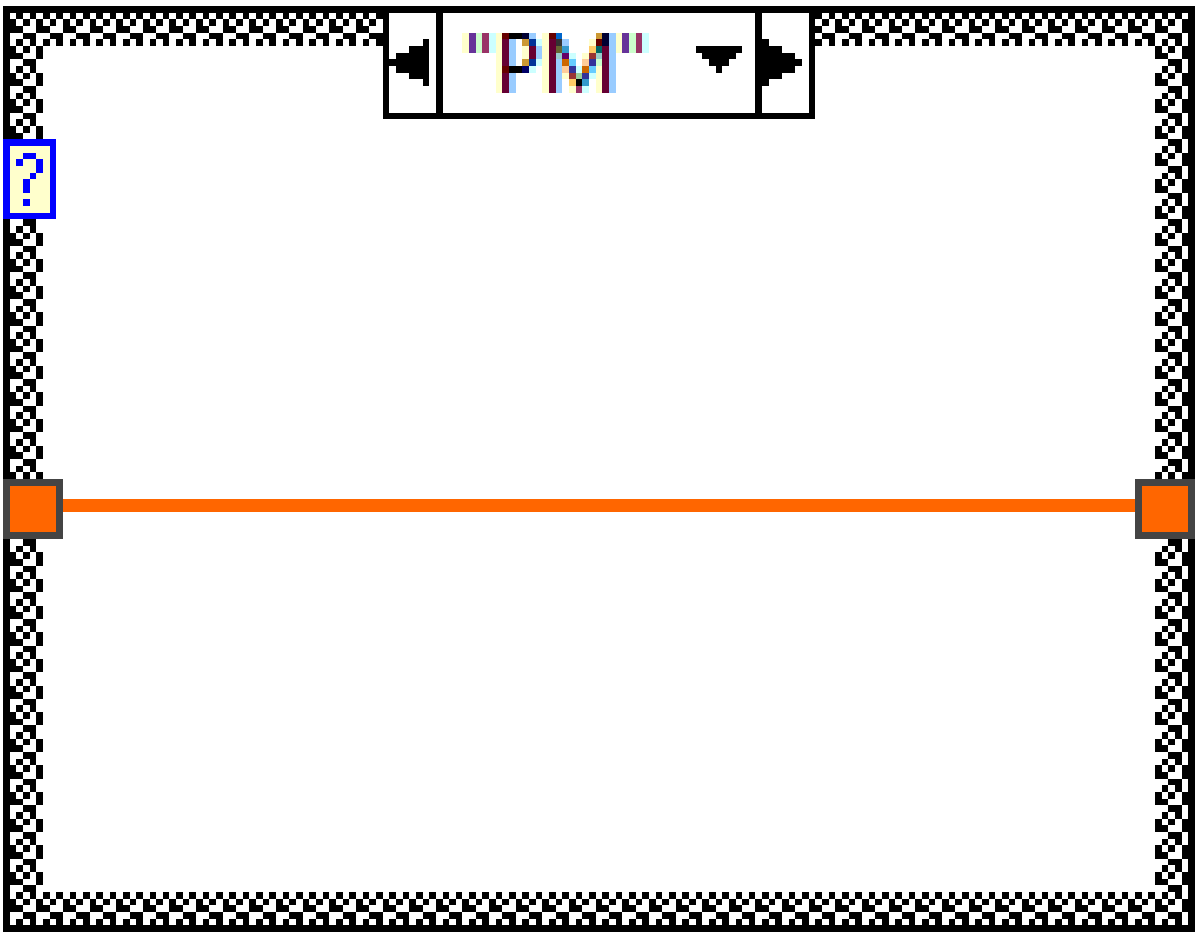


Abbildung 21: .

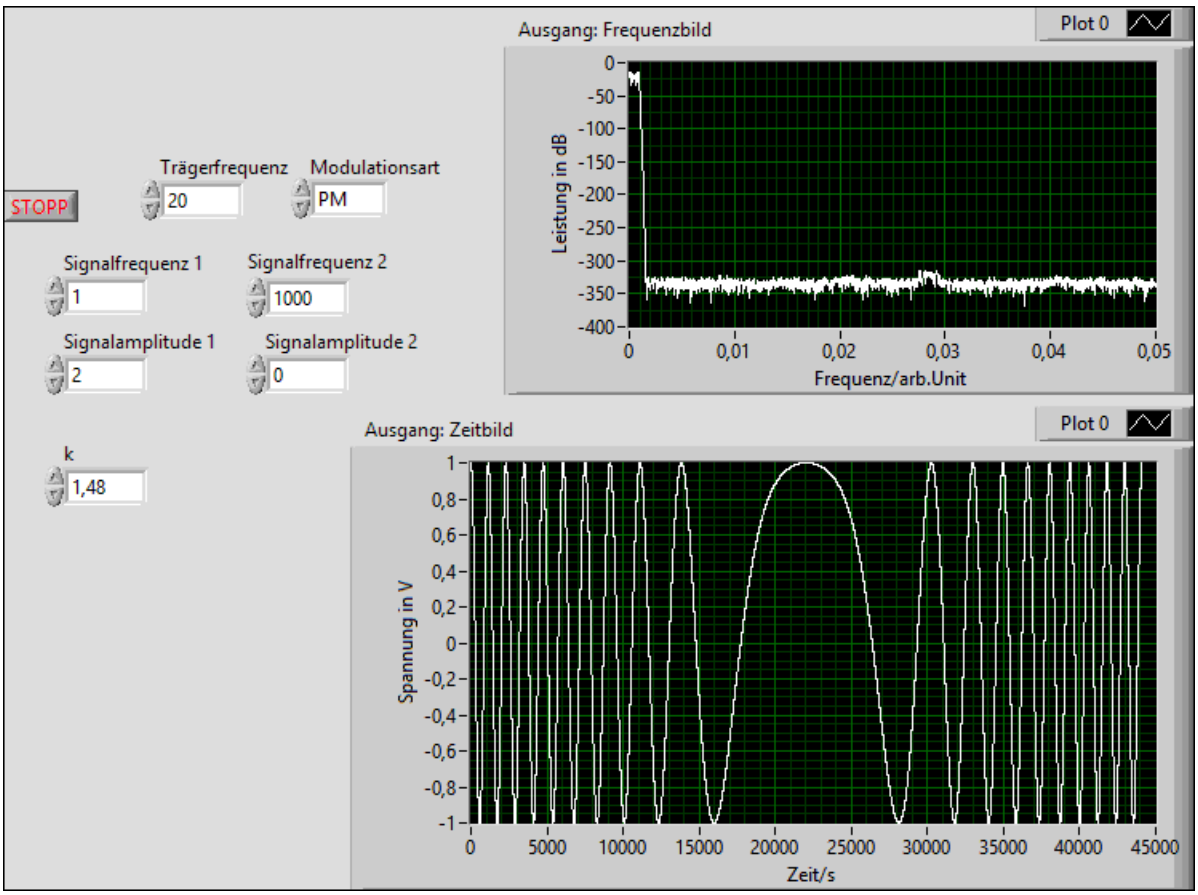


Abbildung 22: .

4.3 Demodulation eines phasen- bzw. frequenzmodulierten Signals

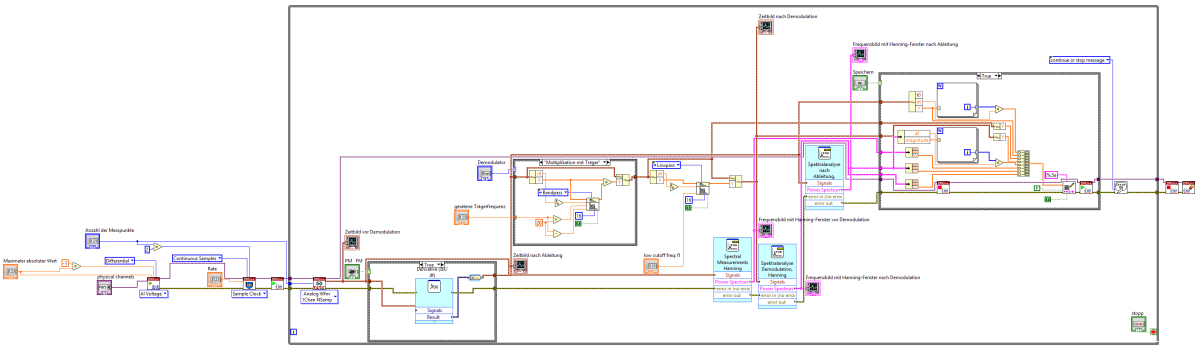


Abbildung 23: .

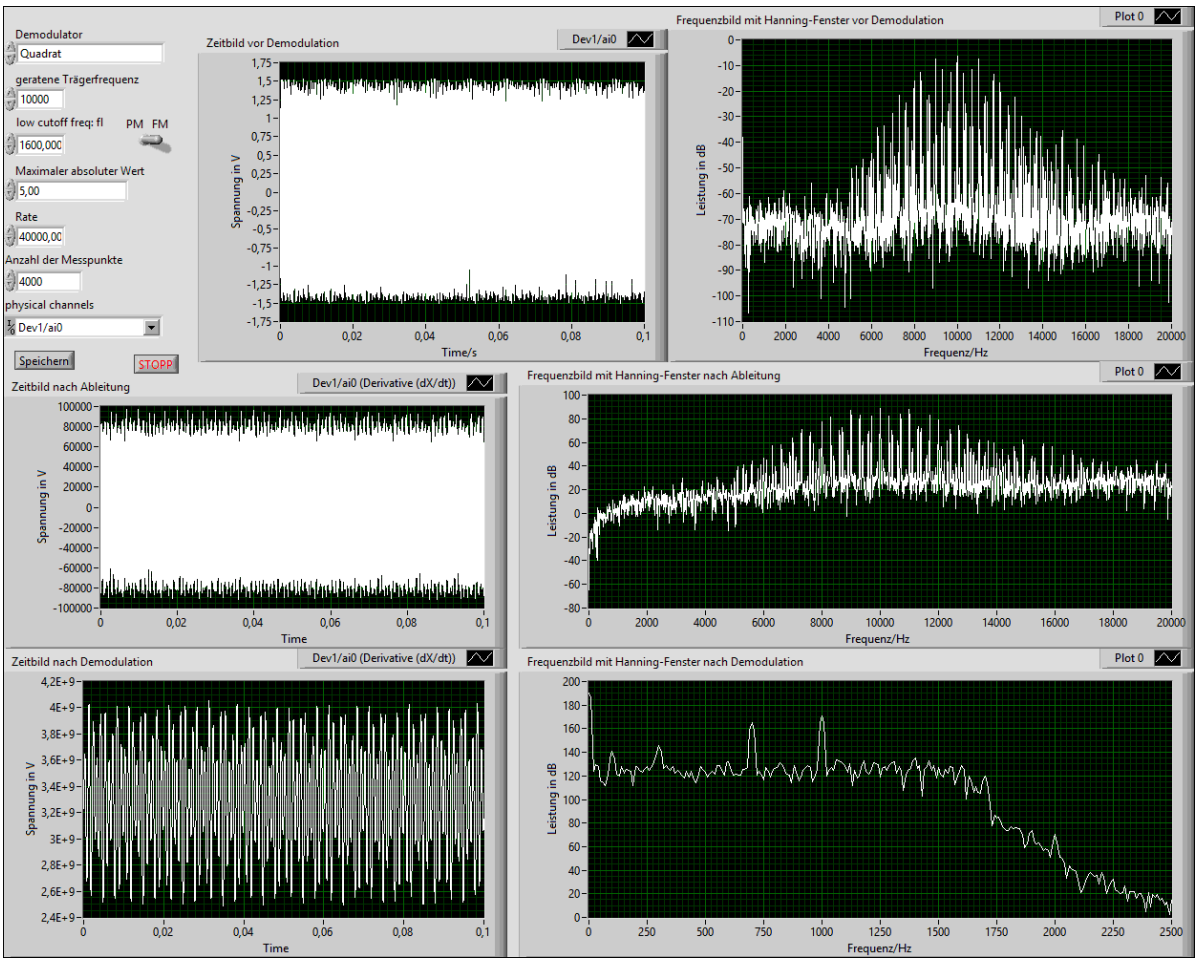


Abbildung 24: .

4.3.1 Erweiterte Demodulation mit Bandpass und zusätzlicher Integration des Signals

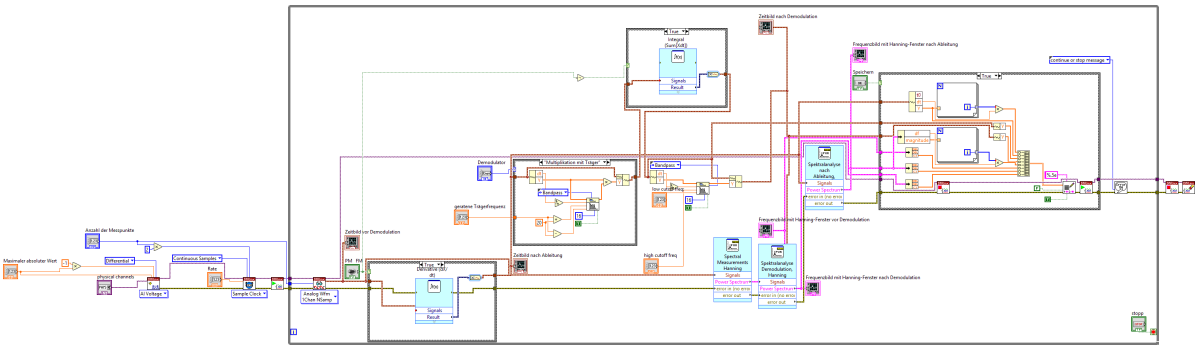


Abbildung 25: .

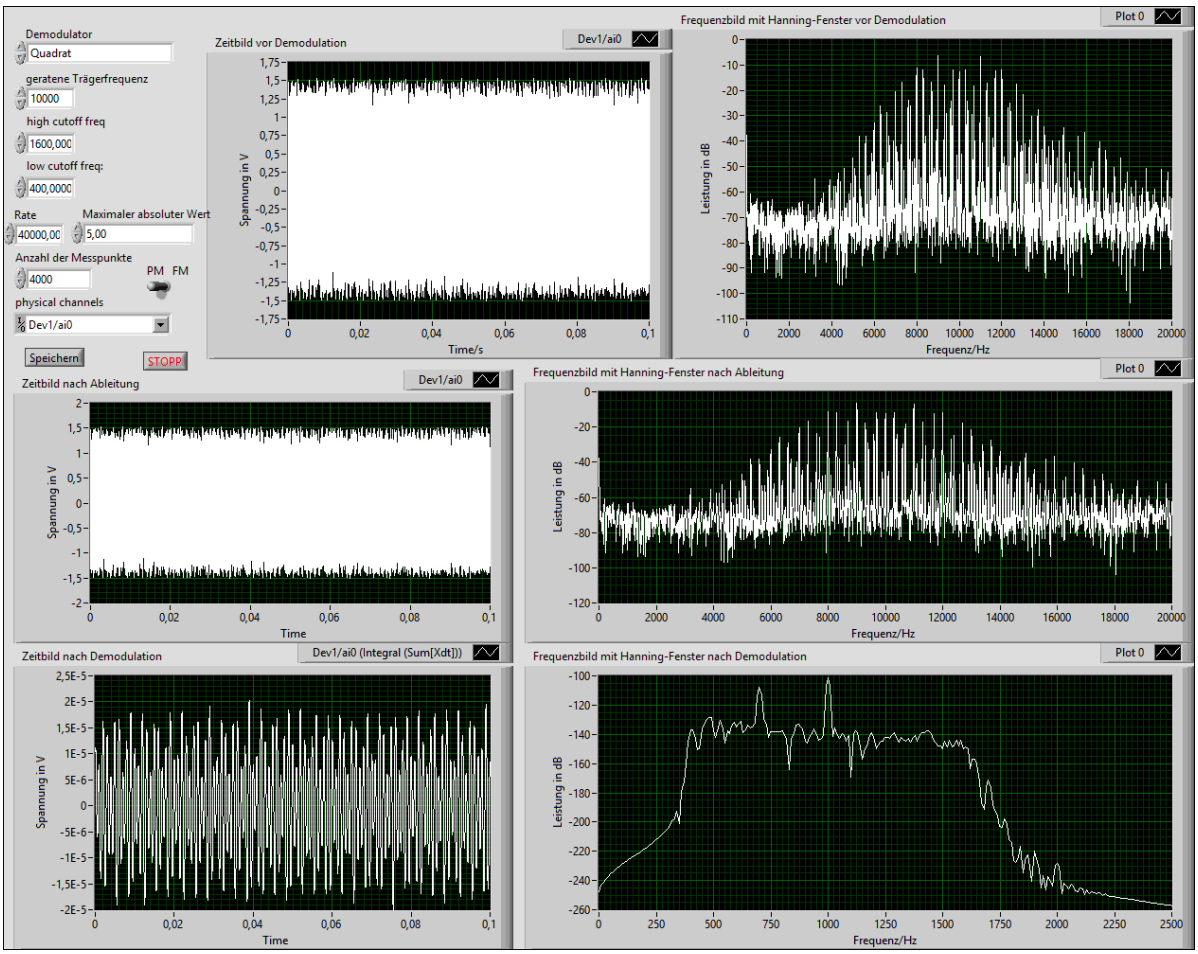


Abbildung 26: .