Allgemein:

Zur Erkennung der Frequenzen und daraus folgend der Noten, wird der Fast Fourier Transform (FFT) Algorithmus verwendet. Dieser ermöglicht die schnelle Durchführung einer Fourier Transformation, was für unseren Anwendungsfall unerlässlich ist.

Was ist eine Fourier Transformation?

Die Fourier Transformation ist eine Integraltransformation, welche 1822 von Jean Baptiste Joseph Fourier eingeführt wurde und genutzt werden kann, um das diskrete Frequenzspektrum periodischer Signale zu ermitteln.

Die Fourier Transformation ist in vielen Bereichen von Wissenschaft und Technik unerlässlich und nahezu jeder nutzt sie täglich beim z.B. Streamen von Videos.

Um sie von einem Computer ausführen zu lassen gibt es die Diskrete Fourier Transformation (DFT) und die Schnelle Fourier Transformation (FFT).

Eigentlich handelt es sich beim FFT-Algorithmus nur um eine Möglichkeit die DFT mit hoher Geschwindigkeit durchzuführen. Deren Wiederentdeckung durch James Cooley und John W. Tukey im Jahr 1965 …  
Wiederentdeckung deshalb, weil Carl Friedrich Gauß den Algorithmus in seiner ersten Form bereits 1805 entdeckte und verwendete, diese aber nie publizierte.

Firmware

Das DSP-Programm, welches am ESP32 läuft, führt eine FFT mit dem Eingangssignal durch. Um ein kontinuierliches Einlesen der Gitarrensignale zu ermöglichen wird der DMA-Controller genutzt.

Wegen ihrer einfachen Anwendung, guten Dokumentation und ihrer hohen Performance wurde die FFT-Bibliothek von Robin Scheibler verwendet. Bei dieser Handelt es um eine Implementierung von radix-2, split-radix und den Basen vier und acht. Diese Basen limitieren die maximal mögliche Größe der FFT.

Erklärung der Firmware anhand des Testfalles



Zuerst werden für NFFT die Größe des FFT-Buffers und für F\_ABT die in der Audiotechnik übliche Abtastrate von 44,1 kHz festgelegt.

Ein Bild, das Text enthält.

Automatisch generierte Beschreibung

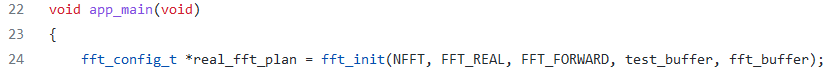
Darauffolgend werden sämtliche benötigte Bibliotheken importiert. Die „freertos“-Bibliotheken werden von Task-System benötigt, welches das Betriebssystem des ESP32 mitliefert. (siehe 3.4…)  
Scheiblers FFT-Bibliothek ist in „fft.h“ enthalten.  
Der Header „processed-Data.h“ enthält die vorbereiteten Testdaten. (siehe 3.3.4.1.1 Testdatenfertigung)

Ein Bild, das Text enthält.

Automatisch generierte Beschreibung

Dann werden all benötigten Buffer und Variablen initialisiert. Das Array „fft\_buffer“ wird nach den ausführen der FFT die Ergebnisse dieser beinhalte. Die drei Darauffolgenden beinhalten jeweils die Magnitude, die Frequenz und die am Piano korrespondierende Tastennummer. Der Float „ratio“ beinhaltet die Auflösung des Amplitudenspektrums in Hertz.

Nun wird der „main-loop“ des Programms gestartet.



Als erstes wird hier die FFT mit den Argument Größe des Eingangsbuffers, Art der FFT, Eingangsbuffer und Buffer für die Ergebnisse konfiguriert.

Danach wird die FFT ausgeführt.



Die Ergebnisse werden anschließend verarbeitet.

Ein Bild, das Text enthält.

Automatisch generierte Beschreibung

Der abgebildete For-Loop ist für die Berechnung der Einzelnen Signifikanten Werte wie etwa Magnitude, Frequenz und die am Piano korrespondierende Tastennummer zuständig.   
Zur Berechnung der Tastennummer wird die Formel

verwendet. Die 440Hz sind der Kammerton A. Hierbei ist zu beachten, dass, um die Midi-Notennummer zu erhalten noch 20 dazu addiert werden müssen, da Midi 128 mögliche Notennummern besitzt und die ersten 20 nicht belegt sind. Mit der If-Verzweigung wird überprüft, ob eine Magnitude einen Schwellenwert übersteigt, was bedeuten würde, dass diese Frequenz im Eingangssignal vorkommt. Das ist insofern wichtig, da bei Audiosignal immer ein Anteil an Rauschen und anderen Schwingungen enthalten ist, welche beispielsweise durch das Kabel, Holz oder Pickup der Gitarre auftreten können.  
Die vereinzelten, auskommentierten „printf“ Funktionen dienen, um die einzelnen Zwischenwerte auszugeben um diese mit zum Beispiel Matlab zu Überprüfen. Im endgültigen Programm sind diese nicht mehr vorhanden da sie rein zum Debuggen dienen.



Zuletzt wird die FFT-Konfiguration zerstört. Dies ist ein Wichtiger Schritt da für jeden Durchlauf des Algorithmus eine neue Konfiguration erstellt werden muss.

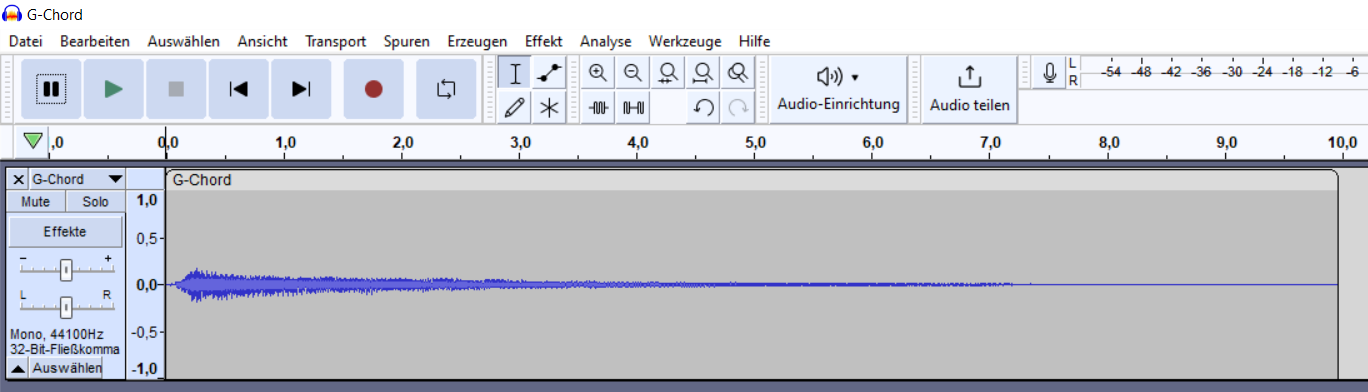
Hiermit ist das Ende des Testprogramms erreicht.

Testdatenfertigung:

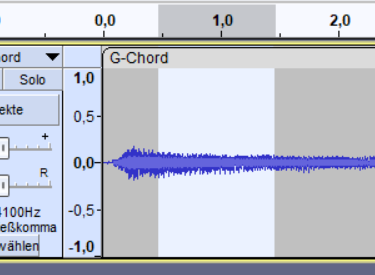
Zur Aufbereitung der Testdaten wurde das Audiobearbeitungsprogramm Audacity und ein eigens entwickeltes Python-Tool genutzt.

Für den Testdurchlauf wurde ein G-Akkord gewählt. Dieser besteht aus den Noten G, B und D. Die Aufnahme erfolgt mit einer Samplerate von 44,1 kHz.

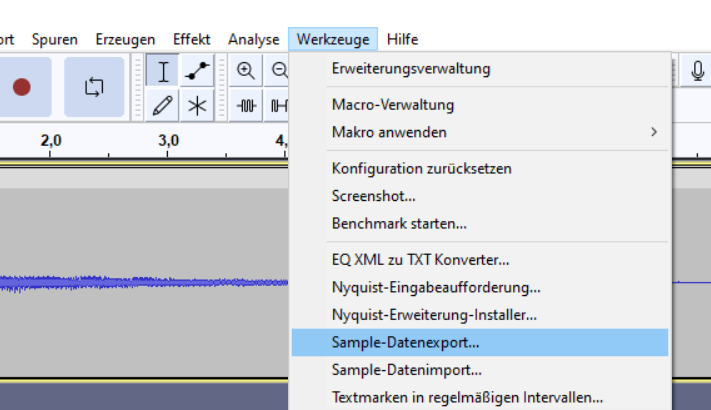
Im Audiobearbeitungsprogramm Audacity sieht das so aus:



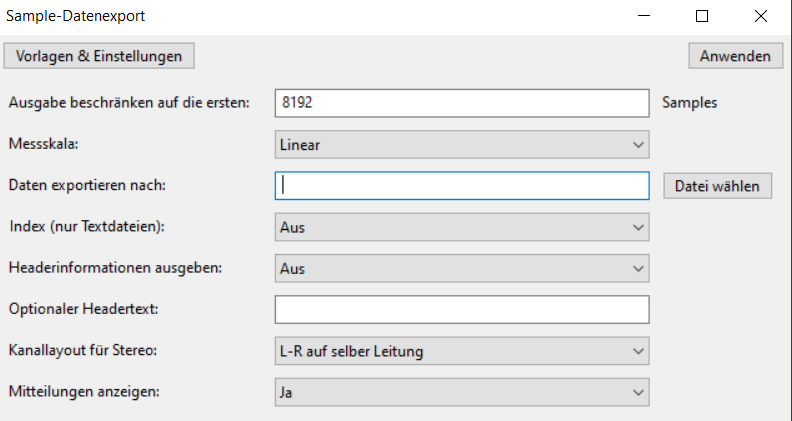
Um nun die Einzelnen Punkte aus dem Signal zu extrahieren, muss ein Bereich im Signal ausgewählt werden.



Anschließend muss in der Menüleiste unter Werkzeuge die Option Sample-Datenexport gewählt werden.



Daraufhin öffnet sich das folgende Fenster.



Hier wird als erste Option die Anzahl der Samplepunkte gewählt. Da im Testprogramm für die maximale Buffergröße NFFT 8192 definiert wurde, wird dies hier ebenfalls gewählt. Die Option Messskala muss auf linear gestellt werden. Als Zielort wird der Pfad zum Python Tool angegeben.

Durch einen Klick auf Anwenden werden die Daten exportiert und das Fenster schließt sich.

Das Python Tool

Am zuvor festgelegten Zielort befindet sich nun die Datei „sample-data.txt“

…

Der Output nach einem Ausführen sieht folgendermaßen aus:

23-th magnitude: 0.539600 => corresponds to 123.815918 Hz

keyNR: 27

46-th magnitude: 0.971720 => corresponds to 247.631836 Hz

keyNR: 39

73-th magnitude: 0.906868 => corresponds to 392.980957 Hz

keyNR: 47

110-th magnitude: 0.510027 => corresponds to 592.163086 Hz

keyNR: 54