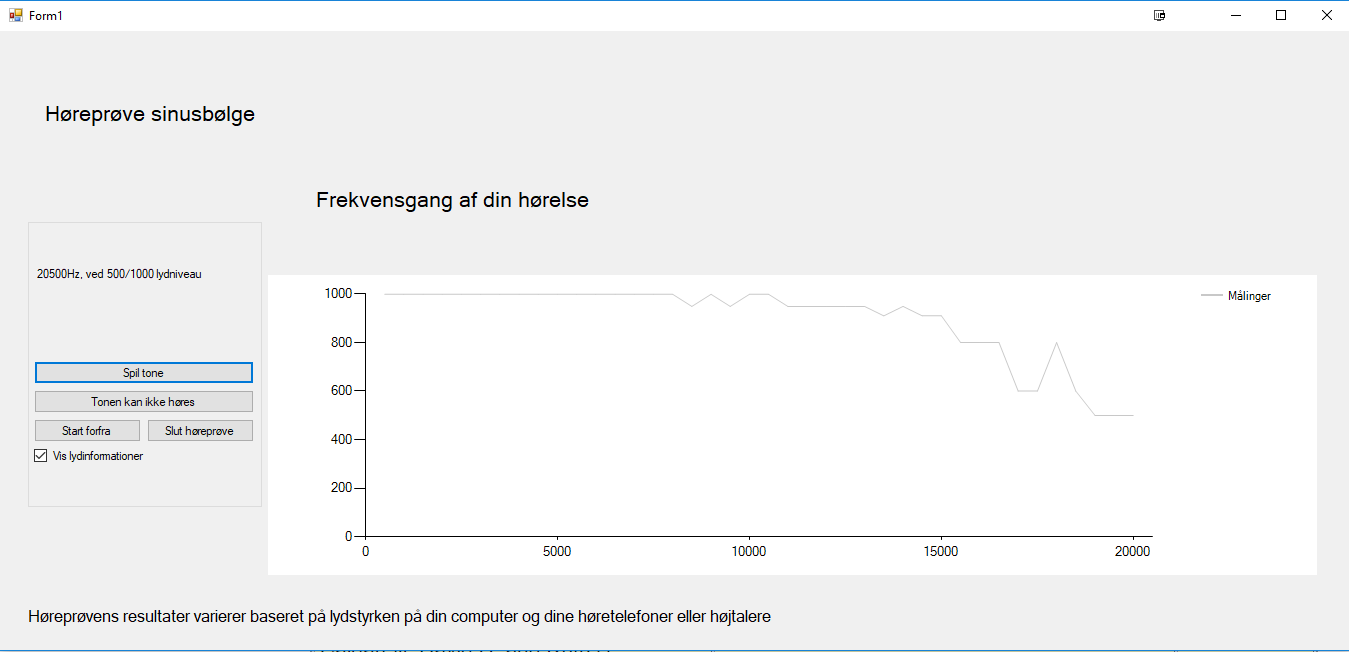
Simpel Høreprøve og sinus-tonegenerator



Lavet af: Markus Repnak Jacobsen

Programmering C: Eksamensprojekt

Skole: H.C. Ørsted Gymnasiet: Lyngby

Lærer: Kristian Krabbe Møller

Klasse: 3.MI2 / 3b2

Afleveres: 22/04/2018

Indholdsfortegnelse

[Synopse 2](#_Toc512196651)

[Indledning 3](#_Toc512196652)

[Projektformulering 3](#_Toc512196653)

[Udviklingsmiljø 3](#_Toc512196654)

[Platform (IDE) og programmeringssproget 3](#_Toc512196655)

[Virkemåde 5](#_Toc512196656)

[Flowchart over koden 5](#_Toc512196657)

[Kode 6](#_Toc512196658)

[Betragtninger og mangler 7](#_Toc512196659)

[Perspektivering 7](#_Toc512196660)

[Konklusion 8](#_Toc512196661)

[Bilag 8](#_Toc512196662)

[Flowchart 9](#_Toc512196663)

[Kode 11](#_Toc512196664)

# Synopse

I projektet har jeg udviklet en sinus-tonegenerator, som kan spille ved forskellige frekvenser og lydniveauer. Tonegeneratoren virker ved at lave array af heltal med et givent ”sample rate” og derefter fylde det op med målinger af den ønskede sinuskurve. Dette konverteres videre til bytes og videre til binært hvor og igen videre til .WAV formatet, som let kan afspilles. Denne tonegenerator er lavet til en funktion og kan derfor let bruges til andre situationer. Jeg har i det her tilfælde lavet en høreprøve til brugeren, som dynamisk laver en frekvensgang af brugerens hørelse. Programmet er lavet i Visual Studio med Windows Forms i C#.

# Indledning

I forbindelse med eksamen i Programmering C, har jeg valgt at lave en tonegenerator. Denne tonegenerator har jeg derefter udbygget, for at lave en simpel sinusbølge-høretest fra 500 til 20000Hz ved forskellige lydniveauer, som resulterer i en ”frekvensgang” af ens egen hørelse i begge ører. Dette er vist på en graf som opdateres dynamisk.

## Projektformulering

I hverdagen er ens hørelse ikke noget som er særlig meget i fokus, trods de mange høje lyde som vi bliver eksponeret for. Dette kan resultere i dårlig hørelse og dermed også degradering i ens evne til at høre forskellige, men specielt, høje frekvenser. I de tidligere år bliver elevers evne til at høre toner ved forskellige frekvenser og lydniveauer afprøvet for at se om der er problemer med hørelsen. Denne mulighed skal man også have senere i livet, uden brug af specialiseret udstyr.

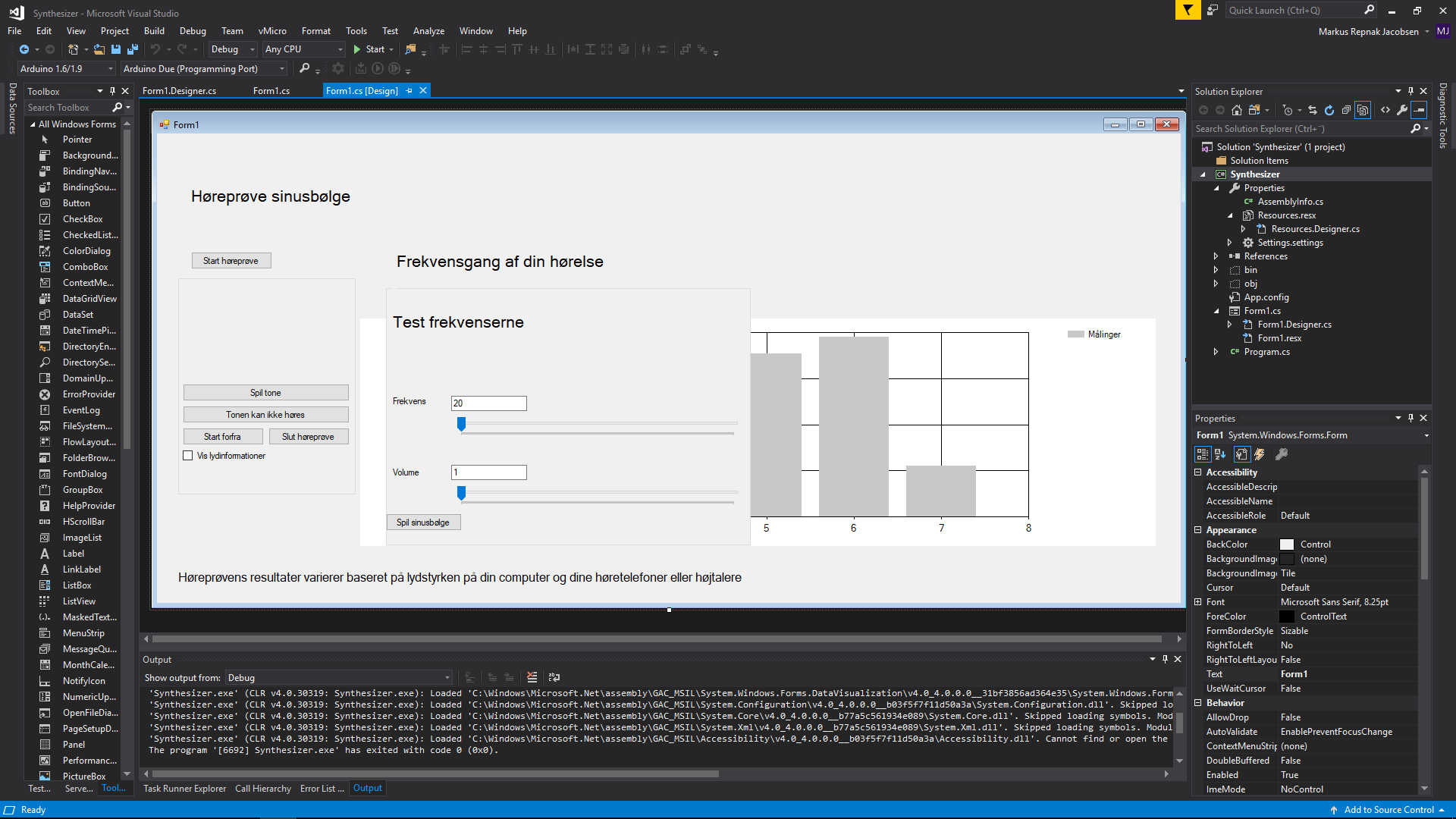
Derfor har jeg udviklet et program som netop gør dette, med ingen krav på specialiseret udstyr. Dette program generer sinusbølger ved forskellige frekvenser og forskellige lydniveauer, for at optimalt vurdere hvor god ens hørelse er.

# Udviklingsmiljø

## Platform (IDE) og programmeringssproget

Programmet er skrevet i Visual Studio[[1]](#footnote-1) i Windows Forms i programmeringssproget C#. C# er et højniveaus-programmeringssprog, som med de mange tilgængelige .NET Framework Class’er i Visual Studio, er meget bredt og lettilgængeligt. Selve koden er selvfølgelig C-baseret.

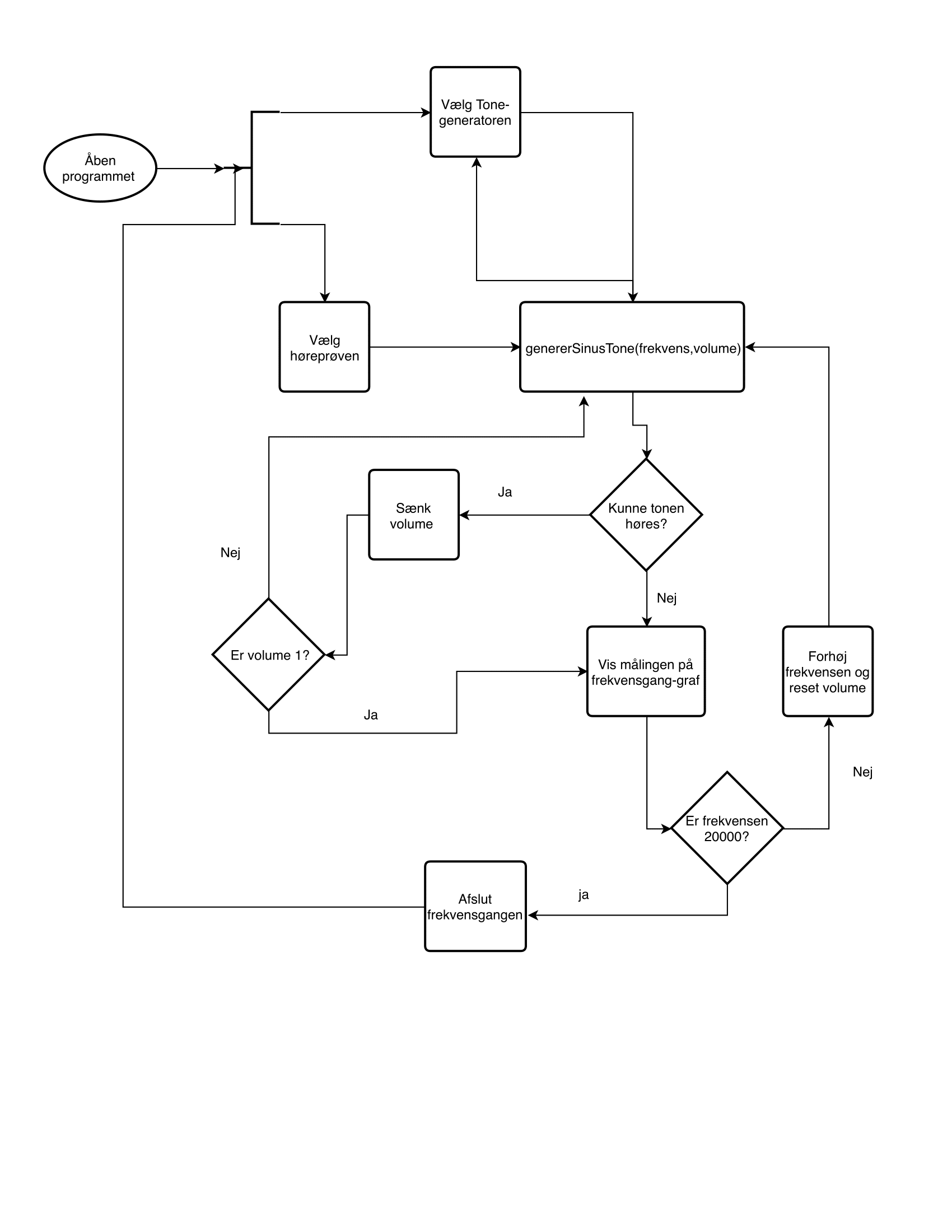
Windows forms giver bred mulighed for at have en grafisk og brugerstyret oplevelse i det endelige program. Forms gør dette ved at give direkte ”drag and drop” muligheder til en række komponenter, f.eks. knapper eller tekstbokse, med automatisk genereret kode i en ekstern, men stadigvæk redigerbar, designer fil. Ud over dette er en lang række andre egenskaber tilgængelige, som f.eks. seriel kommunikation med en mikrokontroller. Visual Studio, Windows Forms og C# er også bredt beskrevet på nettet og gratis at hente. Sproget og økosystemet er også rimelig nyt og derfor et ideelt sprog at kode i.



Figur 1 Screenshot af Visual Studio: Windows Form, i C#s grafiske komponent skærm.

# Virkemåde

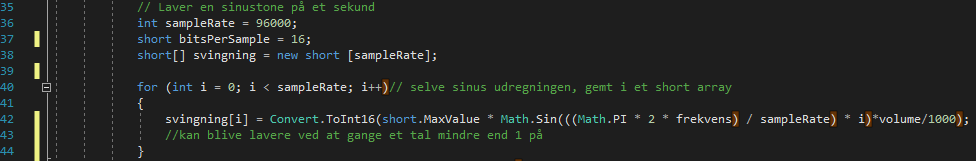
## Flowchart over koden



Programmet startes ved enten at vælge høreprøven eller bruge tonegeneratoren. Hvis tonegeneratoren vælges, skal frekvensen indtastes og lydniveauet vælges. Efter dette kan sinustonen køres ved at trykke på en knap, som kører funktionen: genererSinusTone(frekvens, volume), som bruger de indtastede informationer.

Hvis høreprøven vælges, gemmes tonegeneratoren væk og et nyt layout med passende informationer vises. Herefter skal brugeren trykke på en knap for at høre en tone ved 500Hz. Denne knap skal trykkes indtil tonen ikke kan høres mere eller indtil lydniveauet ikke kan blive lavere. Hvis tonen ikke kan høres, skal brugeren trykke på en knap og dermed bliver frekvensen og lydniveauet indsat i en graf med de givne værdier. Lydniveauet resettes og frekvensen øges med 500Hz. Dette gentages indtil brugeren går ud af testen eller frekvensen er lig med 20000Hz.

## Kode

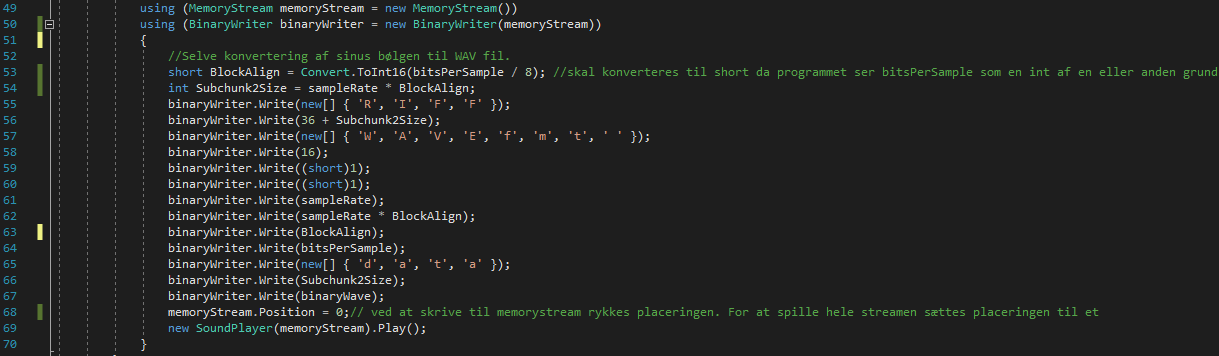
Selve Windows Forms måden at programmere på er lidt ligesom en HTML side med Javascript, koden er altså hovedsageligt ”event” baseret og udfører en række kode ud fra brugerens input. Dette gør jeg vidt brug af i min kode, hovedsageligt til at gemme eller vise elementer eller at køre prædefinerede funktioner. F.eks. når kodens primære funktion: genererSinusTone(frekvens, tone) funktionen bliver kaldet bliver følgende udført:

Figur 2 Matematik algoritmen til generering af sinuskurve

Først vælges hvor mange ”prøver” eller samples der skal tages på et sekund (sampleRate), derefter vælges en opløsning for signalerne (bitsPerSample) og et array som skal indeholde alle disse samples for et minut, som er lige så stort som sampleRate variablen. Derefter bruges en algoritme som bruger den angivne sampleRate, en frekvens og en volume eller lydstyrke. Denne algoritme genererer en sample af gemmer det i variablen ”i”s placering på arrayets plads indtil antal af de angivne samples per minut er udfyldte.

Nu er et array med 96000 pladser fået et heltal i hver plads. Dette skal derefter konverteres til lyd fra heltal. Dette gøres ved at bruge de givne instrukser til at konvertere data til Microsofts WAV lydformat. For at dette kan gøres skal arrayet med heltal først konverteres til bytes:

Figur 3 Konvertering af heltal samples i 16 bit til to opdelte 8 bits

Dette gøres ved at først lave et array af bytes som kan indeholde disse værdier. Dette gøres ved at lave arrayets størrelse de antal samples i heltal arrayet og gange det med hvor mange bytes heltal arrayets opløsning er. I dette tilfælde er en short 16 bit og en byte 8 bit, derfor kan koden oversættes til 96000 \* 2. Derefter bruges Buffer.BlockCopy() til at flytte heltal arrayets værdier til byte arrayets værdier, der angives også offset og hvor mange gange værdierne skal flyttes over.

Figur 4 Konvertering af binært til en WAV fil som derefter afspilles

En MemoryStream åbnes og en BinaryWriter af den MemoryStream laves så binært bliver skrevet ind i den. Nu skrives alle de nødvendige kommandoer[[2]](#footnote-2) for at bruge WAV formatet til BinaryWriteren. MemoryStream placeringen nulstilles og en SoundPlayer bruges til at afspille det binære WAV lydsignal.

Denne funktion bruges videre til at generere toner i høreprøven eller med tonegeneratoren.

Ud over dette består koden mest af en række af forbedringer og rettelser til f.eks. forkerte værdier

# Betragtninger og mangler

## Perspektivering

Selve ideen bag projektet er baseret på ideen om at mennesker gerne vil teste hvor godt de hører og hvilke frekvenser de kan høre. Problemet med denne test er at den ikke fastlåser nogle parametre såsom hvor højt lyden faktisk bliver afspillet eller om ens højtalere filtrerer nogle af frekvenserne ud. Dette kan kun gøres med en dedikeret og kalibreret hardware løsning som ikke kan udsende et potentielt arbitrært lydniveau ud.

Et andet problem med denne test er måske ikke helt kodens skyld, men mere forkert brug af den, som *er* kodens skyld. Lydniveauerne som tonegeneratoren kan udsende, er absurd høje, selv ved lave lydniveauer. Dette resulterer hurtigt i ømme ører, hvilket er ret irriterende. Høreprøven er fastlåst til lydniveau mellem 500 og 1, altså fra 50% til 0,1% og selv her er tonerne meget høje i starten.

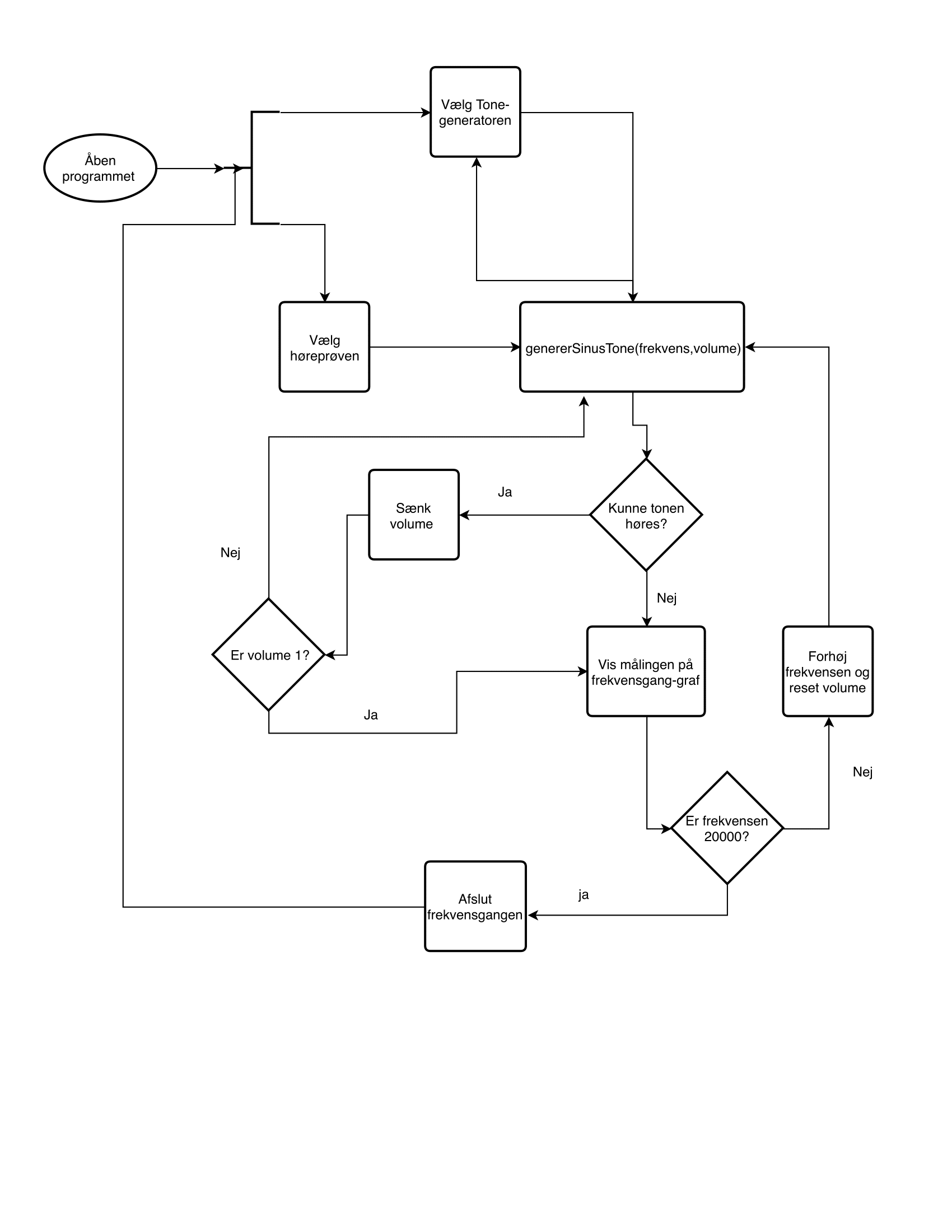
Hvis projektet skulle laves igen, ville jeg undersøge metoder til at kalibrere lydniveauet mere til noget som ikke er arbitrært og mere passende til en ”rigtig” høreprøve.

# Konklusion

Formålet ved projektet var at lave en tonegenerator og derefter implementere den i en høreprøve og dermed også vise resultaterne fra denne prøve i en graf. Dette blev gjort ved at lave et array af værdier fra en sinuskurve ved en given frekvens og lydniveau, efterfulgt af en konvertering af dette array til bytes og derfra videre til binært og ind i WAV lydformatet.

# Bilag

## Flowchart



## Kode

Koden kan findes i arkivet som opgaven ligger i eller på Github, både som Visual Studio projekt og EXE. Dog kan hovedkoden uden rigtig kontekst (Forms.Designer og Forms):

<https://github.com/mj23000/Programmering-Eksamensprojekt>

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using System.Media;

using System.IO;

using System.Windows.Forms.DataVisualization.Charting;

namespace Synthesizer

{

public partial class Form1 : Form

{

public Form1()

{

InitializeComponent();

//Afgrænser grafen

frekvensrespons.ChartAreas[0].AxisX.ScaleView.Zoom(500, 20000);

frekvensrespons.ChartAreas[0].AxisY.ScaleView.Zoom(0, 1000);

frekvensrespons.ChartAreas[0].AxisX.MajorGrid.LineWidth = 0;

frekvensrespons.ChartAreas[0].AxisY.MajorGrid.LineWidth = 0;

frekvensrespons.Series[0].ChartType = System.Windows.Forms.DataVisualization.Charting.SeriesChartType.Line;

}

int frekvensTaeller = 500;

int volume = 500;

public void genererSinusTone(int frekvens, int volume)//frekvens er vilkårlig og volume er maks 1000

{

// Laver en sinustone på et sekund

int sampleRate = 96000;

short bitsPerSample = 16;

short[] svingning = new short [sampleRate];

for (int i = 0; i < sampleRate; i++)// selve sinus udregningen, gemt i et short array

{

svingning[i] = Convert.ToInt16(short.MaxValue \* Math.Sin(((Math.PI \* 2 \* frekvens) / sampleRate) \* i)\*volume/1000);

//kan blive lavere ved at gange et tal mindre end 1 på

}

byte[] binaryWave = new byte[sampleRate \* sizeof(short)];

//En short er 16bit og en byte er 8 bit. for at få 16bit ned i 8bit ganges 16 bit dataen med antal bytes i 16 bits (2), som er sizeof(short)

Buffer.BlockCopy(svingning, 0, binaryWave, 0, sampleRate \* sizeof(short));// igen er hver short delt i to for at få længden i 8bit

//Konverterer svingning (short) til bytes og placerer det i binaryWave

using (MemoryStream memoryStream = new MemoryStream())

using (BinaryWriter binaryWriter = new BinaryWriter(memoryStream))

{

//Selve konvertering af sinus bølgen til WAV fil.

short BlockAlign = Convert.ToInt16(bitsPerSample / 8); //skal konverteres til short da programmet ser bitsPerSample som en int af en eller anden grund

int Subchunk2Size = sampleRate \* BlockAlign;

binaryWriter.Write(new[] { 'R', 'I', 'F', 'F' });

binaryWriter.Write(36 + Subchunk2Size);

binaryWriter.Write(new[] { 'W', 'A', 'V', 'E', 'f', 'm', 't', ' ' });

binaryWriter.Write(16);

binaryWriter.Write((short)1);

binaryWriter.Write((short)1);

binaryWriter.Write(sampleRate);

binaryWriter.Write(sampleRate \* BlockAlign);

binaryWriter.Write(BlockAlign);

binaryWriter.Write(bitsPerSample);

binaryWriter.Write(new[] { 'd', 'a', 't', 'a' });

binaryWriter.Write(Subchunk2Size);

binaryWriter.Write(binaryWave);

memoryStream.Position = 0;// ved at skrive til memorystream rykkes placeringen. For at spille hele streamen sættes placeringen til et

new SoundPlayer(memoryStream).Play();

}

}

void hoereProeve()

{

//Tonegeneratoren gemmes og frekvens og

sinusTestGroupbox.Hide();

startProeve.Hide();

MessageBox.Show("Du har nu startet høreprøven.");

MessageBox.Show("Tryk på: 'spil tone' knappen, indtil du ikke kan høre tonen mere.");

MessageBox.Show("Når du ikke kan høre tonen, tryk på knappen: 'tone kan ikke høres mere'");

hoereproeveGroupbox.Show();

frekvensrespons.Show();

tekstProeve.Show();

proeveFrekvensgang.Show();

frekvensTaeller = 500;

volume = 500;

frekvensrespons.Series[0].Points.Clear();

}

private void startProeve\_Click(object sender, EventArgs e)

{

hoereProeve();

}

private void sinusVolumeBoks\_TextChanged(object sender, EventArgs e)

{

if (sinusVolumeBoks.Text != "")

{

if (Convert.ToInt32(sinusVolumeBoks.Text) > 1000)

{

MessageBox.Show("Den indtastede volumen er for høj. Volumen er sat til 1000");

sinusVolumeBoks.Text = "1000";

}

else sinusVolume.Value = Convert.ToInt32(sinusVolumeBoks.Text);

}

}

private void sinusFrekvens\_Scroll(object sender, EventArgs e)

{

sinusFrekvensBoks.Text = Convert.ToString(sinusFrekvens.Value);

}

private void sinusVolume\_Scroll(object sender, EventArgs e)

{

sinusVolumeBoks.Text = Convert.ToString(sinusVolume.Value);

}

private void sinusFrekvensBoks\_TextChanged(object sender, EventArgs e)

{

if(sinusFrekvensBoks.Text != "")

if (sinusFrekvensBoks.Text != "")

{

if (Convert.ToInt32(sinusFrekvensBoks.Text) > 20000)

{

MessageBox.Show("Den indtastede frekvens er for høj. Frekvensen er sat til 20000");

sinusFrekvensBoks.Text = "20000";

}

else sinusFrekvens.Value = Convert.ToInt32(sinusFrekvensBoks.Text);

}

}

private void sinusKnap\_Click(object sender, EventArgs e)

{

if(sinusVolumeBoks.Text != "" && sinusFrekvensBoks.Text != "") genererSinusTone(Convert.ToInt32(sinusFrekvensBoks.Text), sinusVolume.Value);

}

private void spilTone\_Click(object sender, EventArgs e)

{

genererSinusTone(frekvensTaeller, volume);

}

private void toneHoeres\_Click(object sender, EventArgs e)

{

if (frekvensTaeller > 20000) MessageBox.Show("Du har nu gennemført høreprøven. Se grafen til højre for at se dine resultater");

else

{

if (volume == 1)

{

frekvensrespons.Series[0].Points.AddXY(frekvensTaeller, 1000 - volume);

MessageBox.Show("Du kan høre tonen på alle lydniveauer!");

frekvensTaeller = frekvensTaeller + 500;

volume = 500;

}

if (volume >= 200) volume = volume - 100;

if (volume <= 200) volume = volume - 50;

if (volume <= 50) volume = volume - 10;

if (volume <= 20) volume = volume - 5;

if (volume <= 5) volume = 1;

genererSinusTone(frekvensTaeller, volume);

}

lydInformationProeve.Text = Convert.ToString(frekvensTaeller + "Hz, ved " + volume + "/1000 lydniveau ");

}

private void toneHoeresIkke\_Click(object sender, EventArgs e)

{

if (frekvensTaeller > 20000) MessageBox.Show("Du har nu gennemført høreprøven. Se grafen til højre for at se dine resultater");

else

{

if (volume >= 200) volume = volume + 100;

if (volume <= 200) volume = volume + 50;

if (volume <= 50) volume = volume + 10;

if (volume <= 20) volume = volume + 5;

MessageBox.Show("Resultatet registreret. Tryk videre på spil tone");

frekvensrespons.Series[0].Points.AddXY(frekvensTaeller, 1000 - volume);

frekvensTaeller = frekvensTaeller + 500;

volume = 500;

lydInformationProeve.Text = Convert.ToString(frekvensTaeller + "Hz, ved " + volume + "/1000 lydniveau ");

}

}

private void slutProeve\_Click(object sender, EventArgs e)

{

sinusTestGroupbox.Show();

startProeve.Show();

hoereproeveGroupbox.Hide();

frekvensrespons.Hide();

tekstProeve.Hide();

proeveFrekvensgang.Hide();

frekvensrespons.Series[0].Points.Clear();

}

private void startForfra\_Click(object sender, EventArgs e)

{

frekvensTaeller = 500;

volume = 500;

frekvensrespons.Series[0].Points.Clear();

}

private void checkBox1\_CheckedChanged(object sender, EventArgs e)

{

if (lydinformationerCheckbox.Checked == true) lydInformationProeve.Show();

else lydInformationProeve.Hide();

lydInformationProeve.Text = Convert.ToString(frekvensTaeller + "Hz, ved " + volume + "/1000 lydniveau ");

}

}

}

1. Kraftigt koderedigeringsprogram eller IDE mm. <https://www.visualstudio.com/> [↑](#footnote-ref-1)
2. <http://soundfile.sapp.org/doc/WaveFormat/> [↑](#footnote-ref-2)