|  |  |
| --- | --- |
| UiS%5Fnor%5Fcolor  **DET TEKNISK-NATURVITENSKAPELIGE FAKULTET**  **BACHELOROPPGAVE** | |
| Studieprogram/spesialisering: | Vårsemesteret, 2015  Åpen / ~~Konfidensiell~~ |
| Forfatter:  Cecilie M. Langfeldt  Helene H. Larsen | …………………………………………  (signatur forfatter) |
| Fagansvarlig: Tom Ryen  Veileder(e): Tom Ryen | |
| Tittel på bacheloroppgaven: Musikknotasjon ved bruk av smarttelefon  Engelsk tittel: Music notation using smartphone | |
| Studiepoeng: 20sp | |
| Emneord:    Good  Innovative!  Funs to be had1! | Sidetall: …………………    + vedlegg/annet: …………  Stavanger, ………………..  dato/år |

## Sammendrag

Løsningen på oppgaven ble appen VeryNote for Android. Brukeren blir presentert med et enkelt UI, hvor han/hun starter opptaket, sangen blir analysert og noter/pauser skrevet ut på skjermen i sanntid. Det finnes en meny med innstillinger for metronomen, og brukeren kan velge mellom g-nøkkel og f-nøkkel. Etter opptaket kan notene redigeres som ønsket, tilslutt kan sangen lagres på enheten som en MusicXML fil og/eller deles for eksempel over email eller bluetooth.

## Innledning

### Bakgrunn og problemstilling

Problemstillingen og tittelen for oppgaven er «Musikknotasjon ved bruk av smarttelefon». Målet med oppgaven var å lage en app for Android OS som kunne ta opp sang/plystring fra mikrofonen, finne fundamental frekvens i en gitt tidsperiode og omsette denne frekvensen til en note. Notene skulle så skrives ut på mobilskjermen i et notesystem. Det skulle også være mulig å eksportere notene som en MusicXML fil. Dersom tiden strakk til, var det foreslått og legge til redigering og avspilling av notene etter brukeren har sunget ferdig.

Vi valgte denne oppgaven fordi programmering for Android OS hørtes ut som en spennende utfordring, samtidig som vi kunne få brukt kunnskapene vi har om Java. Vi har begge litt erfaring med musikk og noter fra tidligere.

### Innholdsfortegnelse

[Sammendrag 2](#_Toc418590646)

[1. Innledning 3](#_Toc418590647)

[1.1. Bakgrunn og problemstilling 3](#_Toc418590648)

[1.2. Innholdsfortegnelse 4](#_Toc418590649)

[2. Digital signalbehandling og musikkteori 5](#_Toc418590650)

[2.1. DFT - Discrete Fourier Transform og FFT – Fast Fourier Transform 5](#_Toc418590651)

[2.2. Frekvenser i Musikk 8](#_Toc418590652)

[2.3. Musikkteori 10](#_Toc418590653)

[3. Programmeringsverktøy 14](#_Toc418590654)

[3.1. Android 14](#_Toc418590655)

[3.2. Android Studio 14](#_Toc418590656)

[3.3. Mobiltelefoner for testkjøring 15](#_Toc418590657)

[4. Implementasjon av teori 15](#_Toc418590658)

[4.1. JTransform/PitchDec.java 15](#_Toc418590659)

[4.2. NoteSearch 17](#_Toc418590660)

[4.3. Dynamisk endring av notelengde 18](#_Toc418590661)

[5. Appens oppbygging 18](#_Toc418590662)

[5.1. Tråder og handlers 18](#_Toc418590663)

[5.2. Sentrale funksjoner 19](#_Toc418590664)

[5.3. XML 20](#_Toc418590665)

[5.4. UI 21](#_Toc418590666)

[6. Appens virkemåte 24](#_Toc418590667)

[6.1. Hovedskjerm 24](#_Toc418590668)

[6.2. Redigering 26](#_Toc418590669)

[6.3. Meny 27](#_Toc418590670)

[7. Konklusjon og videre arbeid 30](#_Toc418590671)

[7.1. Konklusjon 30](#_Toc418590672)

[7.2. Videre arbeid 30](#_Toc418590673)

[Vedlegg 32](#_Toc418590674)

[Referanser 33](#_Toc418590675)

## Digital signalbehandling og musikkteori

### DFT - Discrete Fourier Transform og FFT – Fast Fourier Transform

Frekvensanalysen av lyden som spilles inn blir utført ved hjelp av fouriertransformasjon [1][2][5] . Fouriertransformasjon brukes for å gjøre signaler fra tidsdomenet om til frekvensdomenet, og er definert for både kontinuerlige og diskrete signaler. Lyden som tas opp fra mikrofonen er i prinsipp kontinuerlig så lenge mikrofonen er på, men det må plukkes ut en endelig mengde verdier til analyse. DFT, diskret fouriertransformasjon, er metoden som benyttes når det finnes en endelig mengde datapunkt.

DFT er definert ved

FFT, eller Fast Fourier Transform, er en algoritme for å utføre diskrete fouriertransformasjoner mer effektivt. Den gjør akkurat det samme som DFT, men det går mye raskere. Hele operasjonen deles opp i flere små DFT'er. Litt interessant er det at en slik algoritme ble oppdaget allerede i 1805 av Carl Friedrich Gauss, men så lenge det ikke fantes datamaskiner så var det ikke interessant og den ble glemt. I 1965 ble den plukket opp igjen av J.W. Cooley og J. Turkey og igjen lansert i deres artikkel «An algorithm for the machine calculation of complex fourier series».

En vanlig DFT tar O(N2) operasjoner, mens en FFT gjør den samme jobben med O(N*log*N). Dette betyr at med store datasett er tidsbesparelsen enorm.

.....mer om O(n)

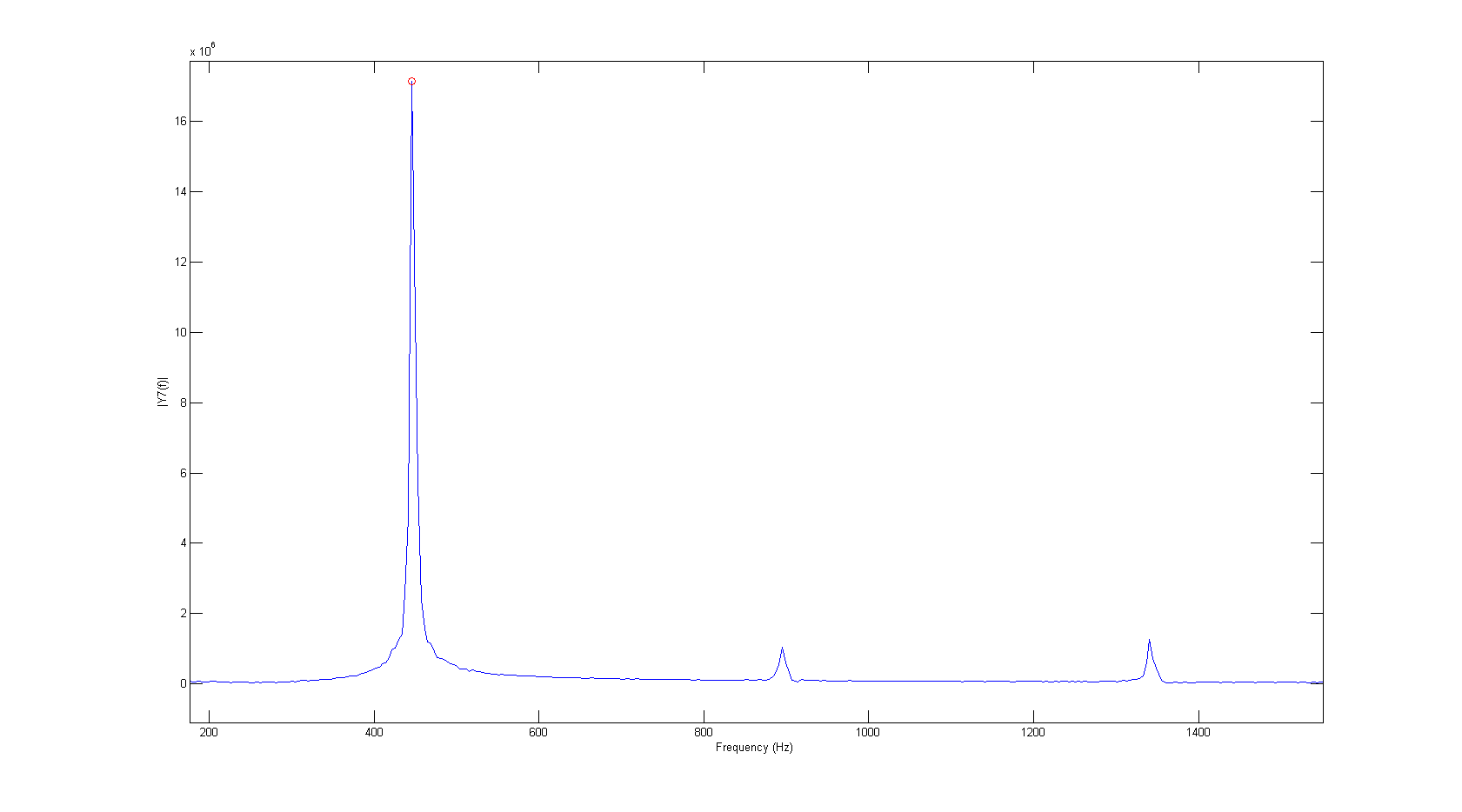
For å fullt utnytte denne algoritmen må antall punkter N være et av tallene 2n hvor n er et heltall. I frekvensanalysen appen gjør jobber den med 4096 (212) datapunkter og det er derfor mulig å utnytte FFT for analysen.

Dataene som appen henter fra mikrofonen deles opp i «chunks», altså en endelig mendge datapunkter, og det utføres en FFT på hver chunk for å finne frekvensen.



Figur 2.1 – Representasjon av dataene fra et typisk lydopptak, 4096 samples over 256ms i tidsdomenet.

Verdier fra et opptak med mobilens mikrofon ble hentet ut og lagret, både før og etter FFT. Disse ble så plottet i grafer ved hjelp av MATLAB (vedlegg 2). Resultatet av dette er grafene i figur 2.1 og 2.2.

DFT tar dataene fra tidsdomenet og gir oss en representasjon av de samme dataene i frekvensdomenet. Figur 2.2 viser funksjonen fra figur 2.1 etter FFT, og man kan se en representasjon av amplituden til de ulike frekvensene fra det originale signalet.

Figur 2.2 – Lydopptak etter FFT, representerer de samme dataene som i figur 2.1 etter transformasjon til frekvensdomenet.

På denne figuren ser det ut som det finnes flere markerte frekvenser. Dette er et resultat av at et lydopptak har en fundamental frekvens, og flere overharmoniske frekvenser. Hvis vi har en lyd som er 440hz, har den overharmoniske frekvenser på 880hz og 1320hz.

Et problem med å bruke FFT til frekvensanalyse er oppløsningen på resultatene, særlig ved lavere frekvenser. Frekvensene som oppfattes som de ulike tonene i skalaen er ikke lineære, men logaritmiske. En forskjell på 100hz ved lavere frekvenser tilsvarer en hel oktav, men ved høyere frekvenser er det over 100hz mellom hver enkelt note. (Se tabell 2.1 for en fullstending oversikt). FFT tar ikke hensyn til dette, derfor er nøyaktigheten mye bedre ved høyere frekvenser enn ved lavere frekvenser. Dette er et problem fordi det er gjerne de lavere frekvensene vi er mest interessert i ved frekvensanalyse av sang.

Et annet problem er at en må gjøre et kompromiss mellom høy oppløsning i frekvensdometet, og høy oppløsning i tidsdomenet. Frekvensanalysen som blir utført i dette tilfellet er utført på 4096 sampler med en samplingsrate på 16000. Dette vil si at hver FFT blir utført over et intervall på (1000ms\*4096)/16000 = 256ms. Frekvensoppløsningen er 16000/4096 ≈ 4hz. De laveste frekvensene det letes etter i denne analysen er 131hz og 139hz, derfor er denne oppløsningen god nok i dette tilfelle.

For et mer nøyaktig resultat i frekvensdomenet, desto flere datapunkter må analyseres og antall sampler må økes. Hvis antall sampler dobles til 8192 gir dette med en samplingsrate på 16000 en frekvensoppløsning på 16000/8192 ≈ 2hz. Kompromisset en da må gjøre er at hver FFT blir utført på et intervall på (1000\*8192)/16000 = 512ms, og blir derfor mindre nøyaktig i tidsdomenet. På den andre siden vil en bedre oppløsning i tidsdomenet gi en dårligere oppløsning i frekvensdomenet. Et lavere antall sampler, som for eksempel 1024 vil gi et tidsintervall på (1000\*1024)/16000 = 64ms, men da blir frekvensoppløsningen 16000/1024 ≈ 16hz som ikke vil være nok for å skille de laveste frekvensene.

Det finnes altså ikke et entydig svar på hvilket antall sampler som gir best resultat i frekvensanalyse, det må gjøres et valg alt etter hvilken type lyd man analyserer. I tilfellet for sang gir 8192 sampler en tilfredsstillende frekvensoppløsning, samtidig som at det ikke gir en stor unøyaktighet i tidsdomenet. Alternativene for samplingsrate er mer begrenset, det avhenger av enheten som gjør lydopptaket. Valget falt på 16000hz som ga nøyaktige nok resultater fra frekvensanalysen, samtidig som det fungerte på alle de forskjellige enhetene som var tilgjengelig for testing.

### Frekvenser i Musikk

Det mennesker oppfatter som lyd, er trykkbølger i luften som oppstår fra vibrasjoner av for eksempel stemmebånd eller en gitarstreng og kalles gjerne lydbølger. Når disse bølgene når trommehinnen i øret, oppfattes de som lyd.

Musikalske lyder kan representeres som en sinusfunksjon, de repeteres periodisk med en frekvens som tilsvarer forskjellige toner. I figur 2.1 kan det være vanskelig å se at det dreier seg om bølger, men zoomet inn kan bølgene tydelig sees.



Figur 2.3 – Zoomet inn område av figur 2.1

Bølgelengde λ er avstanden mellom to bølgetopper. Frekvens *f* er hvor mange ganger en hendelse gjentar seg i løpet av et bestemt tidsrom, i tilfellet med lyd altså hvor mange bølgetopper som passerer per tidsenhet.

Hvis frekvensen av lyden er 440hz oppfattes dette som en A. Hvis frekvensen er 220hz, er dette også en A, men en oktav lavere.

Tabell 2.1 viser en oversikt over frekvensene til notene i de fire oktavene VeryNote er laget for å gjenkjenne.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Lille oktav** | **Enstrøken oktav** | **Tostrøken oktav** | **Trestrøken oktav** |
| **C** | 131 | 262 | 523 | 1047 |
| **C#** | 139 | 277 | 554 | 1109 |
| **D** | 147 | 294 | 587 | 1175 |
| **E♭** | 156 | 311 | 622 | 1245 |
| **E** | 165 | 330 | 659 | 1319 |
| **F** | 175 | 349 | 698 | 1397 |
| **F#** | 185 | 370 | 740 | 1480 |
| **G** | 196 | 392 | 784 | 1568 |
| **G#** | 208 | 415 | 831 | 1661 |
| **A** | 220 | 440 | 880 | 1760 |
| **B♭** | 233 | 466 | 932 | 1864 |
| **H** | 247 | 493 | 988 | 1976 |

Tabell 2.1 – Noter med tilhørende frekvenser i hertz. [3]

### Musikkteori

En kort gjennomgang av musikkteorien [4] brukt i appen.

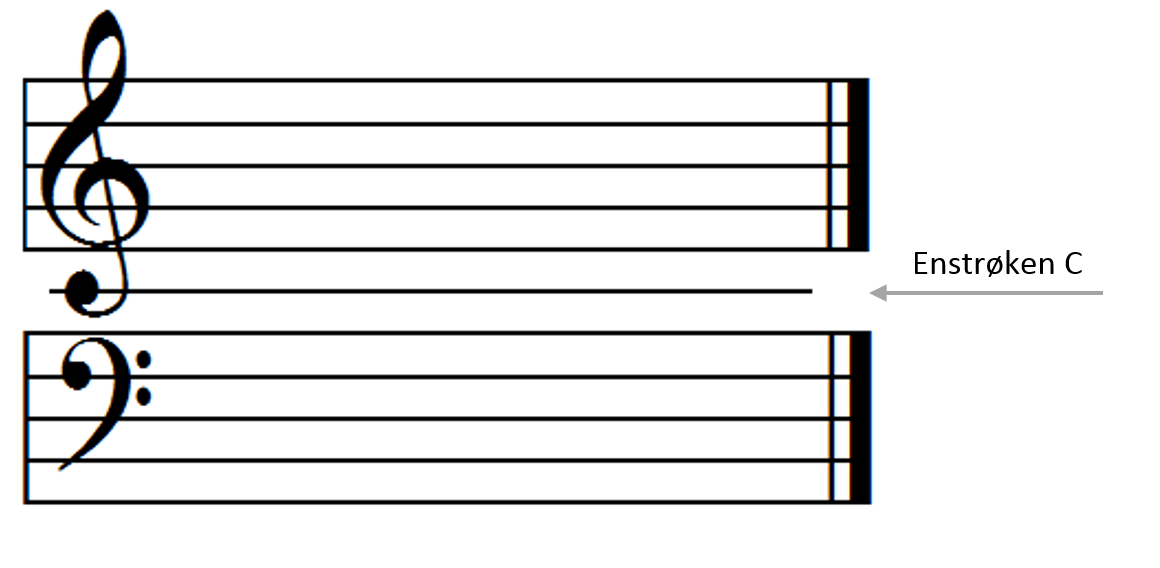
#### Bass/treble

Linjene i notesystemet representerer hvilken note som skal spilles, men bare linjene er ikke nok. Får å vite hvilke noter som ligger i notesystemet, må det være markert med en nøkkel. G-nøkkelen markerer streken for enstrøken G. Figur 2.4 viser et eksempel på dette, alle notene i denne figuren er skrevet på linjen som representerer enstrøken G.



Figur 2.4 – Notesystem med G-noter.

F-nøkkelen markerer med en prikk på hver side av linjen som lille F skrives på. Utfra dette kan vi fort finne frem til hvilke noter som ligger i notesystemet. Hvis man skriver disse notesystemene i forhold til hverandre, ser man at de kun har en hjelpe linje mellom seg, se figur 2.5.



Figur 2.5 – Notesystemer med G- og F-nøkkel.

#### Ulik lengde på noter

Hvilke note som skal spilles blir markert med plasseringen av noten, mens varigheten blir bestemt av utsende. Som det kommer av tabell 2.2 er varigheten til noten relativ til lengden på en hel takt. En takt består av de noter eller pauser som befinner seg mellom to taktstreker.

|  |  |  |
| --- | --- | --- |
| navn | bilde | varighet |
| Sekstendelsnote | s | 1/16 |
| Åttedelsnote | e | 1/8 |
| Firedelsnote | q | 1/4 |
| Halvnote | h | 1/2 |
| Helnote | w | 1/1 |
| Halvpause | H | 1/2 |
| Helpause | W | 1/1 |

Tabell 2.2 – Noter med tilhørende navn og varighet.

Varigheten til noten kan endres enda mer når man tar i bruk punktering. Tabell 2.3 viser et par eksempler på hvordan punktering fungerer. En note som er punktert blir sin egen lengde pluss halvparten av sin egen lengde.

|  |  |  |  |
| --- | --- | --- | --- |
| Punktert note |  | Ny verdi | lengde |
| R | = | w+ h | 3/2 |
| d | = | h+ q | 3/4 |
| j | = | q+ e | 3/8 |
| i | = | e+ s | 3/16 |

Tabell 2.3 – Punkterte noter med tilhørende varighet.

#### Pauser

Stillhet i musikk har selvsagt sine egne markeringer. På samme måte som notene, har de forskjellige tegn for forskjellige varighet. Appen har foreløpig kun implementert to pauser, den som varer en hel takt og den som varer en halv takt. Begge disse pausene blir representert av et lite svart rektangel. Pausen som varer hele takten ligger under en av linjene i notesystemet, mens pausen som varer halve takten ligger oppå denne linjen.

#### Fortegn

En note kan markers med flere fortegn, de tre som brukes I appen er “b”, “#” og “ ♮”. Hvis en note er markert med en «b» skal den notens tonehøyde senkes med en halvnote. Fortegnet «#» derimot, *øker* notens tonehøyde med en halvnote. Disse to fortegnene varer ut den takten de er skrevet i. Figur 2.5 viser et kort «stykke» med 8 noter i. De første to notene er G –er. Den neste noten er markert med en «#» så den er selvsagt en halvnote høyere enn en G, en giss. Siden disse fortegnene påvirker alle de gjenværende notene innenfor den takten, er den siste noten i denne takten også er en giss. På andre siden av taktstreken påvirker ikke fortegnet notene lenger og disse notene er vanlige G –er. Hvis man kun ville hatt én giss i notesystemet i figur 2.5 måtte man tatt i bruk det siste fortegnet, «n». Dette fortegnet setter bort alle tidligere fortegn. Hvis den siste noten i den første takten hadde vært markert med et «n» tegn, ville den også vært en G.

## Programmeringsverktøy

### Android

VeryNote er laget for bruk på mobiltelefoner som kjører Android operativsystem. Den siste versjonen er i skrivende stund Lollipop 5.1 (SDK 22), man må minst ha versjon 4.0 IceCreamSandwich (SDK 14) for å bruke VeryNote.

Hvis man velger å kun utvikle for den siste versjonen så risikerer man at veldig få vil kunne bruke appen, mens hvis man velger en veldig gammel versjon mister man muligheten til å bruke metoder og funksjoner som finnes i nyere API. Rundt 90% av alle enheter som kjører Android har idag versjon 4.0 eller nyere [6], vi valgte derfor å ha dette som minimumsversjon for at appen skulle kunne brukes på flest mulig enheter, samtidig som vi har et nyere API å jobbe med.

### Android Studio

Android Studio er den offisielle IDE for å skrive applikasjoner for Android ved bruk av programmeringsspråket Java. Det fungerer på mange måter som Eclipse for Java, men inneholder i tillegg flere funksjoner som er nyttige i utviklingen av en app. Det finnes en emulator man kan teste apper på uten å måtte koble til en mobiltelefon, og det er enkelt å sette opp en fungerende hello-world app for videre utvikling. Når man jobber med brukergrensesnitt kan man plassere knapper og tekstbokser rett på en forhåndsvisning, så ser man med en gang hvordan det vil se ut.

Det finnes også et verktøy som heter Memory Monitor, hvor man kan se i sanntid hvor mye minne appen bruker. For testing av apper er det mulig å kjøre de på en emulator, men denne har begrensede funksjoner, som for eksempel det finnes ingen mikrofon.

### Mobiltelefoner for testkjøring

For å teste appen underveis ble det brukt 4 forskjellige mobiltelefoner med ulike android operativsystemer. Mobilene hadde også ulike skjermstørrelser og oppløsninger, dette fordi det var veldig viktig å få testet grundig at notene fikk riktig plassering på skjermen i forhold til notelinjene.

Mobilene appen er testet på er:

* Samsung Galaxy Trend GT-S7560 – Android OS 4.0.4 Ice Cream Sandwich
* Samsung Galaxy Note 2 GT-N7100 – Android OS 4.4.2 KitKat
* Samsung Galaxy S4 GT-I9505 – Android OS 4.4.2 KitKat
* Samsung Galaxy S4 GT-I9506 – Android OS 5.0.1 Lollipop

## Implementasjon av teori

### JTransform/PitchDec.java

JTransform [8] er et Java bibliotek med åpen kildekode. Det inneholder fire forskjellige transformasjoner, Discrete Fourier Transform (DFT), Discrete Cosine Transform (DCT), Discrete Sine Transform (DST) og Discrete Hartley Transform (DHT). Vi har benyttet oss av klassen DoubleFFT\_1D og metoden complexForward(double[] a).

PitchDec.java er skrevet av Aleksey Surkov [9] og er skrevet som en del av et program for en gitarstemmer. Denne koden er modifisert av oss for bruk i denne appen.

Når brukeren trykker på Record-knappen begynner mikrofonen og ta opp lyd i 16bit PCM format. PCM, eller Pulse-code modulation, er en digital representasjon av et analogt signal. For å unngå at bakgrunnsstøy blir analysert må amplituden på lyden være over 4400. Når lyden blir gjort om til 16bit PCM, representerer tallene amplituden på lyden, og vi fant ved hjelp av testing at 4400 var et tall som eliminerte bakgrunnstøy uten at en må rope unødvendig høyt inn i mikrofonen for at resultatene skal bli analysert og vises på skjermen.

Opptaket fra mikrofonen blir lagret i et array, som det så blir gjort en FFT på. Vi leter etter den fundamentale frekvensen mellom 131hz og 1976hz. Dette dekker 4 oktaver fra C til H. VeryNote er i hovedsak laget for å analysere opptak av sang, derfor er frekvenser utenfor dette spekteret mindre interessante og ikke inlkudert.

Deretter brukes resultatene fra FFT for å finne frekvensen med høyest amplitude. Dette skjer i filen PitchDec.java. Som sagt tidligere, selv om lyden i prinsipp er et kontinuerlig signal, må den deles opp i biter som kan analyseres. Opptaket blir gjort med en «chunksize» på 4096, og raten er 16000. Det betyr at hver «chunk» går over (1000\*4096)/16000 = 256ms. Hver «chunk» blir så analysert vha FFT. Se figur 2.2. Hvis vi zoomer inn ser man tydelig at den fundamentale frekvensen i dette tilfellet er mellom 440 og 450hz. Se figur 4.1.



Figur 4.1 – Zoomet inn område av figur 2.2

Frekvensen med høyest amplitude blir lagt til en «cluster». Siden vinduet analysene blir utført på er så små, vil det normalt være flere vindu med samme frekvensresultat etter hverandre. Metoden isNearblir kalt for å sjekke om frekvensen er mindre enn 5% forskjellig fra forrige frekvens. Hvis den er mindre enn 5% forskjellig blir den tolket som en fortsettelse av den samme noten og lagt til samme «cluster». Hvis den er mer enn 5% forskjellig blir den tolket som en ny note og en ny «cluster» blir opprettet.

Et normalt FFT resultat vil vise fundamental frekvens, i tillegg til frekvensens overharmoniske frekvenser. Se figur 2.2

For å unngå at overharmoniske frekvenser blir oppfattet som egne toner, slåes «clusterene» som inneholder frekvenser som er overharmoniske sammen vha metoden isHarmonic*.* Denne metoden sjekker harmonisk faktor, og om denne er over 5% forskjellig. Hvis den er mindre enn 5% blir det oppfattet som en overharmoni av den forrige frekvensen og lagt til samme «cluster». Hvis forskjellen er større enn 5% er den ikke overharmonisk og fortsetter og være sin egen «cluster».

Etter at like frekvenser og overharmoniske frekvenser er luket bort finnes den frekvensen med høyest amplitude og denne sendes videre i programmet.

### NoteSearch

Notene blir lagret som objekter med informasjon om blant annet hertz-verdi og om de er sharp/flat. Det blir også laget et hashmap notes i filen NoteSearch.java med frekvensverdier som brukes for å finne hvilken note som er nærmest gitt frekvens. Etter at frekvensen er funnet i PitchDec.java er det ikke sikkert denne er helt lik en eksakt notefrekvens, da kalles metoden findNearestNote som sammenligner differansene mellom funnet frekvens og de som finnes lagret i notes for å finne den noten med minste differanse mellom funnet frekvens og lagret eksakt frekvens. Denne blir så returnert for så å skrives ut på skjermen.

### Dynamisk endring av notelengde

Notens utsende blir bestemt av metoden noteLength og i stor grad variabelen dur, kortform av «duration». Første gang den noten blir registret blir notesOnScreen kalt, denne metoden, blant annet, setter dur lik null og kaller noteLength. Bildet til noten blir satt til å være et gjennomsiktig bilde som er like stort som bildene med noter. Hvis showPitchDetectionResult neste gang blir kalt med den samme noten, kaller den noteLength direkte. noteLength ser på variabelen dur og bestemmer ut ifra dette hvilket bilde som vises på skjermen. Variabelen dur blir sett på i forhold til fullBar som er tiden innenfor en takt. Notens varighet endres derfor med metronomen, når en takt varer lenger må noten holdes lenger får å vises på skjermen.

## Appens oppbygging

### Tråder og handlers

Appen består av to tråder. Det består av en hovedtråd og en tråd for frekvensanalysen kalt pitch\_detector\_thread\_. Hovedtråden styrer oppdateringer av brukergrensesnittet i tillegg til alt som ikke har med frekvensanalysen å gjøre. Siden det konstant må gjøres oppdateringer på UI, er det viktig at disse oppdateringene ikke blir blokkert. For å unngå blokkeringer er det opprettet handlere for kjørbare objekter. Når en ny handler er opprettet er den bundet til tråden som opprettet den. Handlere kan brukes til å kalle metoder etter en gitt tid. Denne funksjonen er blitt brukt i blant annet metronomfunksjonen. Når metronomen blir slått på, kaller den en «runnable» som sørger for at telleren vises på skjermen, og at telefonen vibrerer i riktig intervall. Denne kaller så seg selv igjen ved hjelp av metoden postDelayed etter det gitte tidsintervallet, helt til metronomen slåes av. Dette gjør at metronomen kan kjøre i bakgrunnen uten at hovedtråden blir blokkert. Handlere kan også sende meldinger til hovedtråden. Denne funksjonen brukes etter at frekvensanalysen er gjennomført og notene skal vises på skjermen, metoden post sender resultatene fra ShowPitchDetectionResults til hovedtråden slik at de kan vises på skjermen.

### Sentrale funksjoner

#### dimens.xml

Filen dimens.xml inneholder alle størrelser og posisjoner for alle bilder som skrives ut på skjermen. Når en metode, f.eks setX trenger en størrelse må den vite hvor mange piksler av skjermen den størrelsen er. Siden alle skjermer har forskjellig antall piksler i forhold til både størrelse og oppløsning, kan ikke disse tallene hardkodes. Metoden setX kaller derfor FitToScreenmed hvor stor prosent av skjermen den trenger. Først blir metoden getPercent fra MainActivity.java kalt med navnet til prosent størrelsen. getPercent går inn i dimens.xml og henter ut det elementet med det rette navnet. getPercent henter deretter ut floatverdien, som er en prosentverdi, og legger den i en float som returneres. Metoden sender denne floatverdien til en av to svært like metoder. Disse metodene heter returnViewHeight og returnViewWidth, hvor returnVeiwHeight returnerer piksler i forhold til prosenten av høyden og returnViewWidth returnerer piksler i forhold til prosenten av vidden. setX får da tilbake en størrelse i piksler.

Størrelsene er skrevet i prosent slik at skjermbildet ser likt ut uansett skjermstørrelse og oppløsning. I den første koden var verdiene som ble brukt for størrelser og plasseringer på skjermen lagt inn i piksler. Når appen ble kjørt på en telefon med større skjerm og høyere oppløsning ble alle notene veldig små og plassert i hjørnet av skjermen.

Densisty Indepentent Pixels, eller dp, er en annen enhet hvor størrelsene blir regnet om til piksler ut i fra hvor mange piksler skjermen består av. Dette kunne fungert, hvis plasseringen av notene ikke måtte være helt nøyaktig i forhold til hjelpelinjene.

I den ferdige koden blir alle verdiene nå regnet ut som prosent av skjermen, og notene blir plassert riktig på alle skjermstørrelser og oppløsninger.

#### onFocusChanged

Denne metoden ser om vinduet har fokus. Hvis vinduet mister fokus blir metoden som stopper opptaket kalt, slik at selv om man åpner menyen uten å avslutte innspillingen blir den likevel stoppet. Når vinduet får fokus igjen blir størrelsen satt på linLayout, lowestLayer og bakgrunnsbildet. Dette ligger her motsetning til onCreate pga at når onCreate blir kalt finnes ikke størrelsene enda.

#### Config.java

Filen Config.java består av en klasse som gir tilgang til MainActivity sin context utenfor MainActivity. Dette blir gjort ved at når appen startes og onCreate blir kalt, blir context sendt til Config.java. Context er en abstrakt klasse i Android som gir tilgang til ressurser som for eksempel notebilder, størrelsesforhold og klassene i MainActivity. Dette er derfor en elegant løsning, slik at disse kan brukes av alle klasser og ikke bare i MainActivity.

### XML

XML, eller Extensible Markup Language, er markeringsspråket som brukes for å sette opp brukergrensesnittet i Android apper. XML er en kombinasjon av tekst og markeringer med informasjon om teksten, som for eksempel størrelse og plassering på skjermen. Det fungerer på mange måter som HTML, HyperText Markup Language, hvor flere elementer kan nestes inn i hverandre. Store deler av brukergrensesnittet i appen er forhåndsdefinert i ulike XML-filer, mens noen deler som for eksempel knappene for å redigere noter, blir opprettet direkte i koden.

#### Layouts

Det finnes tre ulike layout-oppsett man kan bruke for å organisere et skjermbilde, de kan også nestes for å oppnå det ønskede resultatet.

Brukergrensesnittet i VeryNote er i hovedsak konstruert ved hjelp av Relative layouts. Innholdet i layouten plasseres relativt i forhold til hverandre, hvert element har sin plassering definert i forhold til andre elementer i layouten. Utenom Relative Layout blir det også brukt Linear Layout, hvor elementene blir plassert etter hverandre i en spesifisert retning, og Frame Layout som gjør det mulig å vise flere RelativeLayouts over hverandre.

#### Scrollview

Scrollview et et felt som kan rulles fram og tilbake over skjermen. Feltet midt på skjermen som viser notene er i starten et statisk felt, men etterhvert som notene blir skrevet ut vil feltet bli rullbart.

#### ImgViews

Brukes for å vise bilder til brukeren. Notene som animeres på skjermen etterhvert som brukeren synger er bilder plassert i ulike ImgView.

#### TextView

Brukes for å vise tekst til brukeren. En variasjon av dette som kalles EditText lar brukeren endre teksten, som for eksempel i feltet hvor brukeren kan lagre et egendefinert navn for filen som lagres.

#### Buttons

XML element for å opprette knapper i brukergrensesnittet. Det finnes flere ulike knapper, i VeryNote finner vi standard knapper som brukes for de ulike redigeringsknappene. Knappen som styrer metronomen er en «Switch», og en «SeekBar» brukes for å la brukeren velge en hastighet for metronomen.

### UI

#### Layers

I activity \_main.xml ligger hovedoppsettet til brukergrensesnittet. Det består av et FrameLayout som holder de tre RelativeLayout som blir referert til i koden. Når RelativeLayout blir lagt til et Frame Layout på denne måten blir de stablet oppå hverandre, dette gjør at noen elementer kan ligge oppå andre elementer. Det Relative Layout som ligger nederst i activity\_main.xml filen er det Relative Layout som ligger øverst i brukergrensesnittet.

#### Scrollview animating

Funksjonen moveLinLay animerer det RelativeLayout linLayout som ligger i scrollView, ved å flytte det og å øke størrelsen på det hvert 70ms. moveLinLay er en «Runnable» som blir kalt med den første noten etter at brukeren har trykket på record-knappen. Det vil si at linLayout ikke beveger seg før den har fått inn en note. Når brukeren trykker på record-knappen neste gang vil linLayout bli animert til slutten selv om brukeren så på begynnelsen av innspillingen.

#### Note plassering

Etter at noten har gått gjennom PitchDec.java blir den sendt til showPitchDetectionResultsom ligger i MainActivity.java. findNearestNotesøker gjennom HashMap-et notes i NoteSearch.java og returnerer det noteobjektet som ligger nærmest den noten som ble returnert av PitchDec.java.

Dette noteobjektet blir deretter sendt til NotesOnScreen. Her blir ImageView-et currentNote og det Relative Layout-et imgLayout lagt klar til den eventuelle noten. Noten blir deretter sendt til noteLength hvor det finnes ut om noten er lang nok til å bli vist på skjermen. Her sjekkes det også hvilken vei noten skal tegnes, om noten skal markeres og om noten trenger ekstra linjer. Hvis noten er lengre en 1/16-delsnote blir noten, fortegnet og de ekstra linjene skrevet til hvert sine ImageView som alle legges til imgLayout.

Videre i noteOnScreen blir currentNote tildelt en id slik at den skal kunne kjennes igjen ved redigering. Noten får også tildelt hvor på skjermen den skal skrives ved å søke gjennom dimens.xml filen etter navnet på noten. En lytter blir lagt til noten for redigeringen senere og imgLayout blir lagt til linLayout.

Hvis findNearestNote returnerer den samme noten flere ganger blir noteLength kalt direkte.

#### Redigering av noter

På hvert bilde som viser en note eller pause ligger en lytter av MyTouchListener. Når brukeren trykker på en note, etter at innspillingen er ferdig, blir noten og alt tilhørende farget blått. Dette skjer ved hjelp av et fargefilter som legges til alle «barnene» til forelderen til noten, dvs alle notens «søsken» og den selv. Når en note blir valgt, blir også en del knapper laget og lagt ut på skjermen. Disse knappene går inn under tre kategorier.

##### Delete Note

Etter at brukeren har trykket på sletteknappen, begynner den med å identifisere noten og dens plass i allNotesForXML, under navnet allNotes i denne filen, og endrer lengden til noten til en tom streng i stedet for den lengden den har. Grunnen for dette er at når arraylisten blir eksportert til MusicXML blir bare de objektene som har en lengde inkludert, slik at når en note ikke har lengde blir den ikke lagt til i XML-filen. Etter det blir alle bildene som var markert med blått slettet fra brukergrensesnittet og knappene blir også gjemt igjen.

##### ♭ /#/n

Når brukeren trykker på en av disse tre knappene, sjekkes det først om noten allerede har dette fortegn og det skal fjernes eller om dette fortegnet skal legges til. Hvis noten allerede har dette fortegnet blir bildet som representerer det fjernet fra brukergrensesnittet. Noten blir funnet i arrayen og noten blir satt til å ikke ha fortegn. Hvis derimot noten ikke har det valgte fortegnet, må dette legges til. Først blir de andre mulige fortegnene sine bilder funnet og fjernet. Noten blir også funnet her og blir satt til å ha det korrekte fortegnet. Etter dette blir et blide med det nye fortegnet laget og lagt til brukergrensesnittet.

##### Down/Up

Når brukeren flytter noten en stamnote opp eller ned, må navnet på noten oppdateres i arraylisten allNotesForXML. På samme måte som de andre knappene blir noten funnet frem til ved hjelp av metoden setNoteName. Denne metoden setter det nye navnet til noten likt den nye id-en til bildet av noten. Hvis noten er markert med # eller ♭ fjernes disse, siden det ikke finnes noen garanti for at den noten eksisterer i denne dur/moll. Knappene for disse fortegnene blir også oppdatert, slik at brukeren kun kan velge alternativer som finnes. Når noten flyttes kalles også metoden fixLines, som sletter alle hjelpelinjene og tegner de inn på nytt, med riktig antall linjer. Den nye plasseringen til noten på skjermen blir funnet ved å søke etter den nåværende plasseringen i yValueSearch.java, og så hente ut enten den foregående (opp) eller den neste (ned) verdien i arrayet. Hvis noten er oppned blir den forskjøvet tilsvarende.

I redigeringen av noten var ideen å bruke setName og setSharp, men pga at alle like noter har samme id, måtte det løses på en annen måte.

#### Meny

Menyen består av seks knapper, hvor Record og A4 alltid er synlige på actionbar såfremt det finnes plass på skjermen, mens de fire gjenståene knappene alltid vises som en pop-up meny. Dette styres av tag’en ifRoom i XML-filen menu\_main.xml. Funksjonaliteten til menyen styres av en switch/case som sjekker hvilken knapp brukeren trykker på, og kaller videre den tilhørende metoden(e) for valgt knapp.

## Appens virkemåte

### Hovedskjerm

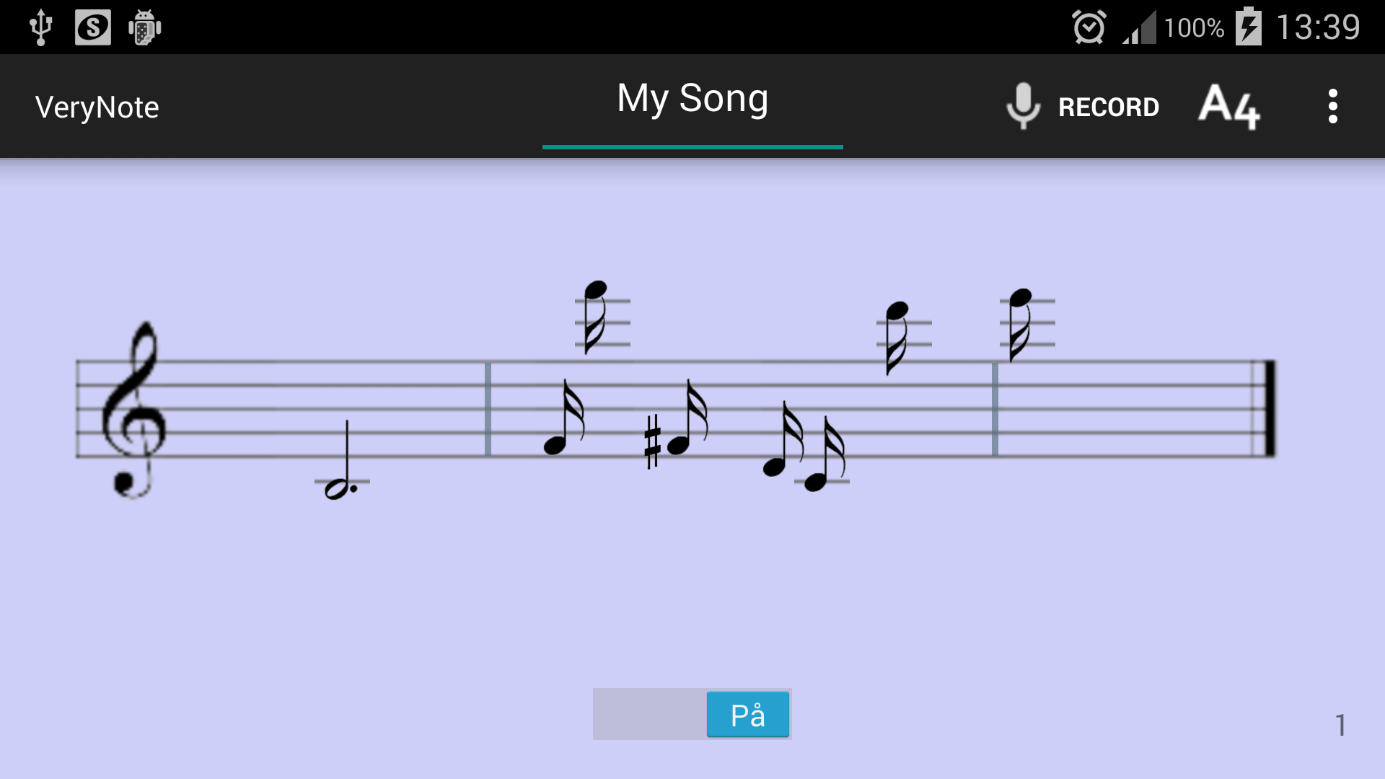
Hovedskjermen består av en «Action-Bar», navnet på appen, filnavnet og flere knapper og funksjoner. Nederst er en knapp for å slå av og på metronomen, og midt på skjermen er et notesystem med G-nøkkel.



Figur 6.1 – Skjermbilde av appens hovedskjerm ved oppstart.

#### Record

Når brukeren trykker på record-knappen, vil appen begynne å analysere lyden den får inn. Mikrofonsymbolet vil endre seg til ett stopp symbol, og teksten record vil endre seg til stop. Appen vil vise hvilke noter brukeren synger ved å skrive noten ut i notesystemet på skjermen. Hvis brukeren ikke holder noten lenge nok til å tilsvare minst en 1/16-dels note vil det vises en liten markør i notesystemet, men ingen noter vil bli skrevet ut. Når brukeren trykker på stop-knappen igjen, vil appen slutte å analysere lyden som kommer inn og symbolet vil endre seg tilbake til record-teksten og mikrofonsymbolet. Brukeren kan scrolle gjennom notene og hvis brukeren trykker på record-knappen igjen kan han fortsette på den samme inspillingen. De nye notene vil bli lagt til å slutten av den forrige.



Figur 6.2 – Skjermbilde av appens hovedskjerm etter innspilling, med metronomen på.

#### A4

Til høyre for record-knappen er det en knapp med et A4-symbol. Denne knappen spiller av en frekvens på 440hz som tilsvarer en A. Denne knappen skal hjelpe brukeren å «stemme seg selv» før de begynner å spille inn en sang. Firetallet representerer oktaven.

#### Tittelfelt

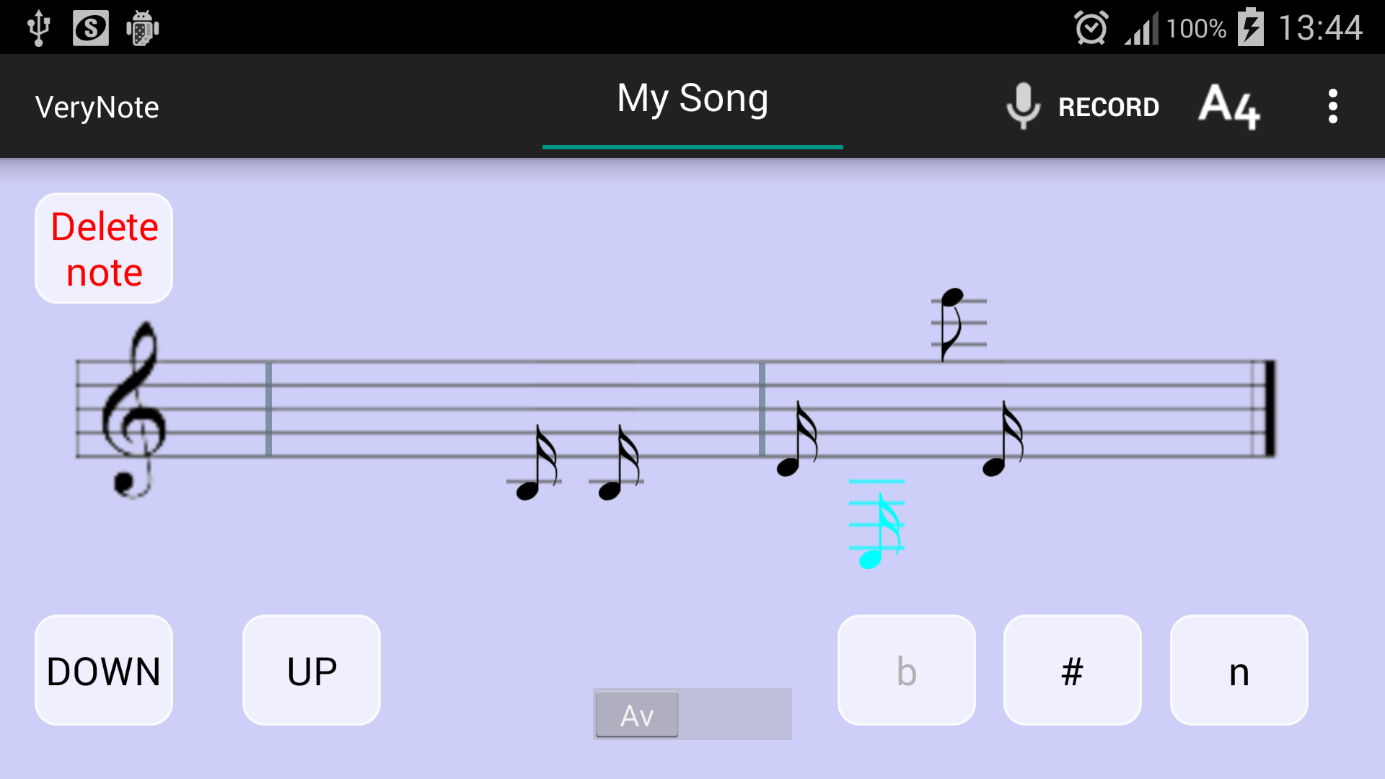
Når appen startes står det «Untitled» øverst på skjermen. Når brukeren trykker på ordet, kan de selv legge til et egendefinert navn på filen de jobber med. Dette navnet blir tittelen på filen om brukeren velger å lagre eller dele filen.

#### Metronom

Nederst på skjermen står en av/på knapp. Denne knappen styrer metronomfunksjonen. Hastigheten er som standard 80BPM, men brukeren kan selv endre dette i settings-menyen. Når brukeren slår på knappen begynner telefonen å vibrere i gitt intervall, og det vises en liten teller nederst til høyre på skjermen.

### Redigering

Etter at brukeren har sunget inn notene sine, kan det være at de vil endre noe. Det er enkelt å legge til/ta bort fortegnene ♭/#/n, eller flytte notene opp og ned. Alle noter og pauser kan også fjernes.



Figur 6.3 – Skjermbilde etter at brukeren har valgt en note for redigering.

#### Valg av noter

Brukeren trykker på den noten de vil endre, noten vil endre farge og telefonen vil vibrere kort for å indikere at en note har blitt valgt. Når brukeren er ferdig å redigere kan de trykke på den samme noten igjen for å velge den bort, eller trykke på en annen note for å velge den istedet.

#### Delete Note

Hvis brukeren har sunget helt feil, eller ombestemt seg kan de trykke på «delete note»-knappen. Denne knappen sletter noten

#### Down/Up

Disse knappene flytter den valgte noten opp eller ned en stamtone.

#### ♭ /#/n

Brukeren kan endre eller fjerne fortegn på den valgte noten. Hvis noten allerede har et fortegn kan dettee fjernes ved å trykke på knappen som tilsier dette fortegnet. F.eks hvis noten er markert med en «♭ » kan brukeren fjerne den ved å trykke på «♭ »-knappen. Den kan selvsagt legges til igjen ved å trykke på knappen en gang til. Hvis brukeren heller vil endre fortegnet for noten fra «♭» til «#» kan brukeren trykke på «#»-knappen. Hvis noten ikke finnes i C-skalaen vil knappen være deaktivert, slik at man ikke kan markere en A med verken «#» eller «♭»

### Meny

Metoden for å åpne menyen varierer etter hvor ny androidtelefonen som appen kjører på er. Eldre telefoner har en «meny» knapp nederst til venstre, nyere nettbrett og telefoner har gjerne ikke denne knappen, da åpnes menyen ved å trykke på de tre prikkene ved siden av A4 knappen.



Figur 6.4 – Skjermbilde av appen med menyen åpen.

#### New Masterpiece

Når brukeren vil starte på nytt, kan de trykke på «new masterpiece»-knappen i menyen. Appen vil bli resatt, alle notene og tidligere innstillinger vil bli slettet. Tittelen vil også bli satt tilbake til «Untitled»

#### Save/Share MusicXML

MusicXML [7] har blitt et populært filformat for å dele musikk. Det er en standisert måte å skrive musikk i digitalt format slik at det enkelt kan deles mellom brukere av forskjellige applikasjoner.

Når brukeren trykker på «Save MusicXML»-knappen vil en xml-fil med alle notene fra appen bli lagret under den tittelen brukeren selv har skrevet inn. Mellomrom vil bli fjernet fra tittelen hvis de finnes før filen lagres. Hvis brukeren ikke har valgt en tittel vil filen bli lagret som «Untitled». Hvis det allerede finnes en fil under samme navn, vil filen ikke bli lagret og bruker en få en melding om at en fil ved dette navnet allerede eksisterer. Det blir da opp til brukeren om de vil gi denne filen ett nytt navn, eller finne den gamle filen og slette den.

«Share MusicXML» knappen åpner en ny meny som lar brukeren velge hvordan de vil dele xml-filen. Den kan for eksempel sendes til email, eller over bluetooth/wi-fi.



Figur 6.5 – Skjermbilde avr delingsmenyen.

#### Settings

Når brukeren trykker på «Settings»-knappen åpnes et pop-up vindu med noen valg. Her kan brukeren velge om de vil synge inn i F eller G- nøkkel og hvilken hastighet de vil ha på meteronomet. Hastigheten på meteronomet påvirker også hvor ofte man ser taktstreker når man spiller inn noter. Når brukeren velger å endre hvilken nøkkel de synger i, vil alle notene bli slettet.



Figur 6.6 – Skjermbilde av Settings-menyen.

## Konklusjon og videre arbeid

### Konklusjon

Appen VeryNote viser at det absolutt er mulig å notere musikk ved bruk av smarttelefon. Brukere av VeryNote blir presentert med et enkelt UI, hvor han/hun starter opptaket, sangen blir analysert og noter/pauser skrevet ut på skjermen i sanntid. Det finnes en meny med innstillinger for metronomen, og brukeren kan velge mellom g-nøkkel og f-nøkkel. Etter opptaket kan notene redigeres som ønsket, tilslutt kan sangen lagres på enheten som en MusicXML fil og/eller deles for eksempel over email eller bluetooth.

Likevel finnes det begrensinger, f.eks mikrofonen. Mikrofonene i dag er laget for å stenge ute støy, og som en konsekvens av dette, lave frekvenser. Dette betyr at selv om koden var tilrettelagt for å ta opp de laveste notene i kjent register, ville ikke appen fått mulighet til å behandle dem.

Dynamisk generering av en MIDI fil slik at brukere kan spille av notene de har lagret var en funksjon vi gjerne ville integrert, men per idag finnes det ikke støtte for dette i Android. Det er ikke umulig, men å implementere dette i appen var ikke et alternativ innenfor tidsrammen.

Til tross for begrensningene oppfyller appen alle funksjonskravene i problemstillingen, i tillegg er det implementert flere funksjoner som for eksempel redigering av notene.

### Videre arbeid

Dette har vært et svært interessant og lærerikt prosjekt. Hadde vi hatt mer tid ville vi selvsagt ha lagt til mer funkjonalitet i appen. Et godt eksempel på noe vi dessverre ikke hadde nok tid til å implementere er midi playback. Siden android ikke støtter javax.sound fant vi ingen løsning på dette som kunne blitt implementert innen tidsrammene.

Hvis tiden hadde strukket ville appen hatt en hjelpeskjerm som kun ble kjørt den første gangen appen ble åpnet. Det ville selvsagt vært mulig å åpne denne hjelpeskjermen opp igjen fra menyen hvis brukeren ville se den igjen. Denne hjelpeskjermen ville bestått av flere gjennomsiktige bilder brukeren kunne ha bladd mellom, med forklaringer av knapper og funksjoner.

Appen ville gitt mulighet til å velge og å vise taktarten. Det ville være mulig å synge inn i forskjellige dur-er og mol-er, og markert hvilken dur/mol som var valgt i margen sammen med nøkkelen og taktarten. Appen ville også hatt større utvalg av noter, både kortere og mer spesielle tilfeller. Den korteste noten tilgjengelig i appen nå er 16-dels note, men det finnes noter så korte som en 256-delsnote. Appen ville hatt bindebuer for å symbolisere at noter skal holdes over taktstreken, og legatobuer som viser at notene skal utføres som én bevegelse (ett pust). Appen ville hatt muligheten til å legge inn doble taktstreker for å sybolisere en ny frase. Den ville også bundet sammen notene hvis flere 16-dels eller 8-dels noter ble registrert etter hverandre.

Det ville også vært naturlig å utvide redigeringsmulighetene. Det ville vært knapper for å endre varigheten til notene og pausene, og det ville vært mulig å flytte notenes og pausenes posisjon. Det ville også vært mulig å legge til noter, ikke bare endre på de man allerede har på skjermen. Hvis brukeren hadde trykket og holdet på skjermen, ville det dukket opp en ny note som man kunne redigert videre med de vanlige redigerings funksjonene.

Vi vurderte også å utvikle et dataprogram som fungerte sammen med appen, slik at man lett kunne få filene sine til dataen og redigere dem videre der. Dette programmet kunne også hatt støtte for å spille tilbake en MIDI-versjon av det brukeren synger siden da kunne man utnyttet javax.sound biblioteket.

Mot slutten av utviklingen av appen ble vi oppmerksom på at om den tok opp lyd over en lengre periode begynte animasjonen og stotre litt. Dette var mer merkbart på den eldste test-telefonen enn på de nyere telefonene. Ved hjelp av MemoryMontior i Android Studio så vi at den eldste telefonen slapp opp for minne, mens de nyere kunne fortsette å allokere nok minne etterhvert som det var nødvendig. Hadde vi hatt mer tid hadde vi vært nødt til å ta en nærmere kikk på dette. Det er ikke et stort problem slik den står nå, selv på den gamle telefonen kan man synge i over et minutt før man merker at animasjonen ikke helt holder følge. (Notene blir fremdeles lagret som de skal, det er bare animasjonen som stopper opp.) På de nyere telefonene kunne man synge i rundt 3 minutt før man merket noe feil med animasjonen.

## Vedlegg

Vedlegg 1: Kildekode skrevet i Android Studio

Vedlegg 2: Matlab kode brukt til å tegne figurer.

Vedlegg 3: Kompilert app.

## Referanser

[1] Arnt Inge Vistnes, *Lærebok i fysikk : Svingninger og bølger*, 2013. Kapittel 4. Tilgjengelig fra: <http://folk.uio.no/arntvi/SvingBolg.html>

[2] THE BOOK

[3] John R. Pierce, *The Science of Musical Sound*, Scientific American Books - W. H. Freeman & Co, 1983. Figur side 21.

[4] Øyvind Risa, *Musikkteori og arrangering*. Universitetsforlaget, 2001.

[5] Phil Burk, Larry Polansky, Douglas Repetto, Mary Roberts, Dan Rockmore, *Music and Computers,* 2011. Tilgjengelig fra: <http://music.columbia.edu/cmc/musicandcomputers/>

[6] <http://www.bidouille.org/misc/androidcharts>

[7] <http://www.musicxml.com/>

[8] <https://github.com/wendykierp/JTransforms>

[9] <https://code.google.com/p/androidtuner/source/browse/src/com/example/AndroidTuner/PitchDetector.java?r=134743f47bfb0a0842f3dc73a75fed52923c8917>

Diverse oppslag:

<http://www.stackoverflow.com>

[http://developer.android.com](http://developer.android.com/guide/topics/ui/declaring-layout.html)

Font brukt for å lage notefigurer: <http://www.fontspace.com/robert-allgeyer/musisync>

Appikon: <http://www.clipartlord.com/free-giant-panda-clip-art/>