|  |  |
| --- | --- |
| UiS%5Fnor%5Fcolor  **DET TEKNISK-NATURVITENSKAPELIGE FAKULTET**  **BACHELOROPPGAVE** | |
| Studieprogram/spesialisering:  Data | 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:    Android  Fast Fourier Transform  MusicXML | Sidetall: …………………    + vedlegg/annet: …………  Stavanger, ………………..  dato/år |

## Sammendrag

I denne oppgaven har vi laget en app for Android OS som utfører frekvensanalyse på sang tatt opp fra mobilens mikrofon, finner hvilken note frekvensen tilsvarer for deretter å tegne notene i et notesystem på skjermen. Appen heter VeryNote, er utviklet for Android OS og er laget ved bruk av Java og XML. VeryNote er skrevet i Android Studio for Android OS versjon 4.0 (Ice Cream Sandwhich) og kan dermed brukes på alle enheter med denne og nyere OS.

Brukeren blir presentert med et enkelt UI, hvor 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. Til slutt kan sangen lagres på enheten som en MusicXML-fil og/eller deles for eksempel over e-post eller Blåtann.

Metoden som er benyttet for frekvensanalysen er Fast Fourier Transform (FFT). Dataene fra mikrofonen blir gjort om fra tidsdomenet til frekvensdomenet, slik at det er mulig å finne den fundamentale frekvensen innen det gitte tidsrom.

Det er mulig å rulle fram og tilbake over notesystemet etter man er ferdig å synge. De enkelte notene og eventuelle pauser kan redigeres ved å trykke på den, den markeres da med en annen farge og redigeringsknappene blir synlige. Noter og pauser kan slettes, notene kan også flyttes opp og ned i notesystemet og fortegnet kan endres eller slettes.

Etter utvikling og testing fungerer appen i henhold til oppgaven og våre egne ambisjoner. Likevel er det noen tilleggsfunksjoner vi ville lagt til hvis den skulle blitt distribuert åpent på Google Play. Mulighet for å spille av sangen etter innspilling og at notene blir plassert riktig i forhold til taktstrekene hadde gitt en bedre brukeropplevelse.

## Innholdsfortegnelse

[Sammendrag 2](#_Toc419195589)

[Innholdsfortegnelse 3](#_Toc419195590)

[Forord 5](#_Toc419195591)

[1. Innledning 6](#_Toc419195592)

[1.1. Bakgrunn 6](#_Toc419195593)

[1.2. Problemstilling 6](#_Toc419195594)

[2. Digital signalbehandling og musikkteori 7](#_Toc419195595)

[2.1. DFT - Diskret Fourier Transform og FFT – Fast Fourier Transform 7](#_Toc419195596)

[2.2. Frekvenser i Musikk 10](#_Toc419195597)

[2.3. Musikkteori 12](#_Toc419195598)

[3. Programmeringsverktøy 16](#_Toc419195599)

[3.1. Android 16](#_Toc419195600)

[3.2. Android Studio 17](#_Toc419195601)

[3.3. Mobiltelefoner for testkjøring 17](#_Toc419195602)

[4. Implementasjon av teori 18](#_Toc419195603)

[4.1. JTransform/PitchDec.java 18](#_Toc419195604)

[4.2. NoteSearch 20](#_Toc419195605)

[4.3. Dynamisk endring av notelengde 20](#_Toc419195606)

[5. Appens oppbygging 21](#_Toc419195607)

[5.1. Tråder og handlers 21](#_Toc419195608)

[5.2. Sentrale funksjoner 21](#_Toc419195609)

[5.3. XML 23](#_Toc419195610)

[5.4. UI 25](#_Toc419195611)

[6. Appens virkemåte 28](#_Toc419195612)

[6.1. Hovedskjerm 28](#_Toc419195613)

[6.2. Redigering 30](#_Toc419195614)

[6.3. Meny 33](#_Toc419195615)

[7. Testing av appen 35](#_Toc419195616)

[8. Konklusjon og videre arbeid 35](#_Toc419195617)

[8.1. Konklusjon 35](#_Toc419195618)

[8.2. Videre arbeid 36](#_Toc419195619)

[Vedlegg 38](#_Toc419195620)

[Referanser 39](#_Toc419195621)

## Forord

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.

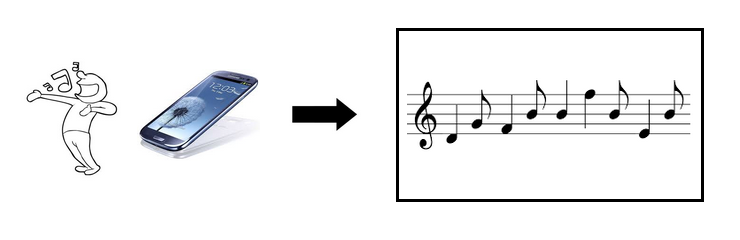
Vi vil gjerne takke Tom Ryen for god veiledning i løpet av arbeidet med denne oppgaven. Vi vil også takke for MATLAB kode brukt til å lage figurer for å illustrere frekvensanalysen i denne rapporten.

## Innledning

### Bakgrunn

### Problemstilling

Problemstillingen og tittelen for oppgaven er «Musikknotasjon ved bruk av smarttelefon». Målet med oppgaven er å lage en app for Android OS som kan ta opp sang/plystring fra mikrofonen, finne fundamental frekvens i en gitt tidsperiode og omsette denne frekvensen til en note. Notene skal så skrives ut på mobilskjermen i et notesystem. Det skal også være mulig å eksportere notene som en MusicXML-fil.



Figur 1,1 – Figur hentet fra oppgavebeskrivelsen, med tilltatelse fra veileder.

## Digital signalbehandling og musikkteori

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

Frekvensanalyse av lyd kan gjøres ved hjelp av fouriertransformasjon [1,2,5] . Fouriertransformasjon endrer signaler fra tidsdomenet om til frekvensdomenet, og er definert for både kontinuerlige og diskrete signaler. Lyd er et kontinuerlig signal, men for å lagre og bearbeide lyd digitalt må det plukkes ut en endelig mengde verdier til analyse. Dette kalles sampling. Diskret fouriertransformasjon (DFT), er metoden som benyttes til frekvensanalyse når det finnes en endelig mengde datapunkt.

DFT er definert ved

Fast Fourier Transform (FFT), er en algoritme som utfører diskret fouriertransformasjon mer effektivt. Det er verdt å nevne at en slik algoritme ble oppdaget allerede i 1805 av Carl Friedrich Gauss, men siden det ikke fantes datamaskiner, så var den ikke interessant, og den ble glemt. I 1965 ble den plukket opp igjen av J.W. Cooley og J. Turkey og lansert i deres artikkel «An algorithm for the machine calculation of complex fourier series» [11].

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.

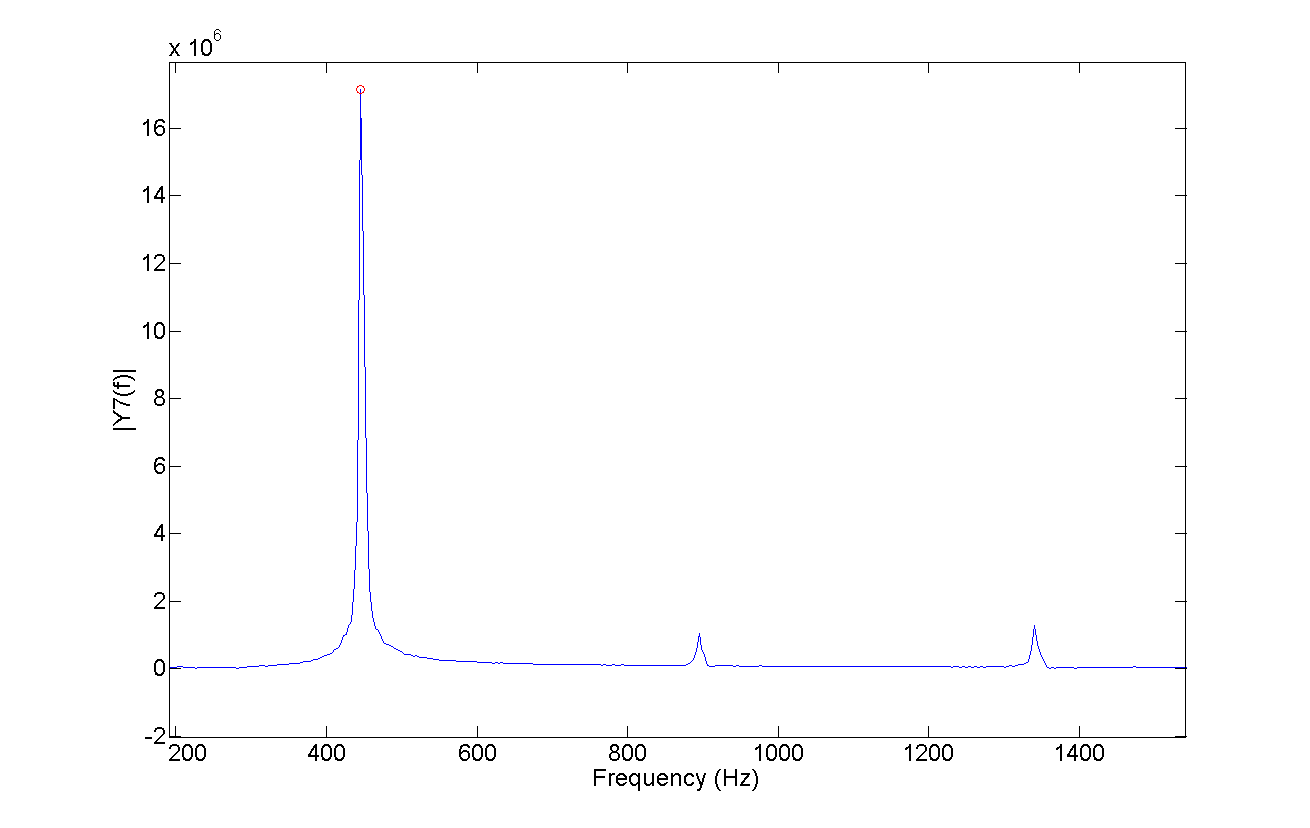
For å fullt utnytte denne algoritmen må antall punkter N være et tall 2n, hvor n er et heltall. I frekvensanalysen VeryNote gjør, jobber den med «chunks» på 4096 (212) datapunkter om gangen og den utnytter derfor FFT fullt ut.



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

Verdier fra et opptak med mobilens mikrofon ble hentet ut og lagret, både før og etter fourieranalysen. Disse ble så plottet i grafer ved hjelp av MATLAB (se vedlegg 2). Resultatet av dette er grafene i figur 2.1, 2.2, 4.1 og 4.2.

I figur 2.1 ser vi en representasjon av dataene fra et lydopptak av sang. Det er 4096 sampler over 256 ms i tidsdomenet. Lyden er tatt opp i 16 bits PCM format. PCM, eller Pulse-code modulation, er en digital representasjon av et analogt signal [12]. Verdiene langs y-aksen kan være mellom minimum 215 og maksimum -215  ved bruk av 16 bits PCM.

Med bruk av FFT får vi en representasjon av de samme dataene i frekvensdomenet. Figur 2.2 viser en representasjon av amplituden til de ulike frekvensene fra det originale signalet i figur 2.1.

Figur 2.2 – Amplitudespekteret til dataene i figur 2.1 etter transformasjon til frekvensdomenet.

På denne figuren ser vi at det finnes flere markerte frekvenser. Dette er et resultat av at menneskestemmer i tillegg til en fundamental frekvens, har flere overharmoniske frekvenser. Vi ser at lyden har en grunnharmonisk frekvens på 440 Hz, og overharmoniske frekvenser på 880 Hz og 1320 Hz. Dette blir forklart nærmere i kap. 2.2.

Et problem med frekvensanalyse av digitale signaler er oppløsningen ved lavere frekvenser. Frekvensene som oppfattes som de ulike tonene i skalaen er ikke lineære, men logaritmiske. En forskjell på 100 Hz ved lavere frekvenser tilsvarer en hel oktav, men ved høyere frekvenser er det over 100 Hz mellom hver enkelt note. (Se tabell 2.1 for en fullstendig oversikt). Med bruk av FFT vil vi ha like mange sampler i frekvensdomenet som i tidsdomenet, for eksempel 4096. Dersom samplingsraten er relativt høy, vil vi få en stor avstand mellom hver frekvens.

Nøyaktigheten er derfor 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 frekvensdomenet, og hvor korte intervaller man skal analysere. 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 (1000 ms\*4096)/16000 = 256 ms. Frekvensoppløsningen er da 16000/4096 ≈ 4 Hz. De laveste frekvensene det letes etter i denne analysen er 131 Hz og 139 Hz, derfor er denne oppløsningen god nok i dette tilfelle.

For et mer nøyaktig resultat i frekvensdomenet, må flere datapunkter analyseres og antall sampler må økes. Hvis antall sampler dobles til 8192 gir dette, med en samplingsrate på 16000, en frekvensoppløsning på 16000/8192 ≈ 2 Hz. Kompromisset en da må gjøre er at hver FFT blir utført over et intervall på (1000 ms\*8192)/16000 = 512 ms. FFT av 8192 sampler vil også ta lengre tid enn FFT av 4096 sampler, fordi N i O(N*log*N) blir dobbelt så stor.

Det finnes altså ikke et entydig svar på hvilket antall sampler som gir best resultat i frekvensanalyse, det må gjøres et valg alt etter hvilke frekvenser man analyserer. I dette tilfellet, hvor fundamental frekvens for sang blir analysert, gir 8192 sampler en tilfredsstillende frekvensoppløsning på relativt korte tidsintervaller. Alternativene for samplingsrate er mer begrenset. Det avhenger av enheten som gjør lydopptaket. Valget falt på 16000 Hz, 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 lydbølger. Når disse bølgene når trommehinnen i øret, oppfattes de som lyd.

Musikalske lyder kan representeres som en sinusfunksjon. Bølgelengden er avstanden mellom to bølgetopper. Frekvensen er hvor mange ganger en hendelse gjentar seg i løpet av et bestemt tidsrom, i tilfellet med lyd blir det hvor mange bølgetopper som passerer per tidsenhet. 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, vi ser tydelig fundamental frekvens.

Som vi så på figur 2.2 består lyden av en menneskestemme av en fundamental frekvens og flere overharmoniske frekvenser. Disse kommer av at den komplekse sinuskurven som representerer lyden er bygget opp av flere enkle sinuskurver. Frekvensene på disse enkle sinuskurvene er heltallige multipler av den fundamentale frekvensen [15].

Alle noter har en frekvens, for eksempel en tone med frekvens 440 Hz blir oppfattet som en A. En frekvens på 220 Hz er 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. Her bruker vi navnene lille oktav til trestrøken oktav, en annen måte å navngi oktavene er nummerering. VeryNote dekker oktav tre til og med seks.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **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]

Disse frekvensene dekker fire oktaver. Appen runder av til nærmeste frekvens hvis brukeren synger litt falskt. Hver oktav dekker notene fra C til H, og mellom hver oktav fordobles frekvensen som tilsvarer noten. En A kan for eksempel være 220, 440, 880 eller 1760 Hz, alt etter hvilken oktav den befinner seg i. Skalaen er altså logaritmisk, ikke lineær.

Mennesker hører bedre forskjell på 100 og 101 Hz (101/100 = 1,01) enn en forskjell mellom 1000 og 1001 Hz (1001/1000=1,001). Jo høyere frekvensen er, desto større må forskjellen i Hz være for at det skal oppfattes som en annen note. Det er forholdet mellom notene som oppfattes, ikke den faktiske forskjellen i Hz. Dette vil si at vi hører den samme forskjellen mellom 440 og 220 Hz som vi hører mellom 880 og 440 Hz, selv om den faktiske forskjellen i Hz er dobbelt så stor.

### Musikkteori

Dette kapittelet beskriver kort musikkteorien [4] brukt i appen.

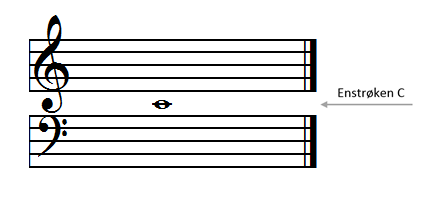
#### G-nøkkel/F-nøkkel

Notesystemet inneholder fem linjer, eventuelt med hjelpelinjer og er 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- og Giss-noter. [13]

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 hjelpelinje mellom seg, se figur 2.5.



Figur 2.5 – Notesystemer med G- og F-nøkkel. [13]

#### Takt

En takt består av de noter eller pauser som befinner seg mellom to taktstreker. Taktarten beskrives av signaturen, en brøk som skrives etter nøkkelen. Nevneren representerer hvor langt ett slag er, mens telleren representerer hvor mange slag det er i hver takt. Dette vil si at med signaturen 3/4 består alle takter av tre fjerdedelsnoter eller noter/pauser som tilsvarer denne lengden. De mest vanlige taktartene er 2/4, 3/4 og 4/4. VeryNote bruker taktarten 4/4.

#### Ulik lengde på noter

Hvilke noter som skal spilles blir markert med plasseringen av noten, mens varigheten blir bestemt av utseende. Som det kommer av tabell 2.2 er varigheten til noten relativ til lengden på en hel takt.

|  |  |  |
| --- | --- | --- |
| 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 med taktart 4/4. [13]

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. [13]

#### Pauser

Stillhet i musikk har selvsagt sine egne markeringer. På samme måte som notene, har de forskjellige tegn for forskjellig 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, se tabell 2.2. 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 markeres med flere fortegn, de tre som brukes i appen er ♭, ♯ og ♮ Hvis en note er markert med en ♭ skal den notens tonehøyde senkes med en halv tone. Fortegnet ♯ øker notens tonehøyde med en halv tone. Disse to fortegnene varer ut den takten de er skrevet i. Figur 2.5 viser et kort stykke med 8 noter. De første to notene er begge G. Den neste noten er markert med en ♯ så den er 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 den andre siden av taktstreken påvirker ikke fortegnet notene lengre 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, ♮. Dette fortegnet setter bort alle tidligere fortegn. Hvis den siste noten i den første takten hadde vært markert med et ♮tegn, ville den også vært en G.

#### Metronom

En metronom er et hjelpemiddel musikere kan bruke for å holde rytmen. Den produserer en lyd, for eksempel et klikk eller pip, et gitt antall ganger i minuttet. I appen er metronomen implementert som en vibrasjon. Hastigheten er satt til 80 slag i minuttet, men kan endres av brukeren.

## 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 ha versjon 4.0 IceCreamSandwich (SDK 14) eller nyere for å bruke VeryNote.

Hvis vi hadde valgt å kun utvikle for den siste versjonen hadde vi risikert at veldig få vil kunne bruke appen. Hvis vi hadde valgt en gammel versjon, mister man muligheten til å bruke metoder og funksjoner som finnes i nyere API. Rundt 90 % [6] av alle enheter som kjører Android har i dag versjon 4.0 eller nyere. 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 [12] er en IDE som kan brukes for å skrive applikasjoner for Android ved bruk av programmeringsspråket Java. Det fungerer svært likt 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 brukergrensesnittet kan man plassere knapper og tekstbokser rett på forhåndsvisningen, og se med en gang hvordan det vil se ut.

Det finnes også et verktøy i Android Studio som heter Memory Monitor, hvor man kan se i sanntid hvor mye minne appen bruker. For testing av apper er det mulig å kjøre dem på en emulator, denne har imidlertid begrensede funksjoner. Det finnes for eksempel ingen mikrofon, som i dette tilfellet gjorde det nødvendig å bruke fysiske enheter i testingen.

### Mobiltelefoner for testkjøring

For å teste appen underveis ble det brukt fire forskjellige mobiltelefoner med ulike Android operativsystemer. Mobilene hadde også ulike skjermstørrelser og oppløsninger, fordi det var viktig for oss 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 musikknotasjonsappen

### 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.

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 16 bits PCM-format. PCM, eller Pulse-code modulation, er en digital representasjon av et analogt signal [12]. For å unngå at bakgrunnsstøy blir analysert må amplituden på lyden være over 4400. Når lyden blir gjort om til 16 bits PCM, representerer tallene amplituden på lyden, er et tall mellom -215 og 215. Vi fant ved hjelp av testing at 4400 var et tall som eliminerte bakgrunnsstøy uten at brukeren må rope unødvendig høyt inn i mikrofonen for at resultatene skal bli analysert og vises på skjermen.

Opptaket fra mikrofonen blir lagret i et tabell, som det så blir utført en FFT på. Det søkes etter den fundamentale frekvensen mellom 131 Hz og 1976 Hz. Dette dekker 4 oktaver fra lille C til trestrøken H. VeryNote er i hovedsak laget for å analysere opptak av sang, derfor er frekvenser utenfor dette spekteret mindre interessante og ikke inkludert.

Deretter brukes resultatene fra FFT for å finne frekvensen med høyest amplitude. Dette skjer i filen PitchDec.java. Opptaket blir gjort med en «chunk»-størrelse på 4096, og samplingsraten er 16000. Det betyr at hver «chunk» går over (1000\*4096)/16000 = 256 ms. Hver «chunk» med sampler blir så analysert ved hjelp av FFT. Se figur 2.2. Hvis vi zoomer inn ser man tydelig at den fundamentale frekvensen i dette tilfellet er mellom 440 og 450 Hz. Se figur 4.1.



Figur 4.1 – Fundamental frekvens er her mellom 440 og 450 Hz.

Metoden AnalyzeFrequencies finner frekvensen med høyest amplitude fra hver «chunk». Denne frekvensen blir så lagt til en klynge. Disse klyngene benyttes for å få bedre resultater fra FFT-analysen. Istedenfor at alle frekvenser funnet blir vist på skjermen, blir det gjort prøver for å finne ut om hver nye frekvens i realiteten er den samme frekvensen, eller en overharmonisk frekvens. 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 frekvensverdien er mindre enn 5% forskjellig fra forrige vindu sin frekvens. Hvis den er mindre enn 5% forskjellig, blir den tolket som en fortsettelse av den samme noten og lagt til samme klynge. Hvis den er mer enn 5% forskjellig blir den tolket som en ny note og en ny klynge blir opprettet.

For å unngå at overharmoniske frekvenser blir oppfattet som egne toner, blir klyngene som inneholder overharmoniske frekvenser slått sammen ved hjelp av 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 klynge. Hvis forskjellen er større enn 5% er den ikke overharmonisk og forblir sin egen klynge.

Når frekvensen med høyest amplitude er funnet, kaller postToUI metoden showPitchDetectionResult med frekvensen som innparameter.

### NoteSearch

Notene blir lagret som objekter med informasjon om blant annet frekvens-verdi og om de har et fortegn. 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 minst 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 eng. duration). Første gang noten blir registrert blir notesOnScreen kalt, denne metoden setter blant annet dur lik null og kaller noteLength. Bildet til noten blir satt til å være et bilde med en prikk der noten blir plassert i notesystemet hvis den blir holdt lenge nok. 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. Når writePause eller notesOnScreen blir kalt igjen, kalles metoden keepLatestNote. Hvis den samme noten ikke er holdt lenge nok erstattes bildet av prikken med et gjennomsiktig bilde. 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 for at den skal vises på skjermen.

### Fortegn

Hver gang en note blir skrevet ut på skjermen, blir navnet lagret i tabellen currentMeasure. Før navnet blir lagret, kalles metoden markNote som returnerer en boolean. Hvis en lik note allerede finnes i takten får ikke denne noten fortegn. Metoden markNote vil også endre noteobjektet fra en umerket note til en ♮ note når det finnes en ♯ eller ♭ versjon av den stamnoten i takten.

## 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 slås 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, for eksempel 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 faste størrelser hardkodes inn i appen. Metoden setX kaller derfor FitToScreenmed hvor stor prosent av skjermen den trenger. Først blir metoden getPercent fra MainActivity.java kalt med navnet til prosentstørrelsen. 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

Til enhver tid kan kun et vindu være aktivt, det vinduet har fokus. Metoden onFocusChanged ser om hovedvinduet har fokus. Hvis hovedvinduet mister fokus blir metoden som stopper opptaket kalt, slik at selv om man åpner menyen uten å avslutte innspillingen blir den likevel stoppet. Når hovedvinduet får fokus igjen blir størrelsen satt på linLayout, lowestLayer og bakgrunnsbildet. Dette ligger her i motsetning til i onCreate på grunn av 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.

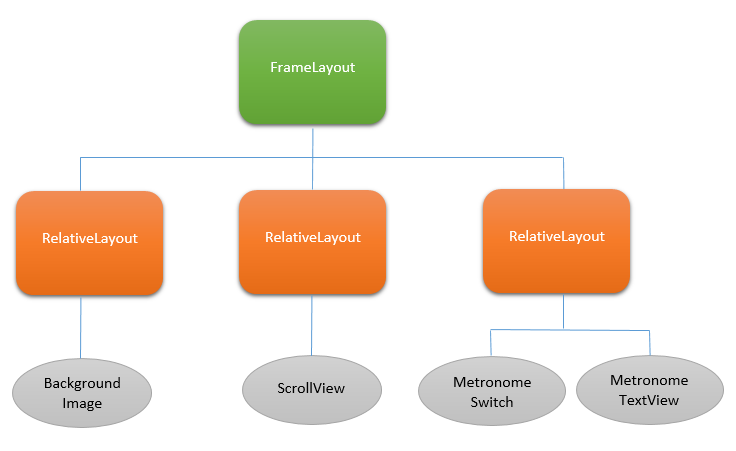
#### Layouts

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.

Alle layout-filene må ha et bunnelement som holder alle de andre elementene. Når bunnelementet er definert kan man legge til layouter eller andre elementer som barn av dette for å skape det elementhierarkiet som er ønsket.

Figur 5.1 viser hvordan hovedlayouten til VeryNote er bygget opp. FrameLayouten er bunnelementet og har tre barn, de tre relative layoutene. Disse tre layoutene er igjen foreldre til flere elementer, i dette tilfellet bakgrunnsbildet, et scroll-view og elementene som utgjør metronomen.



Figur 5.1 – Oppbygging av hovedlayouten i VeryNote

#### Scrollview

Scrollview et felt som kan rulles fram og tilbake over skjermen. Feltet er ikke rullbart før det blir større enn skjermen og er heller ikke rullbart under innspilling.

#### 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 feltet hvor brukeren kan lagre et egendefinert navn for filen som lagres.

#### Buttons

For å opprette knapper i brukergrensesnittet bruker man XML elementer. 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 øke størrelsen på det hvert 70 ms. 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 hadde rullet til begynnelsen av innspillingen.

#### Noteplassering

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 enn 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 eller pause etter at innspillingen er ferdig, blir noten og alt tilhørende farget blått. Dette skjer ved hjelp av at koden finner foreldre-elementet til noten, og legger et fargefilter over alle barnene til dette elementet, det vil si at alle notens «søsken» og den selv. (Se kapittel 5.3.1 for en nærmere forklaring på hierarkiet.) Hvis noten er en ♭/♯/♮ note uten fortegn, vil appen endre navnet til noten slik at den blir sett av koden som en stamnote. 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 tabellisten 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.

##### ♭/♯/♮

Når brukeren trykker på en av disse tre knappene, sjekkes det først om noten allerede har dette fortegnet 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 tabellen 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 tabellisten 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 tabellen. Hvis noten er opp-ned blir den forskjøvet tilsvarende.

I redigeringen av noten var ideen å bruke setName og setSharp, men på grunn av 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 appens actionbar såfremt det finnes plass på skjermen, mens de fire gjenstående 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

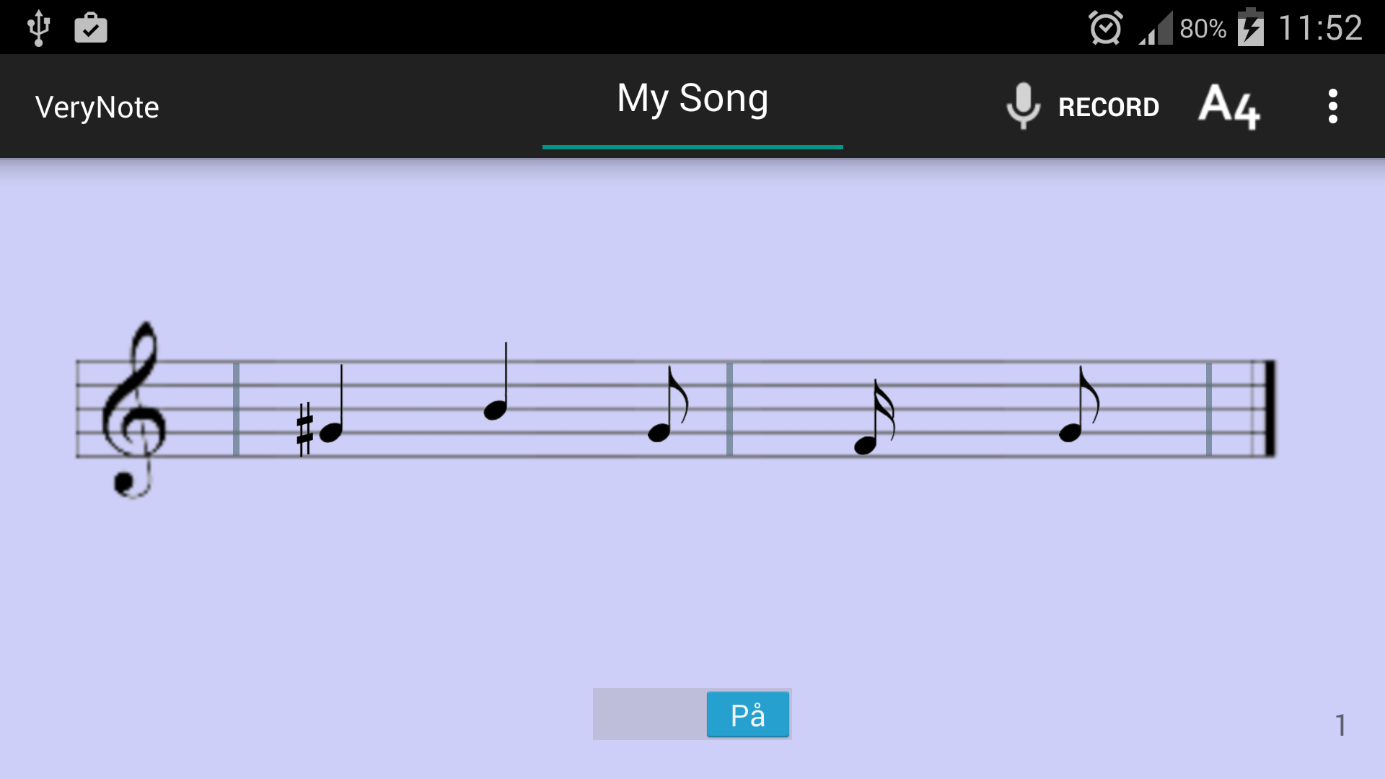
Hovedskjermen består av en «actionbar», navnet på appen, filnavnet og flere knapper og funksjoner. Nederst er en knapp for å slå av og på metronomen, og midt på skjermen er et notesystem med G-nøkkel.



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 et 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 stoppknappen igjen, vil appen slutte å analysere lyden som kommer inn og symbolet vil endre seg tilbake til record-teksten og mikrofonsymbolet. Brukeren kan rulle gjennom notene. Hvis brukeren trykker på record-knappen igjen kan hun fortsette på den samme innspillingen. 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 440 Hz som tilsvarer en enstrøken A. Denne knappen skal hjelpe brukeren å «stemme seg selv» før hun begynner å spille inn en sang. Firetallet representerer oktaven, som tilsvarer det vi kjenner som enstrøken.

#### Tittelfelt

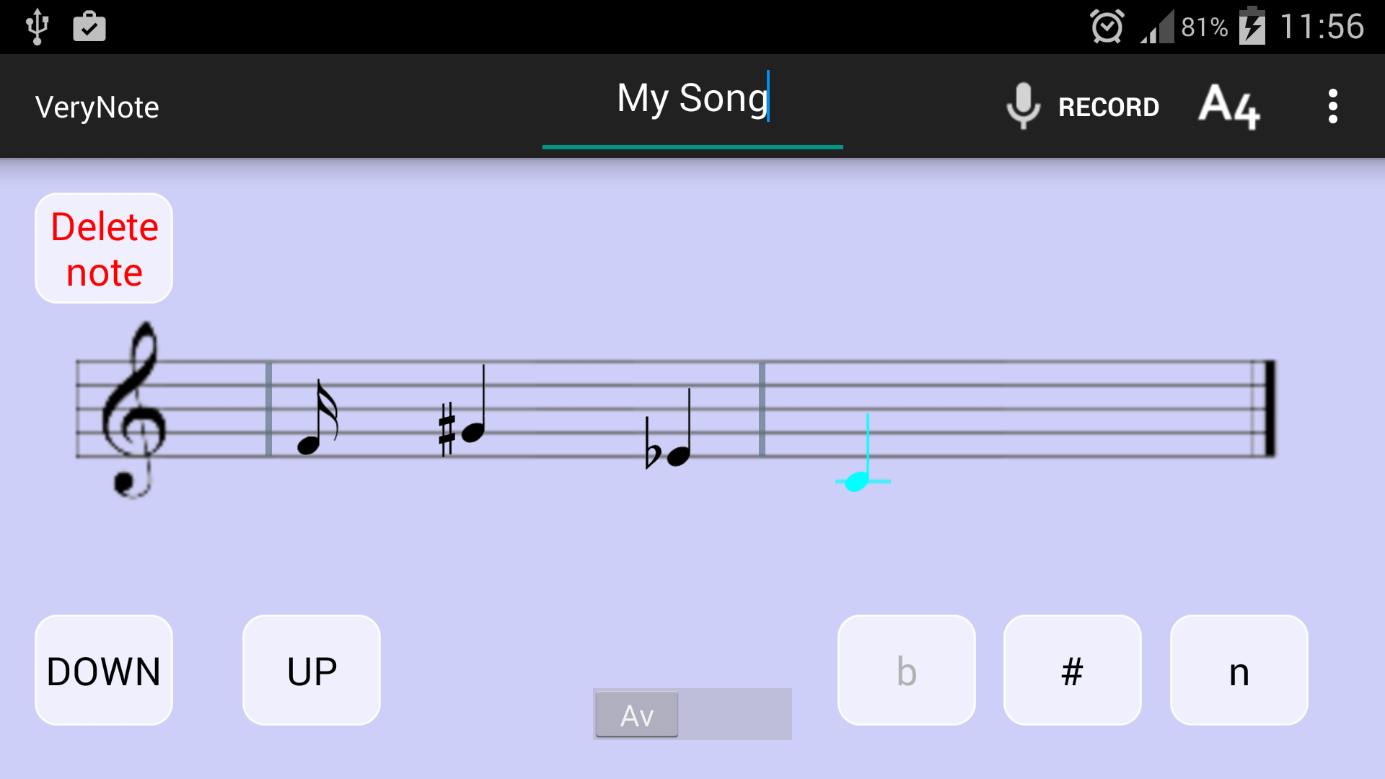
Når appen startes står det «Untitled» øverst på skjermen. Hvis brukeren trykker på ordet, kan de selv legge til en egendefinert tittel på filen de jobber med. Dette navnet blir tittelen på filen om brukeren velger å lagre eller dele filen.

#### Metronom

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

### Redigering

Etter at brukeren har sunget inn notene sine, kan det være at hun vil endre noe. Det er enkelt å legge til/ta bort fortegnene ♭/♯/♮, 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 hun trykke på den samme noten igjen for å velge den bort, eller trykke på en annen note for å velge den isteden.

#### Delete Note

Hvis brukeren har sunget helt feil, eller ombestemt seg kan hun trykke på «Delete note»-knappen. Denne knappen sletter noten fra notesystemet.

#### Down/Up

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

#### ♭/♯/♮

Brukeren kan endre eller fjerne fortegn på den valgte noten. Hvis noten allerede har et fortegn kan dette fjernes ved å trykke på knappen som tilsier dette fortegnet. For eksempel, hvis noten er markert med en ♭ kan brukeren fjerne den ved å trykke på ♭ knappen. Den kan 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 ♯ eller ♭.

### Meny

Metoden for å åpne menyen varierer etter hvor ny enheten 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, og 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 standardisert 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 hun vil gi denne filen et nytt navn, eller finne den gamle filen og slette den.

Share MusicXML-knappen åpner en ny meny som lar brukeren velge hvordan hun vil dele XML-filen. Den kan for eksempel sendes til e-post, eller over Blåtann.



Figur 6.5 – Skjermbilde av delingsmenyen.

#### Settings

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



Figur 6.6 – Skjermbilde av Settings-menyen.

## Test av applikasjon

??

Likevel finnes det begrensinger, for eksempel kvaliteten på opptaket fra mikrofonen. Mikrofonene på telefoner 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.

??

Mot slutten av utviklingen av appen ble vi oppmerksom på at om den tok opp lyd over en lengre periode begynte animasjonen å gå langsommere. Dette var mer merkbart på den eldste testtelefonen 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 at animasjonen begynte å henge etter.

## Konklusjon og videre arbeid

### Konklusjon

Appen VeryNote viser at det er mulig å ta opp og notere musikk ved bruk av smarttelefon. Brukere av VeryNote blir presentert for et enkelt UI, hvor 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 e-post eller Blåtann.

### Videre arbeid

Et godt eksempel på noe vi ikke hadde nok tid til å implementere er dynamisk generering av en MIDI-fil slik at brukeren kan spille av de notene de har lagret. Per i dag støtter ikke Android javax.sound biblioteket for Java [14], og fant vi ingen løsning på dette som kunne blitt implementert innen tidsrammene.

Med noe mer tid ville vi ha kunnet plasser pausene og notene i forhold til taktstrekene, istedenfor i forhold til når noten eller pausen ble begynt. Vi ville også hatt med en hjelpeskjerm som kun ble kjørt den første gangen appen ble åpnet, men også mulighet til å åpne denne hjelpeskjermen opp igjen fra menyen hvis brukeren vil se den igjen. Denne hjelpeskjermen ville bestått av flere gjennomsiktige bilder brukeren kunne ha bladd mellom, med forklaringer av knapper og funksjoner.

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

Vi kunne vi tenkt oss at appen ville gitt mulighet til å velge og å vise taktarten. Det burde være mulig å synge inn i forskjellige tonearter (dur og moll), og markert hvilken toneart som var valgt i margen sammen med nøkkelen og taktarten. Appen ville også hatt større utvalg av noter, både kortere og mer spesielle tilfeller. 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 symbolisere en ny frase. Den ville også bundet sammen notene hvis flere 16-dels eller 8-dels noter ble registrert etter hverandre.

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.

## Vedlegg

*Vedlegg 1: VeryNote* - Kildekode skrevet i Android Studio

All kildekode fra Android Studio utenom PitchDec.java er i helhet skrevet av oss i forbindelse med dette prosjektet. PitchDec.java er skrevet av Aleksey Surkov [9] og er skrevet som en del av et program for en gitarstemmer, modifisert av oss for bruk i dette prosjektet.

* + - Config.java
    - ExportXML.java
    - FitToScreen.java
    - LockableScrollView.java
    - MainActivity.java
    - MyTouchListener.java
    - Note.java
    - NoteNameSearch.java
    - NoteSearch.java
    - PitchDec.java
    - RestListener.java
    - yValueSearch.java
    - fancy\_buttons.xml
    - activity\_main.xml
    - edit\_text.xml
    - settings\_popup.xml
    - menu\_main.xml
    - colors.xml
    - dimens.xml
    - id.xml
    - strings.xml

*Vedlegg 2:* MATLAB kode brukt til å tegne figurer.  
Kildekode fra MATLAB er skrevet av Tom Ryen.

*Vedlegg 3:* Kompilert app.

*Vedlegg 4:* Video av appen i bruk.

## Referanser

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

[2] Lonnie C. Ludeman, *Fundamentals of Digital Signal Processing*. John Wiley & Sons, 1986.

[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>

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

[11] J.W. Cooley, J. Turkey, *An algorithm for the machine calculation of complex fourier series*, 1965. Tilgjengelig fra: <https://web.stanford.edu/class/cme324/classics/cooley-tukey.pdf>

[12] <https://snl.no/PCM>

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

[14] <http://developer.android.com/reference/packages.html>

[15] <https://snl.no/overtoner%2Fmusikk>

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