Afinador musical sobre dispositiu mòbil Android

Escola d'Enginyeria de Terrassa

Marc Torramilans Peidro

Director: Ignasi Esquerra Llucià

30/08/2010

Departament de Teoria del Senyal i Comunicacions

UPC
UNIVERSITAT POLITÈCNICA DE CATALUNYA

Índex

1. Introducció

1.1 Motivació

Abans de decidir quin projecte de final de carrera faria, barrejava vàries opcions, però una cosa la tenia clara: seria un projecte relacionat amb la captura i l'anàlisi de l'àudio. Una de les idees que vaig pensar era fer una espècie de Singstar [3] per ordinador, on l'usuari pogués triar les cançons de la seva pròpia biblioteca i que el programa s'encarregués de processar la cançó i de generar la versió de la cançó en mode Singstar. El Singstar és un videojoc molt famós per la plataforma de Sony Playstation2 i Sony Playstation3 [4] en el qual l'usuari ha de triar una de les cançons de la biblioteca del joc i cantar-la de la manera més semblant possible a la cançó original, fent més aguda o més greu la veu segons la part de la cançó, mitjançant uns micròfons. El programa que volia fer jo, crec que hauria estat perfecte com a projecte de final de carrera, ja que per una banda, seria original: no tothom es sap totes les cançons de les diferents versions del Singstar i no tothom sap cantar-les totes, a tothom li agradaria poder cantar les seves cançons preferides; i per altra banda, hauria ofert una dura però motivant dificultat a l'hora del desenvolupament del projecte. Una de les tasques més difícils hauria estat separar la veu de la música, ja que la majoria de cançons que la gent té als seus ordinadors està en formats d'àudio on veu i música no estan separades, com per exemple en mp3 [5] o wma [6]. Una altra tasca difícil hauria estat lligar la interfície gràfica amb el cos del programa, però amb temps i esforç segur que hauria acabat sortint. A més a més, vaig estar parlant amb una companya de classe que també volia fer una cosa semblant, i si no hagués sigut per la primera dificultat, un s'hauria encarregat de la interfície gràfica i l'altra, del cos del programa. Però el cos del programa era massa complicat.

Davant de la impossibilitat de dur a terme la meva primera idea per al projecte, vaig parlar amb el professor Ignasi Esquerra, el qual ens estava donant una classe de codificació d'àudio i em va proposar de fer un projecte relacionat amb l'Android. Jo en aquell moment no sabia que era l'Android [8], però vaig dir-li que m'ho pensaria durant una setmana. Durant aquella setmana vaig fer dues coses: investigar sobre l'Android i pensar en una

nova idea. La nova idea que vaig pensar era semblant a la idea principal, però aquest cop, enlloc de que l'usuari pogués seleccionar les cançons desde la seva pròpia biblioteca, hauria tingut la opció de comprar-les en format Singstar. Però per fer això, hauria calgut contactar amb molts grups i cantants (òbviament, el benefici hauria anat cap a ells) i això va fer que abandonés de nou la idea. Així que em vaig centrar a investigar sobre l'Android.

Quan vaig veure que l'Android tenia relació amb els Smartphones, em vaig il·lusionar, ja que crec que d'aquí molt poc, igual un parell d'anys, la majoria de la gent tindrà un Smartphone a les seves mans. Per tant, Android té relació amb el futur proper, i això em va motivar, perquè si aconseguia fer una aplicació per Android, potser m'obria algunes portes en el mercat laboral.

Al cap d'aquella setmana, li vaig preguntar al professor quins projectes proposava ell, i quan em va dir de fer un afinador per a guitarra per Android, vaig acceptar quasi de seguida. Dic quasi perquè encara desitjava que em digués que podríem fer la meva primera idea, però un cop em va dir que seria impossible, vaig abandonar les esperances. Un cop vaig acceptar a fer el projecte de l'afinador, ell va acceptar ser el meu tutor del projecte.

1.2 Objectius i Tasques

- Estudiar la plataforma Android a fons per tal de poder desenvolupar aplicacions pròpies i entendre les alienes.
- Estudiar la història de l'Android.
- Investigar sobre les aplicacions d'Android que poden orientar-me amb el projecte.
- Desenvolupar petites aplicacions a mode d'aprenentatge.
- Crear un algorisme capaç de detectar la freqüència dominant en una seqüència d'àudio.
- Implementar un afinador de guitarra per a Android.
- Crear una interfície gràfica que indiqui a l'usuari el grau d'afinació de la corda seleccionada.
- Optimitzar el codi per tal que consumeixi el mínim de recursos possibles.

El projecte té com a objectiu estudiar la plataforma Android i el desenvolupament d'aplicacions en un entorn Open Source per acabar fent l'aplicació d'un afinador per a guitarra.

La primera part és senzilla: definir què és Android i explicar la seva història. Per dur a terme la segona part, l'estudi del desenvolupaments d'aplicacions per a Android, caldrà desenvolupar aplicacions senzilles amb ajuda de guies i tutorials per veure com estan fetes i poder treure'n profit a l'hora de dur a terme la tercera part, el desenvolupament de l'afinador per a guitarra.

1.3 Agraïments

Moltes són les hores que he passat fent aquest projecte i moltes són les persones que s'han ofert a ajudar-me i que sense elles no hauria pogut dur a terme aques projecte.
Per començar, m'agradaria agrair al tutor del projecte Ignasi Esquerra, que m'ha ajudat molt a l'hora de tirar endavant el projecte i a prendre decisions. També agraeixo l'ajuda inestimable dels professors Ramon Morros i Javier Ruiz, que es van prestar a ajudar-me en hores lectives per resoldre'm uns quants dubtes.
De manera particular, m'agradaria als companys de classe Ivó Castells i Pía Muñoz per haver-me ajudat a l'hora de l'aprenentatge de Java i d'Android. Molts altres companys de classe m'han donat suport, així que també els estic agraït.
També m'agradaria agrair a la Júlia Pérez, alumna de Traducció i Interpretació a la UAB, per haver-me ajudat a fer una bona bibliografia.
Per últim, m'agradaria mencionar l'ajuda desinteressada que m'ha proporcionat via e-mail el desenvolupador Aleksey Surkov, ja que cada resposta seva ha sigut d'un valor incalculable.

Moltes gràcies a tots, de debò!

2. Estat de l'art

2.1 Breu introducció a l'Android

Android és un Sistema Operatiu i una plataforma de Software basada en el nucli de Linux [45] , dissenyada principalment per a dispositius mòbils. És una plataforma de codi obert, per tant, qualsevol programador pot crear i desenvolupar aplicacions escrites en C o en Java mitjançant la corresponent API d'Android [9] .

Inicialment, Android va ser desenvolupat per Android Inc. [38], empresa que va ser comprada per Google Inc. [39] a la que se li va unir Open Handset Alliance [40], un consorci de 48 companyies de hardware, software i telecomunicacions les quals havien arribat a un acord per promocionar els estàndards de codi obert per a dispositius mòbils.

Il·lustració 1: Logotip d'Android

Google ha sigut qui ha publicat la majoria del codi font d'Android sota la llicència de software Apache [41], una llicència de software lliure i de codi obert a qualsevol desenvolupador.

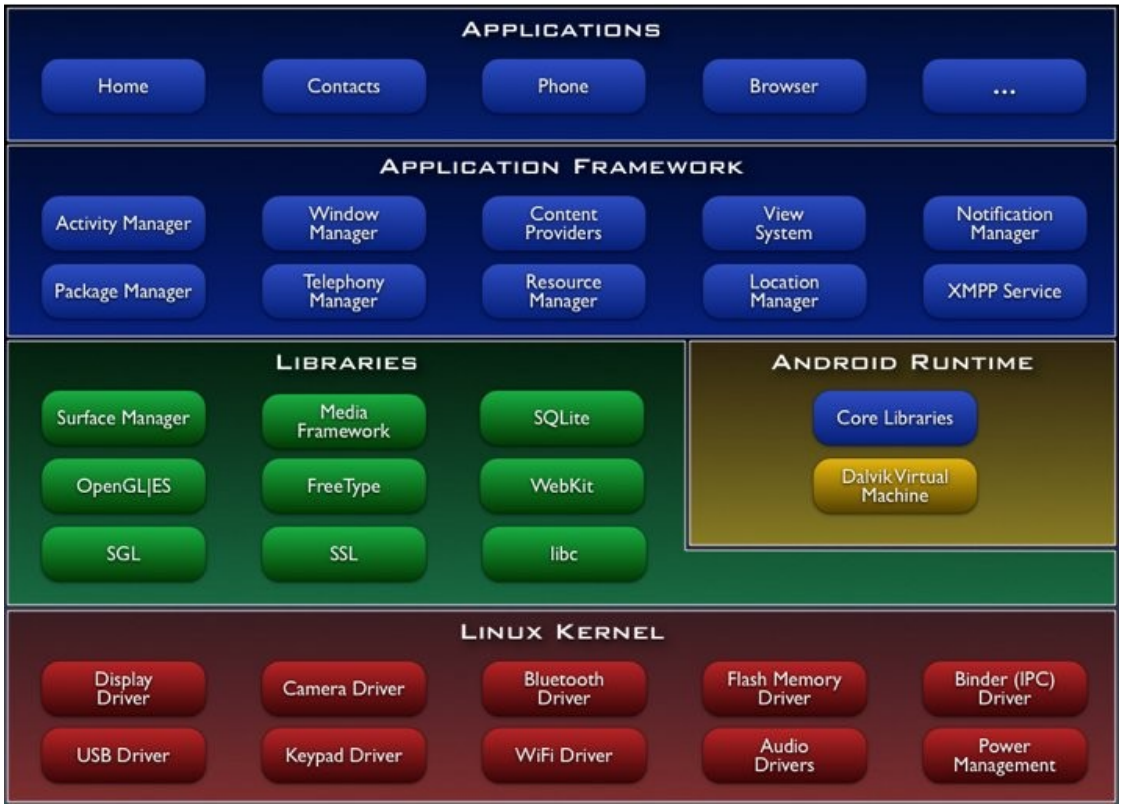
2.2 Característiques principals de la plataforma

De les característiques d'Android [10] que l'han fet situar-se al capdavant dels actuals sistemes operatius per a mòbils, en destaquen les següents:

- Framework d'aplicacions per tal de re-utilitzar i reemplaçar components
- Exporador Web integrat, basat en el motor WebKit de codi obert, que és el que utilitza el Safari, d'Apple.
- Màquina Virtual Dalvik
- Gràfics 2D SGL i 3D OpenGL ES 1.0
- Tecnologia GSM
- SQLite
- Suport de formats d'àudio, vídeo i imatges (H.264, MPEG4, AAC, AMR, MP3, JPG, GIF, PNG)
- Connexions Bluetooth, EDGE, 3G i WiFi
- Càmera, acceleròmetre, GPS i bruixola
- SDK, que permet crear aplicacions als desenvolupadors de manera gratuïta, i inclou un emulador i eines per a la depuració de les aplicacions

2.3 Arquitectura Android

A continuació es mostren les components bàsiques del sistema Android [10]:



Il·lustració 2: Esquema de l'arquitectura d'Android

Aplicacions: Són les aplicacions bàsiques incloses per defecte, tals com un client de mail, programa per enviar i rebre SMS, per navegar per Internet, per veure mapes i posicionar-se sobre ells, una agenda de contactes, etc.

Framework d'aplicacions: A l'hora de desenvolupar es té accés a totes les APIs del Framework que es fan servir a les aplicacions. L'arquitectura s'ha dissenyat per tal de facilitar la reutilització de components; qualsevol aplicació pot publicar les seves capacitats amb una regla de seguretat, i totes les aplicacions que compleixin aquesta regla podran accedir a elles i utilitzar-les. Aquest mateix
mecanisme permet que els components puguin ser substituïts per l'usuari. Una capa de serveis disponibles per a les aplicacions inclou:

- Un conjunt molt complet i ampliable d'objectes i vistes tals com llistes, graelles, caixes de text, caixes d'imatges, botons o fins I tot un navegador de webs.
- Proveïdors de continguts que permeten compartir les dades entre aplicacions, com per exemple obrir la llibreta de contactes en una aplicació.
- Un administrador de recursos que proporciona accés a recursos com cadenes, gràfics o arxius.

Llibreries: Android inclou un conjunt de llibreries de C/C++ que s'utilitzen per diverses components del sistema. Aquestes llibreries es poden accedir a través del Framework de les aplicacions. Algunes de les llibreries del nucli són:

- "System C library": Una variant de BSD (Berkeley Software Distribution) de l'estàndard de llibreries de sistema de C (libc).
- "Media Libraries": Suporta la reproducció i enregistrament de diversos formats d'àudio imatge i vídeo com MPEG4, H.264, MP3, AMR, AAC, JPG i PNG.
- "Surface Manager": Gestiona l'accés al subsistema de visualització i permet manegar sense gaires problemes capes 2D i 3D de les aplicacions.
- "SGL (Scene Graphics Library)": És el motor de gràfics 2D.
- "3D Libraries": Una implementació basada en APIs OpenGL ES 1.0 que utilitza l'acceleració 3D del hardware si en disposa.
- "FreeType": Permet la renderització de vectors i Bitmaps.
- "SQLite": Una potent base de dades relacionals de gran abast i lleuger disponible per a totes les aplicacions.

"Runtime" d'android: Android inclou un conjunt de biblioteques bàsiques que proporcionen la major part de la funcionalitat disponibles a les llibreries base del llenguatge de programació Java.
Cada aplicació d'Android s'executa sobre el seu propi procés, amb una instància pròpia de la màquina virtual Dalvik. Dalvik s'ha dissenyat per que un mateix dispositiu pugui executar diverses màquines virtuals eficientment.

Linux Kernel: Android es basa en la versió Linux 2.6 per als serveis del nucli del sistema, tals com el sistema de seguretat, la gestió de memòria, la gestió de processos, la pila de

xarxa o el model del driver. El nucli també actua com una capa d'abstracció entre el maquinari i la resta de la pila de programari.

2.4 Filosofia de disseny d'aplicacions

Les aplicacions desenvolupades per a dispositius mòbils s'assemblen bastant a les aplicacions que s'utilitzen en sistemes més gran, com els ordinadors de sobretaula, però la seva capacitat de processat és molt més baixa. Per tal que les aplicacions siguin eficients, Android proposa una sèrie de característiques [10] com a base per a qualsevol aplicació per mòbils. Una aplicació ha de ser:

1. Ràpida: és a dir, eficient, que consumeixi el mínim de recursos possibles.
2. Interactiva: és a dir, que es comuniqui amb l'usuari en tot moment perquè aquest sàpiga què està passant tota l'estona.
3. Fluïda: és a dir, que les aplicacions es comuniquin entre elles per tal d'agilitzar els processos, depenent de la prioritat d'aquests.

2.5 Particularitats d'una aplicació per a Android

Una aplicació per a Android es pot observar com un conjunt de blocs que interaccionen entre ells. Cada bloc du a terme una funció diferent dins de les diferents aplicacions del sistema [2]. Els blocs són els següents:

- **Activities:** cada Activity representa una de les pantalles amb les que l'usuari pot interaccionar. Les Activities tenen un cicle de vida semblant al de les pàgines d'un explorador web, de manera que es pot anar endavant i endarrere.
  Cada cop que es genera una nova Activity, l'anterior es pausa i passa a l'historial, tot i que l'usuari pot decidir si guardar-la a l'historial o esborrar-la. A més a més, existeixen activitats anomenades SubActivities que no es guarden a l'historial, ja que són Activities que només interactuen amb l'Activity que les ha cridat.
- **Intents:** un Intent és una classe que permet executar una acció, com per exemple

passar d'una pantalla a una altra. Internament, manifesten quina acció vol executar l'usuari per tal que el sistema utilitzi les aplicacions adients. Per tal d'efectuar aquestes accions, les aplicacions defineixen uns elements anomenats IntentFilters en l'arxiu anomenat manifest.xml. Els IntentFilters indiquen les accions que cada Activity ha de fer. Si dues Activities volen fer servir el mateix IntentFilter, la farà servir l'Activity amb una prioritat més gran, i si la prioritat és la mateixa, el sistema farà escollir a l'usuari.

- **Intent Recievers:** els Intent Recievers són classes que s'executen automàticament com a reacció d'un esdeveniment extern, com per exemple un SMS o una trucada. Per informar a l'usuari, han d'interaccionar amb el Notification Manager, que s'encarrega de indicar a l'usuari les accións que pot fer, com per exemple contestar la trucada o llegir el SMS.
- **Services:** Els Services són processos de llarga duració que no fan ús de la interfície gràfica, per exemple un reproductor de música, que permet a l'usuari escoltar cançons mentre executa qualsevol altra aplicació.
- **Content Providers:** Es tracten de gestors d'informació perquè les aplicacions puguin compartir informació entre elles. Implementen un conjunt de mètodes (copiar, eliminar, inserir...) perquè les aplicacions puguin guardar i recuperar dades. Aquests blocs s'utilitzen en conjunt amb el gestor de bases de dades SQLite, tot i que es poden fer servir altres sistemes de gestió, com per exemple fitxers XML o altres gestors de bases de dades.

2.6 Android Manifest

L'Android Manifest és un document XML anomenat AndroidManifest.xml [43] que han d'incloure totes les aplicacions desenvolupades per tal de poder ser utilitzades a la plataforma Android.

Aquest document es troba al directori arrel de cada aplicació i descriu els components de l'aplicació (Activities, Services, IntentFilters...) i les classes que els implementen. També es declara quin tipus de dades poden tractar i el moment en el que es pot fer ús de cadascun

dels seus components.

També serveix perquè l'aplicació demani els permisos adients per a utilitzar un determinat element del sistema.

Exemple de l'Android Manifest de l'aplicació record_audio, explicada a la secció d'aplicacions provades:

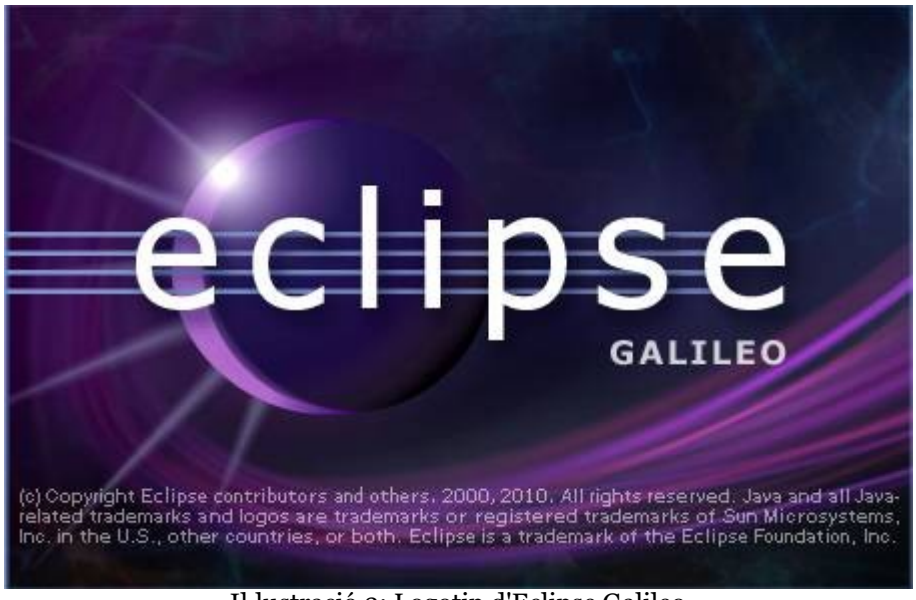
```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
      package="com.example.record_audio"
      android:versionCode="1"
      android:versionName="1.0">
    <application android:icon="@drawable/icon" android:label="@string/app_name">
        <activity android:name=".record_audio"
                  android:label="@string/app_name">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>

    </application>
      <uses-permission android:name="android.permission.RECORD_AUDIO"></uses-permission>
      <uses-permission  android:name="android.permission.WRITE_EXTERNAL_STORAGE"></uses-
permission>


</manifest>
```

2.7 Entorn de desenvolupament

2.7.1 Per començar

Per començar a desenvolupar una aplicació per Android, el primer que cal fer es descarregar-se la versió corresponent de l'Android SDK (Software Development Kit), tenint en compte el sistema operatiu en el que treballarem (Windows, Mac OS o Linux) [11]. També cal tenir en compte si el nostre sistema operatiu compleix tots els requeriments de software i hardware necessaris per al bon funcionament de l'Android SDK, cosa que podem comprovar des de la mateixa web d'Android Developers [7].

Per desenvolupar una aplicació es poden fer servir diferents entorns de programació, però es recomana fer servir l'Eclipse [12]. Aquest projecte ha estat realitzat amb Eclipse Galileo 3.5.2, en un sistema operatiu Mac Os X.



Il·lustració 3: Logotip d'Eclipse Galileo

2.7.2 Android Developement Tools (ADT)

Si es vol fer servir Eclipse, és molt recomanable descarregar-se també l'ADT Plugin (Android Development Tools), dissenyat per a integrar l'entorn de l'Eclipse per desenvolupar aplicacions per a Android [13]. Millora la capacitat de l'Eclipse a l'hora de crear nous projectes d'Android, afegir projectes ja existents, depurar aplicacions fent servir les eines de l'Android SDK... En definitiva, fa que quan volguem desenvolupar una aplicació d'Android fent servir Eclipse, vagi millor i més ràpid.

2.7.3 Android SDK

Com ja hem vist, l'Android SDK és la eina bàsica per a poder desenvolupar aplicacions per Android. Per començar a programar una aplicació, el primer que cal fer és crear una AVD (Android Virtual Device) fent servir l'eina anomenada android, a la carpeta tools de l'SDK. Una AVD defineix la imatge del sistema que farà servir l'emulador. L'eina anomenada android és l'Android SDK And AVD Manager, que a part de permetre'ns crear AVDs, serveix per actualitzar l'SDK per tenir-lo sempre al dia. A l'hora de crear una nova AVD, podem triar si volem una SD Card, una memòria virtual que ens permetrà guardar fitxers generats amb les aplicacions de l'emulador, com per exemple cançons gravades, imatges o fitxers de text. També podem triar l'estètica de l'emulador i definir la seva resolució.

Existeixen diferents versions de l'SDK, tant per a Windows com per a Mac Os o Linux. També existeixen diferents nivells de plataformes, començant per les mes antigues (1.0) fins a les de l'actualitat (2.2). Cada nivell té un màxim API Level (de l'1 al 8), que determina quines llibreries es fan servir, quins Intents es permeten i com s'ha de fer el manifest.

L'SDK ve amb les llibreries necessàries per programar amb el ADT, amb el AVD Manager, un emulador i codis d'exemple.

2.7.4 Emulador

A l'hora de provar les aplicacions, l'Android SDK proporciona un emulador que corre sobre un ordinador. L'emulador emula un dispositiu Android real, de manera que permet veure al programador com queda el seu programa. Existeixen versions diferents de l'emulador, depenent del sistema operatiu (Windows, Mac Os i Linux).

Depenent de la plataforma en la que estiguem programant, es poden seleccionar unes "skins" o unes altres, per representar per pantalla un dispositiu mòbil real, amb els botons de menú, tornar enrere, pujar i baixar el volum i els botons que fan de fletxes direccionals, així com d'un teclat qwerty [42].

Per executar-lo, cada sistema operatiu te la seva manera de fer-ho, però en tots tres es pot executar fent servir el connector de l'Eclipse, cada cop que es fa run o debug. Cada AVD creada té el seu propi emulador, i si les aplicacions creades a l'Eclipse fan servir AVDs diferents, es poden executar diferents emuladors alhora sense problema.

Per interactuar amb l'emulador, a part de fer servir els botons, fent servir el ratolí, també es pot clickar damunt la pantalla, que representa que es tàctil, amb el ratolí, com si aquest fos el dit.

Il·lustració 4: Vista de l'emulador d'Android

2.7.5 Android NDK

L'Android NDK (Native Development Kit) és una eina complementària per al SDK que permet fer servir llibreries i funcions escrites en altres llenguatges de programació, com per exemple C o C++, en les aplicacions per a Android de manera que no calgui traduir a Java codis escrits en altres llenguatges [14]. Una cosa que cal tenir en compte és hi ha diferents versions de l'NDK, però són independents a la plataforma Android i amb la API Level amb la que es treballi.
Al igual que l'Android SDK, l'NDK ve amb un paquet de codis d'exemple per veure com funciona. Per a utilitzar l'NDK cal posar els codis escrits en altres llenguatges dins el directori <ndk>/sources/<my_src>/. Dins d'aquest directori s'ha de ficar un arxiu de descripció Android.mk i també s'ha d'afegir un arxiu de descripció que enllaci amb el SDK que ha d'estar a <ndk>/apps/<my_src>/Android.mk. Per acabar d'enllaçar s'ha de teclejar en el directori arrel de <ndk> la següent instrucció: $make APP=<my_app>.

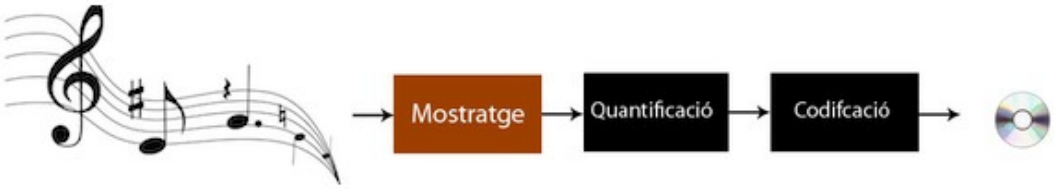
En aquest projecte s'han fet probes amb l'NDK, explicades més endavant.

3. Base teòrica

Per entendre com funciona la detecció de la freqüència dominant, abans cal estudiar uns quants conceptes teòrics, com per exemple què és la digitalització i què és l'anàlisi freqüencial. Posteriorment cap estudiar les diferents maneres d'obtenir la freqüència fonamental d'un senyal d'àudio.

3.1 Digitalització del senyal

Per a digitalitzar un senyal continu en el temps, es segueixen els següents passos: Mostreig > Quantificació > Codificació. A continuació s'expliquen els trets més importants de cada pas.



Il·lustració 5: Passos de la digitalització

3.1.1 Mostreig

La freqüència és la mesura de la quantitat de vegades que una ona completa un període dins un temps d'un segon [16]. Es mesura en Hertz (Hz). 1 Hz vol dir que en un segon, una ona completa exactament un període, sent el període de la ona la inversa de la freqüència:

$$f = \frac{1}{T}$$

on T és el període i f la freqüència.

El mostreig és el primer pas que cal fer per a digitalitzar un senyal temporal (analògic). És una operació que consisteix en discretitzar, és a dir, obtenir mostres espaiades en el temps mitjançant una freqüència de mostreig que indica el nombre de mostres per segon que es

prenen. Aquesta freqüència de mostreig ve condicionada per l'ample de banda del senyal analògic, tal com indica el teorema de Nyquist [15]: “La freqüència de mostreig mínima que es requereix per a realitzar una gravació digital de qualitat ha de ser superior al doble de la freqüència d'àudio més alta del senyal analògic que s'intenta digitalitzar.”
Si la freqüència més alta continguda en el senyal analògic xa(t) és:

$$Fmax = B$$

i la freqüència de mostreig Fs és:

$$Fs > 2\mathrm{Fmax} = 2\mathrm{B}$$

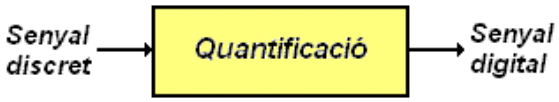
llavors xa(t) es pot recuperar totalment mitjançant la funció d'interpolació:

$$g(t) = \frac{\sin(2\pi Bt)}{7\pi Bt}$$

El no compliment del teorema de Nyquist introdueix un error anomenat Aliasing, que consisteix en un encavalcament del senyal que provoca que quan es vulgui recuperar el senyal no s'obtingui exactament l'original.

3.1.2 Quantificació

Un cop discretitzat el senyal, cal quantificar-lo de manera que el rang d'amplituds passi de ser continu a discret, ja sigui per arrodoniment o per truncament dels valors. Amb aquest pas aconseguim no s'hagin de representar totes les amplituds i per tant que el nombre de bits a utilitzar sigui menor. De fet, el nombre de nivells que s'empren per a la quantificació depèn del nombre de bits de quantificació.



Il·lustració 6: Esquema de la quantificació

En el procés de quantificació [17].es dóna un valor finit d'amplitud a cada mostra obtinguda al procés de mostreig, seleccionat per aproximació dins un marge de nivells prèviament fixat.

Els paràmetres bàsics de la quantificació són:

1. El nombre de bits de quantificació, Nbits.
2. El nombre de nivells de quantificació, $Nnivells = 2^{Nbits}$, que indica el nombre de nivells en que dividim el marge dinàmic.
3. El marge dinàmic. En analògic, és la diferència entre la amplitud màxima i la mínima, és a dir:
   $MD\,analog = Vmax - Vmin$

   En digital, ve donat per la fórmula:
   $MD\,digital = 20\log_{10}(Nnivells) = 20\log_{10}(2^{Nbits}) = 20\log_{10}(2)\cdot Nbits = 6{,}0206\cdot Nbits$
4. El pas de quantificació és la diferència d'amplitud entre dos nivells de quantificació.

$$Triangle = \frac{MD}{Nnivells}$$

Procediment:

El marge dinàmic és segmenta en N nivells, depenent del nombre de bits que s'emprin. Després es mesuren les amplituds de les mostres del senyal d'entrada i se'ls hi assigna per arrodoniment un valor finit, el del nivell de quantificació que estigui més a prop.

Per culpa d'aquest arrodoniment és inevitable que hi hagi un error de quantificació. Es pot definir com la diferència entre el senyal mostrejat, continu en el seu rang de valors, i el senyal quantificat, discret en el seu rang de valors. Aquest error és com a màxim la meitat del pas de quantificació. Com més gran sigui el nombre de bits utilitzats per a la quantificació, més petit serà l'error de quantificació.

Hi ha diferents tipus de quantificació:

1. Quantificació uniforme: La diferència entre els nivells de quantificació és sempre la mateixa. La taxa de bits és constant per a tot el marge de valors del senyal.
2. Quantificació no uniforme: S'assignen més o menys bits depenent de l'entropia (l'estudi de les probabilitats) del senyal. La diferència entre els nivells de quantificació és variable, així com la taxa de bits. S'assignen més bits en les parts del senyal on es vulgui tenir menys error de quantificació.
3. Quantificació logarítmica: Es passa el senyal per un compressor logarítmic i després s'aplica una quantificació uniforme. Es fa servir normalment èr a senyals de veu, com per exemple en la telefonia.

4. Quantificació vectorial: Es basa en la quantificació de les mostres en blocs de mostres, on cada bloc es tracta com un vector. Es quantifiquen els vectors, obtenint així una major eficiència.

3.1.3 Codificació

Hi ha diferents algorismes de codificació, depenent de l'origen del senyal i del seu ample de banda, però tots consisteixen en donar un codi binari a cada valor diferent obtingut al pas de la quantificació. Alguns algorismes, a més a més, comprimeixen el senyal de manera que ocupi menys [18].

Els algorismes de codificació que comprimeixen es poden subdividir en dues grans famílies: els codificadors amb pèrdues i els codificadors sense pèrdues.

Els codificadors sense pèrdues aprofiten la redundància del senyal per reduir la mida del flux de dades. Si el senyal te patrons repetitius, és fàcil predir-lo i eliminar la redundància. Alguns exemples d'algorismes de codificació sense pèrdues són: la codificació de Huffman, la codificació aritmètica i el Run Length Coding (RLE).

Els codificadors amb pèrdues aprofiten les limitacions del sistema auditiu humà per comprimir el flux de dades. Aquestes limitacions són el llindar d'audició, l'emmascarament temporal i l'emmascarament freqüencial.

En els temes de captura de so, l'Android codifica les dades amb la codificació PCM (Pulse Code Modulation), és a dir, no hi ha compressió.

3.2 Anàlisi freqüencial

L'anàlisi freqüencial permet obtenir informació sobre les freqüències que conté un senyal. Hi ha diverses tècniques per fer-la, però la majoria es basen en les equacions de Fourier [19]. Com que en el projecte es parteix de senyals discretes, explicaré les diferents

tècniques per fer les anàlisis freqüencials de senyals discretes.

3.2.1 Transformada Discreta de Fourier (DFT)

La Transformada Discreta de Fourier (DFT, de l'anglès Discrete Fourier Transform) [20] és una operació que permet obtenir la versio discreta de la Transformada de Fourier d'un senyal discret (DTFT, de l'anglès Discrete Time Fourier Transform), que permet passar un senyal del domini temporal al domini freqüencial. El senyal resultant de la DTFT és un senyal continu i de variable real, ja que s'aplica a un senyal discret:

$$X(\omega)=\sum_{n=-\infty}^{\infty} x[n]e^{-j\omega n}$$

A partir de l'expressió $\frac{\omega}{2\pi}=\frac{k}{n}$, es pot trobar la freqüència angular d'un senyal discret, substituint ω per $\frac{2\pi k}{n}$, obtenint un senyal discret de N mostres, la DFT, que ve donada per la fórmula:

$Xn[\omega]=\sum_{n=0}^{N-1} x[n]e^{\frac{-j2\pi kn}{N}}$, on ω és la freqüència angular, N el número de mostres desitjades, k la variable independent del senyal obtingut i n la variable independent del senyal original.

3.2.2 Transformada Ràpida de Fourier (FFT)

La Transformada ràpida de Fourier (FFT, de l'anglès Fast Fourier Transform) és un conjunt d'algorismes que permeten calcular de forma molt ràpida i eficient la DFT [21].
En termes d'eficiència, la DFT té un cost temporal de O(n2) i la FFT de O(nlog n) [22]. En transformades de poques mostres la diferència és quasi inapreciable però en transformades de, per exemple, 1024 mostres, per fer la DFT es necessitarien aproximadament 10^6 operacions. Per fer la FFT només caldrien al voltant de 5000 operacions.

És degut a aquesta diferència de cost temporal i a la necessitat de passar del domini temporal al freqüencial en temps real que en el món de les telecomunicacions la FFT és de les transformades més emprades.

Un valor que cal determinar a l'hora de fer la FFT és el nombre de mostres (punts) de la FFT. Amb un nombre de punts més elevat, la resolució de la FFT també ho serà. Els nombres de punts de la FFT més emprats són 512, 1024, 2048 i 4096.

3.3 Obtenir la Freqüència Fonamental d'un senyal d'àudio

Es defineix com a freqüència fonamental d'un senyal, la freqüència més baixa de l'espectre freqüencial del senyal, la freqüència que nosaltres percebem. Per calcular-la, els següents mètodes són els més emprats:

1. Detecció de creuaments per zero.
2. Detecció de pics.
3. Autocorrelació.
4. Anàlisi Cepstral.

3.3.1 Detecció de creuaments per zero

Per determinar la freqüència fonamental a partir dels creuaments per zero, cal tenir en compte la freqüència de mostratge amb la que s'ha mostrejat el senyal i el valor ZCR (Zero Crossing Rate), que indica el nombre de creuaments per zero per mostra. El ZCR dóna una indicació de si una trama és periòdica (sons sonors) o sorollosa (sons sords i silencis). Amb aquests dos valors, la freqüència fonamental Fo val:

$$F0 = \frac{ZCR \cdot Fm}{2}$$

3.3.2 Detecció de pics

A partir de l'espectre del senyal, es poden observar pics en algunes freqüències. La freqüència fonamental és la freqüència amb més amplitud. No obstant, pot ser que el senyal tingui harmònics amb més energia que la de la freqüència fonamental. Per tal d'evitar que els harmònics influeixin a la detecció de la freqüència fonamental, és necessari un filtrat pas-baix del senyal. El filtrat pas baix redueix les amplituds de les freqüències més altes, per tant les amplituds dels harmònics.
Un altre problema que pot sortir és que la freqüència amb més amplitud sigui 0 Hz, és a dir, que predomini el silenci. Per tant, cal fer l'anàlisi eliminant la component continua.

3.3.3 Autocorrelació

L'autocorrelació és una funció matemàtica que consisteix en la correlació creuada d'un senyal amb ell mateix [23]. Mesura la similitud d'un senyal quan se li aplica un desplaçament relatiu, per tant no cal fer l'anàlisi freqüencial del senyal.
L'autocorrelació discreta R amb un desfasament j per a un senyal Xn i un valor esperat m de Xn és:

$$R(j) = \sum_{n} (X_n - m)(X_{(n-j)} - m)$$

Proporciona informació sobre les periodicitats del senyal i la freqüència fonamental.

3.3.4 Anàlisi Cepstral

El cepstrum d'un senyal és el resultat de calcular la Transformada de Fourier del logaritme de la Transformda de Fourier del senyal [24]. També s'anomena espectre de l'espectre.
La part real del cepstrum fa servir la funció algorítmica que es fa servir per a funcions reals i proporciona magnituds de l'espectre, mentre que la part complexa del cepstrum fa servir la funció algorísmica per calcular la Transformada de Fourier per a valors complexos i conté informació sobre el valor i la fase inicial de l'espectre.

El cepstrum permet calcular la freqüència fonamental d'un senyal, generalment de veu, ja que l'anàlisi cepstral es fa servir molt en aplicacions de veu.

3.4 Afinació d'una guitarra

Perquè totes les cordes d'una guitarra estiguin correctament afinades, cal que cadascuna vibri amb una determinada freqüència. Aquestes freqüències són les següents [25]:

(començant de baix a dalt)

- 1era corda (E/mi): 329.63 Hz
- 2ona corda (A/la): 246.94 Hz
- 3era corda (D/re): 196.0 Hz
- 4rta corda (G/sol): 146.83 Hz
- 5ena corda (B/si): 110.0 Hz
- 6ena corda (E/mi): 82.41 Hz

Per aconseguir-ho, hi ha diferents mètodes. El mètode tradicional és fer-ho d'oïda, amb l'ajuda d'un diapasó. (foto d'un diapasó) La freqüència amb la que vibra un diapasó és de 440 Hz, és a dir, la segona octava del La de 110 Hz de la cinquena corda. D'oïda, s'ha de sentir que el so del diapasó i el de la cinquena corda a l'aire ha de ser la mateixa nota, malgrat no soni igual ja que el so del diapasó és més agut. Si no donen la mateixa nota, s'ha d'anar girant les clavilles en un sentit o en un altre per tensar o destensar la corda. Un cop aconseguit, cal afinar les demés cordes, fent servir el següent mètode:

- La sisena corda amb un dit al cinquè trast ha de sonar igual que la cinquena corda a l'aire.
- La cinquena corda amb un dit al cinquè trast ha de sonar igual que la quarta corda a l'aire.
- La quarta corda amb un dit al cinquè trast ha de sonar igual que la tercera corda a l'aire.

- La tercera corda amb un dit al cinquè trast ha de sonar igual que la segona corda a l'aire.
- La segona corda amb un dit al quart trast ha de sonar igual que la primer corda a l'aire.

Una altra manera d'afinar les cordes d'una guitarra és mitjançant un afinador electrònic. Cada afinador té les seves instruccions, però tots es basen en una sèrie de passos: capturar el so, determinar quina nota es vol tocar i indicar si està ben afinada o no, dient quants Hertzs per sota o per sobre es troba la freqüència de la corda que s'ha tocat. Gràcies a això, la persona que afina la guitarra sabrà si ha de tensar o destensar la corda.

3.5 Distància entre semitons

En la música moderna occidental es fa servir el terme temperament igual per nombrar un sistema d'afinació per afinar un interval, per exemple una octava, en 12 parts igual logarítmicament, és a dir, en 12 semitons [26]. Per tant, la distància proporcional entre semitons és la dotzena arrel de dos, és a dir 1.0594... Si prenem el valor 1.06, l'error no és gaire gran. Per tant, podem dir que la distància entre un semitò A i un semitò B que està just a sobre del semitò A en l'escala musical és A x 1.06, és a dir, la freqüència de B és un 6% més gran que la de A.

Per determinar si una corda està ben afinada, els afinadors han de determinar si la freqüència fonamental del so que capten està dins l'interval que va des de la freqüència fonamental teòrica menys un 2.9 % fins a la freqüència fonamental teòrica més un 2.9%.

4. Creació del codi

A continuació s'explicarà tots els passos que vaig seguir a l'hora de crear el codi per a un afinador de guitarra.

4.1 Funcions i classes necessàries

Per a fer una aplicació per a Android que permeti a l'usuari afinar una guitarra, el primer que cal fer es fer-se un esquema amb el que es vol. Jo tenia clar que el que necessitava era una manera de capturar so i escriure les dades en un buffer, de manera que després es pogués accedir a les dades i fer-ne l'anàlisi freqüencial. Les dues classes d'Android que permeten capturar àudio són [27]:

- MediaRecorder [1], que permet gravar el so en la SD Card virtual. A més a més, després permet reproduir el so gravat.
- AudioRecord [28], que permet capturar so i escriure les dades en un vector.

Així doncs, la classe que jo necessitava era la AudioRecord. Més endavant, a l'explicació del codi, estan explicats el constructor de la classe i els mètodes de la classe utilitzats.

Una altra funció que tenia clar que hauria de fer servir és la FFT. Vaig intentar implementar-la de moltes maneres, explicades més endavant, però finalment vaig optar en fer servir una classe FFT que ja estava feta i que vaig trobar per Internet [46].

A més a més de la FFT, també vaig creure necessari crear una funció que detectés la freqüència dominant de cada buffer de dades d'àudio per tal de comparar-la amb els valors de freqüència ideals de cada corda d'una guitarra i que digués si una corda està ben afinada o no.

4.2 Interacció amb l'usuari

Un dels avantatges de fer un afinador de guitarra per a Android és que la aplicació pot interactuar amb l'usuari i l'usuari amb aquesta. La aplicació interactua amb l'usuari indicant-li què pot fer i quin és l'estat actual en el que es troba, i l'usuari interactua amb la aplicació prement botons per indicar-li què vol fer.

Així doncs, l'aplicació havia de constar de botons amb opcions perquè l'usuari trii què vol fer. Vaig pensar que constaria d'un menú principal amb tres botons: un per sortir de l'aplicació, un per començar-la i un que fes sortir un text amb les instruccions.
Un cop començada l'aplicació, la idea principal era que l'usuari veiés les sis cordes d'una guitarra i pogués seleccionar-ne una. D'aquesta manera, l'aplicació sabria quina corda vol afinar l'usuari. Després, l'usuari tocaria una corda de la seva guitarra i l'aplicació li diria si està ben afinada o no, i en cas que no ho estigués, si havia de tensar o destensar la corda, tot això a temps real i mitjançant unes fletxes cap amunt o cap avall.

4.3 Buscant l'algorisme

Un cop tenia clar el que volia, vaig decidir que primer faria proves amb codis ja creats, obtindria experiència i després em llençaria a fer el meu propi codi. A continuació hi ha una llista amb totes les aplicacions i projectes provats, amb una breu explicació del que fan i amb el seu estat.

4.3.1 Aplicacions i projectes provats

Android Sudoku [2]:
És una aplicació per a Android que ens permet jugar al Sudoku. Permet guardar partides, triar el nivell de dificultat, activar i desactivar sons i pistes per a la resolució del puzzle i inclou un mode per a la resolució vertical.
Estat: Funciona correctament, excepte el mode per a la resolució vertical.

Audio [2]:

És una aplicació senzilla que reprodueix fitxers d'audio quan apretem una tecla determinada de la terminal. Els fitxers d'audio es troben en una carpeta de la aplicació, concretament la carpeta Raw.

Estat: Funciona correctament.

ApiDemos [11]:

És una aplicació que ve amb el SDK que inclou petites aplicacions de mostra per a Android, com per exemple dibuixar amb el dit, reproduir música de la SD Card, enregistrar sons a la SD Card, crear gràfics, fer llistes... La llàstima és que no es pot veure el codi de totes les aplicacions.

Estat: Funcionen gairebé totes les aplicacions.

Jet Boy [29]:

És un joc per Android on el personatge es una nau que es mou sola i ha d'intentar destruir el màxim nombre de meteorits possibles, disparant-los amb la tecla corresponent. Suposadament el més important del joc és la banda sonora, que s'adapta al jugador de manera que si ho fa bé la música s'anima.

Estat: L'aplicació s'executa i es pot disparar als meteorits, però no es pot escoltar la música.

MusicDroid [30]:

És una aplicació que permet reproduir música des de la sdcard. Permet passar a la següent cançó, a la anterior, pausar i continuar.

Estat: Hi ha errors que no he sabut resoldre, per tant no he comprovat si funciona.

Grabarmusica:

És una aplicació que permet gravar i reproduir veu des del micròfon de l'Android.

Estat: No he aconseguit fer que funcioni perquè no entenc l'error.

GuitarTuner [31]:

És una aplicació que consisteix en un afinador de guitarres per a Android. Funciona de manera que l'usuari escull una corda per pantalla, toca aquella corda en la seva guitarra, l'aplicació grava el so, el processa i indica per pantalla si cal tensar o destensar la corda. Realment és el que buscava.

Estat: No he aconseguit fer que funcioni, no he aprés a executar fitxers de C amb l'Eclipse.

record_audio [32]:

És una aplicació que permet gravar àudio des del micròfon de l'Android a la SD Card i després reproduir-lo. Només permet gravar un cop, per tornar a gravar cal tornar a executar l'aplicació. Consta de 3 botons, un per gravar, un per parar de gravar i un per reproduir el so.

Estat: Funciona correctament.

Recorder [33]:

És una aplicació que permet gravar àudio des de micròfon de l'Android a la SD Card i després reproduir-lo. Permet gravar múltiples vegades.

Estat: S'executa però dóna error al començar.

4.3.2 Proves d'aplicacions amb l'Eclipse

De les anteriors aplicacions, n'hi ha tres de les quals vaig aprendre moltes coses útils i vaig dedicar molt de temps a entendre-les i a intentar que funcionessin. Aquestes tres aplicacions són:

- Audio
- record_audio
- GuitarTuner

4.3.2.1 Audio

L'aplicació Audio ve completament detallada al llibre Hello Android, d'Ed Brunette, però com que a mi em va servir de molt, ja que va ser de les primeres aplicacions que vaig provar i que funcionaven, quan encara no sabia absolutament res d'Android, també l'explicaré de manera detallada, a la meva manera.
Es tracta d'una aplicació que crea dos objectes de la classe MediaPlayer i a cadascun li assigna un arxiu d'àudio mp3 que es troba dins la carpeta Raw del projecte. Després, l'usuari pot triar quin objecte reproduir, prement les fletxes cap amunt o cap avall del teclat de l'emulador. Per tant, es tracta d'una espècie de reproductor multimèdia, tot i que per pantalla només es veu el text "Hello World, Audio!" i per aquest motiu, no he posat cap captura de pantalla de la aplicació.

El codi, al ser curt, l'he posat a continuació, on es poden apreciar la creació de la Activity i dels dos objectes MediaPlayer, el **song1** i el **song2**, així com la assignació de les cançons de dins la carpeta Raw a aquests dos objectes. També es pot veure com se li assigna la funció de reproduir aquests dos objectes al prémer les fletxes al teclat del dispositiu Android.

```
package com.example.audio;

import android.app.Activity;
import android.media.MediaPlayer;
import android.os.Bundle;
import android.view.KeyEvent;

public class Audio extends Activity {
   private MediaPlayer song1, song2;

   @Override
   public void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.main);

      // Native rate is 44.1kHz 16 bit stereo, but
      // to save space we just use MPEG-3 22kHz mono
      song1 = MediaPlayer.create(this, R.raw.song1);
      song2 = MediaPlayer.create(this, R.raw.song2);

   }
```

```
@Override
  public boolean onKeyDown(int keyCode, KeyEvent event) {
     MediaPlayer mp;
     switch (keyCode) {
     case KeyEvent.KEYCODE_DPAD_UP:
        mp = song1;
        break;
     case KeyEvent.KEYCODE_DPAD_DOWN:
        mp = song2;
        break;

     default:
        return super.onKeyDown(keyCode, event);
     }
     mp.seekTo(0);
     mp.start();
     return true;
  }
```

4.3.2.2 record_audio

L'aplicació record_audio és una versió que vaig adaptar jo mateix d'una aplicació trobada per Internet [32] ja que aquella aplicació em donava errors i per tant, vaig haver de veure què era el que fallava i reescriure-ho. L'aplicació, com ja he explicat anteriorment, consta de 3 botons, un per començar a gravar, un per parar de gravar i un per escoltar el que s'ha gravat. El que grava ho escriu a la SD Card mitjançant la classe MediaRecorder, i per tant, al reproduir el que s'ha gravat ho llegeix d'allà també. Només permet gravar un cop, doncs no permet reescriure al damunt de l'arxiu guardat. Al reproduir el so, sembla que dupliqui les mostres i per tant l'arxiu reproduït dura el doble que el que s'ha enregistrat. A més a més, degut a aquest fet, se sent distorsionat, per exemple si es grava una persona parlant, al reproduir el que ha dit, sembla que aquesta persona estigui sota els efectes de l'alcohol. Tanmateix, a mi em va servir per fer proves amb la classe MediaRecorder.

Una altra cosa que cal tenir en compte perquè l'aplicació funcioni correctament és que l'AVD que fem servir consti d'una SD Card. Si no, donaria errors ja que l'aplicació no sabria on gravar les dades.

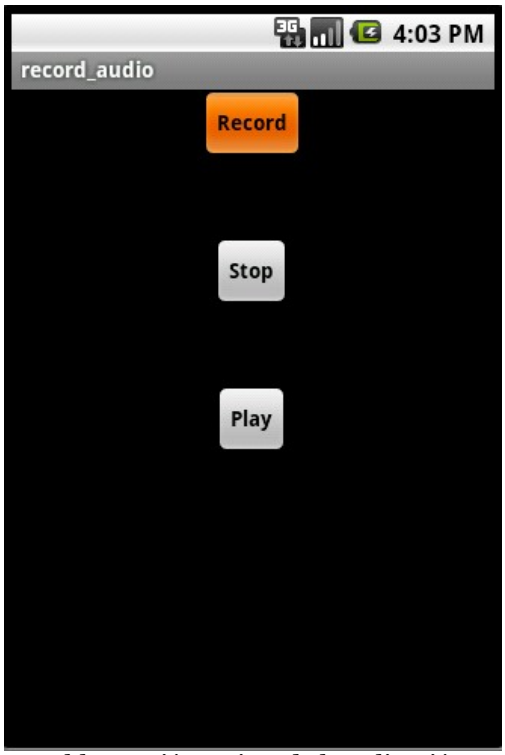
A continuació he posat el codi, que tot i no ser tant curt com l'anterior, ajuda a entendre

conceptes nous que no s'han pogut veure en l'aplicació d'exemple anterior, com per exemple la creació dels tres botons i de l'objecte de la classe MediaRecorder per enregistrar àudio a la SD Card. També es pot veure com es posen a punt tots els paràmetres que l'objecte MediaRecorder necessita, com són la font d'àudio, en aquest cas el micròfon; el format d'àudio, la codificació i finalment la ruta per guardar l'àudio a la SD Card. A més a més, es pot veure la posada a punt de l'objecte MediaRecorder, és a dir, els mètodes start(), stop() i release() i el mètode per reproduir l'objecte MediaRecorder, setDataSource(), prepare() i start(). Finalment, es pot veure com se'ls assignen les funcions a cada botó.

```
package com.example.record_audio;

import java.io.IOException;

import android.app.Activity;
import android.content.ContentValues;
import android.media.MediaPlayer;
import android.media.MediaRecorder;
import android.os.Bundle;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;


public class record_audio extends Activity implements OnClickListener {

     Button record;
     Button stop;
     Button play;
     MediaRecorder recorder = new MediaRecorder();
     ContentValues values = new ContentValues(3);


    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);


        record = (Button)this.findViewById(R.id.record);
        record.setOnClickListener(this);

        stop = (Button)this.findViewById(R.id.stop);
        stop.setOnClickListener(this);

        play = (Button)this.findViewById(R.id.play);
        play.setOnClickListener(this);
    }
```

```
public void record(){
    recorder.setAudioSource(MediaRecorder.AudioSource.MIC);
    recorder.setOutputFormat(MediaRecorder.OutputFormat.THREE_GPP);
    recorder.setAudioEncoder(MediaRecorder.AudioEncoder.AMR_NB);
    recorder.setOutputFile("/sdcard/test.3gpp");
    try {
            recorder.prepare();
      } catch (IllegalStateException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
      } catch (IOException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
      }
    recorder.start();
    }

    public void stop(){
     recorder.stop();
     recorder.release();
    }

        public void play() throws IllegalArgumentException, IllegalStateException,
IOException{
     MediaPlayer mp = new MediaPlayer();
        mp.setDataSource("/sdcard/test.3gpp");
        mp.prepare();
        mp.start();
    }
     @Override
     public void onClick(View v) {
          // TODO Auto-generated method stub
          if (v==record) {
               try {
                  record();
              } catch (IllegalArgumentException e) {
                  // TODO Auto-generated catch block
                  e.printStackTrace();
              } catch (IllegalStateException e) {
                  // TODO Auto-generated catch block
                  e.printStackTrace();
              }
          }
          if (v==stop){
               try {
                  stop();
              } catch (IllegalArgumentException e) {
                  // TODO Auto-generated catch block
                  e.printStackTrace();
              } catch (IllegalStateException e) {
                  // TODO Auto-generated catch block
                  e.printStackTrace();
              }
          }
```

```
        if (v==play){
            try {
                play();
            } catch (IllegalArgumentException e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
            } catch (IllegalStateException e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
            } catch (IOException e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
            }
        }
        }
}
```



Il·lustració 8: Vista de la aplicació record_audio

4.3.2.3 GuitarTuner

Es tracta d'un afinador per a guitarra ja implementat. El meu objectiu era fer alguna cosa semblant a aquest, però amb el meu propi codi, ja que el codi d'aquesta aplicació és bastant complex i és difícil entendre'l del tot, sobretot quan un mai ha fet Java i comença de 0. Tot i així, d'aquesta aplicació vaig aprendre moltes coses i em va ajudar a aclarir les idees.
Aquesta aplicació grava àudio a temps real mitjançant la classe AudioRecord. A temps real vol dir que ho fa tota l'estona, i va omplint uns buffers de dades als que després els fa l'anàlisi freqüencial i en determina la freqüència fonamental de cadascun i treu per pantalla el resultat. Per tant, retorna en tot moment, a temps real, la freqüència del so que està enregistrant.
Una peculiaritat que té el codi, es que fa servir una funció FFT externa, escrita en C, mitjançant l'NDK. Em vaig posar en contacte amb el creador de la aplicació, l'Aleksey Surkov [44], per preguntar-li perquè ho havia fet així, i em va explicar que d'aquesta manera la FFT és més ràpida, consumeix menys recursos i per tant, permet fer l'anàlisi freqüencial en temps real. De fet, no només em vaig posar en contacte amb ell per preguntar-li això, sinó que li vaig demanar ajuda en tot allò que necessitava per entendre el codi millor, i ell, molt amablement, me la va donar.
Degut a algun problema a l'hora de fer servir la funció externa, no vaig poder comprovar si l'aplicació funcionava perquè em donava error, però al final vaig arribar a entendre el codi molt millor i em va servir d'aprenentatge.

4.3.3 Proves amb el Matlab per detectar la freqüència fonamental

Per fer un afinador de guitarra, una de les funcions que havia de fer era una que detectés la freqüència fonamental del so que produeixen les cordes de la guitarra. Per això, abans de llençar-me a fer proves directament amb l'Eclipse, vaig creure oportú fer proves amb el Matlab. Enlloc de fer una funció que a més a més, enregistrés àudio, vaig decidir gravar 18 arxius WAV [34] amb els sons de les 6 cordes de la guitarra, 6 arxius amb les cordes ben afinades, 6 arxius amb les cordes afinades un semitò per sobre i 6 arxius amb les cordes

afinades un semitò per sota, tots de 3 segons de duració. Els vaig gravar fent servir el programa Guitar Pro [35], en format MIDI [36] i posteriorment vaig fer servir un conversor que vaig trobar per Internet anomenat Free Mp3 Wma Converter [37], que permet reconvertir els fitxers d'àudio i triar una sèrie de paràmetres. D'aquesta manera, vaig triar que la freqüència de mostreig fos 8000 Khz, que el canal fos Mono i que els bits per mostra fossin 16. Aquests paràmetres són els mateixos amb els que està configurada la classe AudioRecord d'Android.
Un cop tenia els 18 arxius, vaig provar els següents algorismes:

- FFT
- Autocorrelació
- Anàlisi cepstral

Com que l'algorisme de la autocorrelació no em va acabar de funcionar mai degut a alguns problemes amb la funció del Matlab, no he pogut obtenir un resultat bo, i per tant, no he inclòs la explicació a l'informe.

4.3.3.1 FFT

Aquest script conté un algorisme per detectar la freqüència fonamental d'un arxiu d'àudio. Com que no funciona correctament, explicaré el codi per sobre i no entraré en detalls.
El codi llegeix un arxiu WAV i l'escriu al vector **x**. Guarda el màxim del vector a la variable **Maxtotal**. Després divideix el vector en trames de 250 ms i guarda el màxim de la **trama** en la variable **Maxtrama**. Si Maxtrama és igual a Maxtotal, aquella trama passa a dir-se **tramabona**. Després passa la freqüència de mostreig **fm** i tramabona a la funció **f_fonamental_proba**, on es fa la FFT de la trama i es busca el màxim, que es desa a **f0**. Aquesta f0 és la freqüència fonamental de l'arxiu d'àudio. Aquest codi es basa amb un algorisme que serveix per detectar quins números s'han pitjat en un telèfon a partir del so que emeten els botons al ser pitjats.

proba.m

```
[x,fm]=wavread('E6.wav');

Maxtotal = max(abs(x));
longtrama=250;   % (ms)
L = fix(fm*longtrama/1000); % mostres que dura una trama
N = length(x);   % nombre mostres
Nt =  fix(N/L);   % nombre trames
for t = 1:Nt
  m1 = (t-1)*L + 1;
  m2 = (m1 + L)-1;
  if (m2<N)
    trama = x(m1:m2);

    Maxtrama = max(abs(trama))
    if (Maxtrama == Maxtotal)
       tramabona = trama;
    end
  else
    trama = [ x(m1:N) zeros(m2-N,1)];
  end
end
 f0=f_fonamental_proba(tramabona,fm) % Obtenim la freqüència de la trama
```

f_fonamental_proba.m

```
function [f0]=f_fonamental_proba(trama,fm) % Funció per trobar la fo
w = fft(trama);
W = 20*log10(abs(w));
%f01,f02,f03,f04,f05,f06,f07

if (max(W) >= 0.1)
nfft = length(W);
kref = 1000*nfft/fm;
[Y,k1] = max(W(1:kref));
[Y,k2] = max(W(kref:(kref*2)));
[Y,k3] = max(W((kref*2):(kref*3)));
[Y,k4] = max(W((kref*3):(kref*4)));

v = [k1,k2,k3,k4];
kmax = max(v);
f0 = (kmax*fm)/nfft;

else
f0 = 0;
end
```

4.3.3.2 Anàlisi cepstral

Aquest script retorna la freqüència fonamental d'un arxiu d'àudio. Està pensat perquè aquest arxiu d'àudio contingui el so d'una corda de guitarra, ja que els paràmetres que s'han pres estan pensats perquè així sigui. Funciona perfectament, per tant és el codi que després vaig intentar passar a Java per a Android.

El codi llegeix un arxiu **WAV** i guarda les mostres d'àudio en el vector **x** i la freqüència de mostreig a la variable **fs**. En les probes que he fet, la freqüència de mostreig és de 8000 Hz, la mateixa amb la que la funció d'Android encarregada de capturar àudio captura mostres. Les cordes de la guitarra a l'aire estan dins el rang de 80 – 350 Hz, per tant una freqüència de mostreig de 8000 Hz és suficient, perquè si la freqüència més alta és de 350 Hz, és a dir, 350 períodes per segon, tenim 8000/350 = 22.8 mostres per representar cada període, la qual cosa és suficient.

Després es defineixen dos límits determinats per la freqüència de mostreig i el rang de freqüències que abarca les 6 cordes de la guitarra a l'aire. Aquests límits són **ms1** i **ms20**.

Tot seguit es defineix una trama de 500 mil·lisegons, que servirà per analitzar el vector d'àudio per trames i trobar la freqüència fonamental de cada trama. Aquests 500 mil·lisegons són suficients per determinar la freqüència fonamental, ja que per exemple, per a la freqüència més petita, 80 Hz (T = 1/80 s), en 500 mil·lisegons hi caben 0.5 s / T = 40 períodes.

Seguidament es defineixen les variables **L** (que indica quantes mostres conté una trama de 500 mil·lisegons), **N** (que indica la longitud total del vector d'àudio) i **Nt** (que indica el nombre de trames total de 500 mil·lisegons que hi ha en el vector d'àudio). Després hi ha un for que es recorre totes les trames del vector **x** (i la última, si no coincideix amb el final del vector, s'omple amb zeros) i si el màxim absolut de la trama sobrepassa una amplitud de 0.1, és a dir, no hi ha silenci, fa la FFT de la trama i guarda el resultat en el vector **Y**. Tot seguit fa l'anàlisi cepstral del vector resultant i guarda a la variable **fx** el punt amb més amplitud de l'anàlisi cepstral que hi ha entre ms1 i ms20. Per obtenir la freqüència fonamental (**fz**) a partir de fx, es fa **fz = fs / (ms1 + fx − 1)**. Després es guarda cada freqüència fonamental obtinguda dins el vector **Fo(t)**. El màxim d'aquest vector serà la freqüència fonamental (**freq_fon**) del senyal d'àudio.

```
[x,fs]=wavread('E6.wav')

ms1=fs/400;                 % La freqüència de la primera corda a l'aire, la de més alta
freqüència, està                    % sobre els 350 Hz. Per tant, 400 Hz és un bon límit.

ms20=fs/80;                 % La freqüència de la sisena corda a l'aire, la de més baixa
freqüència, està                    % sobre els 80 Hz. Per tant, 80 Hz és un bon límit.

longtrama = 500;    % Es processarà el senyal per trames de 500 ms.
L = fix (fs*longtrama/1000);    % Mostres que conté una trama de 500 ms.
N = length (x);         % Longitud total del senyal.
Nt = fix(N/L);          % Nombre  de trames.

for t =1:Nt
   m1 = (t-1)*L + 1;
   m2 = (m1+L) -1;
   if (m2<N)
      trama = x(m1:m2);
   else
      trama = [x(m1:N) zeros(m2-N,1)];
   end

% FFT de la trama
   if (max(abs(trama)) > 0.1)
      Y=fft(trama);

% El cepstrum és la DFT de l'espectre.

   C=fft(log(abs(Y)));

   [c,fx]=max(abs(C(ms1:ms20)));

   fz = fs/(ms1+fx-1);
   else
      fz = 0.0;
   end

   Fo(t) = fz;
end
plot (Fo)
freq_fon = max ((Fo))
```

4.3.4 Proves amb l'Eclipse per detectar la freqüència fonamental

Mentre escrivia el codi definitiu per a Android, basat en el codi de la anàlisi cepstral per a Matlab, vaig fer proves amb Java directament, fent servir el mateix Eclipse. D'aquesta manera, vaig poder debugar el codi i vaig poder veure per pantalla el valor de les dades en tot moment, per saber què feia exactament el codi. Vaig comprovar que tenia força errors, com per exemple que no feia bé la conversió de double a short, o que no agafava la trama que m'interessava. Gràcies a poder veure el que feia el codi en tot moment, vaig poder retocar unes quantes coses fins que finalment, em va funcionar la part de comparar la freqüència obtinguda amb la que jo li hagués dit. Per fer proves per saber si em funcionava, vaig omplir un vector de 32000 mostres, cada quart del vector amb un valor diferent, i vaig anar mostrant resultats per pantalla. No funcionava del tot bé, ja que no feia proves amb un senyal real, però vaig pensar que funcionava prou bé. No he posat cap captura de pantalla ni res del codi ja que el codi final és exactament el mateix però passat a Android.

De totes maneres, crec que aquest pas ha sigut molt important a l'hora de comprovar si el codi feia el que jo volia que fes.

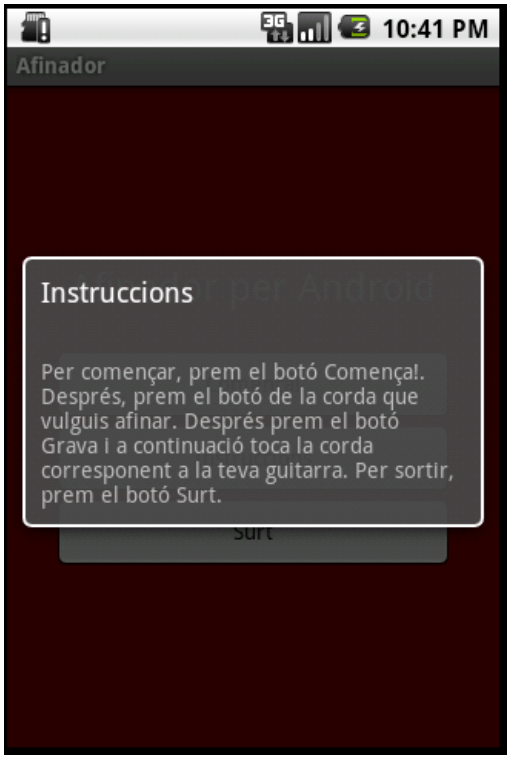
4.4 Aplicació complerta

Un cop acabades les proves, presento l'aplicació complerta i explico les seves característiques, alhora que mostro unes quantes captures de pantalla.

4.4.1 Interfície gràfica

L'aplicació consta d'un menú principal amb tres botons, tal com tenia pensat en un principi: un per sortir de l'aplicació (Surt) , un per començar-la (Comença!) i un altre que fa aparèixer un text amb les instruccions de com funciona la aplicació (Instruccions). El fons es de color vermell, els botons són els que crea l'Eclipse per defecte i el text amb les instruccions deixa veure el que hi ha a sota.

Il·lustració 9: Vista del menú principal

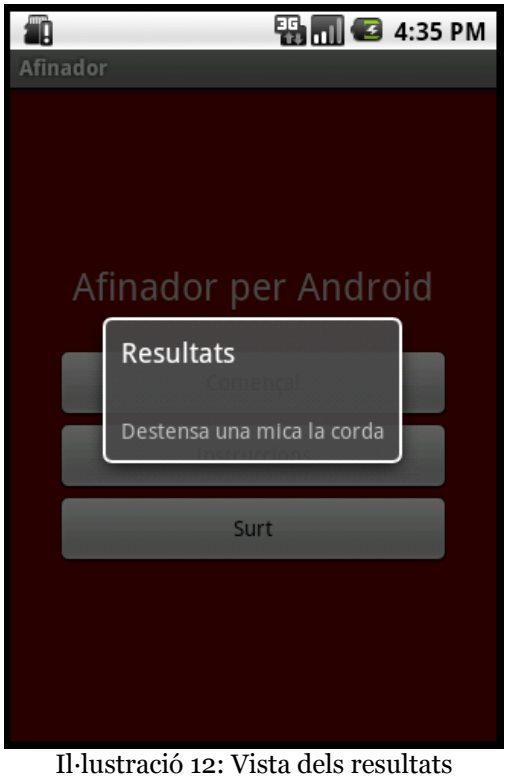


Il·lustració 10: Vista de les instruccions

Dins el menú Comença! hi ha vuit botons, sis d'ells corresponen a les cordes de la guitarra, mostrant la posició de la corda, la nota que dóna la corda i la seva freqüència fonamental. Un altre botó serveix per començar a gravar (Grava) i un altre per tornar enrere al menú

principal (Torna enrere). Quan l'usuari ha seleccionat una corda i ha donat al botó Grava, al cap d'uns segons, mentre es captura el so i es processen les dades, apareix un text dient si la corda està ben afinada o no, també deixant veure el que hi ha a sota.

Il·lustració 11: Vista del menú "Comença!"



Il·lustració 12: Vista dels resultats

4.4.2 Captura d'àudio

La part de captura d'àudio, ha resultat ser de una tasca més complicada del que m'esperava, ja que la documentació que hi ha costa d'entendre per una persona que no està acostumada a escriure en Java, i els codis d'exemple que hi ha no donen una configuració exacta per a cada cas. Gràcies a les proves que vaig fer d'altres aplicacions, finalment vaig aconseguir, mitjançant la classe AudioRecord, guardar les dades d'àudio que entren pel micròfon en un vector de tipus Short.

El constructor d'aquesta classe requereix una sèrie de paràmetres, descrits a continuació:

- int audioSource: defineix la font d'àudio. En aquest cas el micròfon. S'inicialitza com a AudioSource. MIC, que ve definit a la classe MediaRecorder.AudioSource.

- int sampleRateInHz: defineix la freqüència de mostratge en hertzs. En aquest cas, 8000 Hz.
- int channelConfig: defineix la configuració dels canals d'àudio. En aquest cas mono. S'inicialitza com a AudioFormat.CHANNEL_CONFIGURATION_MONO, que ve definit a la classe AudioFormat.
- int audioFormat: defineix el format de les dades d'àudio. En aquest cas, PCM de 16 bits, 2 Bytes per mostra. S'inicialitza com a AudioFormat.ENCODING_PCM_16BIT, que ve definit a la classe AudioFormat.
- int bufferSizeInBytes: defineix la mida total en Bytes del buffer on es guarden les dades d'àudio. Com que vull que es gravin 4 segons d'àudio, tenint en compte que ho demana en Bytes i la freqüència de mostratge, 4000 ms x 8000 Hz x 2 Bytes per mostra, el buffer ha de ser de 64000 mostres.

A l'objecte de la classe AudioRecord l'he anomenat **recorder**.

També cal crear el vector de Shorts on es guardaran les dades d'àudio. No cal que sigui un vector de tipus Shorts, doncs els mètodes de la classe també permeten guardar les dades en vectors de tipus Byte o del tipus ByteBuffer, però he escollit la de tipus Short perquè com que les dades ocupen 2 Bytes, i 1 Short = 2 Bytes, així tindré una mostra a cada Short. La mida del vector, doncs, ha de ser la meitat de 64000 mostres, 32000. He anomenat al vector **audio_data**.

Un cop construït un objecte de la classe amb la inicialització anterior, i el vector de tipus Short on aniran les dades d'àudio, només cal cridar les funcions corresponents de la classe per omplir el vector. Aquestes funcions són:

- recorder.startRecording(): serveix per començar a gravar.
- recorder.read(): serveix per escriure les dades d'àudio en el vector seleccionat. Els paràmetres d'entrada d'aquesta funció són: el vector tipus Short on s'escriuen les dades, en aquest cas el vector audio_data; un int offsetInShorts, que serveix per definir un desfasament respecte l'inici del vector de tipus Short, és a dir, que comenci a escriure a partir del Short número offsetInShorts, en aquest cas 0; la mida del vector on s'escriu, int sizeInShorts, en aquest cas la mateixa mida que el vector audio_data, però no té perquè coincidir, ja que es pot segmentar el vector

audio_data en dues parts, una del principi a la meitat i l'altra de la meitat al final, i fer dues crides a aquesta funció. De ser així, el vector audio_data continuaria tenint 32000 mostres, però aquest int sizeInShorts hauria de valer la meitat, 16000.

- recorder.stop(): serveix per parar de gravar.
- recorder.release(): serveix per alliberar els recursos que consumeix la classe AudioRecord.

4.4.3 Afinació

La part de la afinació ha sigut sense cap mena de dubte la part que més mals de cap ha portat. Un cop obtingut el vector de dades d'àudio, per saber si una corda està correctament afinada primer calia trobar la freqüència fonamental de la mostra d'àudio capturada amb el micròfon i posteriorment comparar-la amb el valor teòric de freqüència fonamental de cada corda. Per aconseguir el que volia, com he explicat anteriorment, vaig fer un munt de proves, amb Matlab, Java i provant altres aplicacions per a Android a l'Eclipse. La base del codi està explicada a la secció de proves amb Matlab i proves amb l'Eclipse, per tant ara explicaré molt per sobre algunes de les parts de les funcions que he fet per a Android. El codi sencer es pot trobar a l'Annex.

La funció que determina si una corda està afinada s'anomena **afinador4**, ja que era la quarta que intentava. És una funció de tipus int que es crida just després de passar les dades d'àudio al vector audio_data. Els dos paràmetres d'entrada són precisament el vector audio_data i frequencia_corda, la freqüència fonamental teòrica de la corda seleccionada. Fa l'anàlisi cepstral per trames del vector audio_data i en determina la freqüència fonamental, i després la compara amb la freqüència fonamental teòrica de la corda. No la compara directament amb aquesta freqüència, sinó que comprova si està dins l'interval que va del -2.9% d'aquesta freqüència fins al +2.9% o si està per sobre o per sota d'aquest interval.

També em va fer falta fer una funció que determinés el valor màxim d'una trama, així que vaig crear la funció **max** que es crida a la funció afinador4. Es tracta d'una funció a la que

se li passen un vector de Doubles, un int per indicar l'inici de la trama i un int per indicar el final de la trama. Es fa servir tres cops, un cop per saber si la trama no està buida, és a dir, que el màxim de la trama és més gran que 0.1; una altra vegada per determinar el màxim després de l'anàlisi cepstral d'una trama; i un últim cop per determinar la freqüència més rellevant dins el vector de freqüències després de fer l'anàlisi cepstral de totes les trames.

4.5 Valoració dels resultats

Els resultats del projecte no han estat els desitjats, ja que l'aplicació no funciona correctament. He intentat fer-la de moltes maneres diferents i he provat un munt de coses, fins que vaig topar amb una manera que vaig creure que funcionaria i fins l'últim dia he intentat que funcionés, però m'he continuat trobant amb problemes que m'han costat de solucionar, i tot i que finalment crec que he trobat quin és l'error, he intentat de solucionar-lo però no hi ha hagut manera. Per tant, l'aplicació del projecte que presento és una aplicació inacabada, que tot i que funciona, funciona malament. No fa bé la part de la transformada de Fourier. He comprovat que l'error ve d'aquí, fent contínues proves amb el codi en Java, comprovant pas a pas què fallava i l'error ve a l'hora de fer la FFT, tant al primer cop com per fer l'anàlisi cepstral.

5. Conclusions i propostes de millora

A continuació exposaré les conclusions que he tret al acabar el projecte així com una sèrie de millores que crec que farien que l'aplicació fos òptima.

5.1 Conclusions

Val a dir que no estic satisfet amb el projecte. El fet de no haver pogut fer el que em vaig proposar en un principi, és a dir, una aplicació que detectés la freqüència dominant del so d'una corda de guitarra i determinés si estava ben afinada, és el que fa que em quedi un mal sabor de boca.

De totes maneres, he complert uns objectius dels quals si que n'estic satisfet. He estudiat a fons la plataforma Android i ara em veig capaç de crear altres aplicacions i a entendre perfectament aplicacions fetes per altres persones. He provat un munt d'aplicacions i crec que això m'ha donat una certa experiència.

El fet d'haver fet moltes aturades a l'hora de treballar, ja sigui per temporada d'exàmens i d'altres treballs, per vacances o per haver estat malalt setmanes seguides, ha fet que cada cop que em volia reenganxar a la feina hagués de tornar a entendre-ho tot. Això és dolent per una banda, perquè vol dir que no he treballat tot el que hauria pogut, però també opino que és bo, ja que així cada cop estudiava a fons una cosa que ja havia fet anteriorment, i això ajuda a reforçar els coneixements.

Com a conclusió final, em queda la decepció de no haver pogut completar l'aplicació, però em queda l'alegria d'haver treballat a fons, d'haver estudiat la plataforma Android i el llenguatge Java, i d'haver acabat de redactar el projecte a temps.

5.2 Propostes de millora a l'aplicació

Encara que no hagi acabat l'aplicació, si l'hagués acabat li faltaria molt per ser òptima, per la qual cosa podria millorar-se en molts aspectes. Proposo tres aspectes de millora per si algú, en el futur, vol fer una aplicació semblant o vol millorar aquesta. Per dur-les a terme caldria un estudi més exhaustiu sobre la arquitectura Android.

5.2.1 Creació d'una interfície gràfica

Una de les moltes possibles millores de la aplicació és la creació d'una interfície gràfica. La aplicació en té una, però és la més bàsica possible: són botons predefinits amb un fons vermell. A més a més, els resultats surten per pantalla en forma de missatge.
Una bona interfície gràfica s'aconseguiria amb uns botons més atractius, com per exemple un dibuix d'una guitarra amb cordes, on cada corda fos un botó que es pogués tocar, i quan estiguin tocats, s'il·luminin per fer saber a l'usuari que aquella corda està seleccionada. També estaria bé que a l'hora de mostrar els resultats, es fes amb un dibuix, com a la majoria d'afinadors de guitarra.
També podrien haver-hi més millores respecte a la interfície gràfica, però crec que aquestes 3 ja farien que la aplicació fos molt més atractiva.

Il·lustració 13: Exemple d'interfície gràfica millorada

5.2.2 Afinar a temps real

Si en lloc de tenir un botó per començar a gravar so, l'aplicació anés gravant el so i mostrant per pantalla la afinació a temps real, la aplicació seria sens dubte gairebé òptima. Per fer això, caldria dominar una mica més d'Android. De totes maneres, de ser així, el terminal mòbil consumiria més recursos, però la aplicació seria un autèntic afinador de guitarra.

Il·lustració 14: Exemple d'afinació a temps real

5.2.3 Utilitzar l'NDK

Una altra millora per a la aplicació, seria utilitzar una funció externa que no fos escrita en Java, per exemple en C, i cridar-la mitjançant l'NDK. Això seria necessari en cas de que la aplicació afinés en temps real, ja que les funcions en C són molt més ràpides al executar-se que les funcions en Java.

6. Bibliografia

[1] Google. *Android Developers: MediaRecorder Class* [en línia]. URL: http://developer.android.com/reference/android/media/MediaRecorder.html (Darrera consulta: 8 d'agost de 2010)

[2] Burnette, Ed. *Hello, Android: Introducing Google's Mobile Development Platform*. 2ª ed. EE.UU.: Pragmatic Bookshelf, 2009. 250 p. ISBN-10: 1934356492

[3] Sony Computer Entertainment Europe Limited . Sing Star [en línia]. Londres, 2007. URL: http://www.singstargame.com/es-es/Espana-What-is-SingStar/ (Darrera consulta: 8 d'agost de 2010)

[4] Sony Computer Entretainment Europe Limited. *Play Station* [en línia]. Londres, 2010. URL: http://es.playstation.com/ (Darrera consulta 5 d'agost de 2010)

[5] Savatier, Tristan. *MPEG* [en línia]. Actualització 2010. URL: http://www.mpeg.org/ (Darrera consulta 8 d'agost de 2010)

[6] Microsoft corporation. *Windows Media Audio Codecs* [en línia]. Actualització 2010. URL: http://www.microsoft.com/windows/windowsmedia/forpros/codecs/audio.aspx (Darrera consulta 6 d'agost de 2010)

[7] Google. *Android Developers* [en línia]. URL: http://developer.android.com/index.html (Darrera consulta: 12 de març de 2010)

[8] Google. *Android* [en línia]. California, 2007. URL: http://www.android.com/ (Darrera consulta 12 de març de 2010)

[9] Überbin. *Celularis: weblog sobre tecnología móvil* [en línia]. Actualització 2010. URL: http://www.celularis.com/software/historia-android.php (Darrera consulta 5 d'agost de 2010)

[10] Vílchez Moreno, Javier. *Configurar Equipos* [en línia]. Còrdova, gener 2002. URL: http://www.configurarequipos.com/doc1107.html (Darrera consulta 6 d'agost de 2010)

[11] Google. *Android Developers: Android SDK* [en línia]. URL: http://developer.android.com/sdk/index.html (Darrera consulta: 12 de març de 2010)

[12] The Eclipse Foundation. *Eclipse Galileo* [en línia]. Ontario, gener 2004, actualització 2010. URL: http://www.eclipse.org/galileo/ (Darrera consulta: 7 d'agost de 2010)

[13] Google. *Android Developers: ADT Plugin for Eclipse* [en línia]. URL: http://developer.android.com/sdk/eclipse-adt.html (Darrera consulta: 12 de març de 2010)

[14] Google. *Android Developers: Download the Android NDK* [en línia]. URL: http://developer.android.com/sdk/ndk/index.html (Darrera consulta: 14 de març de 2010)

[15] Wikipedia. *Wikipedia-Teorema de mostreig de Nyquist-Shannon* [en línia]. Actualització 2 de març de 2010. URL: http://ca.wikipedia.org/wiki/Teorema_de_mostreig_de_Nyquist-Shannon (Darrera consulta: 6 d'agost de 2010)

[16] Wikipedia. *Wikipedia-Freqüència* [en línia]. Actualització 29 d'agost de 2010. URL: http://ca.wikipedia.org/wiki/Freqüència (Darrera consulta: 29 d'agost de 2010)

[17] Wikipedia. *Wikipedia-Quantificació (processament de senyal)* [en línia]. Actualització 14 d'agost de 2010. URL: http://ca.wikipedia.org/wiki/Quantificació_(processament_de_senyal) (Darrera consulta: 29 d'agost de 2010)

[18] Wikipedia. *Wikipedia-Còdec d'àudio* [en línia]. Actualització 8 de maig de 2010. URL: http://ca.wikipedia.org/wiki/Codificació_d'àudio (Darrera consulta: 10 d'agost de 2010)

[19] Wikipedia. *Wikipedia-Sèrie de Fourier* [en línia]. Actualització 18 d'abril de 2010. URL: http://ca.wikipedia.org/wiki/Sèrie_de_Fourier (Darrera consulta: 10 d'agost de 2010)

[20] Wikipedia. *Wikipedia-Transformada Discreta de Fourier* [en línia]. Actualització 12 d'abril de 2010. URL: http://ca.wikipedia.org/wiki/Transformada_discreta_de_Fourier (Darrera consulta: 10 d'agost de 2010)

[21] Wikipedia. *Wikipedia-Transformada Rápida de Fourier* [en línia]. Actualització 5 de juliol de 2010. URL: http://es.wikipedia.org/wiki/Transformada_rápida_de_Fourier (Darrera consulta: 10 d'agost de 2010)

[22] Wikipedia. *Wikipedia-Tranformada Ràpida de Fourier* [en línia]. Actualització 20 d'abril de 2010. URL: http://ca.wikipedia.org/wiki/FFT (Darrera consulta: 10 d'agost de 2010)

[23] Wikipedia. *Wikipedia-Autocorrelació* [en línia]. Actualització 6 de juny de 2010. URL: http://ca.wikipedia.org/wiki/Autocorrelació (Darrera consulta: 10 d'agost de 2010)

[24] Wikipedia. *Wikipedia-Cepstrum* [en línia]. Actualització 31 de desembre de 2010. URL: http://ca.wikipedia.org/wiki/Cepstrum (Darrera consulta: 10 d'agost de 2010)

[25] Miller, Cory. *Blog.creacciona.com* [en línia]. Actualització 1 d'octuble de 2008. URL: http://blog.creacciona.com/afinacion-estandar-de-la-guitarra/ (Darrera consulta: 8 de març de 2010)

[26] Wikipedia. *Wikipedia-Semitono* [en línia]. Actualització 28 d'agost de 2010. URL: http://es.wikipedia.org/wiki/Semitono (Darrera consulta: 29 d'agost de 2010)

[27] Google. *Android Developers: Android media package* [en línia]. URL: http://developer.android.com/reference/android/media/package-summary.html (Darrera consulta: 12 de març de 2010)

[28] Google. *Android Developers: AudioRecord class* [en línia]. URL: http://developer.android.com/reference/android/media/AudioRecord.html (Darrera consulta: 12 de març de 2010)

[29] Google. *Android Developers: JetBoy* [en línia]. URL: http://developer.android.com/resources/samples/JetBoy/index.html (Darrera consulta: 14 de març de 2010)

[30] Why Android. *MusicDroid Audio Player* [en línia]. Actualització 2010. URL: http://whyandroid.com/android/119-musicdroid-audio-player-part-iii.html (Darrera consulta: 18 d'abril de 2010)

[31] Google code. *Android-guitar-tuner – Revision 10:/* [en línia]. URL: http://android-guitar-tuner.googlecode.com/svn/ (Darrera consulta: 18 d'abril de 2010)

[32] phpBB Group. *anddev.org* [en línia]. URL: http://www.anddev.org/viewtopic.php?p=17382 (Darrera consulta: 18 d'abril de 2010)

[33] phpBB Group. *anddev.org* [en línia]. URL: http://www.anddev.org/viewtopic.php?p=22820 (Darrera consulta: 18 d'abril de 2010)

[34] Microsoft Corporation. *WAVE and AVI codec Registries* [en línia]. Juny de 1998. URL: http://tools.ietf.org/html/rfc2361 (Darrera consulta: 29 d'agost de 2010)

[35] Arobas Music. *Guitar Pro 6* [en línia]. França, actualització 2010. URL: http://www.guitar-pro.com/en/index.php (Darrera consulta: 20 d'abril del 2010)

[36] Hoofjaw Media. *Hoofjaw: midi collective* [en línia]. Agost 2006, actualització 2010. URL: http://www.hoofjaw.com/Default.aspx (Darrera consulta: 15 d'abril de 2010)

[37] Media Ingea SL. *Free mp3 wma converter* [en línia]. Desembre 2002, actualització 2010. URL: http://free-mp3-wma-converter.uptodown.com/ (Darrera consulta: 15 d'abril de 2010)

[38] And.roid.es. *Android Inc* [en línia]. Barcelona, 2010. URL: http://and.roid.es/android-inc.html (Darrera consulta: 25 de març de 2010)

[39] Google. *Informació corporativa* [en línia] 2010. URL: http://www.google.es/corporate/ (Darrera consulta: 25 de març de 2010)

[40] Google. *Open Handset Alliance* [en línia]. Actualització 29 de juliol de 2010. URL: http://www.openhandsetalliance.com/ (Darrera consulta: 25 de març de 2010)

[41] The Apache Software Foundation. *The Apache Software Foundation: community-led development since 1999* [en línia]. Actualització 2010. URL: http://www.apache.org/ (Darrera consulta: 25 de març de 2010)
[42] Wikipedia. *Wikipedia-QWERTY* [en línia]. Actualització 29 d'agost de 2010. URL: http://en.wikipedia.org/wiki/QWERTY (Darrera consulta: 29 d'agost de 2010)

[43] Google. *Android Developers: The Android Manifest* [en línia]. URL: http://developer.android.com/guide/topics/manifest/manifest-intro.html (Darrera consulta: 14 de març de 2010)

[44] Androlib. *Aleksey Surkov developer's page* [en línia]. 2009, actualització 2010. URL: http://es.androlib.com/android.developer.aleksey-surkov-DtE.aspx (Darrera consulta: 13 de juny de 2010)

[45] Linux Online Inc. *Linux Online* [en línia]. 1994, actualització 2010. URL: http://www.linux.org/ (Darrera consulta: 14 de març de 2010)

[46] Java2s. *Java Source Code – FFT* [en línia]. 2000, actualització 2009. URL: http://www.java2s.com/Open-Source/Java-Document/6.0-JDK-Modules/java-3d/com/db/media/audio/dsp/monitors/FFT.java.htm (Darrera consulta: 13 de juny de 2010)

7. Annex 1: Codi complet de l'aplicació

7.1 Classes de Java

A continuació hi ha el codi de les 5 classes de Java que fa servir l'aplicació.

7.1.1 Codi.java

```
package com.example.afinador;


import android.app.Activity;
import android.content.Intent;
import android.media.AudioFormat;
import android.media.AudioRecord;
import android.media.MediaRecorder.AudioSource;
import android.os.Bundle;
import android.view.View;
import android.view.View.OnClickListener;
import com.example.afinador.FFT;
import com.example.afinador.Frequencia_fonamental;


public class Codi extends Activity implements OnClickListener {

	private static final int LOG2_FFTSIZE = 12;
	private static final int FFT_SIZE        = 1 << LOG2_FFTSIZE;
	double frequencia_corda;
	public static int afinacio = -1;
	int RATE = 8000;
	int mida_buffer_MS = 4000;
	int mida_buffer_bytes = 2 * RATE * mida_buffer_MS/ 1000 ;//  64000 les mostres
estan en 16BIT, cada mostra son 2 Bytes
	int mida_chunk_MS = 500;
	int mida_chunk_short = RATE * mida_chunk_MS/ 1000;//  4000
	short[] audio_data = new short[mida_buffer_bytes/2]; //dividit per 2 perque es
en Short, i 1 Short  = 2 Bytes

	AudioRecord recorder = new AudioRecord(AudioSource.MIC, RATE,
AudioFormat.CHANNEL_CONFIGURATION_MONO,
		AudioFormat.ENCODING_PCM_16BIT, mida_buffer_bytes);

	 @Override
   protected void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.cordes);
```

```
// Set up click listeners for all the buttons

      View corda1 = findViewById(R.id.corda_1);
      corda1.setOnClickListener(this);
      View corda2 = findViewById(R.id.corda_2);
      corda2.setOnClickListener(this);
      View corda3 = findViewById(R.id.corda_3);
      corda3.setOnClickListener(this);
      View corda4 = findViewById(R.id.corda_4);
      corda4.setOnClickListener(this);
      View corda5 = findViewById(R.id.corda_5);
      corda5.setOnClickListener(this);
      View corda6 = findViewById(R.id.corda_6);
      corda6.setOnClickListener(this);

      View recordButton = findViewById(R.id.record_button);
      recordButton.setOnClickListener(this);

      View backButton = findViewById(R.id.back_button);
      backButton.setOnClickListener(this);

  }

 //Funció que retorna el màxim d'una trama
public double max(double[] x, int ini, int fin){
          double max = 0.0;
          for (int i=ini; i < fin; i++)
          {
                if (Math.abs (x[i]) >= max)
                {
                      max = Math.abs (x[i]);
                }
          }



          return max;
   }
```

```
// Funció que retorna la afinació
   public int afinador4 (short[] x, double fr){


          int ms1 = RATE/400;
          int ms20 = RATE/80;
          int longtrama = 500;
          int L = FFT_SIZE;
          int N = x.length;
          int Nt = N/L;
          int m1,m2;
          double fz = 0, fx = 0, fr_a_mirar = 0;
          double[] trama = new double[L];
          double[] realArray = new double[L];
          double[] imagArray = new double[L];
          double[] C = new double[L];
          double[] Fo = new double[Nt];
          FFT fft = new FFT(LOG2_FFTSIZE);

          for (int t = 0; t < Nt; t++)
          {
                 m1 = (t)*L + 1;
                 m2 = (m1+L) -1;
                 if (m2<N)
                 {
                        for (int i=0, m=m1; i<L; m++, i++)
                        {
                               trama[i]=x[m];
                        }
                 }

                 if (max(trama,0,L) > 0.1)
                 {
                        if (trama.length < FFT_SIZE)
                        {
                               realArray[1] = -1.0;
                        }

                            // Posar les dades de la trama en els vectors.
                            // Les dades no tenen part imaginària.
                            for(int i=0; i < FFT_SIZE; i++)
                            {
                                   realArray[i] = trama[i];
                                   imagArray[i] = 0.0;
                            }
```

```
// Fer la fft.
                        fft.doFFT(realArray, imagArray, false);

                        for(int i=0; i < FFT_SIZE; i++)
                        {
                               C[i] =  Math.abs (Math.log (trama[i]));
                        }

                        fft.doFFT(C, imagArray, false);

                        fx = max (C,ms1,ms20);

                        fz = RATE/(ms1+fx-1);
                }
                else
                {
                       fz = 0.0;
                }
                Fo[t] = fz;
        }


        fr_a_mirar = max(Fo,0,Nt);

        if (fr_a_mirar>=fr-fr*0.3 && fr_a_mirar<=fr+fr*0.3) //Entre to i semitò hi ha
un 6%, per afinar cal un 3%
        {
               afinacio = 0;
        }
        else if (fr_a_mirar<=fr-fr*0.3)
        {
               afinacio = -1;
        }
        else
        {
               afinacio = 1;
        }

        return afinacio;

   }
```

```
//Comprova quin botó s'ha pitjat
 public void onClick(View v) {
          switch (v.getId()) {
          case R.id.back_button:
             finish();
             break;
          case R.id.corda_1:
              frequencia_corda=329.63;
              break;
          case R.id.corda_2:
               frequencia_corda=246.94;
              break;
          case R.id.corda_3:
               frequencia_corda=196.0;
              break;
          case R.id.corda_4:
               frequencia_corda=146.83;
              break;
          case R.id.corda_5:
               frequencia_corda=110.0;
              break;
          case R.id.corda_6:
               frequencia_corda=82.41;
              break;

          case R.id.record_button:
                    if (recorder.getState() != AudioRecord.STATE_INITIALIZED) {
                           ShowError("Can't initialize AudioRecord");
                           return;
                    }
           recorder.startRecording();
              recorder.read(audio_data, 0, mida_buffer_bytes/2); //En Shorts
                    recorder.stop();
                    recorder.release();

                    afinador4 (audio_data,frequencia_corda);


                    Intent i = new Intent(this, EstatCorda.class);
               startActivity(i);
               finish();

               break;

  }
}
       private void ShowError(String string) {
              // TODO Auto-generated method stub

       }
}
```

7.1.2 FFT.java

```
// Fast Fourier Transform (FFT) Code
// Java implementation by: Craig A. Lindley
// Last Update: 02/27/99

package com.example.afinador;


/* libfft.c - fast Fourier transform library
**
** Copyright (C) 1989 by Jef Poskanzer.
**
** Permission to use, copy, modify, and distribute this software and its
** documentation for any purpose and without fee is hereby granted, provided
** that the above copyright notice appear in all copies and that both that
** copyright notice and this permission notice appear in supporting
** documentation.  This software is provided "as is" without express or
** implied warranty.
*/

public class FFT {
	/**
	 * This is a Java implementation of the fast Fourier transform
	 * written by Jef Poskanzer. The copyright appears above.
	 */

	private static final double TWOPI = 2.0 * Math.PI;

	// Limits on the number of bits this algorithm can utilize
	private static final int LOG2_MAXFFTSIZE = 15;
	private static final int MAXFFTSIZE = 1 << LOG2_MAXFFTSIZE;

	/**
	 * FFT class constructor
	 * Initializes code for doing a fast Fourier transform
	 *
	 * @param int bits is a power of two such that 2^b is the number
	 * of samples.
	 */
```

```
public FFT(int bits) {

		this.bits = bits;

		if (bits > LOG2_MAXFFTSIZE) {
			System.out.println("" + bits + " is too big");
			System.exit(1);
		}
		for (int i = (1 << bits) - 1; i >= 0; --i) {
			int k = 0;
			for (int j = 0; j < bits; ++j) {
				k *= 2;
				if ((i & (1 << j)) != 0)
					k++;
			}
			bitreverse[i] = k;
		}
	}

	/**
	 * A fast Fourier transform routine
	 *
	 * @param double [] xr is the real part of the data to be transformed
	 * @param double [] xi is the imaginary part of the data to be transformed
	 * (normally zero unless inverse transofrm is effect).
	 * @param boolean invFlag which is true if inverse transform is being
	 * applied. false for a forward transform.
	 */
	public void doFFT(double [] xr, double [] xi, boolean invFlag) {
		int n, n2, i, k, kn2, l, p;
		double ang, s, c, tr, ti;

		n2 = (n = (1 << bits)) / 2;

		for (l = 0; l < bits; ++l) {
			for (k = 0; k < n; k += n2) {
				for (i = 0; i < n2; ++i, ++k) {
					p = bitreverse[k / n2];
					ang = TWOPI * p / n;
					c = Math.cos(ang);
					s = Math.sin(ang);
					kn2 = k + n2;

					if (invFlag)
						s = -s;

					tr = xr[kn2] * c + xi[kn2] * s;
					ti = xi[kn2] * c - xr[kn2] * s;

					xr[kn2] = xr[k] - tr;
					xi[kn2] = xi[k] - ti;
					xr[k] += tr;
					xi[k] += ti;
				}
			}
```

```
n2 /= 2;
            }

            for (k = 0; k < n; k++) {
                  if ((i = bitreverse[k]) <= k)
                        continue;

                  tr = xr[k];
                  ti = xi[k];
                  xr[k] = xr[i];
                  xi[k] = xi[i];
                  xr[i] = tr;
                  xi[i] = ti;
            }

            // Finally, multiply each value by 1/n, if this is the forward
            // transform.
            if (!invFlag) {
                  double f = 1.0 / n;

                  for (i = 0; i < n ; i++) {
                        xr[i] *= f;
                        xi[i] *= f;
                  }
            }
      }
      // Private class data
      private int bits;
      private int [] bitreverse = new int[MAXFFTSIZE];
}
```

7.1.3 About.java

```
package com.example.afinador;

import android.app.Activity;
import android.os.Bundle;

public class About extends Activity {
   @Override
   protected void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.about);
   }
}
```

7.1.4 Afinador.java

```
package com.example.afinador;

import com.example.afinador.R;
import com.example.afinador.About;

import android.app.Activity;
import android.os.Bundle;
import android.content.Intent;
import android.view.View;
import android.view.View.OnClickListener;


public class Afinador extends Activity implements OnClickListener {
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);


     // Set up click listeners for all the buttons
        View startButton = findViewById(R.id.start_button);
        startButton.setOnClickListener(this);
        View aboutButton = findViewById(R.id.about_button);
        aboutButton.setOnClickListener(this);
        View exitButton = findViewById(R.id.exit_button);
        exitButton.setOnClickListener(this);
    }

        public void onClick(View v) {
            switch (v.getId()) {
            case R.id.about_button:
               Intent i = new Intent(this, About.class);
               startActivity(i);
               break;
            case R.id.start_button:
                Intent i2 = new Intent(this, Codi.class);
                startActivity(i2);
                break;
            case R.id.exit_button:
               finish();
               break;



    }
}
}
```

7.1.5 EstatCorda.java

```
package com.example.afinador;


import android.app.Activity;
import android.os.Bundle;
import com.example.afinador.Codi;


public class EstatCorda extends Activity {
   @Override
   protected void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      if (Codi.afinacio == 0) //corda ben afinada
            {
         setContentView(R.layout.resultat_bo);

            }
            if (Codi.afinacio == -1) //corda destensada
            {
                  setContentView(R.layout.resultat_destensada);
            }
            if (Codi.afinacio == 1) //corda massa tensada
            {
                  setContentView(R.layout.resultat_massa_tensada);
            }
     // setContentView(R.layout.resultat);
   }
}
```

7.2 Arxius xml de la carpeta layout

7.2.1 about.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<ScrollView
   xmlns:android="http://schemas.android.com/apk/res/android"
   android:layout_width="fill_parent"
   android:layout_height="fill_parent"
   android:padding="10dip">
   <TextView
      android:id="@+id/about_content"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:text="@string/about_text" />
</ScrollView>
```

7.2.2 cordes.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
        xmlns:android="http://schemas.android.com/apk/res/android"
    android:background="@color/background"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:padding= "30dip"
    android:orientation="horizontal"
    >
    <LinearLayout
      android:orientation="vertical"
      android:layout_height="wrap_content"
      android:layout_width="fill_parent"
      android:layout_gravity="center">
   <Button
      android:id="@+id/corda_1"
      android:layout_width="fill_parent"
      android:layout_height="wrap_content"
      android:text="@string/corda_1_label" />
      <Button
      android:id="@+id/corda_2"
      android:layout_width="fill_parent"
      android:layout_height="wrap_content"
      android:text="@string/corda_2_label" />
         <Button
      android:id="@+id/corda_3"
      android:layout_width="fill_parent"
      android:layout_height="wrap_content"
      android:text="@string/corda_3_label" />
         <Button
      android:id="@+id/corda_4"
      android:layout_width="fill_parent"
      android:layout_height="wrap_content"
      android:text="@string/corda_4_label" />
       <Button
      android:id="@+id/corda_5"
      android:layout_width="fill_parent"
      android:layout_height="wrap_content"
      android:text="@string/corda_5_label" />
         <Button
      android:id="@+id/corda_6"
      android:layout_width="fill_parent"
      android:layout_height="wrap_content"
      android:text="@string/corda_6_label" />
           <Button
      android:id="@+id/record_button"
      android:layout_width="fill_parent"
      android:layout_height="wrap_content"
      android:text="@string/record_label" />
           <Button
      android:id="@+id/back_button"
      android:layout_width="fill_parent"
      android:layout_height="wrap_content"
      android:text="@string/back_label" />
  </LinearLayout>
  </LinearLayout>
```

7.2.3 main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
       xmlns:android="http://schemas.android.com/apk/res/android"
    android:background="@color/background"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:padding=  "30dip"
    android:orientation="horizontal"
    >
    <LinearLayout
      android:orientation="vertical"
      android:layout_height="wrap_content"
      android:layout_width="fill_parent"
      android:layout_gravity="center">
<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="@string/main_title"
    android:layout_gravity="center"
    android:layout_marginBottom="25dip"
    android:textSize="24.5sp"
    />
   <Button
      android:id="@+id/start_button"
      android:layout_width="fill_parent"
      android:layout_height="wrap_content"
      android:text="@string/start_label" />
   <Button
      android:id="@+id/about_button"
      android:layout_width="fill_parent"
      android:layout_height="wrap_content"
      android:text="@string/about_label" />
   <Button
      android:id="@+id/exit_button"
      android:layout_width="fill_parent"
      android:layout_height="wrap_content"
      android:text="@string/exit_label" />
  </LinearLayout>
</LinearLayout>
```

7.2.4 resultat_bo.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<ScrollView
   xmlns:android="http://schemas.android.com/apk/res/android"
   android:layout_width="fill_parent"
   android:layout_height="fill_parent"
   android:padding="10dip">
   <TextView
      android:id="@+id/resultat_bo_content"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:text="@string/resultat_bo_text" />
</ScrollView>
```

7.2.5 resultat_destensada.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<ScrollView
   xmlns:android="http://schemas.android.com/apk/res/android"
   android:layout_width="fill_parent"
   android:layout_height="fill_parent"
   android:padding="10dip">
   <TextView
      android:id="@+id/resultat_destensada_content"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:text="@string/resultat_destensada_text" />
</ScrollView>
```

7.2.6 resultat_massa_tensada.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<ScrollView
   xmlns:android="http://schemas.android.com/apk/res/android"
   android:layout_width="fill_parent"
   android:layout_height="fill_parent"
   android:padding="10dip">
   <TextView
      android:id="@+id/resultat_massa_tensada_content"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:text="@string/resultat_massa_tensada_text" />
</ScrollView>
```

7.3 Arxius xml de la carpeta values

7.3.1 colors.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<resources>
   <color name="background">#6E0000</color>
</resources>
```

7.3.2 strings.xml

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string name="app_name">Afinador</string>
    <string name="main_title">Afinador per Android</string>
    <string name="start_label">ComenÃ§a!</string>
    <string name="about_label">Instruccions</string>
    <string name="exit_label">Surt</string>
    <string name="about_title">Instruccions</string>
       <string name="about_text">\ Per començar, prem el botó "Comença!". Després, prem
el botó de la corda que vulguis afinar. Després prem el botó Grava i a continuació toca
la corda corresponent a la teva guitarra. Per sortir, prem el botó "Surt".


                                        </string>
       <string name="resultat_title">Resultats</string>
       <string name="resultat_bo_text">\ La corda està ben afinada </string>
       <string name="resultat_destensada_text">\ Tensa una mica la corda </string>
       <string name="resultat_massa_tensada_text">\ Destensa una mica la corda </string>
       <string name="corda_1_label">1ra (E): 329,63 Hz</string>
       <string name="corda_2_label">2na (B): 246,94 Hz</string>
       <string name="corda_3_label">3ra (G): 196,00 Hz</string>
       <string name="corda_4_label">4ta (D): 146,83 Hz</string>
       <string name="corda_5_label">5na (A): 110,00 Hz</string>
       <string name="corda_6_label">6na (E): 82,41 Hz</string>
       <string name="record_label">Grava</string>
       <string name="back_label">Torna enrere</string>
</resources>
```

7.4 AndroidManifest.xml

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
      package="com.example.afinador"
      android:versionCode="1"
      android:versionName="1.0">
    <application android:icon="@drawable/icon" android:label="@string/app_name">
        <activity android:name=".Afinador"
                  android:label="@string/app_name">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
          <activity android:name=".About"
                          android:label="@string/about_title"
                          android:theme="@android:style/Theme.Dialog" >
          </activity>
           <activity android:name=".Codi"
           android:label="@string/app_name">
          </activity>
           <activity android:name=".EstatCorda"
           android:label="@string/resultat_title"
           android:theme="@android:style/Theme.Dialog">
          </activity>

    </application>
       <uses-permission android:name="android.permission.RECORD_AUDIO"></uses-
permission>
       <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"></uses-
permission>

</manifest>
```