SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

ZAVRŠNI RAD br. 577

Aplikacija za mobilne uređaje za štimanje gitare

Fran Jelavić

Zagreb, lipanj 2022.

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

Zagreb, 11. ožujka 2022.

ZAVRŠNI ZADATAK br. 577

Pristupnik:	**Fran Jelavić (0036523987)**
Studij:	Elektrotehnika i informacijska tehnologija i Računarstvo
Modul:	Računarstvo
Mentor:	prof. dr. sc. Ivan Đurek

Zadatak: **Aplikacija za mobilne uređaje za štimanje gitare**

Opis zadatka:

Objasnite princip titranja žica na gitari i frekvencijski raspon gitare. Opišite frekvencijski spektar titranja žica gitare uključujući harmonike. Projektirajte softversku aplikaciju za detekciju istaknutih frekvencija titranja žica gitare. Na mobilnom uređaju testirajte aplikaciju i njen rad usporedite s profesionalnim sustavima za štimanje gitara.

Rok za predaju rada: 10. lipnja 2022.

Zahvala mentoru prof. dr. sc. Ivanu Đureku što je pružio mogućnost vlastitog izbora teme i dopustio rad vlastitim tempom.

SADRŽAJ

1. Uvod

Digitalna obrada signala često se koristi za obradu zvuka u glazbenoj umjetnosti. Svaki glazbeni instrument na drugačiji način proizvodi zvuk, ali svaki glazbeni instrument funkcionira po istim principima titranja. Njihova konzistentnost u proizvodnji frekvencija titranja omogućava jednostavniju obradu zvučnog signala iz zvučnih snimki instrumenata.

Kako je svaka glazbena nota odsvirana određenom frekvencijom, tako svaki instrument ima svoj frekvencijski raspon. Kako bi se instrumenti naštimali na određene note, korisno je imati sustav koji će moći prepoznati odsvirane note te ih usporediti s poželjnim notama. Budući da je većina zvučnih signala sačinjena od većeg skupa frekvencija, prepoznavanje note u zvučnom signalu nije toliko jednostavno.

U ovom radu bit će predstavljena ideja iza detekcije frekvencija snimljenog zvučnog signala te problemi koji se javljaju pri snimanju i obradi signala.

Ovaj rad temelji se na detekciji frekvencija titranja u stvarnome vremenu uz primjer gitare. Jedan od mogućih načina implementacija detekcije frekvencija i obrade zvučnog signala bit će prikazana pomoću aplikacije za štimanje gitare na mobilnim uređajima.

2. Frekvencija i visina tona

Zvučni val, kao svaki longitudinalni val, unosi se u medij titranjem objekta. Izvor vala je objekt koji titra, a medij je okruženje kojim val prolazi. Titrajući objekt u kontekstu zvučnih valova mogu biti glasnice čovjeka, žica koja titra, zvučnik itd. Medij u ovom kontekstu je zrak, ljudsko uho, i slično. Bez obzira koji objekt stvara zvučni val, čestice medija kroz koji val prolazi gibaju se naprijed-nazad (sabijaju i razrjeđuju) pri određenoj frekvenciji.

Frekvencija vala odnosi se na to koliko često čestice medija vibriraju kada val prođe kroz medij. Mjeri se kao broj potpunih titraja (kompresija) čestice medija u jedinici vremena. Drugim riječima, mjeri se u titrajima po sekundi, tj. hercima (Hz).

Kako se zvučni val kreće kroz medij, svaka čestica medija titra pri istoj frekvenciji, budući da svaka čestica titra radi gibanja svog najbližeg susjeda. Isto tako, frekvencija kojom titra svaka čestica vala jednaka je frekvenciji izvora zvučnog vala. Primjerice, žica gitare koja titra pri frekvenciji od 500 Hz će uzrokovati da čestice u okruženju titraju pri istoj frekvenciji od 500 Hz, što će rezultirati prenošenjem zvučnog vala frekvencije od 500 Hz do uha slušatelja.

Poznato je da se raspon frekvencija koju mogu raspoznati ljudi proteže između 20 Hz do 20 kHz.

2.1. Visina tona

Visina tona (eng. *pitch*) je termin koji se često koristi kao sinonim za frekvenciju. Iako su vrijednosti izravno povezane i proporcionalne, frekvencija mjeri titraj fizičkog vala u jedinici vremena, dok se visina tona koristi za ljudsku percepciju zvuka (najčešće u glazbi).[1]

Viša frekvencija odgovara višoj visini tona, kao što i niža frekvencija odgovara nižoj visini tona. Većina ljudi može raspoznati dva istovremena zvuka s razlikom frekvencija od 7 Hz, dok neki i do 2 Hz.

3. Frekvencija u glazbi

Ljestvica u zapadnjačkoj koncertnoj glazbi temelji se na oktavi, što je razlika između dva zvuka čije su frekvencije u omjeru 2:1. Dva tona odsvirana istovremeno s razlikom u oktavi zvuče ugodno, stoga oktava ima široku primjenu u glazbi.

Kromatska ljestvica (eng. *chromatic scale* ili *twelve-tone scale*) sačinjena je od jedne oktave podijeljene na 12 jednakih intervala, pri čemu svi intervali imaju jednaki omjer frekvencija od $\sqrt[12]{2}$, odnosno svaki interval je udaljen za jedan poluton (eng. *semitone*).[2]

3.1. Glazbene note

U glazbi se termin nota koristi za reprezentaciju određene visine (i duljine u zapisu) tona. Nota se ujedno koristi i za skupinu visine tona koje spadaju u isti razred (eng. *pitch class*), tj. visine tonova koje su odvojene oktavom. U regijama u kojima se govori engleski uglavnom se koristi zapis nota kojim su note predstavljene pomoću prvih sedam slova latinske abecede (A, B, C, D, E, F i G). Osma nota, nazvana oktavom, zapisuje se istim nazivom (slovom) kao prva, ali ima dvostruko višu frekvenciju.

Kako bi se razlikovale dvije note istog razreda, ali različitih oktava, sustav znanstvenog zapisa visine tona (eng. *scientific pitch notation*) kombinira naziv note s brojem koji označava određenu oktavu. Primjerice, E_2 s frekvencijom od 82,41 Hz je za jednu oktavu viša od E_1 s frekvencijom od 41,20 Hz, a za dvije oktave viša od E_0 s frekvencijom od 20,60 Hz.

Uz broj, nazivu note može se pripisati predznak akcidental (eng. *accidental*).[3] Pod akcidentale spada znak povisilice (eng. *sharp* - ♯) koji podiže notu na frekvenciju koja je $\sqrt[12]{2}$ puta veća od trenutne te znak snizilice (eng. *flat* - ♭) koji spušta notu na frekvenciju $\sqrt[12]{2}$ puta manju od trenutne. Drugim riječima, povisilica (♯) podiže notu za jedan poluton, dok snizilica (♭) spušta notu za jedan poluton.[4] Primjerice, $F_4^\sharp$ s frekvencijom od 369,99 Hz je za jedan poluton viši od F_4 s frekvencijom od 349,23 Hz, ali isto tako za jedan poluton niži od G_4 s frekvencijom od 392 Hz. Stoga se nota

$F_4^\sharp$ da zapisati i kao $G_4^\flat$.

Osim povisilice ($\sharp$) i snizilice ($\flat$), pod akcidentale se ubraja i razrešnica (eng. *natural* - $\natural$), koji se pridodaje tzv. prirodnoj noti čija visina tona ostaje nepromijenjena, odnosno nije ni povišena ni snižena za poluton.[5] Ovaj akcidental se uglavnom ne koristi jer se bez njegovog dodatka nazivu note sporazumijeva da je nota prirodna.

3.2. Kromatska ljestvica

Konačno, uz navedene informacije može se prikazati kromatska ljestvica, konstruirana po noti C.

C $C^\sharp$ ili $D^\flat$ D $D^\sharp$ ili $E^\flat$ E F $F^\sharp$ ili $G^\flat$ G $G^\sharp$ ili $A^\flat$ A $A^\sharp$ ili $B^\flat$ B

3.3. Koncertna visina tona

Koncertna visina tona (eng. *concert pitch*) je referenca prema kojoj se skupina glazbenih instrumenata štima za nastup. Koncertna visina tona može varirati od ansambla do ansambla, a često se mijenjala kroz povijest. Najčešći moderni standard za štimanje koristi 440 Hz za notu A_4, prema kojoj se u odnosu na nju ostale note postavljaju.

3.4. Intervali u glazbi

U glazbi se termin intervala koristi za razliku u visini tona između dva zvuka.[6] Najmanji interval kromatske ljestvice je poluton. Pri početku poglavlja spomenuto je da su frekvencije dviju nota s razlikom u oktavi u omjeru 2:1. To znači da se uzastopnim povećanjem visine tona za isti interval ujedno eksponencijalno povećava frekvencija, iako ljudsko uho to percipira kao linearno povećanje visine tona. Iz tog razloga, intervali se često mjere u centima (eng. *cent*).[7] Cent je jedinica dobivena dijeljenjem polutona na 100 jednakih dijelova, odnosno dijeljenjem oktave na 1200 jednakih dijelova. Formulu za cent moguće je izraziti logaritmom omjera frekvencija:

$$n = 1200 * \log_2(\frac{f_1}{f_2}) \tag{3.1}$$

pri čemu je n broj izražen u centima (cent), a f_1 i f_2 frekvencije izražene u hercima (Hz).

3.5. Harmonici

Većina zvukova mješavina je različitih frekvencija. Pregibi ili harmonici (eng. *harmonics*) nastaju pri zvukovima s jednom dominantnom, odnosno temeljnom frekvencijom (eng. *fundemental frequency*), kao frekvencije višekratnika temeljne frekvencije.[8] Temeljna frekvencija je najniža frekvencija kojom objekt titra i uglavnom je najvećeg intenziteta.[9] Primjerice, ako je F temeljna frekvencija, harmonike zvuka bi se moglo prikazati na ovaj način:

$$a * F + b * 2F + c * 3F + ... \tag{3.2}$$

pri čemu su a, b, c koeficijenti intenziteta. Koeficijent a je uglavnom najveći, budući da je F temeljna frekvencija. Razliku u koeficijentima čine različiti izvori zvuka. Osim intenziteta harmonika, razliku u zvuku između dvaju različita izvora čine i slabije popratne frekvencije, tzv. alikvotni tonovi (eng. *overtones*) koji tonu daju dodatnu boju.[10] Iz tih razloga će nota odsvirana na dva različita instrumenta zvučati drugačije, ali jednake visine tona.

Više o harmonicima objašnjeno je u sljedećem poglavlju, na primjeru titranja žica gitare.

4. Frekvencije na primjeru gitare

Ovaj rad služi se isključivo primjerom standardno naštimane (eng. *standard tuning*) akustične gitare, prema standardnoj koncertnoj visini tona (440 Hz). *Standard tuning* podrazumijeva da su žice, od gornje prema donjoj, redom naštimane prema notama E_2, A_2, D_3, G_3, B_3 i E_4.

4.1. Frekvencijski raspon

Frekvencijski raspon gitare ovisi o tome kako je gitara štimana i o samom tipu gitare (klasična, akustična, električna) budući da različiti tipovi gitare imaju različitu građu (posebice duljinu vrata).

Frekvencijski raspon akustične gitare proteže se od note E_2 s frekvencijom od 82,41 Hz do note $C_6^\sharp$ s frekvencijom od 1108,73 Hz. Valja napomenuti da je najniža frekvencija postignuta sviranjem "otvorene" (nepritisnute) gornje E_2 žice, a najviša frekvencija postignuta sviranjem donje E_4 žice, pritisnute na 21. polju (eng. *fret*) vrata gitare.

Nota/oktava	2	3	4	5	6
C		130,8	261,6	523,3	1047
$C^\sharp$		138,6	277,2	554,4	1109
D		146,8	293,7	587,3	
$D^\sharp$		155,6	311,1	622,3	
E	82,41	164,8	329,6	659,3	
F	87,31	174,6	349,2	698,5	
$F^\sharp$	92,50	185,0	370,0	740,0	
G	98,00	196,0	392,0	784,0	
$G^\sharp$	103,8	207,7	415,3	830,6	
A	110,0	220,0	440,0	880,0	
$A^\sharp$	116,5	233,1	466,2	932,3	
B	123,5	246,9	493,9	987,8	

Tablica 4.1: Frekvencijski raspon akustične gitare u Hz.

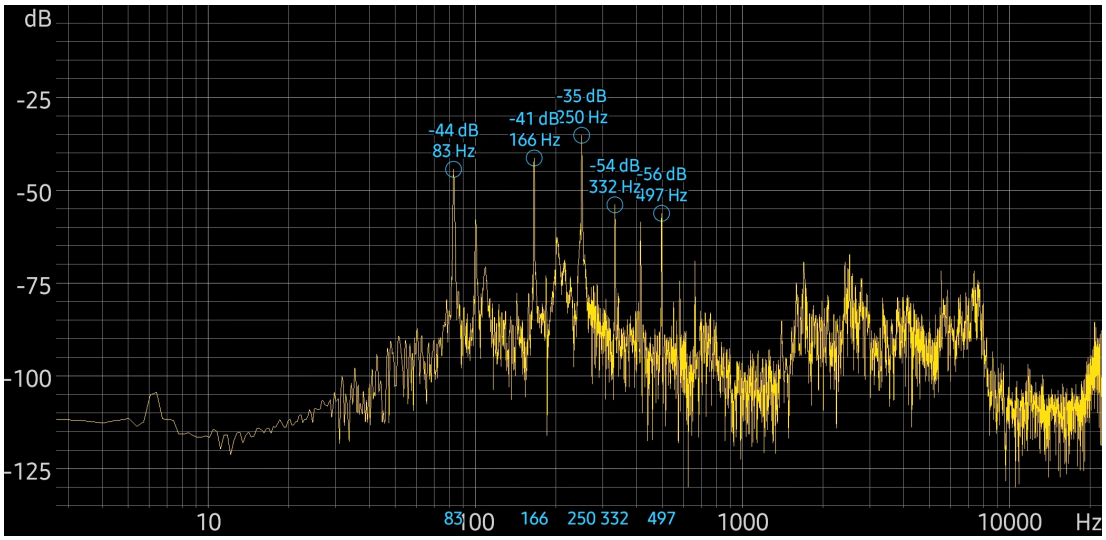
4.2. Frekvencijski spektri

Harmonici se lako mogu istaknuti prikazima frekvencijskih spektara snimljenih zvukova. Slike 4.1 do 4.6 predstavljaju spektrograme frekvencija žica gitare, snimljene pomoću aplikacije *Spectroid*.[15] Lako je uočiti kako se među frekvencijama ističe temeljna frekvencija i njezini harmonici.

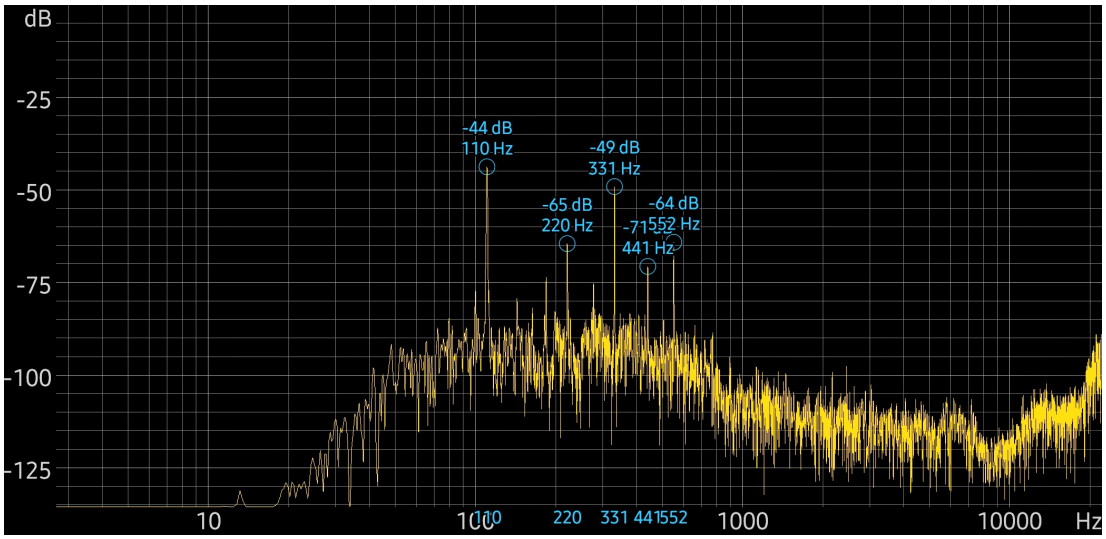
Primjerice, slika 4.5 vrlo jasno prikazuje ovaj fenomen, s istaknutom dominantnom frekvencijom od 247 Hz, što je upravo temeljna frekvencija note B_3. Temeljnu frekvenciju redom slijede istaknute frekvencije od 494 Hz, 739 Hz, 985 Hz te 1478 Hz, kao (približni) višekratnici temeljne frekvencije. Valja napomenuti da intenzitet harmonika nije uvijek silazan, što se vidi s harmonikom od 1478 Hz koji je jačeg intenziteta od prethodnih dva harmonika.

Drugačiji primjer pokazuje slika 4.1. Frekvencija od 83 Hz note E_2 je istaknuta, ali nije najistaknutija frekvencija. To je česti problem pri detekciji frekvencija u zvučnim signalima. Ovisno o podatcima snimljenog signala, temeljna frekvencija može, ali ne mora biti najistaknutija.[11] Više o ovom problemu nastavlja se u poglavlju 6.2.

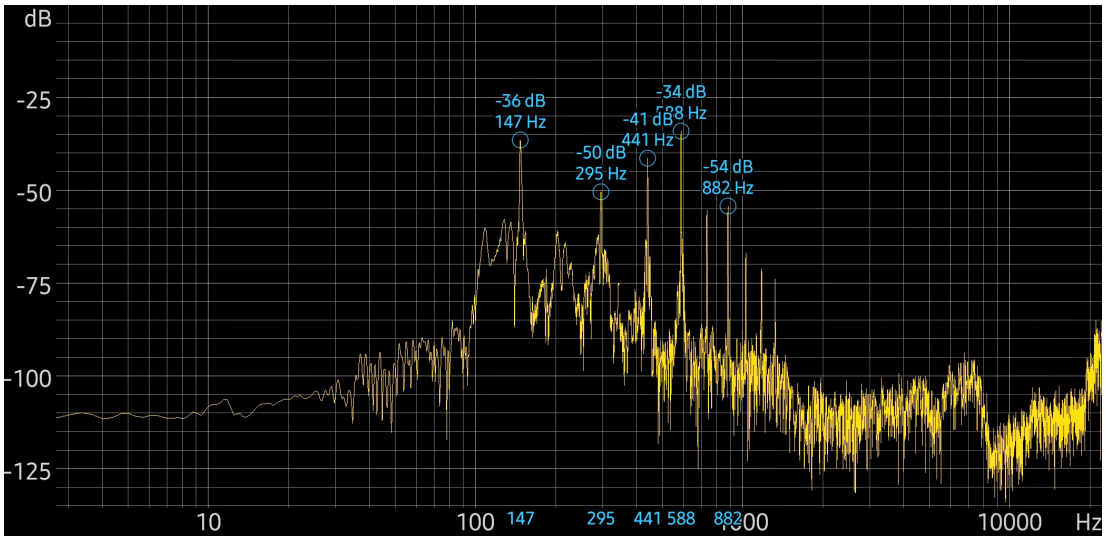
Zbog neidealnih uvjeta pri mjerenju u sklopu ovoga rada dolazi do slabe devijacije u vrijednostima višekratnika, ali je pojava harmonika svejedno jasno vidljiva.



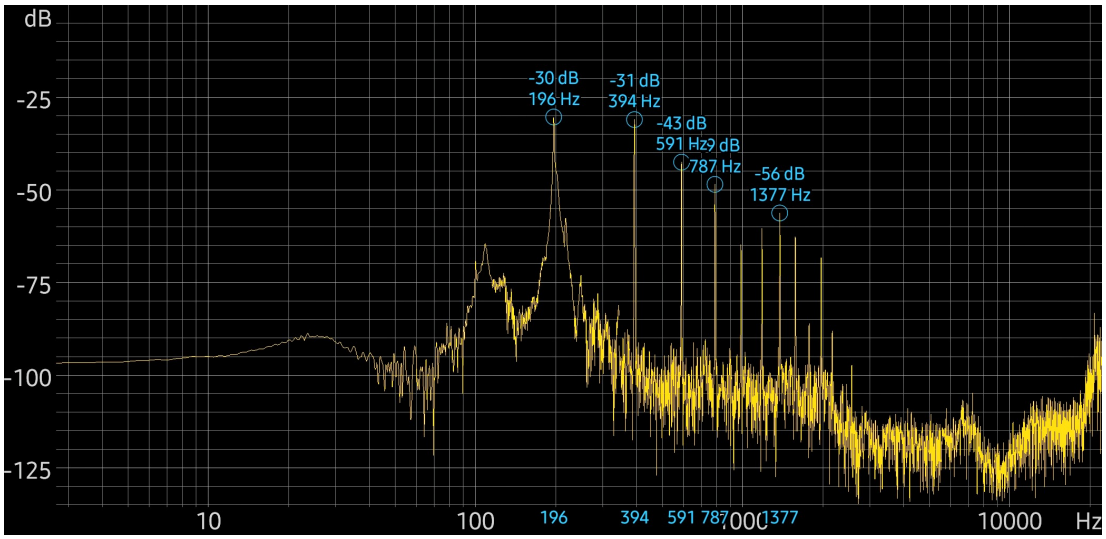
Slika 4.1: Frekvencijski spektar žice E_2



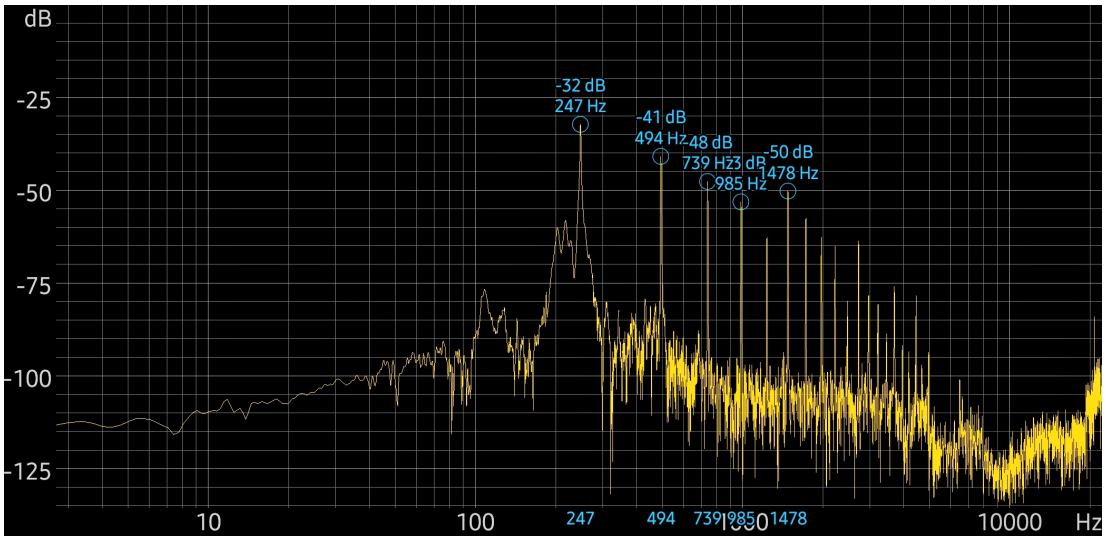
Slika 4.2: Frekvencijski spektar žice A_2



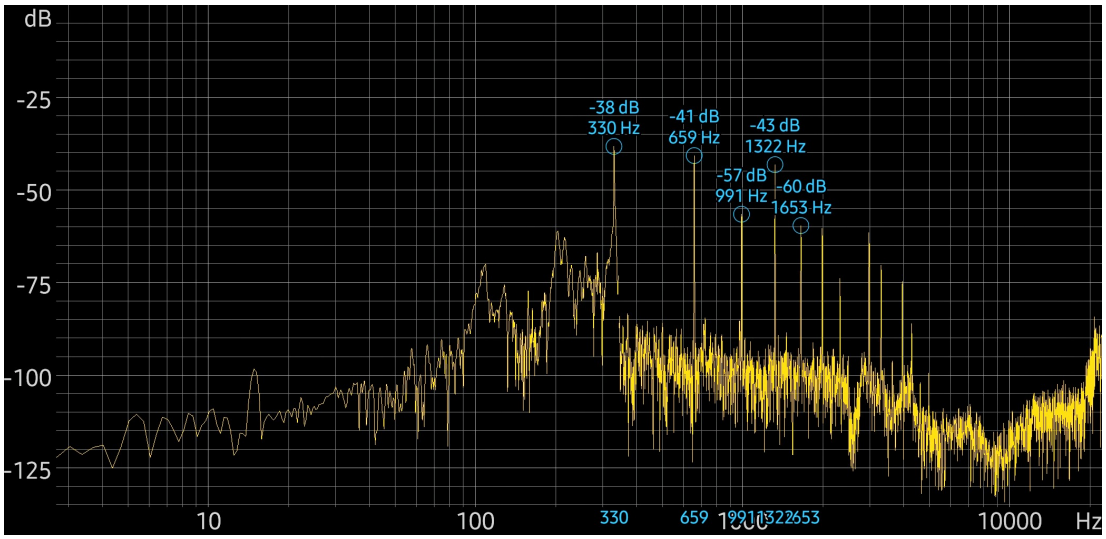
Slika 4.3: Frekvencijski spektar žice D_3



Slika 4.4: Frekvencijski spektar žice G_3



Slika 4.5: Frekvencijski spektar žice B_3



Slika 4.6: Frekvencijski spektar žice E_4

5. Digitalna obrada signala

Digitalna obrada signala (eng. *Digital Signal Processing - DSP*) koristi procesore signala koji preuzimaju snimljene signale iz stvarnog svijeta poput zvuka, govora, video prikaza, temperature, tlaka, radara ili ostalih signala koji su digitalizirani te nad njima izvode matematičke operacije kako bi ih oblikovali.

Signale je potrebno obraditi kako bi se informacija koju sadrže mogla bolje prikazati, analizirati ili pretvoriti u drugu vrstu signala koja bi mogla biti od koristi. Kako bi signal uopće bio digitaliziran, potrebno je pretvoriti snimljeni analogni signal u digitalni binarni zapis, što se postiže analogno-digitalnim pretvaračima (eng. *Analog-to-digital converter - ADC*).

Primjena digitalne obrade signala omogućuje mnoge prednosti u odnosu na analognu obradu, kao što je kompresija podataka ili otkrivanje i ispravljanje pogrešaka pri prijenosu podataka.[12]

U ovome radu digitalna obrada signala odnosi se isključivo na obradu snimljenog zvučnog signala.

5.1. Uzorkovanje

U kontekstu obrade zvučnog signala, uzorkovanje (eng. *sampling*) je pretvorba zvučnog vala u slijed "uzoraka". Uzorak (eng. *sample*) je vrijednost signala u točki u vremenu i/ili prostoru. Sklop za uzorkovanje (eng. *Sampler*) je operacija ili sustav koji izvlači uzorke iz "kontinuiranog" signala. Cilj sklopa za uzorkovanje je zabilježiti uzorke ekvivalentne njihovim trenutnim vrijednostima u signalu.

Frekvencija uzorkovanja (eng. *sample rate* ili *sampling frequency*) je prosječni broj dobavljenih uzoraka u jednoj sekundi te se mjeri u hercima (Hz). Frekvencija uzorkovanja je najčešće iznad 40 kHz radi teorema Nyquista i Shannona, koji tvrdi da je rekonstrukcija signala moguća tek kada je frekvencija uzorkovanja dvostruka veća od frekvencije signala koji se obrađuje.[13] Zvučni signali snimljeni su uglavnom do 20 kHz, budući da je to gornja granica ljudskog sluha.

Najčešće frekvencije uzorkovanja su 44,1 kHz, 48 kHz, 88,2 kHz ili 96 kHz.[14] Koriste se uglavnom frekvencije uzorkovanja do 50 kHz jer vrijednosti više od toga ne pridonose zvuku za ljudsko uho, a zauzimaju veću širinu pojasa (eng. *bandwidth*).

6. Algoritam brze Fourierove transformacije - FFT

Aplikacija *Spectroid* za detekciju frekvencija koristi algoritam brze Fourierove transformacije (eng. *Fast Fourier Transform - FFT*).[15] Algoritam FFT pretvara signal u skup brojeva pomoću kojeg se da raspoznati najzastupljenije frekvencije u signalu. Sposobnost FFT-a da podijeli zvučni signal u frekvencije od kojih se sastoji ga čini jako korisnom metodom koja se često koristi u praksi obrade zvučnih signala.[11]

6.1. Način rada FFT-a

Obradom signala pomoću FFT-a, na izlazu se dobiva slijed brojeva u nizu koji predstavlja domenu frekvencija obrađenog signala, npr.:

```
fft_output_array = [23, 43, 65, 443, 321, 54, 56 ...]
```

Svaki indeks ovog niza naziva se *bin* te predstavlja određen raspon frekvencija. Broj na svakom indeksu je intenzitet frekvencija u rasponu koji predstavlja. Što je broj veći, to je veća zastupljenost frekvencija unutar svoga raspona. Raspon frekvencija u pri svakom indeksu ovisi o rezoluciji (eng. *resolution*) koja se može izraziti na sljedeći način:

```
resolution = sampling_rate / fft_size
```

pri čemu je `fft_size` veličina spremnika za uzorke (eng. *buffer size*).[16] Veličina *buffer*-a povezana je s vremenom koje je potrebno za obradu signala, pri čemu manje veličine stvaraju manje kašnjenje, ali predstavljaju veći teret pri procesiranju i veću mogućnost grešaka.[17]

Kako bi se raspoznale frekvencije po *bin*-ovima, može se koristiti sljedeća formula:

```
start_frequency = bin_number * resolution
```

pri čemu je `start_frequency` početna frekvencija raspona.[11] Raspon frekvencija bi time bio od `start_frequency` do `start_frequency + resolution`.

Primjerice, s frekvencijom uzorkovanja od 48 kHz i veličinom *buffer*-a od 1024 uzoraka dobiva se rezolucija od 46,875 Hz po uzorku; što znači da svaki *bin* predstavlja raspon frekvencija dugačak 46,875 Hz. Prošlom formulom moguće je izračunati početnu frekvenciju raspona 12. *bin*-a, a ta frekvencija je 562,5 Hz. Time se može zaključiti da se raspon 12. *bin*-a proteže od 562,5 Hz do 609,375 Hz.

Kako bi algoritam mogao raspoznati dvije različite note, rezolucija mora biti manja ili jednaka razlici njihovih frekvencija. U navedenome primjeru izračunata je rezolucija od 46,875 Hz po uzorku. Ako se uspoređuju note najnižih frekvencija akustične gitare (E_2 od 82,41 Hz i F_2 od 87,31 Hz) gdje je razlika u frekvencijama najniža (4,9 Hz), rezolucija neće biti dovoljno niska da se note mogu raspoznati. Stoga je potrebno sniziti rezoluciju na vrijednost od 4,9 Hz po uzorku ili niže. To je moguće postići snižavanjem vrijednosti frekvencije uzorkovanja ili povećanjem *buffer*-a, po cijeni dodatnog opterećenja procesora i vremena za punjenje *buffer*-a, što u krajnju ruku smanjuje koliko su rezultati u stvarnom vremenu.

6.2. Problemi FFT-a

U poglavlju 4.2 spomenut je problem pri detekciji temeljne frekvencije signala. Na slici 4.1 temeljna frekvencija note E_2 je 82,41 Hz, ali nije najvećeg intenziteta. Stoga se ne može pouzdati na prepoznavanje temeljne frekvencije preko one s najvećim intenzitetom. Ova pojava može izazvati veliki problem u sustavu koji bi trebao prepoznati visinu tona odsvirane note.

Budući da algoritam FFT sam po sebi nije dovoljno dobar alat za cilj koji ovaj rad nastoji ostvariti, mora se koristiti kompleksniji algoritam. U primjeru ovog rada, koristit će se algoritam YIN.

7. Algoritam YIN

YIN (nadahnuto filozofijom *yin* i *yang*) algoritam temelji se na FFT i metodi autokorelacije (eng. *autocorrelation function - ACF*) s brojnim izmjenama koje se kombiniraju kako bi se spriječila pojava grešaka. Algoritam ima nekoliko velikih prednosti. Učestalost grešaka je oko tri puta manja od najboljih konkurentskih metoda; algoritam je prikladan za glazbu i visoke tonove te je jednostavan za implementaciju, s niskim kašnjenjem (eng. *latency*).[18]

7.1. Metoda autokorelacije

Autokorelacija je korelacija signala s odgođenom kopijom samog sebe kao funkcija kašnjenja.[18] Koristi se kao alat za pronalaženje ponavljajućih obrazaca, kao što je identificiranje temeljne frekvencije u signalu među njegovim harmonicima. U obradi signala metoda unakrsne korelacije (eng. *cross-correlation*) koristi se za usporedbu dvaju signala i određivanje njihove sličnosti. Autokorelacija je slična, samo što usporedbu čini nad jednim te istim signalom.

8. Aplikacija ZRTuner

Kao primjer primjene digitalne obrade signala, odnosno detekcije temeljne frekvencije zvuka, u sklopu ovog rada razvijena je aplikacija za štimanje žica gitare za mobilne uređaje (eng. *guitar tuner*).

Aplikacija "ZRTuner" (ili punim nazivom "Završni Rad Tuner") za detekciju odsvirane note koristi YIN algoritam te prikazuje odsviranu notu i relativnu udaljenost njene visine tona s visinom tona najbliže note na kromatskoj ljestvici. Pisana je u integriranom razvojnom okruženju (eng. *Integrated Development Environment - IDE*) Android Studio-u u programskom jeziku Java te je namijenjena isključivo za Android platforme.

Cilj aplikacije je omogućiti prepoznavanje visine tona odsvirane note te time uputiti korisnika kako naštimati žicu gitare do najbliže note, uključujući oktavu.

8.1. TarsosDSP

Za implementaciju YIN algoritma ZRTuner koristi TarsosDSP, Java biblioteku (eng. *library*) za obradu zvuka.[19]

Cilj biblioteke je pružiti jednostavno sučelje za implementaciju algoritama za obradu zvuka u glazbi pisano u Javi i bez ikakvih drugih vanjskih ovisnosti. Nastoji uravnotežiti kompleksnu funkcionalnost za uspješnu primjenu s jednostavnosti za demonstraciju kako algoritmi digitalne obrade signala funkcioniraju.[20]

8.2. Značajke sučelja

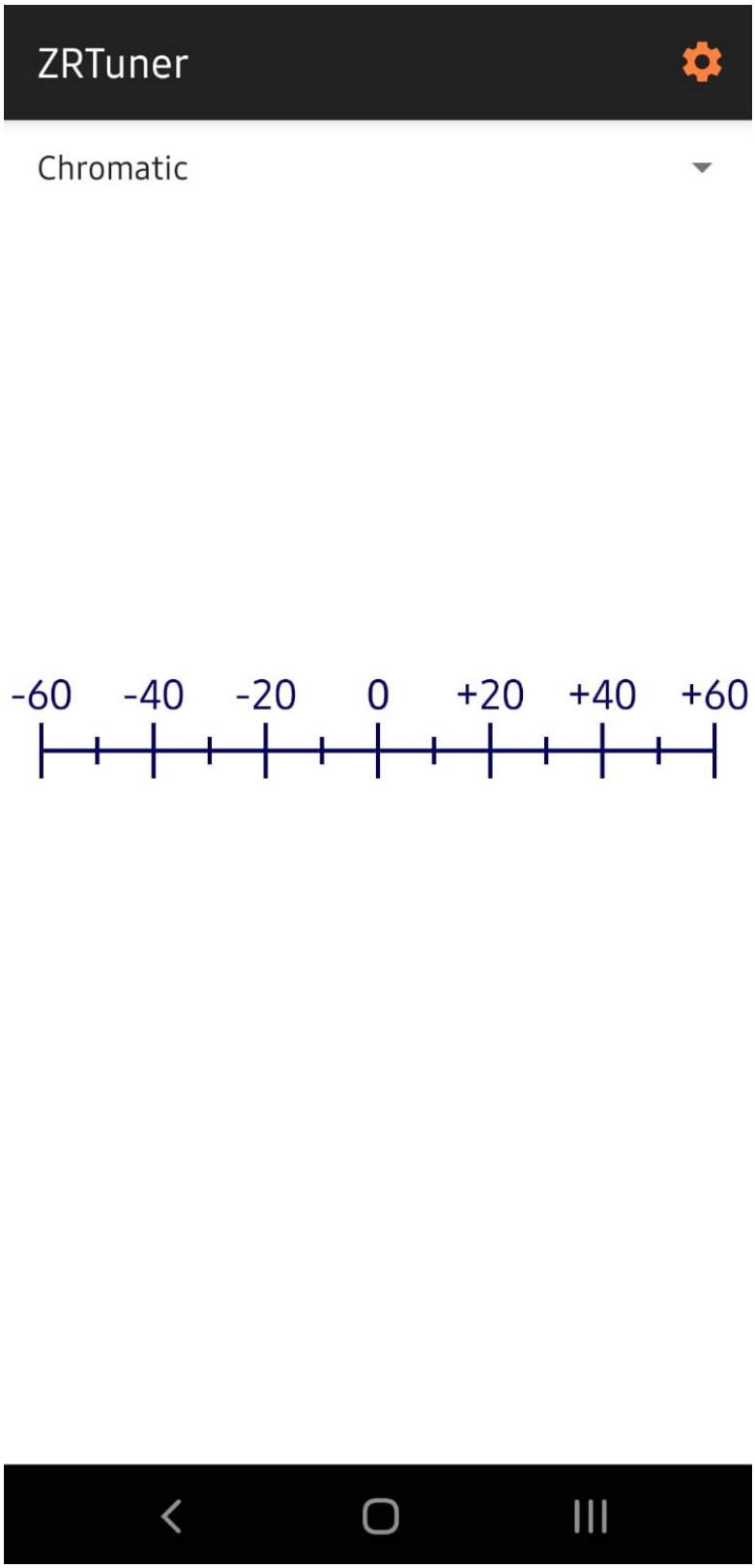
Pri ulasku u aplikaciju prezentirano je sučelje koje se sastoji od navigacijske trake, padajućeg popisa za izbor načina štimanja (eng. *tuning*) i mjerača visine tona.

Navigacijska traka pri vrhu sučelja sastoji se od naslova aplikacije i navigacijskog gumba za postavke. U postavkama moguće je pomoću klizača podesiti koncertnu visinu tona, koja je po standardu zadane vrijednosti od 440 Hz.

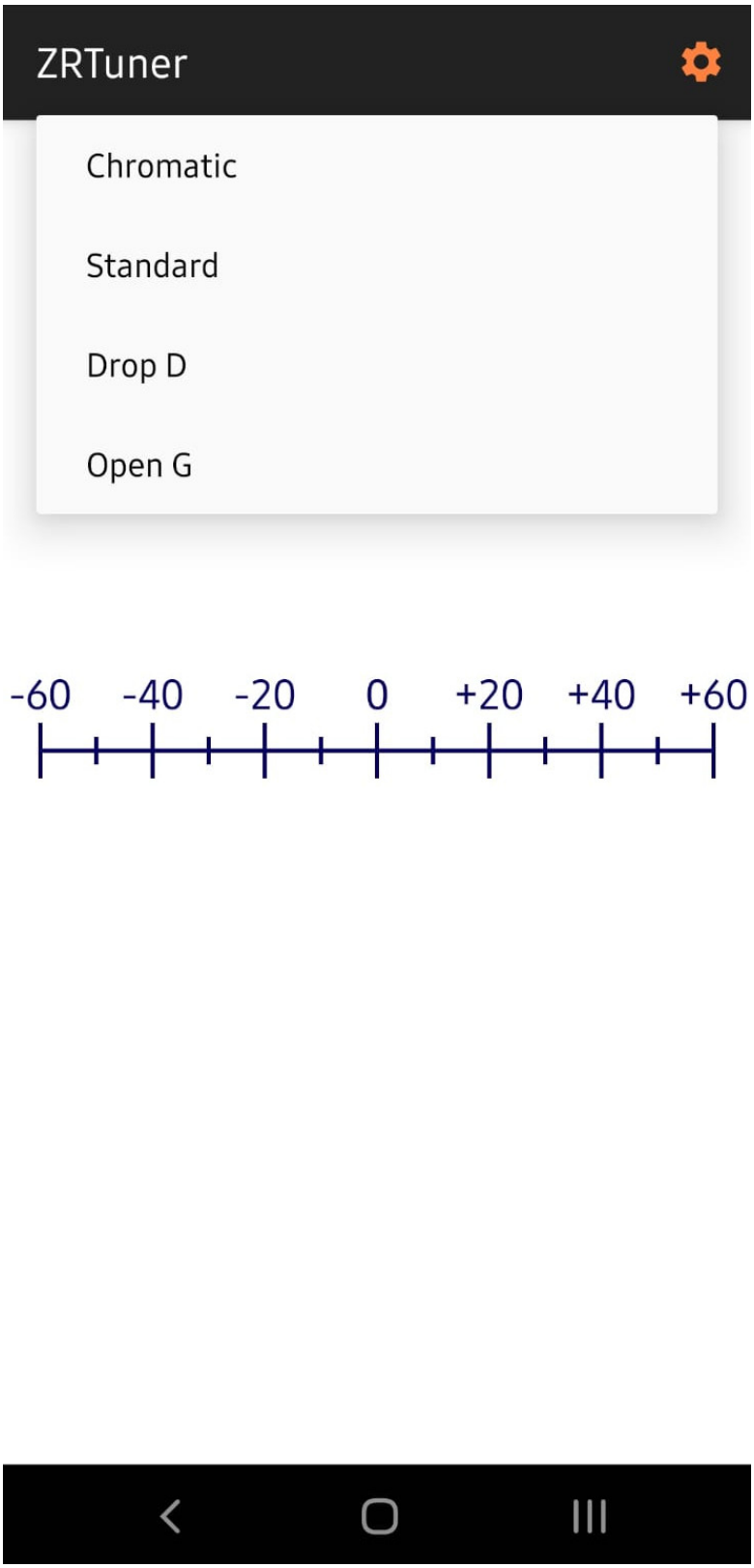
Pod navigacijskom trakom nalazi se padajući popis načina štimanja među koje spadaju kromatski (eng. *chromatic tuning*), standardni (eng. *standard tuning*), *drop D* i *open G*. Izbor načina štimanja implementiran je s ciljem da se tražene note mogu lakše izolirati te da se smanji mogućnost pojave grešaka pri prepoznavanju note.

Glavni dio sučelja čini mjerač koji prikazuje razliku između snimljene visine tona i visine tona najbliže note te je razlika izražena u cjelobrojno zaokruženim centima. Mjerač se sastoji od vodoravne trake podijeljena na cjeline negativnih (lijevo) i pozitivnih (desno) vrijednosti, s vrijednosti od 0 u sredini.

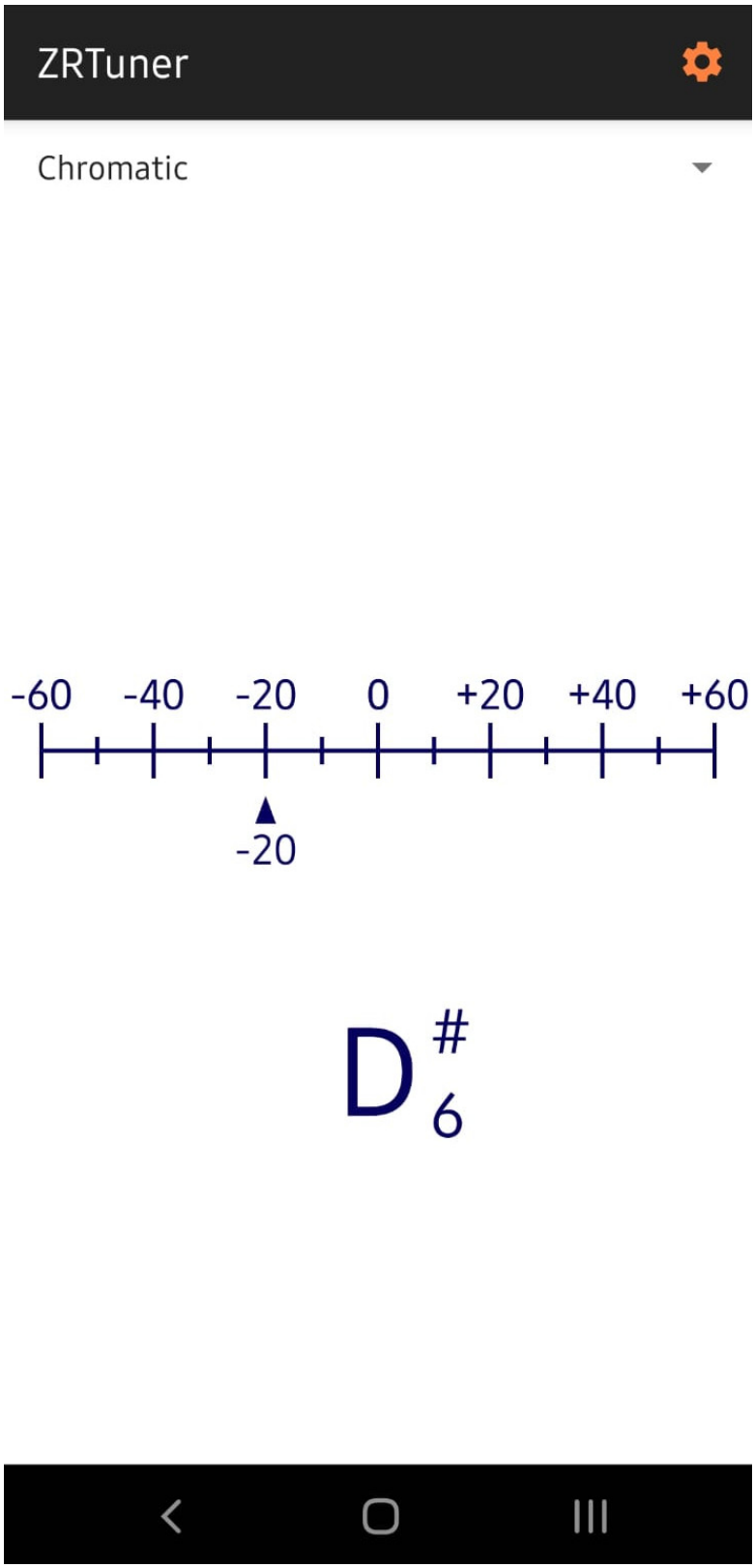
Pri prepoznavanju visine tona snimljenog zvuka, mjeraču se pridodaje pokazivač ispod trake koji razliku u visini tonova prikazuje brojem ispod sebe te pokazuje na prikladnu vrijednost na traci. Uz pokazivač, pri dnu sučelja prikazuje se puni naziv note s najbližom visinom tona. U slučaju da je razlika u visini tonova manja ili jednaka 5 centi za duljinu jedne sekunde, boja trake pretvorit će se u zelenu, kako bi se impliciralo na to da je žica relativno precizno naštimana.



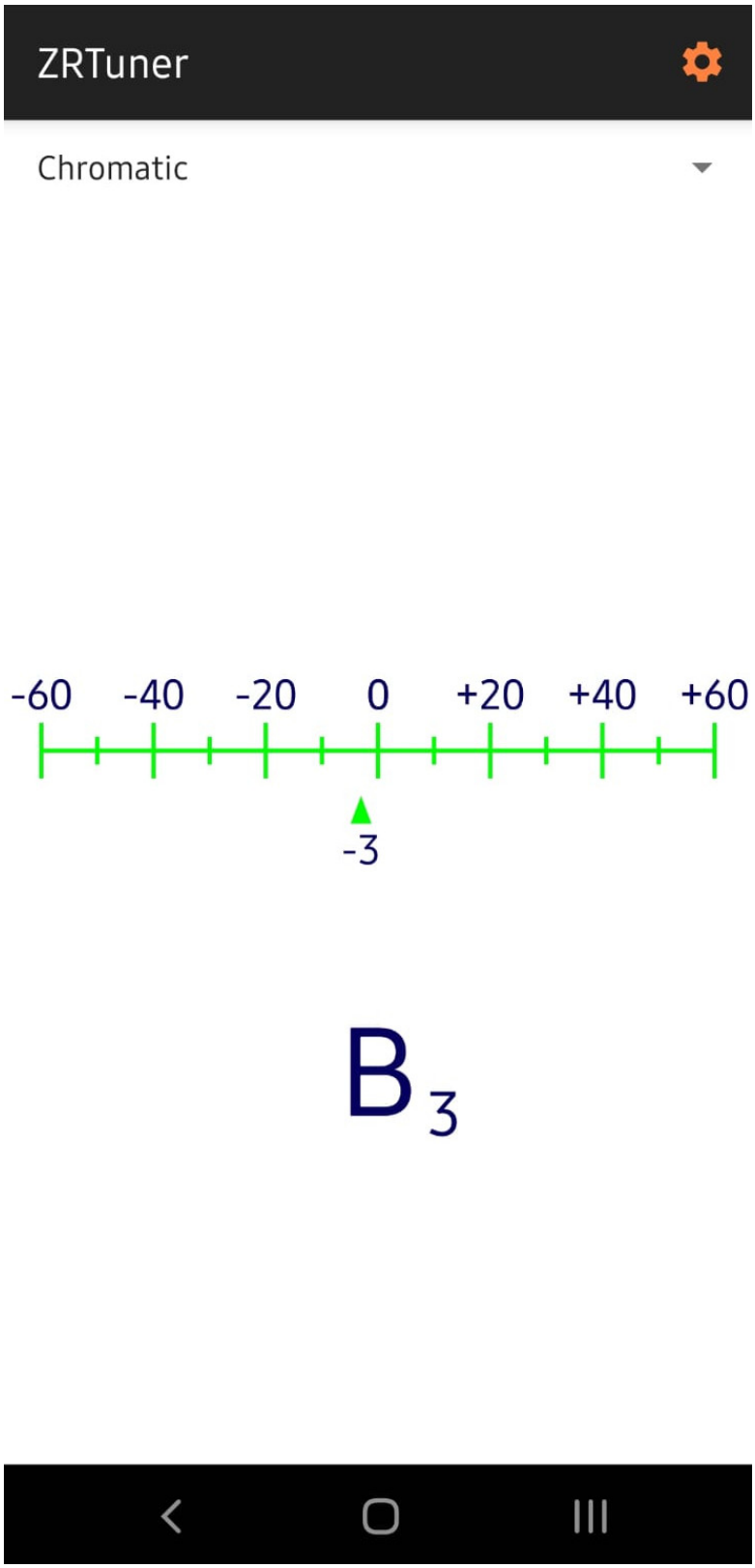
Slika 8.1: Početno sučelje aplikacije



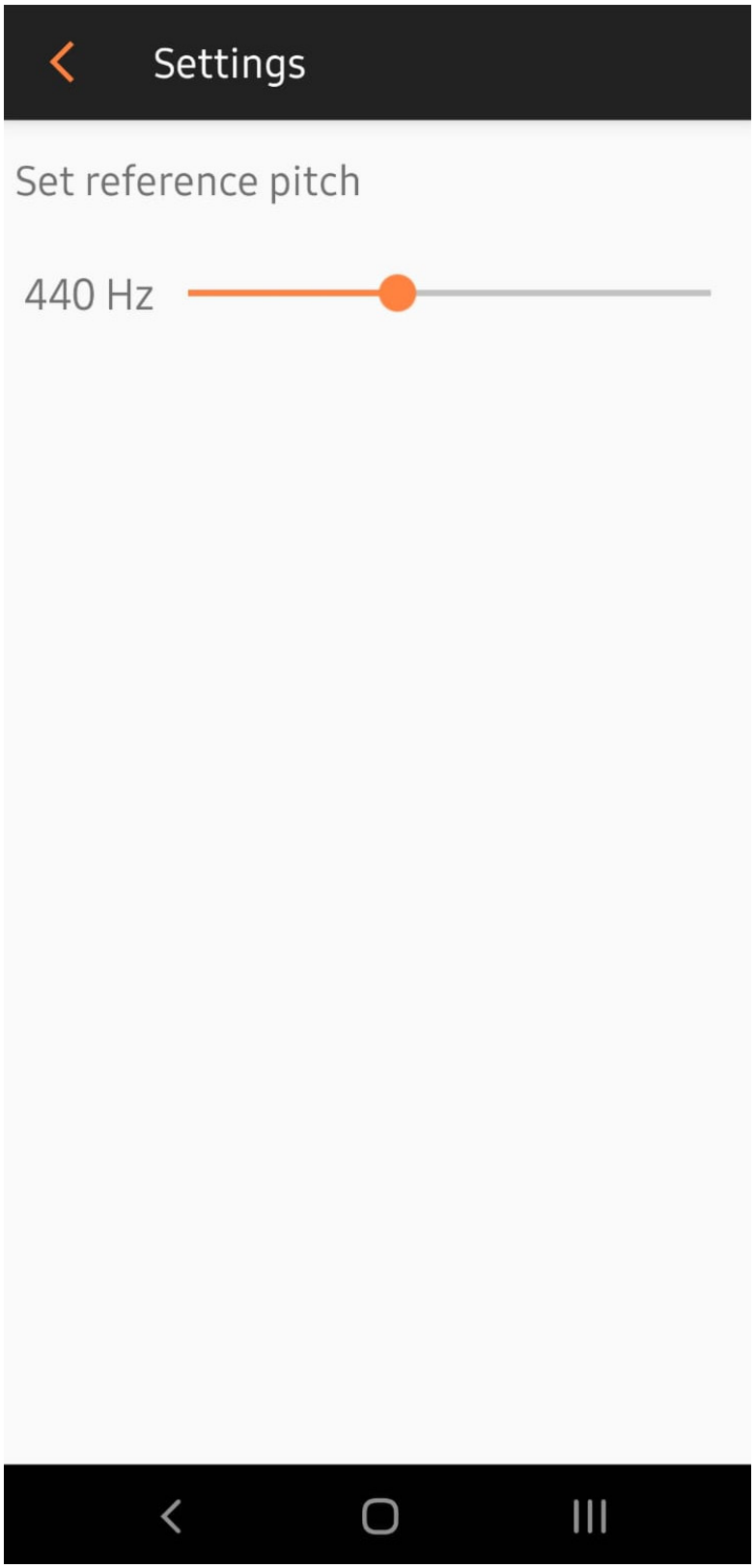
Slika 8.2: Sučelje padajućeg popisa načina štimanja



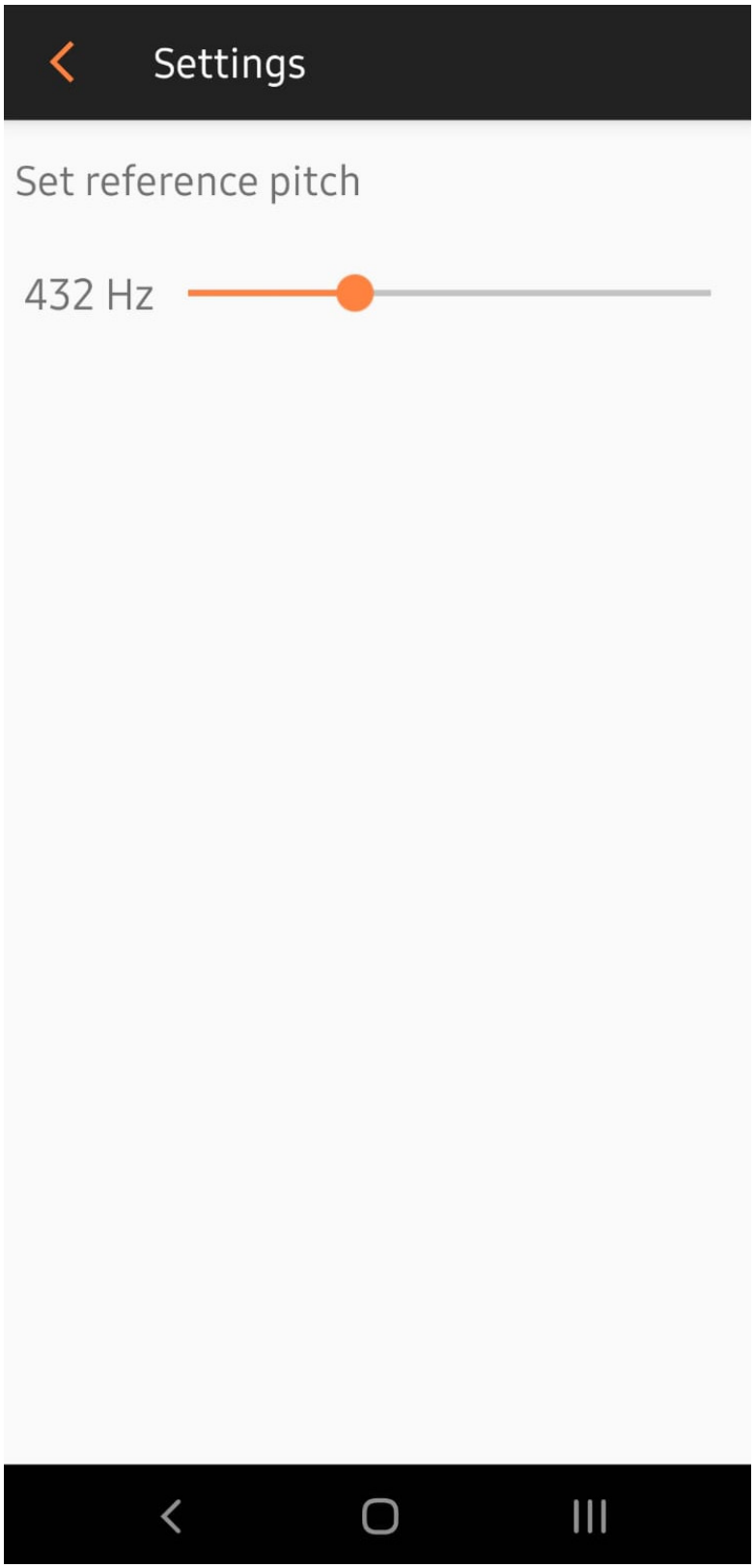
Slika 8.3: Sučelje mjerač pri prepoznatoj visini tona



Slika 8.4: Sučelje mjerača pri maloj razlici u visinama tona



Slika 8.5: Sučelje postavki s klizačem za koncertnu visinu tona



Slika 8.6: Sučelje postavki s promijenjenom koncertnom visionom tona

8.3. Snimanje i obrada zvuka

Aplikacija pri prvom dizanju šalje upit korisniku za dopuštenje snimanja zvuka. Dopuštenjem snimanja zvuka aplikacija kreće u rad, zove metodu `startRecording()` u glavnoj klasi `MainActivity` te pokreće snimanje i obradu zvuka.

8.3.1. Stvaranje fragmenta

Metoda `startRecording()` instancira objekt klase `FragmentManager` pomoću kojeg će moći instancirati i upravljati objektom klase `ListenerFragment`,[21] koji će u sklopu klase `MainActivity` služiti za pozivanje operacija potrebnih za detekciju visine tona. Aktivnosti (eng. *Activity*) u Android aplikacijama najčešće su predstavljene kao prozori sa sučeljem koji su usredotočeni na obavljanje specifičnog zadatka.

Fragmenti u Android aplikacijama predstavljaju dio korisničkog sučelja s višekratnom uporabom koji služi za obavljanje specifičnih funkcija. Za dodavanje, brisanje i zamjenu fragmenata te upravljanje fragmentima koristi se klasa `FragmentManager`. Klasa `ListenerFragment` nasljeđuje klasu `Fragment` te pomoću objekta klase `FragmentManager` pri inicijalizaciji svog objekta dohvaća instancu svog fragmenta preko oznake "listener_fragment".

Budući da pri prvome instanciranju fragment te oznake neće biti pronađen (jer još nije instanciran), uz pomoć metode `beginTransaction()` započet će proces stvaranja novog fragmenta s oznakom "listener_fragment". Metodom `add()`, u glavnoj aktivnosti `MainActivity` stvara se fragment `ListenerFragment`, a metodom `commit()` se taj proces izvršava i završava.

```
private static final String TAG_LISTENER_FRAGMENT = "listener_fragment";

private void startRecording() {
    FragmentManager fragmentManager = getSupportFragmentManager();
    ListenerFragment listenerFragment =
            (ListenerFragment) fragmentManager.findFragmentByTag(TAG_LISTENER_FRAGMENT);

    if (listenerFragment == null) {
        listenerFragment = new ListenerFragment();
        fragmentManager.beginTransaction()
                .add(listenerFragment, TAG_LISTENER_FRAGMENT)
                .commit();
    }
}
```

Kod 8.7: Stvaranje fragmenta za snimanje zvuka u klasi `MainActivty`

8.3.2. Pozadinski proces

Stvaranjem fragmenta `ListenerFragment` pokreće se instanca pozadinskog procesa privatne klase `PitchListener`. `PitchListener` nasljeđuje apstraktnu klasu `AsyncTask` kako bi uz metodu `doInBackground` aplikacija u pozadini mogla obrađivati snimljeni zvuk.

Metoda `doInBackground` započinje instanciranjem objekta sučelja za rukovanje visinom tona `PitchDetectionHandler` iz biblioteke TarsosDSP te lambda izrazom definira metodu `handlePitch()`.

Klasom `PitchDetectionResult` pomoću metode `getPitch()` dohvaća se detektirana visina tona iz snimljenog bloka zvučnog signala. U slučaju detektirane visine tona instancira se objekt klase `PitchDifference`,[21] koji predstavlja devijaciju snimljene visine tona od njoj najbliže note. Objekt `pitchDifference` preko klase `PitchComparator`[21] metodom `retrieveNote()` dohvaća najbližu notu i devijaciju u visini tona izraženu u centima te ih kao objekt tipa `PitchDifference` stavlja u niz `pitchDifferences`. U trenutku kada se ovaj postupak ponovi barem `MIN_PITCH_COUNT` puta (po potrebi izabrano 8 puta), izračunat će se prosječna visina tona niza `pitchDifferences` te će metodom `publishProgress()` ta vrijednost biti proslijeđena za prikaz korisniku. Navedeni prosjek računa se metodom `calculateAverageDifference` klase `Sampler`.[21]

Instanciranjem objekta `pitchDetectionHandler`, njegova vrijednost prosljeđuje se kao argument za instanciranje objekta klase `PitchProcessor` iz biblioteke TarsosDSP, zajedno s enumeracijom YIN algoritma `FFT_YIN`, frekvencijom uzorkovanja `SAMPLE_RATE` tipa `int` i veličinom *buffer*-a `BUFFER_SIZE` tipa `int`.

Aplikacija koristi frekvenciju uzorkovanja od 44,1 kHz i veličinu *buffer*-a od 4096 uzoraka. Time postiže rezoluciju od 10,77 Hz po uzorku, čime dobiva na brzini, ali gubi na preciznosti među najnižim frekvencijama u rasponu gitare. Ovaj kompromis je prihvatljiv budući da se mogućnost pojave grešaka pri nižim frekvencijama smanjuje na način da se uzastopno mjeri prosjek snimljenih blokova frekvencija.

Klasa `PitchProcessor` kao implementacija sučelja `AudioProcessor` pruža funkcije namještanja izabranog algoritma za detekciju visine tona. U ovom slučaju se instancira objekt klase `FastYin` koja u biblioteci TarsosDSP implementira YIN algoritam.

Iza instance `pitchProcessor` instancira se objekt klase `AudioDispatcher` koja služi za pretvorbu snimljenih zvučnih signala ili zapisa u obradive blokove. Za specifikaciju izvora zvuka koristi se metodama klase `AudioDispatcherFactory`, u ovom slučaju metode `fromDefaultMicrophone()`, budući da aplikacija obrađuje zvuk snimljen u stvarnom vremenu preko mikrofona na mobilnom uređaju. Kao argumenti joj se proslijeđuju `SAMPLE_RATE`, `BUFFER_SIZE` i veličina preklapanja `OVERLAP` tipa `int`. Veličina preklapanja u ovom slučaju je 3072 uzoraka, što je 75% veličine *buffer*-a (4096 uzoraka). Drugim riječima, svakih snimljenih 1024 uzoraka, započet će istovremeno snimanje novih 4096 uzoraka. Kao odabir procesora za obradu zvuka, `audioDispatcher` postavlja `pitchProcessor` metodom `addAudioProcessor`. Konačno, snimanje i obrada signala započinje naredbom `audioDispatcher.run()`.

```
private static class PitchListener extends AsyncTask<Void, PitchDifference, Void> {

    private AudioDispatcher audioDispatcher;

    @Override
    protected Void doInBackground(Void... params) {
        PitchDetectionHandler pitchDetectionHandler = (pitchDetectionResult, audioEvent) -> {

            if (isCancelled()) {...}

            if (!IS_RECORDING) {...}

            float pitch = pitchDetectionResult.getPitch();

            if (pitch != -1) {
                PitchDifference pitchDifference = PitchComparator.retrieveNote(pitch);

                pitchDifferences.add(pitchDifference);

                if (pitchDifferences.size() >= MIN_PITCH_COUNT) {
                    PitchDifference average =
                            Sampler.calculateAverageDifference(pitchDifferences);

                    publishProgress(average);

                    pitchDifferences.clear();
                }
            }
        };

        PitchProcessor pitchProcessor = new PitchProcessor(
                PitchProcessor.PitchEstimationAlgorithm.FFT_YIN, SAMPLE_RATE,
                BUFFER_SIZE, pitchDetectionHandler);

        audioDispatcher = AudioDispatcherFactory.fromDefaultMicrophone(SAMPLE_RATE,
                BUFFER_SIZE, OVERLAP);
        audioDispatcher.addAudioProcessor(pitchProcessor);
        audioDispatcher.run();

        return null;
    }
```

Kod 8.8: Privatna klasa `PitchListener` i metoda `doInBackground()` u klasi `ListenerFragment`

8.4. Usporedba sa sličnim sustavima

Što se tiče mobilnih aplikacija za štimanje gitare, danas ih je tržište puno. To su aplikacije koje uglavnom funkcioniraju na sličan način, bez velike razlike u njihovom načinu rada ili performansama. Radi njihove funkcionalnosti i zadovoljstva među korisnicima, nema velike potrebe za unaprjeđenjem tehnologije koju aplikacije koriste.

Budući da nema prevelike razlike u njihovom načinu rada, najčešće se oslanjaju na dodatne značajke. To su uglavnom implementacije podrške za širi opseg instrumenata, pružanje tečaja za sviranje instrumenata ili mogućnost većeg izbora u izgledu sučelja aplikacije. Stoga aplikacije često služe kao platforma za reklamaciju tuđih ili vlastitih usluga.

Ideja iza aplikacije ZRTuner nije bila samo predstaviti primjer digitalne obrade signala, nego i poslužiti kao jednostavna aplikacija za obavljanje vremenski kratkog zadatka. Motivacija za razvoj aplikacije proizašla je upravo iz većih sličnih aplikacija na tržištu koje otežavaju taj jednostavan zadatak reklamacijom.

U usporedbi s profesionalnim sustavima za štimanje gitara, aplikacije nisu toliko precizne, ali je razlika relativno premalena (do jednog centa) da se one ne bi mogle smatrati jednako dobrom alternativom. Profesionalni sustavi koji su namijenjeni za štimanje uglavnom imaju bolji mikrofon od mobilnih uređaja, građen specifično za tu ulogu. Stoga su profesionalnim sustavima zvučni signali snimljeni bolje. Isto tako, uglavnom imaju manje kašnjenje od njihovih alternativa na mobilnim uređajima.

9. Zaključak

Digitalna obrada signala koristi se u širokom opsegu područja informacijskih tehnologija, od obrade zvučnih signala u glazbi do satelitske industrije. Algoritmi poput FFT razvijaju se i modificiraju od druge polovice 20. stoljeća te se sve više primjenjuju u prijenosu informacija.

Primjena digitalne obrade signala u snimanju ili uređivanju glazbe pokazala se djelotvornom. Problemi poput uklanjanja nepoželjnog šuma ili manipulacija zvuka izabranih instrumenata riješeni su detekcijom frekvencija koje se ističu pri sviranju zvuka. Budući da svaki izvor zvuka ima svoj frekvencijski raspon, detekcijom temeljne frekvencije i harmonika zvuka se poželjni tonovi zvuka mogu bolje podesiti, a nepoželjni stišati.

Svaka primjena digitalne obrade zvučnog signala može se razlikovati u zadanim vrijednostima frekvencije uzorkovanja, veličini *buffer*-a i preklapanja pri samom snimanju zvuka. Ove vrijednosti bitno je uskladiti za zadatak koji se obavlja, kako bi se postigla poželjna ravnoteža brzine i preciznosti.

Kao primjer implementacije razvijena je aplikacija za štimanje žica gitare, odnosno detekciju visine tona odsvirane žice. Budući da za ovu namjenu algoritam FFT sam po sebi nije bio dovoljan, uz pomoć YIN algoritma za detekciju visine tona, kako je implementiran u biblioteci TarsosDSP, postignuta je brzina i preciznost za prikaz devijacije snimljene visine tona od visine tona svoje najbliže note.

LITERATURA

[1] Stanley S Stevens and John Volkmann. The relation of pitch to frequency: A revised scale. *The American Journal of Psychology*, 1940.

[2] Johan Sundberg. *In Tune or Not?: a study of fundamental frequency in music practise*. na, 1982.

[3] Vlastimir Pericic. Visejezicni recnik muzickih termina. 2008.

[4] Hrvatska enciklopedija (1941.-1945.), Apr 2020.

[5] B Benward and M Saker. Introduction. the materials of music: Sound and time. *Music in theory and practice, pp. xii–25). NY, NY: McGraw-Hill*, 2003.

[6] Ebenezer Prout. *Harmony: its theory and practice*. Cambridge University Press, 2011.

[7] Dave Benson. *Music: A mathematical offering*. Cambridge University Press, 2006.

[8] William A Sethares. *Tuning, timbre, spectrum, scale*. Springer Science & Business Media, 2005.

[9] Fundamental and harmonic frequencies, May 2020.

[10] Thomas Christensen and Jean-Philippe Rameau. Eighteenth-century science and the "corps sonore:" the scientific background to rameau's "principle of harmony". *Journal of Music Theory*, 31(1):23–50, 1987.

[11] Jeremy Gustine. Guitar tuner - pitch detection for dummies, Jul 2021.

[12] James D Broesch. *Digital signal processing: instant access*. Elsevier, 2008.

[13] Emmanuel J Candès and Michael B Wakin. An introduction to compressive sampling. *IEEE signal processing magazine*, 2008.

[14] Douglas Self. *Audio engineering explained*. Routledge, 2012.

[15] Carl Reinke. Spectroid, 2018.

[16] Mary Lourde and Anjali Kuppayil Saji. A digital guitar tuner. *arXiv e-prints*, pages arXiv–0912, 2009.

[17] Sample rate, bit depth and buffer size explained – focusrite audio ..., Jan 2022.

[18] Alain De Cheveigné and Hideki Kawahara. Yin, a fundamental frequency estimator for speech and music. *The Journal of the Acoustical Society of America*, 111(4):1917–1930, 2002.

[19] Joren Six, Olmo Cornelis, and Marc Leman. TarsosDSP, a Real-Time Audio Processing Framework in Java. In *Proceedings of the 53rd AES Conference (AES 53rd)*, 2014.

[20] Joren Six, Olmo Cornelis, and Marc Lerman. Tarsosdsp: A real-time audio processing framework in java.

[21] Gstraube. Cythara: A musical instrument tuner for android.

Aplikacija za mobilne uređaje za štimanje gitare

Sažetak

U ovom radu predstavljeno je rješenje za detekciju visine tona zvuka i primjer primjene tog rješenja u obliku aplikacije za mobilne uređaje. Objašnjeni su problemi s kojima se susreće ovo područje digitalne obrade signala te kompromisi koje mora poduzeti kako bi uravnotežilo brzinu i prezicnost.

Ključne riječi: Digitalna obrada signala, FFT, frekvencijski spektar, harmonici, visina tona, aplikacija za štimanje gitare

Guitar tuner application for mobile devices

Abstract

This paper presents a solution for the detection of pitch and an example of its application as a mobile application. The problems faced by this field of digital signal processing and the trade-offs it must make to balance speed and precision are explained.

Keywords: Digital signal processing, FFT, frequency spectrum, harmonics, pitch, guitar tuner