

Aplikacija za mobilne uređaje za štimanje gitare

Fran Jelavić

Cilj završnog rada

- Predstaviti :
 - princip titranja žica na gitari
 - frekvencijski raspon gitare
 - frekvencijski spektar titranja žica gitare
- Projektirati **softversku aplikaciju** za detekciju frekvencija

Frekvencija i visina tona

- Titranje čestica zvučnog vala → frekvencija
 - Broj titraja (kompresija) čestica medija u jedinici vremena
- **Visina tona** (eng. *pitch*) ≈ **frekvencija**
 - Ljudska percepcija frekvencije
- Ljudi raspoznaju:
 - frekvencije od 20 Hz do 20 kHz
 - između dva zvuka s razlikom frekvencija od 7 Hz do 2 Hz

Frekvencija u glazbi

- Oktava

- Razlika između dvaju zvuka s omjerom frekvencija 2:1

- Kromatska ljestvica (eng. *chromatic scale*)

- Oktava podijeljena na 12 jednakih intervala → **poluton** (eng. *semitone*)
- Omjer frekvencija susjednih intervala $\sqrt[12]{2} \approx 1.0595$

C C[#] / D^b D D[#] / E^b E F F[#] / G^b G G[#] / A^b A A[#] / B^b B

Glazbene note

- **Nota** → reprezentacija visine tona
 - Skupina visine tona koje spadaju u isti razred (eng. *Pitch class*)
 - Razredi odvojeni po oktavama
- **Naziv note** (A, B, C, D, E, F, G)
 - Osma nota → oktava (zapisana kao prva)
- **Razred** (broj oktave)
 - Npr. E_3 je za oktavu viši od E_2
- Predznak **akcidental** (eng. *accidental*)
 - Povećilica \sharp (eng. *sharp*), snizilica \flat (eng. *flat*)
 - Dižu ili spuštaju notu za poluton

F_2^\sharp

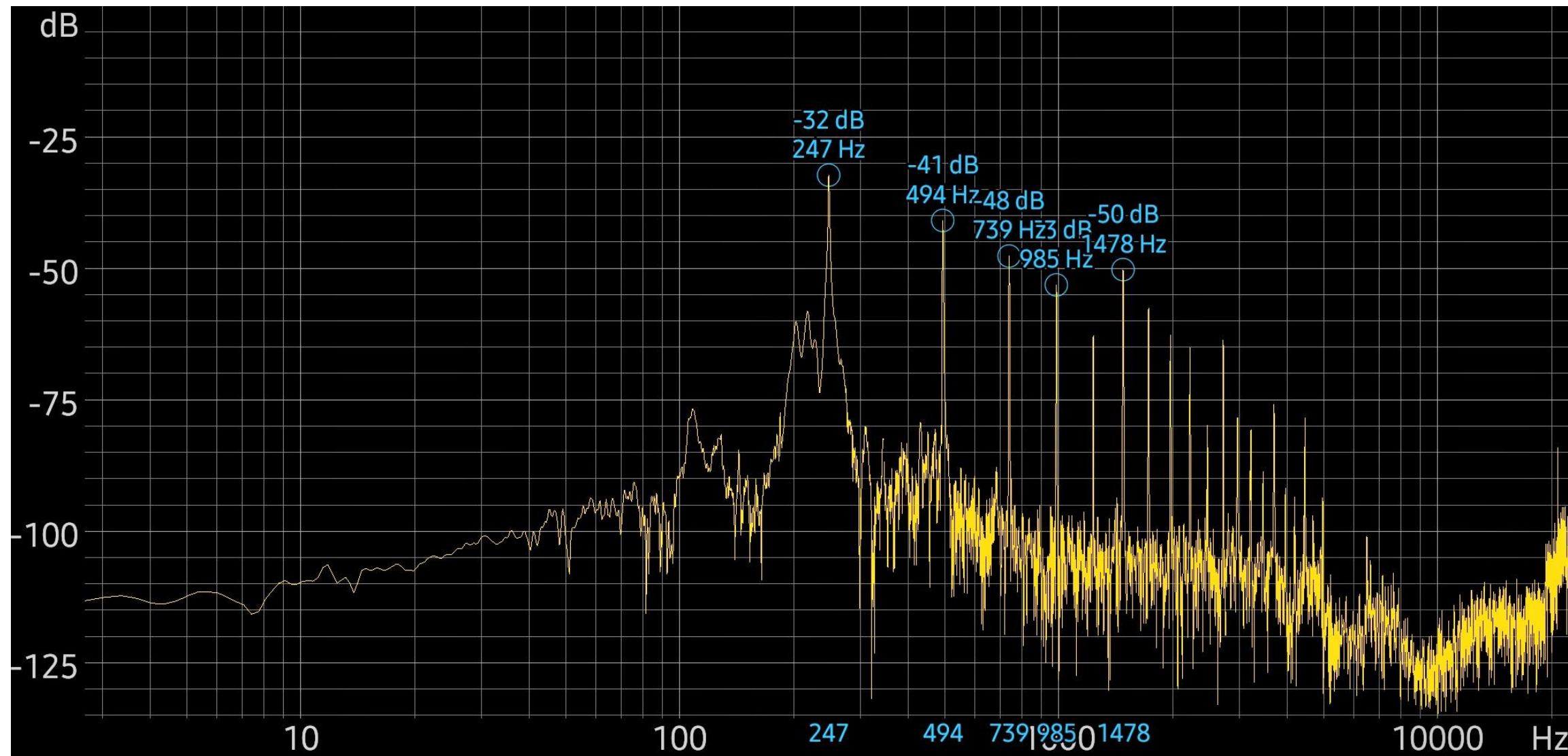
Raspon gitare

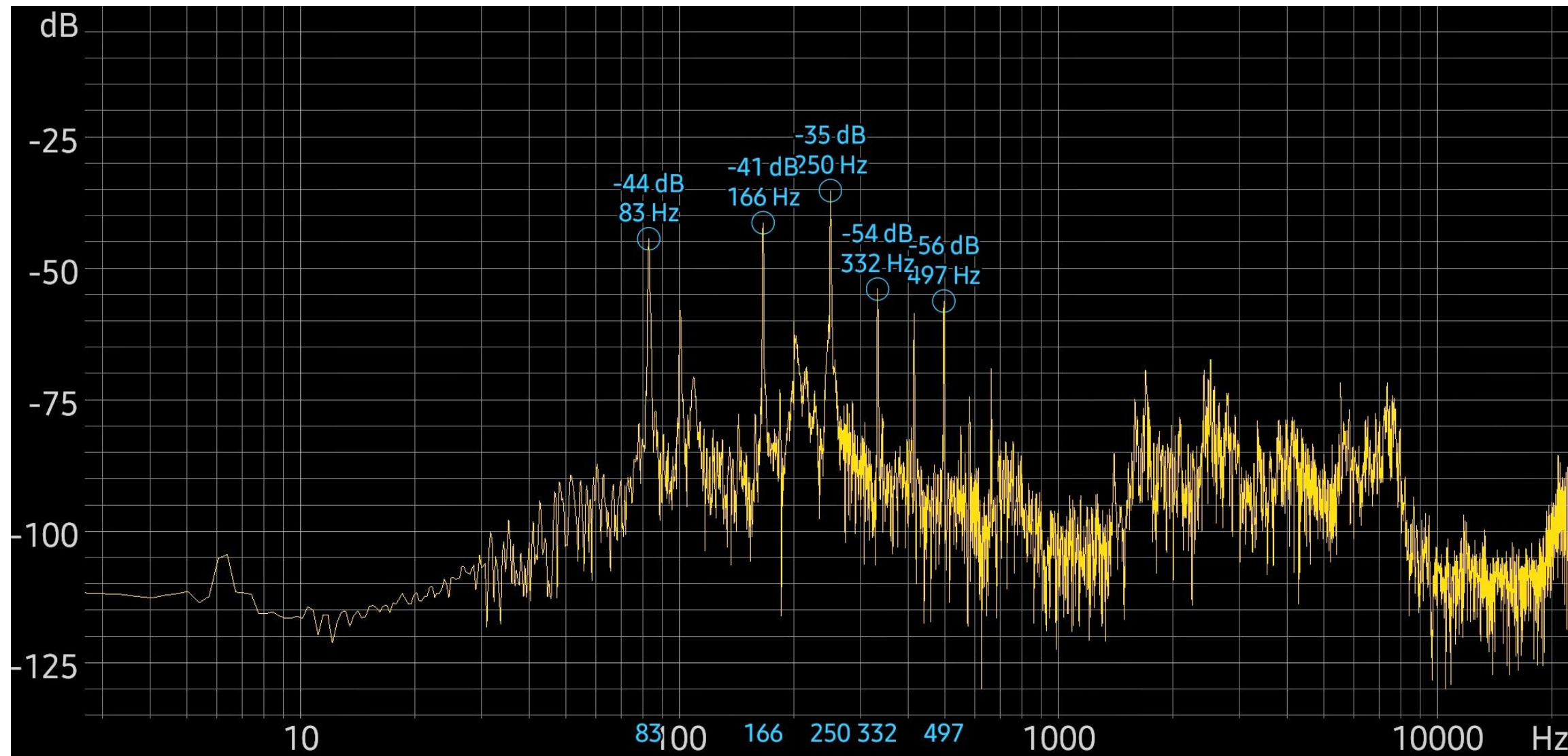
- **Koncertna visina tona**
 - Referenca za štimanje instrumenata
 - Standard $f(A_4) = 440 \text{ Hz}$
- **Način štimanja**
 - *Standard tuning*
 - $E_2 A_2 D_3 G_3 B_3 E_4$
- **Duljina vrata**
 - 19 do 24 polja (eng. *fret*)

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	

Harmonici

- **Temeljna frekvencija** i njezini **višekratnici**
- **Koeficijenti intenziteta frekvencija** → razlikuju se po izvoru zvuka
 - Popratne frekvencije (eng. *overtones*)
- **Objekt titra temeljnom frekvencijom**
 - Najčešće dominantna frekvencija (najvećeg intenziteta)





Digitalna obrada signala

- Snimljeni signali → **digitalizacija** → oblikovanje operacijama
 - Digitalizacija analogno-digitalnim pretvaračima (eng. *ADC*)
- Prednosti digitalizacije:
 - Kompresija podataka
 - Otkrivanje pogrešaka pri prijenosu
 - Ispravljanje pogrešaka pri prijenosu

Uzorkovanje (eng. *sampling*)

- Zvučni signal → slijed uzoraka
 - Vrijednost signala u točki u vremenu i/ili prostoru
- **Frekvencija uzorkovanja** (eng. *sample rate*)
 - Najčešće iznad 40 kHz
 - Teorem Nyquista i Shannona
 - Frekvencija uzorkovanja dvostruko veća od frekvencije signala koji se obrađuje
 - 44,1 kHz, 48 kHz, 88,2 kHz, 96 kHz
 - Povećanjem se povećava širina pojasa (eng. *bandwidth*)

Algoritam FFT

- Zvučni signal → **skup brojeva** kojim se raspoznaju najzastupljenije frekvencije u signalu
 - Svaki *bin* predstavlja određen raspon frekvencija
 - Veći broj u *bin*-u → veći intenzitet tog raspona frekvencija
- Raspon *bin*-a
 - **Rezolucija** = frekvencija uzorkovanja / veličina *buffer*-a
 - Raspoznavanje nota → **rezolucija** \leq **razlika frekvencija**

Aplikacija za štimanje gitare (eng. *tuner*)

- Ciljevi:
 - Detekcija visine tona odsvirane žice
 - Usmjeravanje korisnika prema visini tona najbliže note
- **Problem : FFT nije dovoljno dobar alat za ovakav zadatak**
 - Ne može se pouzdati na ispravno prepoznavanje temeljne frekvencije
 - YIN algoritam

Algoritam YIN

- FFT
- Metoda autokorelacije (eng. *autocorrelation function*)
 - Korelacija signala s odgođenom kopijom samog sebe kao funkcija kašnjenja
 - Služi za identificiranje temeljne frekvencije među harmonicima
- Mala učestalost grešaka
- Nisko kašnjenje (eng. *latency*)
- Prikladan za glazbu i visoke tonove

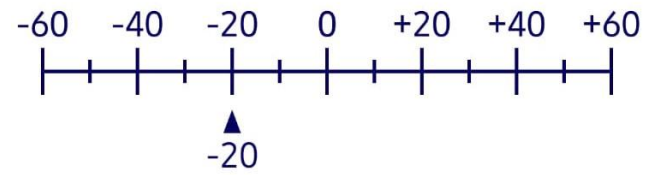
Aplikacija „ZRTuner”

- *IDE* Android Studio
 - Java
- TarsosDSP biblioteka (Java)
 - Implementacija algoritama za obradu zvuka
 - Bez vanjskih ovisnosti
 - Relativno jednostavna implementacija
- Sličan projekt : Gstraube - Cythara Tuner

ZRTuner



Chromatic



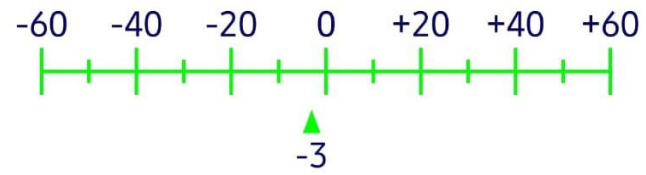
D[#]₆



ZRTuner



Chromatic



B₃



< Settings

Set reference pitch

440 Hz



Stvaranje fragmenta

```
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();
    }
}
```

Pozadinski proces

```
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();
                }
            }
        };
    }
};
```

Pozadinski proces

```
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;
}
```

Usporedba sa sličnim sustavima

- Tržište puno aplikacijama za štimanje gitare
 - Bez velike razlike, slično funkcioniranje, slične performanse
- Bez potrebe za unaprjeđenjem tehnologije
 - Dodavanje nepotrebnih ili dodatnih nepovezanih *feature-a*
 - Reklamacija → usporavanje cijelog procesa štimanja
 - Inspiracija za završni rad
- Usporedba s profesionalnim sustavima
 - Premale razlike u brzini i preciznosti za prosječne korisnike

Hvala!