BAB 3
ANALISIS DAN PERANCANGAN SISTEM

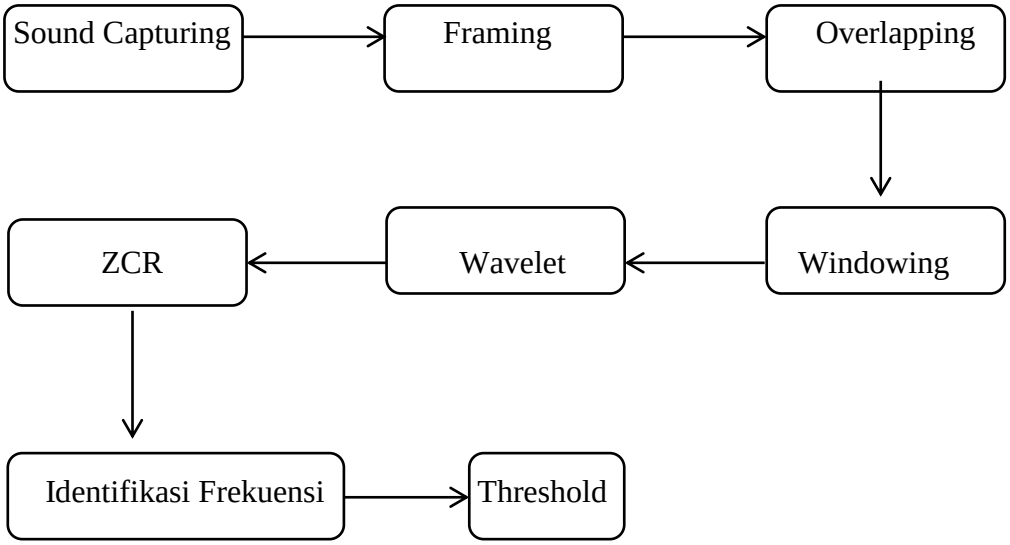
3.1 Analisis Masalah

Indera pendengaran manusia tidak dapat mengetahui secara pasti jenis nada apa yang didengar olehnya, terkecuali para pemusik profesional. Hal ini amatlah penting bagi seorang pemusik untuk mengetahui apakah gitarnya sudah menghasilkan nada-nada yang tepat. Menala gitar dapat dilakukan menggunakan alat penala gitar, namun harga penala gitar tidak terjangkau karena cukup mahal. Proses penalaan gitar menggunakan alat penala gitar bagi pemula aan menjadi sulit karena harus mengetahui terlebih dahulu kunci pada gitar dan pada saat proses penalaan gitar harus dilakukan ditempat yang hening dari keramaian agar tidak terjadinya *noise* atau suara-suara yang bukan dihasilkan oleh gitar tidak ikut terekam.

Aplikasi penala gitar berbasis *mobile* diperlukan untuk memberikan kemudahan bagi pemula dan lebih terjangkau. Aplikasi penala gitar berbasis *mobile* dapat melakukan penalaan nada gitar dengan menghitung sinyal digital. Hal ini membutuhkan suatu metode analisa sinyal digital yang mampu mengkompresi dan memfilter sinyal secara efisien.

3.2 Analisis Sistem Aplikasi

Analisis proses kerja aplikasi digunakan untuk mengetahui alur proses dari sebuah metode yang digunakan dapat diterapkan kedalam aplikasi yang dibangun. Aplikasi ini menggunakan metode *wavelet daubechies* untuk melakukan dekomposisi sinyal asli menjadi band-band frekuensi. Adapun gambaran umum aplikasi sebagai berikut :



Gambar 3.1 Proses Metode

a. *Sound Capturing*

   Sound capturing adalah proses awal menangkap sinyal yang dikeluarkan oleh suara gitar.

b. *Framing*

   Framing untuk mengelompokkan semua sinyal.

c. *Overlapping*

   Overlapping untuk pemotongan 75% semua *frame* yang telah menampung *sampling*.

d. *Windowing*

   Windowing untuk penghalus pada setiap *frame* dengan melakukan *overlapping* antara satu *frame* dengan *frame* yang lain

e. *Wavelet Daubechies*

   Wavelet daubechies adalah tahap-tahap perhitungan amplitude dengan nilai koefisien-koefisien *wavelet daubehcies* dB4.

f. *Zero Crossing Rate*

adalah manipulasi sederhana dari besarnya sinyal dapat menyediakan sejumlah property berguna.

g. *Threshold*

Threshold adalah menentukan nilai ambang batas agar bisa mengetahui ketepatan kunci yang akan ditampilkan.

3.2.1 Analisa Framing

Framing adalah pembagian sinyal audio menjadi beberapa frame yang nantinya dapat memudahkan dalam perhitungan dan analisa sinyal, satu frame terdiri dari beberapa sampel tergantung tiap berapa detik suara akan disampel dan berapa besar frekuensi samplingnya. Dalam contoh kasus ini ada enam *sampling* yang harus dikelompokkan (*framing)*. Proses *framing* adalah sebagai berikut :

● ● ● ● ● ● = 6 total *sampling*

● ● ● ● ● ●

Contoh kasus tersebut misalkan setiap *sampling* mempunyai nilai [2000, 1800, 2500, 1500, 3000, 1500], sebagai berikut :

2000 1800 2500 4000 3000 1500

Karena ada enam total *sampling* maka dikelompokkan (*framing*) menjadi dua *frame* yang terdiri dari tiga *sampling* dari setiap *frame*.

3.2.2 Analisis Overlapping

Overlapping adalah proses yang diawali dengan membandingkan panjang data yang akan dioverlapping dari maisng-masing frame dari beberapa frame yang akan digabungkan. Setelah itu akan dibandingkan dan dicari mana yang paling pendek dari keduanya akan dijadikan patokan dalam penentuan banyak data yang akan diproses *overlapping*. Proses *overlapping* 75% sebagai berikut :



Setelah proses *overlapping* secara otomatis jumlah *frame* akan bertambah, jumlah *frame* setelah proses *overlapping* adalah empat *frame*.

3.2.3 Analisis Windowing

Dalam melakukan pemrosesan sinyal, maka dari input yang dimasukkan akan terbentuk sinyal yang amplitudonya bervariasi pada awal maupun akhir *frame.* Hal tersebut dapat menghambat pemrosesan sinyal dan menghasilkan keluaran yang kurang akurat. Untuk itu perlu diaplikasikan suatu *window* penghalus pada setiap dengan melakukan *overlapping* antara satu *frame* dengan *frame* yang lain, sehingga dapat dibangkitkan suatu *feature* yang lebih halus sepanjang durasi waktu tersebut. Contoh kasus diatas setelah *overlapping*, setiap *sampling* pada *frame* akan diproses oleh fungsi *hamming windowing.* Prosesnya adalah sebagai berikut :

Proses perhitungan *sampling* pertama pada *frame pertama*.

$$w(n) = 0.54 - 0.46 \,.\, cos\left(\frac{2 \,.\, 3.14 \,.\, 2000}{3-1}\right) \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad (3.1)$$

$$w(n) = 0.54 - (-0.432)$$

$$w(n) = 0.54 + 0.432$$

$$w(n) = 0.972$$

Perhitungan tersebut terus dilakukan sampai *sample* ketiga pada setiap *frame*. Hingga setiap *sampling* pada *frame* mempunyai nilai sebagai berikut :

Frame 1 = 0.972, 0.682, 0.383

Frame 2 = 0.682, 0.383, 0.188

Frame 3 = 0.383, 0.188, 0.938

Frame 4 = 0.188, 0.938, 0.972

Setelah setiap *frame* mempunyai nilai *sample* yang sudah diproses oleh *hamming window*, setiap *frame* akan masuk pada proses *wavelet daubechies filter lowpass* dB4.

3.2.4 Analisis Metode Wavelet Daubechies

Transformasi *wavelet daubechies* merupakan metode yang digunakan untuk mengolah sinyal *nonstationer* dan untuk menganalisa biosignal yang selalu mengalami perubahan. Transformasi *wavelet daubechies* dapat memberikan representasi waktu dan frekuensi secara bersamaan sehingga dapat mempresentasikan sinyal yang dimaksud. Berikut contoh kasus transformasi *wavelet daubechies* pada *frame* pertama dengan tiga total *sample* dan koefisien *lowpass* dB4 :

X(n) = sinyal keluaran yang telah melewati *pre processing*.

= [0.972, 0.682, 0.383]

H(n) = nilai koefisien *lowpass* dB4

= [0.2, 0.7, 0.6, -0.02, -0.18, 0.03, 0.03, -0.01]

Konvolusi = X(n) * H(n) …… (3.2)

Tabel 3.1 Perkalian frame pertama dan nilai koefisien

H(n) / X(n)	**0.2**	**0.7**	**0.6**	**-0.02**	**-0.18**	**0.003**	**0.003**	**-0.01**
0.972	0.194	0.680	0.583	-0.019	-0.174	2.916	2.916	-9.72
0.682	0.136	0.477	0.409	-0.013	-0.122	2.046	2.046	-6.82
0.383	0.076	0.268	0.229	-7.66	-0.068	1.149	1.149	-3.83

Pada Tabel 3-1 adalah hasil perkalian dari nilai-nilai koefisien *lowpass* dB4 dengan nilai *sample* pada *frame* pertama yang telah masuk pada tahap *pre processing*. Perkalian tersebut terus dilakukan hingga *frame* keempat untuk mendapatkan nilai konvolusi. Nilai dari hasil perkalian tersebut akan dijumlahkan secara diagonal, penjumlahan nilai secara diagonal sebagai berikut :

Tabel 3.2 Penjumlahan nilai frame pertama secara diagonal

H(n) / X(n)	**0.2**	**0.7**	**0.6**	**-0.02**	**-0.18**	**0.003**	**0.003**	**-0.01**
0.972	0.194	0.680	0.583	-0.019	-0.174	2.916	2.916	-9.72
0.682	0.136	0.477	0.409	-0.013	-0.122	2.046	2.046	-6.82
0.383	0.076	0.268	0.229	-7.66	-0.068	1.149	1.149	-3.83

Diagonal pertama = 0.194

Diagonal kedua = 0.136 + 0.680 = 0.816

Diagonal ketiga = 0.076 + 0.477 + 0.583 = 1.136

Diagonal keempat = 0.268 + 0.409 + (-0.019) = 0.658

Diagonal kelima = 0.229 + (-0.013) + (-0.174) = 0.042

Diagonal keenam = -7.66 + (-0.122) + 2.916 = -4.866

Diagonal ketujuh = -0.068 + 2.046 + 2.916 = 4.894

Diagonal kedalapan = 1.149 + 2.046 + (-9.72) = -6.525

Diagonal kesembilan = 1.149 + (-6.82) = -5.671

Diagonal kesepuluh = -3.83

Jadi, hasil perhitungan *wavelet daubechies* setiap *frame* adalah sebagai berikut :

Frame 1 = [0.194, 0.816, 1.136, 0.658, 0.042, -4.866, 4.894, -6.525, -5.671, -3.83]

Frame 2 = [0.136, 0.553, 0.714, 0.347, -7.67, -1.782, 3.164, -0,031, 1.81, -1.88]

Frame 3 = [0.136, 0.484, 0.727, 0.755, -3.32, 1.995, 7.518, 1.634, 0.934, -9.38]

Frame 4 = [0.037, 0.318, 0.962, -2.518, 0.532, 5.453, 8.28, 3.85, -6.464, -9.72]

3.2.5 Analisis Zero Crossing Rate (ZCR)

Zero crossing rate adalah manipulasi sederhana dari besarnya sinyal dapat menyediakan sejumlah property berguna. Beberapa pengukuran ini secara tradisional termasuk energi, rata-rata ukurannya, *zero crossing*, dan fungsi otokorelasi. Properti ini dapat digunakan untuk analisa yang lebih kompleks, seperti estimasi lapangan. Dalam konteks sinyal diskrit, *zero crossing* dikatakan terjadi jika sampel mempunyai tanda aljabar yang berbeda. Ini berarti bahwa ada hubungan antara *zero crossing* dan konten frekuensi sinyal. Formula zero crossing rate adalah sebagai berikut :

$$\mathrm{ZCR} = \frac{1}{2N}\sum_{n=1}^{N} |sgn\ x(n) - sgn\ x(n-1)| \quad \ldots\ldots\ldots \quad (3.3)$$

Pada contoh kasus *wavelet daubechies* setiap *frame* yang telah mendapatkan hasil penjumlahan secara diagonal akan diproses oleh formula *zero crossing rate*, sebagai berikut :

Hasil *wavelet daubechies* pada *frame* pertama adalah [0.194, 0.816, 1.136, 0.658, 0.042, -4.866, 4.894, -6.525, -5.671, -3.83]

Perhitungan tersebut dilakukan dari *frame* pertama sampai dengan *frame* keempat, hingga hasil perhitungan setiap *frame* sebagai berikut :

Frame 1 = 1.540

Frame 2 = 1.426

Frame 3 = 1.621

Frame 4 = 1.660

Nilai dari setiap *frame* diatas akan dinormalisasi, sebagai berikut :

$$Z = \frac{1.540 + 1.426 + 1.621 + 1.660}{4} = 1.561$$

Jadi, hasil perhitungan dari tahap-tahap yang terurai diatas menghasilkan nilai frekuensi 1.561Hz.

3.2.6 Identifikasi Frekuensi

Identifikasi frekuensi yaitu dengan cara membandingkan selisih terkecil dari hasil pengurangan antara x dengan y, formula untuk identifikasi kunci adalah sebagai berikut :

$$|x - y|$$

Diamana :

x adalah frekuensi yang telah ditetapkan.

y adalah frekuensi yang didapat.

Tabel 3.3 Frekuensi

	0	1	2	3	4	5	6	7	8
C	16.35	32.70	65.41	130.81	261.63	523.25	1046.50	2093.00	4186.01
C#	17.32	34.65	69.30	138.59	277.18	554.37	1108.73	2217.46	4434.92
D	18.35	36.71	73.42	146.83	293.66	587.33	1174.66	2349.32	4698.64
D#	19.45	38.89	77.78	155.56	311.13	622.25	1244.51	2489.02	4978.03
E	20.60	41.20	82.41	164.81	329.63	659.26	1318.51	2637.02	5274.04
F	21.83	43.65	87.31	174.61	349.23	698.46	1396.91	2793.83	5587.65
F#	23.12	46.25	92.50	185.00	369.99	739.99	1479.98	2959.96	5919.91
G	24.50	49.00	98.00	196.00	392.00	783.99	1567.98	3135.96	6271.93
G#	25.96	51.91	103.83	207.65	415.30	830.61	1661.22	3322.44	6644.88
A	27.50	55.00	110.00	220.00	440.00	880.00	1760.00	3520.00	7040.00
A#	29.14	58.27	116.54	233.08	466.16	932.33	1864.66	3729.31	7458.62
B	30.87	61.74	123.47	246.94	493.88	987.77	1975.53	3951.07	7902.13

Formula identifikasi frekuensi untuk menampilkan kunci gitar adalah sebagai berikut :

$$= |x - y|$$

$$= |16.35 - 1.561|$$

$$= |14.78|$$

Hasil pengurangan frekuensi C kolom 0 adalah 14.78Hz, langkah ini dilakukan untuk semua frekuensi. Pada tabel 3.4 adalah hasil akhirnya :

Tabel 3.4 Hasil identifikasi frekuensi

	0	1	2	3	4	5	6	7	8
C	14.78	31.13	63.84	129.24	260.06	521.68	1044.93	2091.43	4184.44
C#	15.75	34.49	67.73	137.02	275.61	552.80	1107.16	2215.89	4433.35
D	16.78	35.14	71.85	145.26	292.09	585.76	1173.09	2347.75	4697.07
D#	17.88	37.32	76.21	153.99	309.56	620.68	1242.94	2487.45	4976.46
E	19.03	39.63	80.84	163.24	328.06	657.69	1316.94	2635.45	5272.47
F	20.26	42.08	85.74	173.04	347.66	696.89	1395.34	2792.26	5586.08
F#	21.56	44.68	90.93	183.43	368.42	738.42	1478.41	2958.39	5918.34
G	22.93	47.43	96.43	194.43	390.43	782.42	1566.41	3134.39	6270.36
G#	24.39	50.34	102.26	206.08	413.73	829.04	1659.65	3320.87	6643.31
A	25.93	53.43	108.43	218.43	438.43	878.43	1758.43	3518.43	7038.43
A#	27.57	56.70	114.97	231.51	464.59	930.76	1863.09	3727.74	7457.05
B	29.30	60.17	121.90	245.37	492.31	986.20	1973.96	3949.50	7900.56

Tabel 3.4 frekuensi diatas adalah hasil pengurangan antara frekuensi acuan dengan frekuensi 1.651Hz yang dikeluarkan oleh senar gitar. Frekuensi yang terkecil pada tabel 3.4 adalah 14.78Hz yang terletak pada kunci C nada 0.

3.2.7 Analisis Threshold

Tahapan selanjutnya adalah *thresholding* atau penentuan nilai ambang batas untuk memastikan kunci apa yang terdeteksi. Proses penentuan nilai *threshold* didapatkan melalui proses *trial* and *error* pada hasil identifikasi frekuensi. Nilai *Threshold* adalah sebagai berikut :

a. *Threshold* = <3 jika frekuensi kurang dari 3 maka akan terdeteksi.
b. *Threshold* = >3 jika frekuensi lebih dari 3 maka tidak akan terdeteksi.

Penentuan nilai *threshold* adalah <3, maka frekuensi 14.78Hz pada tabel 3.4 belum memenuhi nilai *threshold* yang telah ditentukan sehingga harus melakukan

penalaan ulang agar menampilkan kunci yang tepat dan memenuhi nilai *threshold* yang telah ditentukan.

3.3 Spesifikasi Kebutuhan Perangkat Lunak

Spesifikasi perangkat lunak adalah analisis spesifikasi perangkat lunak yang akan dibangun. Spesifikasi perangkat lunak dibutuhkan untuk menetapkan layanan apa yang akan dituntut pada oprasi dan pengembangan perangkat lunak. Berikut spesifikasi perangkat lunak yang akan dibangun :

Tabel 3.5 Kebutuhan Non-Fungsional

Nomor	Keterangan
SKPL-NF-01	Sistem dibangun dalam bentuk aplikasi *mobile* Android
SKPL-NF-02	Sistem dapat dijalankan pada ponsel android minimal versi 3.0 (*HoneyCom*).
SKPL-NF-03	Minimal memori 500mb

Tabel 3.6 Kebutuhan Fungsional

Nomor	Keterangan
SKPL-F-01	Aplikasi dapat mendeteksi suara.
SKPL-F-02	Aplikasi dapat menampilkan kunci.

3.4 Analisis Kebutuhan non-Fungsional

Analisis kebutuhan non-fungsionalitas merupakan analisis yang dibutuhkan untuk menentukan spesifikasi yang diperlukan. Spesifikasi ini juga meliputi elemen atau komponen-komponen apa saja yang dibutuhkan untuk aplikasi yang akan dibangun. Kebutuhan non-fungsionalitas yang dibutuhkan untuk

membangun aplikasi ini terdiri dari tiga hal, yaitu : analisis pengguna, analisis kebutuhan perangkat keras dan analisis kebutuhan perangkat lunak.

3.4.1 Analisis Pengguna

Aplikasi yang dirancang hanya untuk pengguna yang ingin menggunakan aplikasi ini untuk menala gitar, dengan kebutuhan spesifikasi pengguna sebagai berikut :

Spesifikasi	**:**Pengguna
Hak Akses	**:**Menala gitar
Tingkat Keterampilan	**:**Dapat mengoperasikan *mobile* berbasis android dan dapat mengoperasikan gitar.
Usia	**:**10^{th} +

3.4.2 Analisis Kebutuhan Perangkat Keras

Perangkat keras merupakan salah satu kebutuhan yang paling penting untuk pembangunan aplikasi ini. Perangkat keras akan mempengaruhi kinerja pembangunan aplikasi ini, semakin tinggi spesifikasi dari perangkat keras yang digunakan maka akan semakin cepat proses pembangunan apliaksinya. Perangkat keras yang digunakan pada pembangunan aplikasi ini yaitu seperti pada Tabel 3.7.

Tabel 3.7 Spesifikasi Perangkat Keras

No	Nama Perangkat	Spesifikasi
1	Processor	Intel Core i3 1.40 GHz
2	Monitor	Monitor 14 inch (1366 x 768)
3	Memori	RAM 2 GB DDR3
4	Hardisk	320 GB SATA 7200 RPM

3.4.3 Analisis Kebutuhan Perangkat Lunak

Perangkat lunak yang digunakan untuk membangun aplikasi ini seperti pada Tabel 3.8.

Tabel 3.8 Spesifikasi Perangkat Lunak

No	Nama Perangkat	Spesifikasi
1	Sistem Operasi	Windows 8 Pro 64 bit
2	JDK (Java Development Kit)	JDK 1.7.0
3	JRE (Java Runtime Environment)	JRE 7
4	Eclipse	Juno
5	ADT	22.2.1

Sedangkan minimal kebutuhan perangkat lunak pada *mobile* android untuk menjalankan aplikasi ini seperti pada Tabel 3.9.

Tabel 3.9 Spesifikasi Minimal Kebutuhan Perangkat Lunak

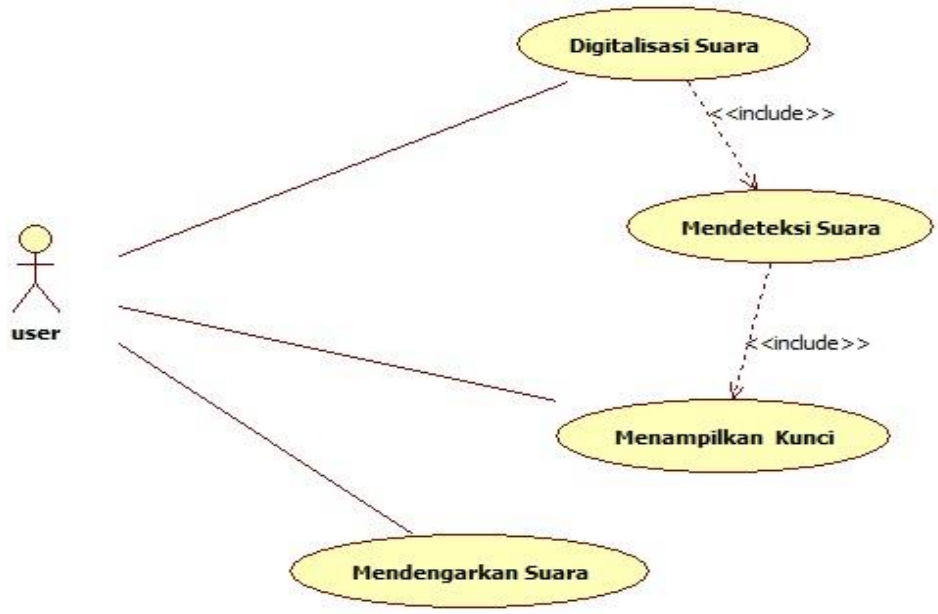
No	Nama Perangkat	Spesifikasi
1	Sistem Operasi	Android 2.2 Froyo
2	Memori	384 MB
3	Monitor	240 x 320 pixels
4	Hardisk	160 MB

3.5 Analisis Kebutuhan Fungsional

Analisis kebutuhan fungsional adalah segala bentuk data yang dibutuhkan oleh aplikasi ini agar dapat berjalan sesuai dengan prosedur yang dibangun. Aplikasi yang dibangun akan dimodelkan menggunakan *unified modeling language* (UML) dan *tools* yang akan digunakan yaitu *use case* diagram, *activity* diagram, *sequence* diagram dan *class* diagram.

3.5.1 Use Case Diagram

Use case diagram digunakan untuk mengetahui apa saja yang dapat dilakukan oleh pengguna terhadap fungsionalitas yang yang terdapat pada aplikasi yang dibangun. *Use case* diagram pada aplikasi ini terlihat pada gambar 3.2.



Gambar 3.2 Use case diagram aplikasi penala gitar

3.5.2 Skenario Use Case

Skenario *use case* merupakan penjelasan secara rinci setiap *use case* yang terdapat pada diagram *use case*. Skenario *use case* pada aplikasi ini yaitu sebagai berikut:

3.5.2.1 Skenario Digitalisasi Suara

Skenario digitalisasi sebagai berikut :

Tabel 3.10 Skenario Digitalisasi Suara

Use case name	Digitalisasi Suara
Actor	*User*
Use case description	Digunakan untuk mengubah sinyal analog menjadi sinyal digital
Precondition	*User* harus melakukan tuning gitar
Trigger	*Button tuning*
Scenario :	
1. *User* melakukan *tuning* gitar 2. Aplikasi akan melakukan digitalisasi suara gitar	
Exception :	

3.5.2.2 Skenario Mendeteksi Suara

Skenario mendeteksi suara sebagai berikut :

Tabel 3.11 Skenario Mendeteksi Suara

Use case name	Mendeteksi Suara
Use case description	Digunakan untuk mendeteksi suara digitalisasi
Precondition	Suara digital
Trigger	
Scenario :	
1. Aplikasi konvolusi *sampling* menggunakan *wavelet daubechies* 2. Aplikasi normalisasi *sampling* menggunakan *zero crossing rate*	
Exception :	

3.5.2.3 Skenario Menampilkan Suara

Skenario menampilkan suara sebagai berikut :

Tabel 3.12 Skenario Menampilkan Suara

Use case name	Menampilkan Suara
Actor	*User*
Use case description	Aplikasi menampilkan kunci pada *user*
Precondition	Aplikasi mendapatkan frekuensi
Trigger	
Scenario :	
1. Aplikasi menampilkan kunci	
Exception :	

3.5.2.4 Skenario Mendengarkan Suara

Skenario mendengarkan suara sebagai berikut :

Tabel 3.13 Skenario Mendengarkan Suara

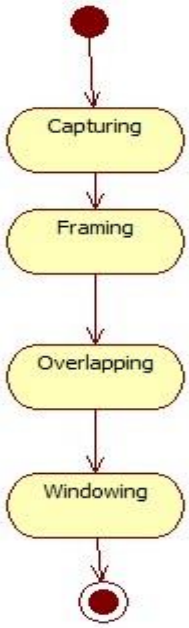
Use case name	Mendengarkan Suara
Actor	*User*
Use case description	Aplikasi menampilkan suara kunci
Precondition	Menampilkan form *key*
Trigger	*Button Key*
Scenario :	
1. Aplikasi menampilkan kunci	
Exception :	

3.5.3 Activity Diagram

Activity diagram menggambarkan berbagai alur aktivitas dalam sistem yang sedang dirancang, bagaimana masing-masing alur berawal, *decision* yang mungkin terjadi dan bagaimana mereka berakhir. *Activity* diagram yang terdapat pada aplikasi ini yaitu sebagai berikut :

3.5.3.1 Activity Diagram Digitalisasi Suara

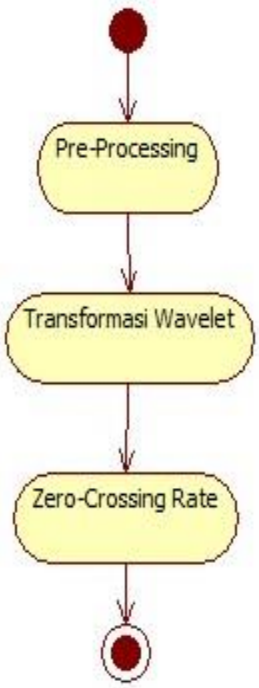
Activity diagram digitalisasi suara menggambarkan alur aktivitas *capturing* sinyal analog kemudian dirubah menjadi sinyal digital. Sinyal yang telah dirubah menjadi sinyal digital akan dideteksi nilai-nilai *sampling*, dapat dilihat pada gambar 3.3 :



Gambar 3.3 Activity Diagram Digitalisasi Suara

3.5.3.2 Activity Diagram Mendeteksi Suara

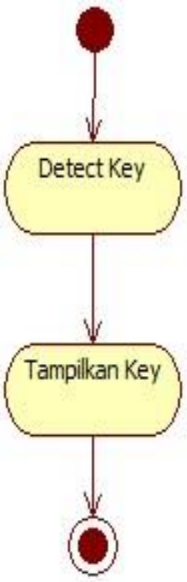
Activity diagram mendeteksi suara menggambarkan alur nilai-nilai *sampling* yang telah terdeteksi akan masuk pada proses *filtering*, seperti pada gambar 3.4 :



Gambar 3.4 Activity Diagram Mendeteksi Suara

3.5.3.3 Activity Diagram Menampilkan Suara

Activity diagram menampilkan suara menggambarkan alur nilai-nilai *sampling* yang telah diproses *filtering* akan mendapatkan nilai frekuensi, seperti pada gambar 3.5 :



Gambar 3.5 Activity Diagram Menampilkan Suara

3.5.3.4 Activity Diagram Mendengarkan Suara

Activity diagram mendengarkan suara yaitu alur jika user ingin mendengarkan suara kunci A sampai dengan kunci G, seperti pada gambar 3.6 :



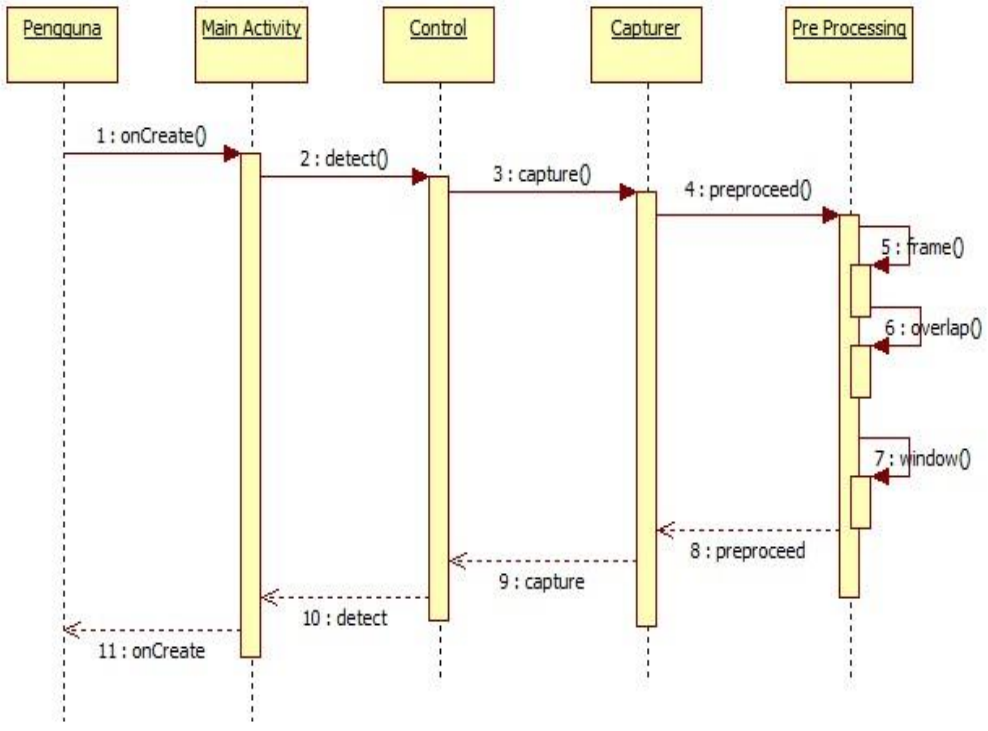
Gambar 3.6 Activity mendengarkan suara

3.5.4 Sequence Diagram

Sequence diagram menggambarkan interaksi antar objek didalam dan disekitar sistem termasuk pengguna, *display* dan sebagainya yang digambarkan terhadap waktu. *Sequence* diagram yang terdapat pada aplikasi ini yaitu sebagai berikut :

3.5.4.1 Sequence Diagram Digitalisasi Suara

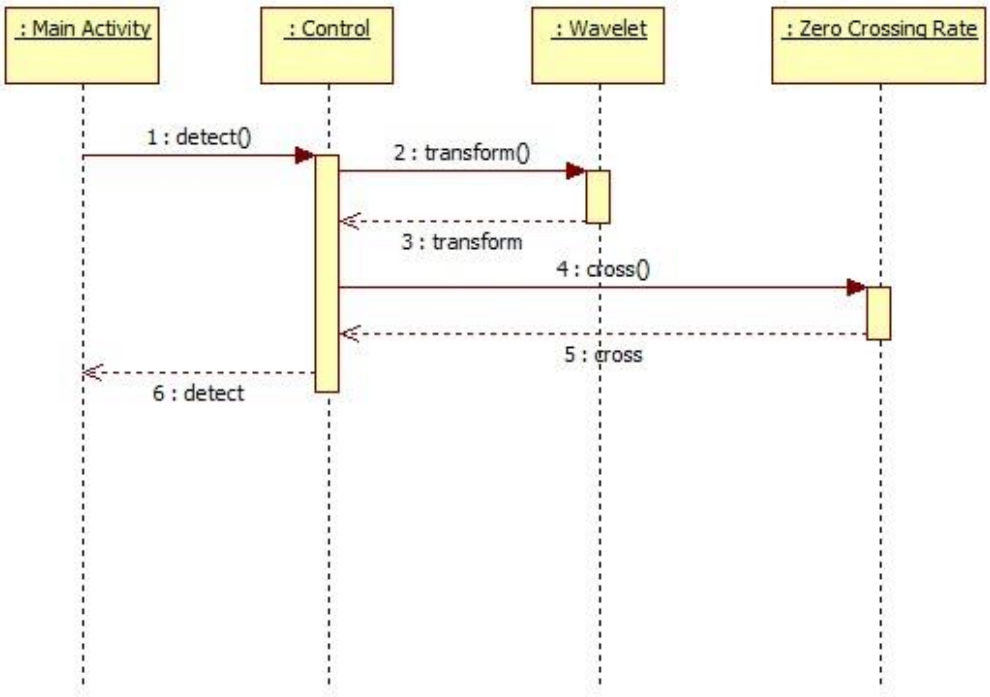
Sequence diagram digitalisasi suara menggambarkan interaksi pengguna dengan suara yang akan diproses digitalisasi, seperti pada gambar 3.7 :



Gambar 3.7 Sequence Diagram Digitalisasi Suara

3.5.4.2 Sequence Diagram Mendeteksi Suara

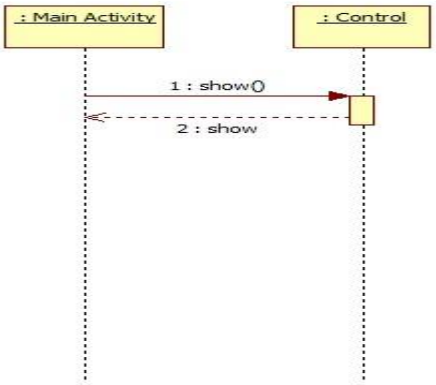
Sequence diagram mendeteksi suara menggambarkan alur sistem saat *filtering* nilai-nilai *sampling* yang terdeteksi, seperti pada gambar 3.8 :



Gambar 3.8 Sequence Diagram Mendeteksi Suara

3.5.4.3 Sequence Diagram Menampilkan Suara

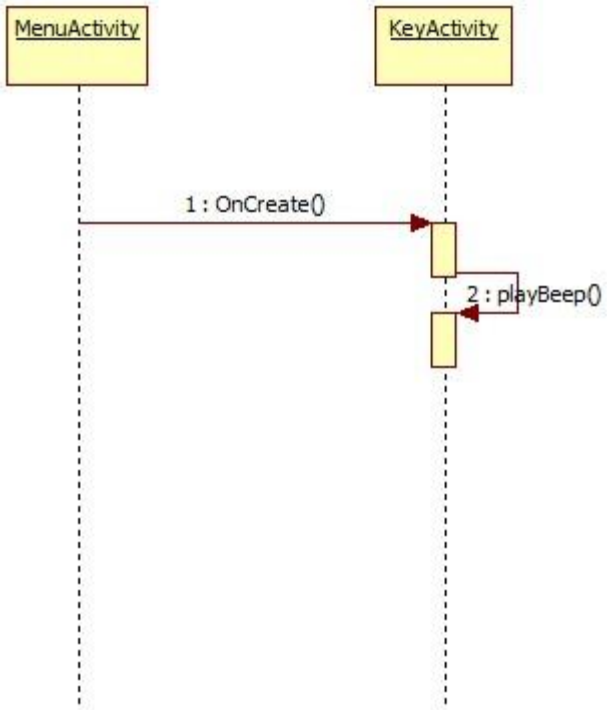
Sequence diagram menampilkan suara menggambarkan sistem untuk menampilkan suara pada *user*, seperti pada gambar 3.9 :



Gambar 3.9 Sequence Diagram Menampilkan Suara

3.5.5 Sequence Diagram Mendengarkan Suara

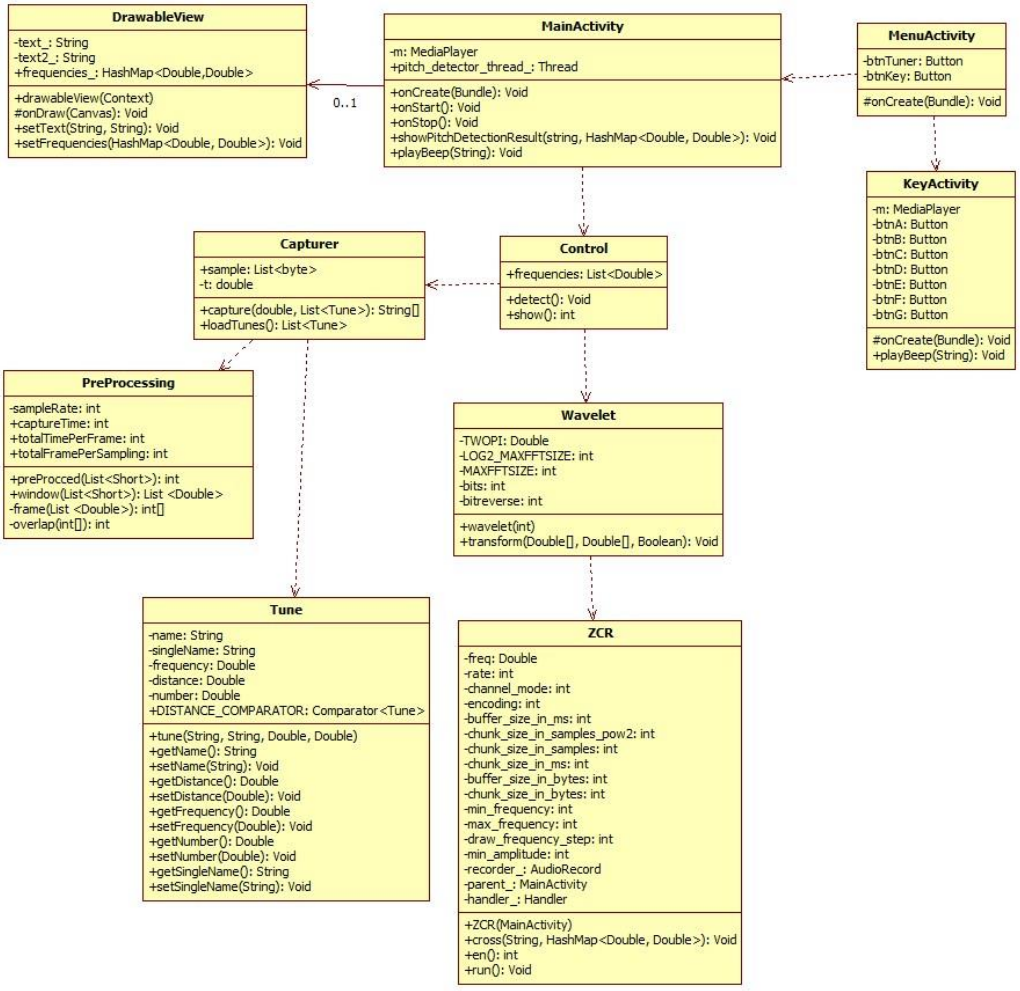
Sequence diagram mendengarkan suara menggambarkan sistem untuk user yang ingin mendengarkan suara kunci A sampai dengan kunci G.



Gambar 3.10 Sequence Diagram Mendengarkan Suara

3.5.6 Class Diagram

Class diagram digunakan untuk menggambarkan keadan suatu sistem dengan menjelaskan keterhubungan antara suatu *class* dengan *class* yang lain yang terdapat pada sistem, seperti pada gambar 3.11 :



Gambar 3.11 Class Diagram

Tabel 3.14 Jenis Class

Nama Class	Jenis Class
Main Activity	Boundary
Control	Boundary
Capturer	Entity
Pre Processing	Control
Wavelet	Control
Zero Crossing rate	Control
Menu Activity	Boundary
Tune	Control
Key Activity	Control
Drawable View	Boundary

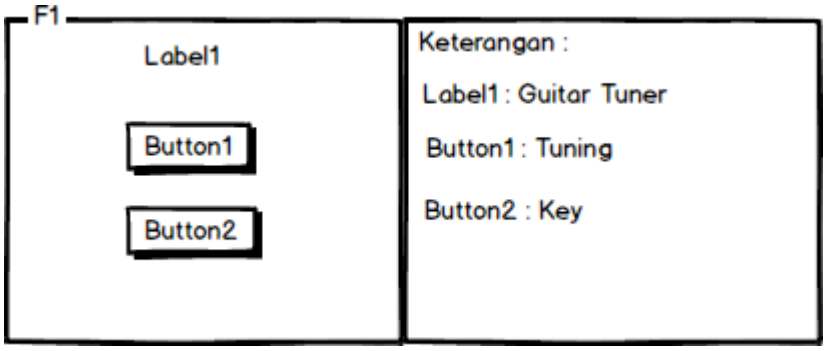
3.6 Perancangan Sistem

Perancangan sistem merupakan suatu proses yang menggambarkan bagaimana suatu sistem dibangun untuk memenuhi kebutuhan pada fase analisis.

3.6.1 Perancangan Antar Muka

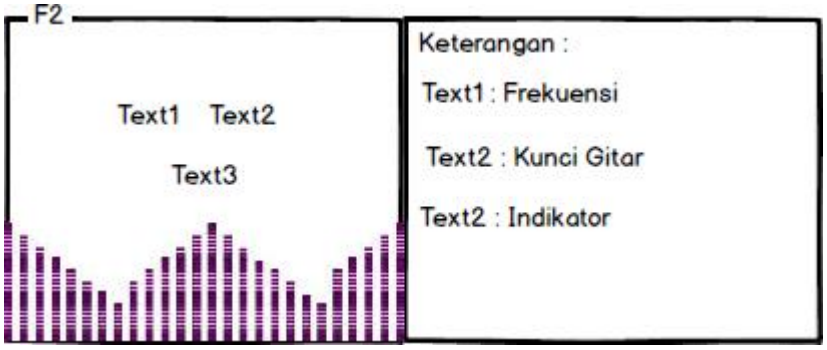
Perancangan antarmuka adalah tahapan pembuatan rancangan antarmuka untuk digunakan pada pembangunan aplikasi penala gitar yang dibagi menjadi tiga bagian yaitu antarmuka menu utama, antarmuka *detect* dan antarmuka tutorial, sebagai berikut :

a. Perancangan Antar Muka Menu Utama



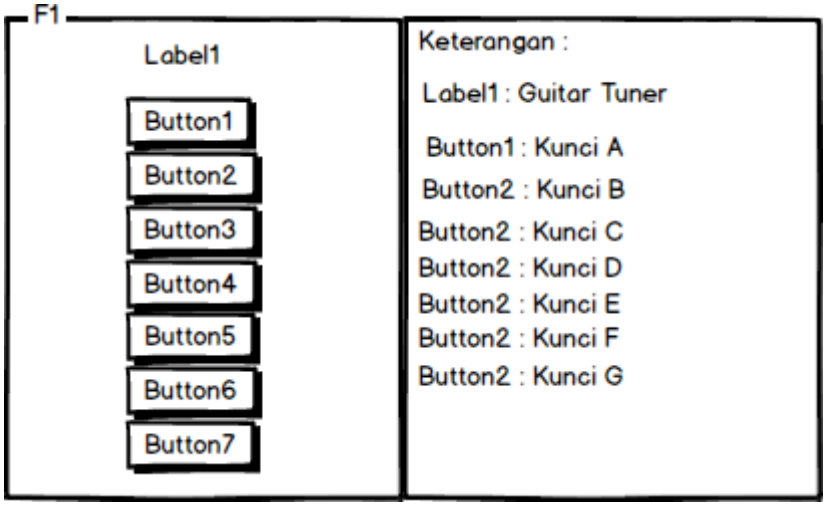
Gambar 3.12 Antar Muka Menu Utama

b. Perancangan Antar Muka *Tuning*



Gambar 3.13 Antar Muka Tuning

c. Perancangan Antar Muka *Key*



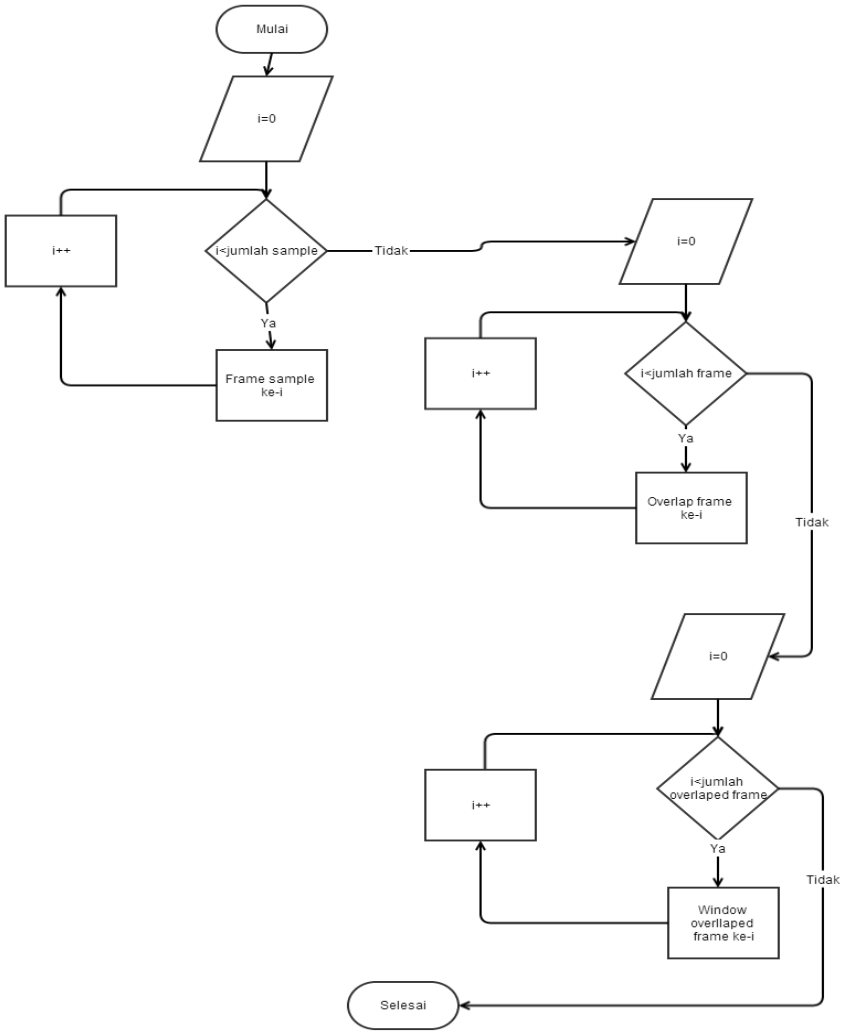
Gambar 3.14 Antar Muka Key

3.6.2 Perancangan Method

Perancangan komponen method merupakan perancangan yang dibuat setelah perancangan antarmuka. Perancangan ini berfungsi untuk mendeskripsikan method yang berada di dalam aplikasi penala gitar. Adapun perancangan method yang terdapat dalam aplikasi penala gitar yang dibangun adalah sebagai berikut :

a. Perancangan Method preProcced()

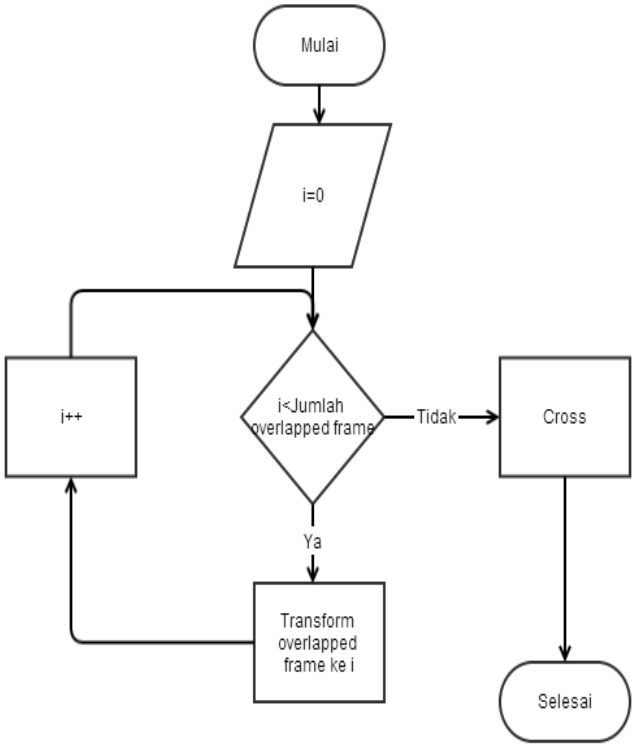
   Perancangan method preProcced() pada menu *tuning* dipanggil untuk digitalisasi sinyal.



Gambar 3.15 Perancangan Method preProcced()

b. Perancangan Method Transform() dan Cross()

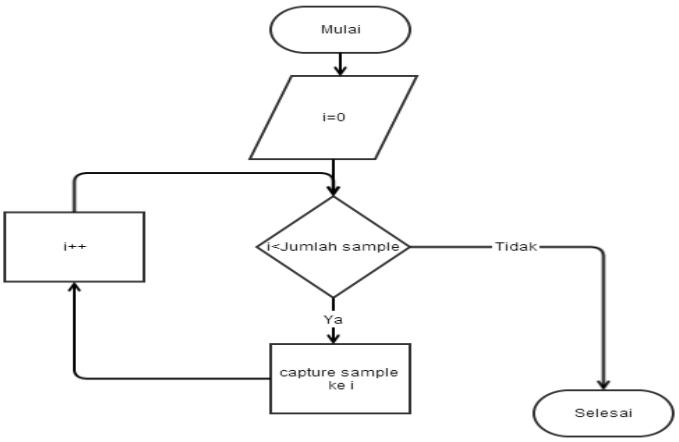
Perancangan method transform() dan Cross() pada menu *tuning* dipanggil untuk *filtering* sinyal dan untuk mendeteksi frekuensi.



Gambar 3.16 Perancangan Method Transform() dan Cross()

c. Perancangan Method playBeep()

Perancangan method playBeep() pada menu *chord* dipanggil untuk mendengarkan suara kunci gitar.



Gambar 3.17 Perancangan Method playBeep()