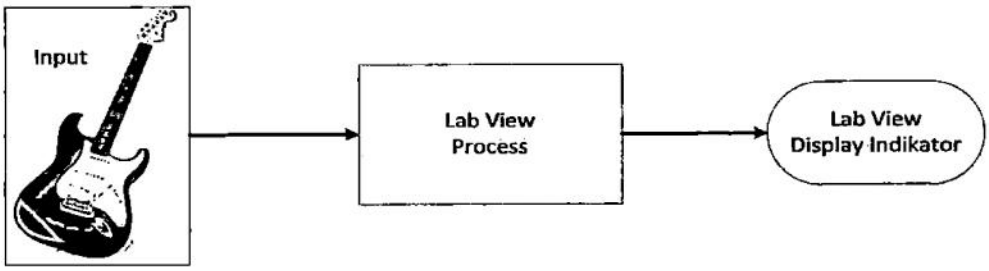
# BAB III
# PERANCANGAN DAN PEMBUATAN

## 3.1 Perancangan Penala Gitar pada Lab View

Sistem penala gitar, sebagaimana sistem pengolahan sinyal pada umumnya, terdiri atas Input, Proses dan Output seperti ditunjukkan pada diagram blok berikut:




**Gambar 3.1** Diagram blok sistem penala gitar

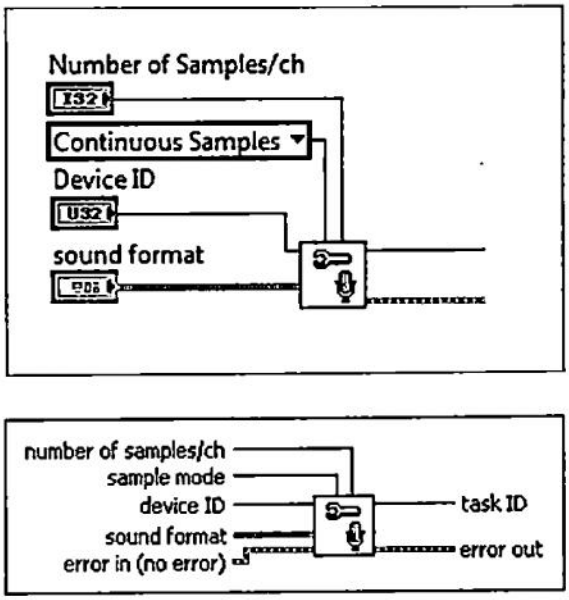
Input di sini berupa getaran dawai gitar yang menghasilkan nada (gelombang suara dengan frekuensi tertentu), akan dideteksi oleh sistem yang dibuat pada LabVIEW untuk diolah dan ditampilkan hasilnya berupa informasi mengenai nada tersebut.

## 3.2 Rancangan Penala Gitar pada Lab View

Rancangan penala gitar pada LabVIEW dibagi menjadi 3 bagian, yaitu rangkaian input, rangkaian proses, dan rancangan output.

### 3.2.1 Rangkaian Input

Rangkaian input difungsikan sebagai penerima sinyal gelombang suara yang masuk melalui *sound card* dari komputer yang kemudian dikonversi menjadi data dengan beberapa komponen input yang terdapat pada *library* LabVIEW.




**Gambar 3.2** Diagram blok rangkaian Input

Rangkaian input terdiri atas beberapa komponen pada *library* LabVIEW, yaitu:

1. *Sound Input Configure VI* berfungsi untuk mengkonfigurasi perangkat input suara untuk memperoleh data dan mengirim data ke buffer.

2. *Number of Samples/ch* berfungsi untuk menentukan jumlah sampel per channel dalam *buffer*. Untuk operasi yang kontinyu maka diperlukan jumlah sampel yang banyak.

3. *Sample Mode* berfungi untuk menentukan apakah VI memperoleh sampel hanya sekali (*Finite Samples*) atau terus menerus (*Continuous Samples*). Dalam mode *Finite Samples*, sound input dibaca hanya sampai pada jumlah

sampel yang ditentukan pada *Number of Samples/ch.* Dalam mode *Continuous Samples*, sound input akan dibaca berulang kali.

4. *Device ID* adalah perangkat input atau output untuk akses operasi suara. Nilai *default device ID* adalah 0. Nilai berkisar dari 0 sampai n-1, dimana n adalah jumlah input atau perangkat output pada komputer.

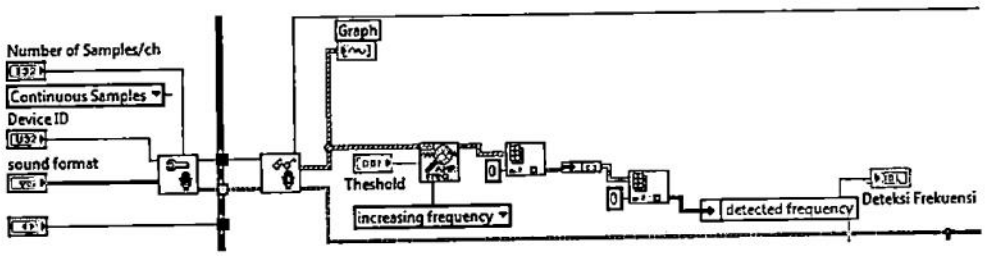
5. *Sound Format* berfungsi menetapkan nilai yang didapat, jumlah channel, dan bit per sampel dari operasi suara. Nilai untuk masing-masing kontrol ini tergantung *sound card* pada komputer.

6. *Task ID* berfungsi untuk kembali mengidentifikasi jumlah yang terkait dengan konfigurasi pada perangkat tertentu.

7. *Error Out* berfungsi untuk menyampaikan informasi tentang kesalahan. Output ini memberikan standard *Error Out* fungsionalitas.

### 3.2.2 Rangkaian Proses

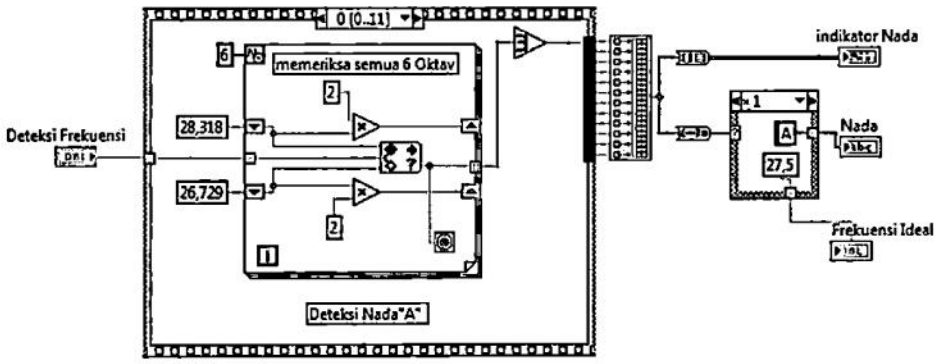
Pada rangkaian proses ini, semua diagram rangkaian dibingkai dalam struktur *While Loop* yang berfungsi agar rangkaian itu dapat berjalan berulang-ulang. Data yang dikirim dari rangkaian input akan dibaca oleh *Sound Input Read VI*, dimana data yang masuk akan divisualisasikan sebagai bentuk gelombang dengan *Waveform Graph VI* sebagai indikator sinyal input. Sinyal input yang masih dalam domain waktu akan diubah ke domain frekuensi oleh *Extract Multiple Tone Information VI* yang kemudian hasilnya akan disimpan di *index array* agar frekuensi input dapat dideteksi.




**Gambar 3.3** Diagram blok proses data input

Untuk mendeteksi frekuensi yang masuk, maka digunakan subVI yang dirancang untuk mendeteksi frekuensi nada dan menampilkan nada yang didapat dari sinyal input dengan beberapa indikator dan sub VI yang dirancang untuk menentukan akurasi nada pada frekuensi ideal dan ditampilkan dengan indikator.

Data frekuensi yang didapat dari input akan melewati sebuah sturktur *stacked sequence* yang berfungsi untuk mengeksekusi beberapa subdiagram rangkaian secara berurutan dan *for loop* untuk mengeksekusi subdiagram rangkaian untuk didibandingkan dengan menggunakan *in range and source function* yang menentukan apakah frekuensi input berada dalam kisaran batas maksimal dan batas minimal frekuensi input yang memaksa nilai jatuh dalam jangkauan, kemudian nilai tersebut disimpan dengan *build array* untuk ditampilkan sebagai keluaran yang berupa indikator nada.




**Gambar 3.4** Diagram blok proses deteksi frekuensi

Dari keluaran subVI untuk mendeteksi frekuensi nada, selanjutnya adalah menentukan apakah frekuensi nada tersebut sudah ideal dengan frekuensi nada yang telah ditetapkan dengan subVI untuk menentukan frekuensi ideal.

Sinyal input yang telah dideteksi akan dideteksi kembali dan diukur apakah frekuensi tersebut sudah sesuai dengan frekuensi ideal atau belum, dengan adanya toleransi maka akurasi dalam pengukuran dapat ditingkatkan untuk mendekati nilai frekuensi idealnya.

Dalam subVI untuk menentukan frekuensi ideal ini, sinyal input akan melewati *case structure* yang fungsinya membuat keputusan apakah input yang diterima sudah benar atau salah (*true or false*), dalam hal ini jika sinyal input bernilai 0 yang artinya tidak ada frekuensi yang terdeteksi oleh program maka *case structure* akan menganggap salah (*false*) yang berarti tidak ada frekuensi nada yang teridentifikasi, dan jika sinyal input bernilai 1 yang artinya ada frekuensi nada yang terdeteksi maka *case structure* akan menganggap benar (*true*) yang berarti frekuensi yang terdeteksi akan segera diproses oleh program.

Setelah melewati *case structure* sinyal input akan dikirim oleh struktur *for loop* yang berfungsi untuk mengeksekusi subdiagram rangkaian yang ada di dalamnya yang berupa rangkaian untuk memeriksa 6 oktav dari frekuensi ideal dan mencari apakah frekuensi yang masuk berada dalam jangkauan frekuensi ideal, input disini terhubung dengan beberapa komponen yaitu *multiplay*, *add*, *subtract* dan *case structure*. *Multiplay* adalah operasi perkalian, input($x$) yang masuk dikalikan dengan jangkauan frekuensi ideal (*fr*), *add* adalah operasi penjumlahan dan *subtract* adalah operasi pengurangan.

$$y = x(fr) + x \text{ dan } y = x(fr) - x$$

$x$ =input

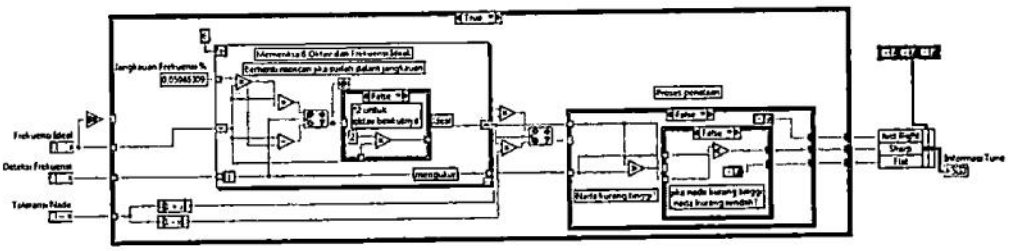
*fr* = jangkauan frekuensi ideal

$y$ =output

Hasil dari persamaan tersebut kemudian akan dibandingkan lagi untuk mengetahui apakah frekuensi sudah berada dalam jangkauan frekuensi yang ideal.

Selain melalui persamaan, input juga melewati *case structure* yang bertugas untuk memeriksa apakah frekuensi masukan sudah ideal atau belum, jika ideal maka pada *case structure* akan bernilai benar (*true*) yang artinya input akan diteruskan untuk dibandingkan dengan hasil persamaan tadi dan jika belum ideal akan bernilai salah (*false*) yang berarti frekuensi masukan akan dikalikan 2 untuk mencari frekuensi nada di oktav berikutnya sampai bisa dikatakan ideal dan kemudian dibandingkan lagi dengan hasil persamaan.

Perbandingan antara hasil persamaan dan frekuensi ideal kemudian masuk *case structure* lagi untuk mendapatkan nilai yang nantinya akan ditampilkan pada indikator, jika frekuensi nada sudah ideal maka *case structure* akan memberikan nilai benar (*true*) yang berarti nilai dari frekuensi nada sudah ideal dan akan langsung menyalakan lampu indikator tengah yang menginformasikan bahwa nada sudah tepat (*tune*), akan tetapi jika tidak maka *case structure* akan memberikan nilai salah (*false*) yang berarti nada belum tepat, untuk mengetahui apakah nada itu kurang tinggi atau kurang rendah di dalam *case structure* terdapat komponen *greater* yang berfungsi untuk menyeleksi 2 frekuensi nada yang masuk, untuk menjadikannya sebuah tampilan maka digunakan lagi *case structure* yang akan menilai keluaran dari *greater*, jika *case structure* bernilai benar (*true*) maka *case structure* akan langsung menghubungkan *greater* ke tampilan lampu indikator bagian kiri yang akan menginformasikan bahwa nada kurang tinggi, dan jika *case structure* bernilai salah (*false*) maka *case structure* akan menghubungkan 2 frekuensi nada yang masuk tadi ke dalam komponen *less* yang keluaranya terhubung dengan tampilan lampu indikator bagian kanan yang akan menginformasikan bahwa nada kurang rendah.



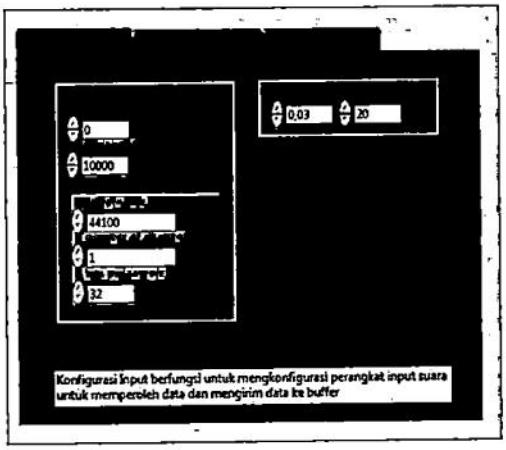
**Gambar 3.5** Diagram blok proses penalaan

### 3.2.3 Rancangan Output

Output Penala Gitar dengan LabVEW adalah berupa tampilan yang dibagi menjadi 3 bagian yang dipisahkan dengan kontrol tab, yaitu Konfigurasi Input, Indikator Sinyal Input dan Indikator Penalaan.

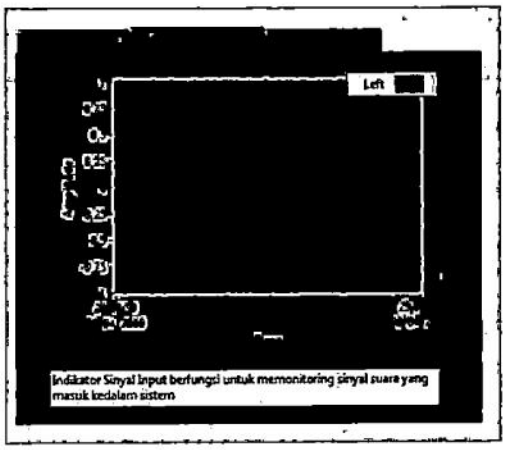
Pada Konfigurasi Input terdapat 2 kotak tampilan, kotak tampilan sebelah kiri menampilkan *device id* yang berfungsi sebagai informasi perangkat input atau output untuk akses operasi suara. Nilai *default device ID* adalah 0. Kemudian *Number of Samples/ch* yang berfungsi untuk menentukan jumlah sampel per channel dalam buffer, untuk operasi yang kontinyu maka diperlukan jumlah sampel yang banyak, dan selanjutnya adalah *sound format*, berisi informasi *sample rate*, *number of chanel* dan *bit per sample* yang berfungsi menetapkan nilai yang didapat, jumlah channel, dan bit per sampel dari operasi suara.

Kotak tampilan di sebelah kanan menampilkan informasi *Threshold* yang berfungsi untuk menentukan amplitudo minimum dari sinyal suara yang masuk dan Toleransi untuk menentukan jangkauan frekuensi ideal.



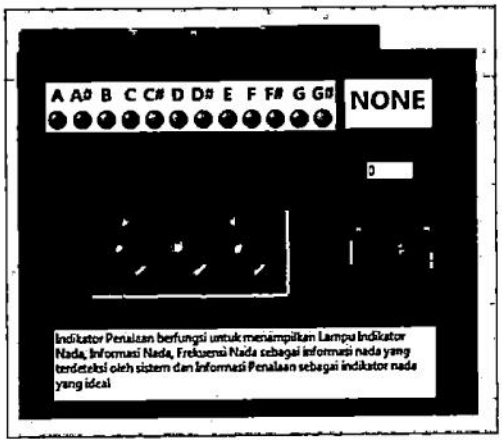

**Gambar3.6** Konfigurasi Input

Tampilan kedua adalah Indikator Sinyal Input yang menampilkan gelombang suara berdasarkan amplitudo (sumbu vertikal) dan waktu (sumbu horizontal) berfungsi untuk memonitor sinyal suara yang masuk ke dalam sistem.




**Gambar 3.7** Indikator Sinyal Input

Tampilan yang ke-3 adalah Indikator Penalaan yang menampilkan lampu indikator nada, informasi nada, frekuensi nada sebagai informasi nada yang terdeteksi oleh sistem dan informasi penalaan sebagai indikator penala nada yang ideal.

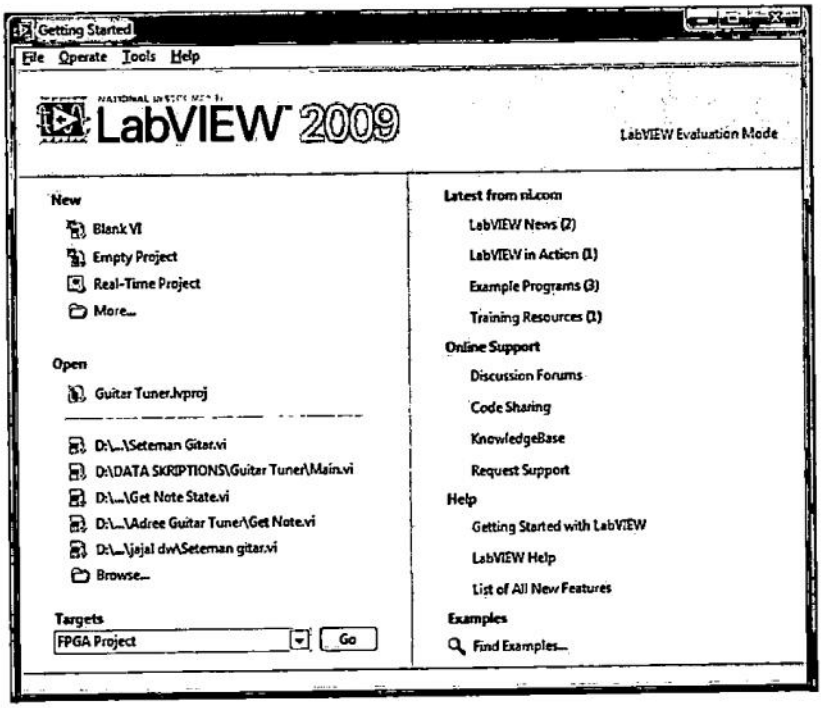



Gambar 3.8 Indikator Penalaan

## 3.3. Pembuatan Penala Gitar pada Lab View

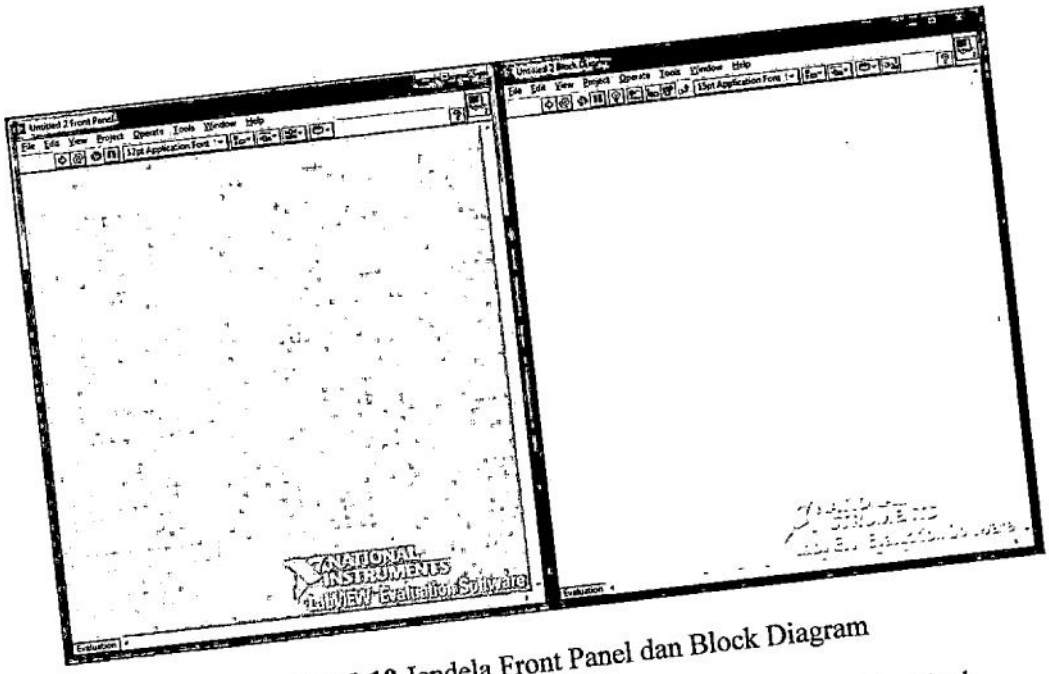
Dalam pembuatan Penala Gitar pada Lab View, langkah-langkah yang dilakukan adalah sebagai berikut:

1. Jalankan aplikasi LabVIEW pada desktop, maka akan muncul jendela seperti gambar di bawah ini :



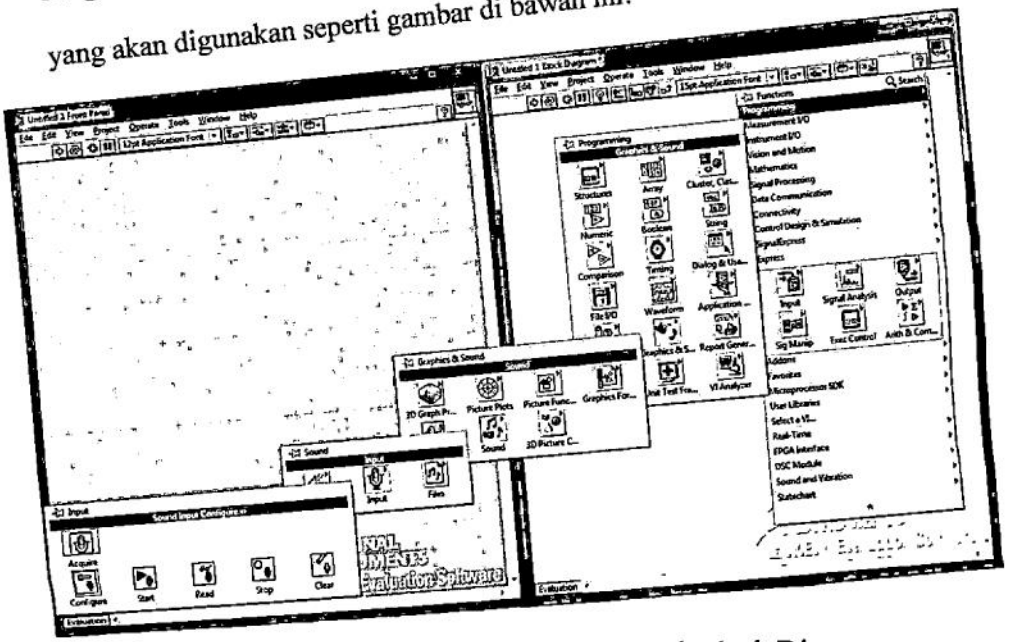

Gambar 3.9Tampilan awal LabVIEW

2. Setelah itu klik *File →New VI* maka ada 2 jendela yang muncul, jendela sebelah kiri adalah jendela front panel dan jendela sebelah kanan adalah Block Diagram seperti gambar di bawah ini :



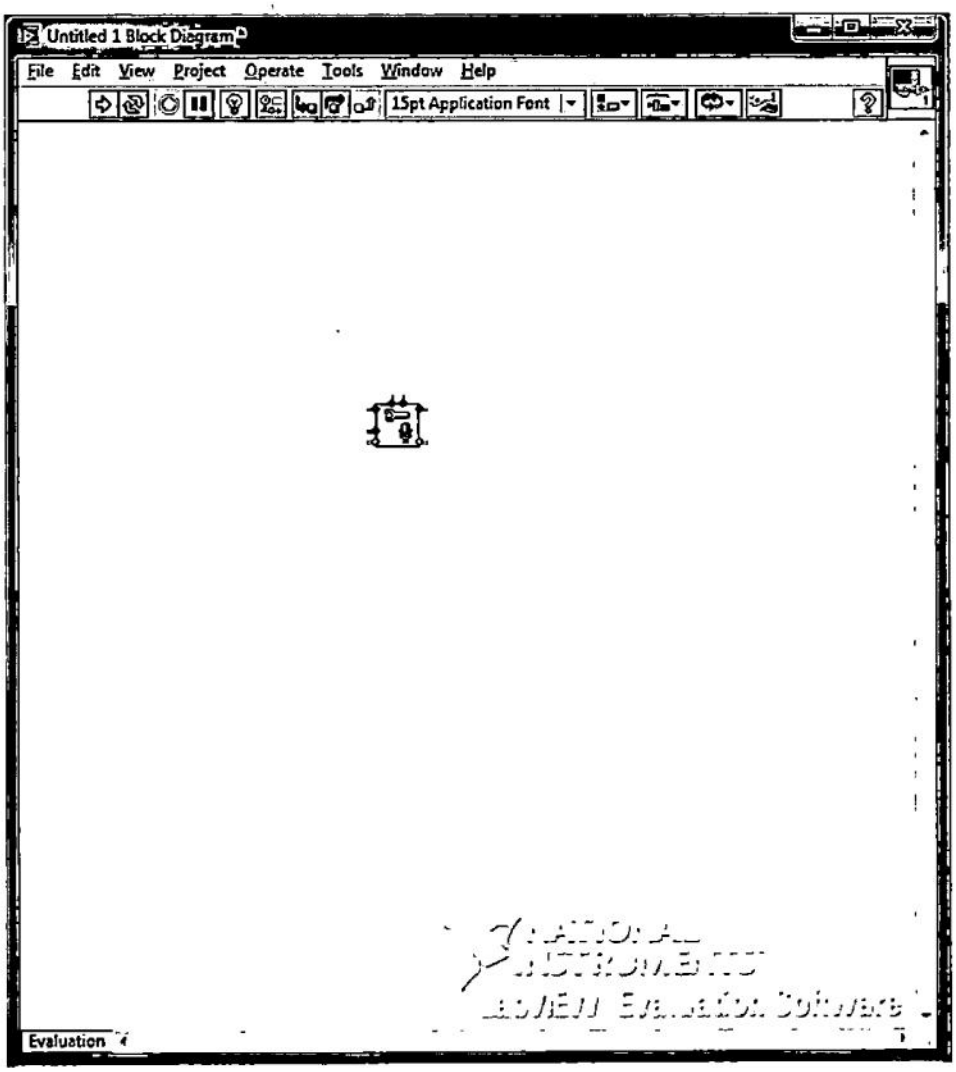
**Gambar 3.10** Jendela Front Panel dan Block Diagram

3. Untuk memulai membuat rangkaian maka klik kanan pada jendela Block Diagram maka akan muncul tampilan untuk memilih komponen-komponen yang akan digunakan seperti gambar di bawah ini:



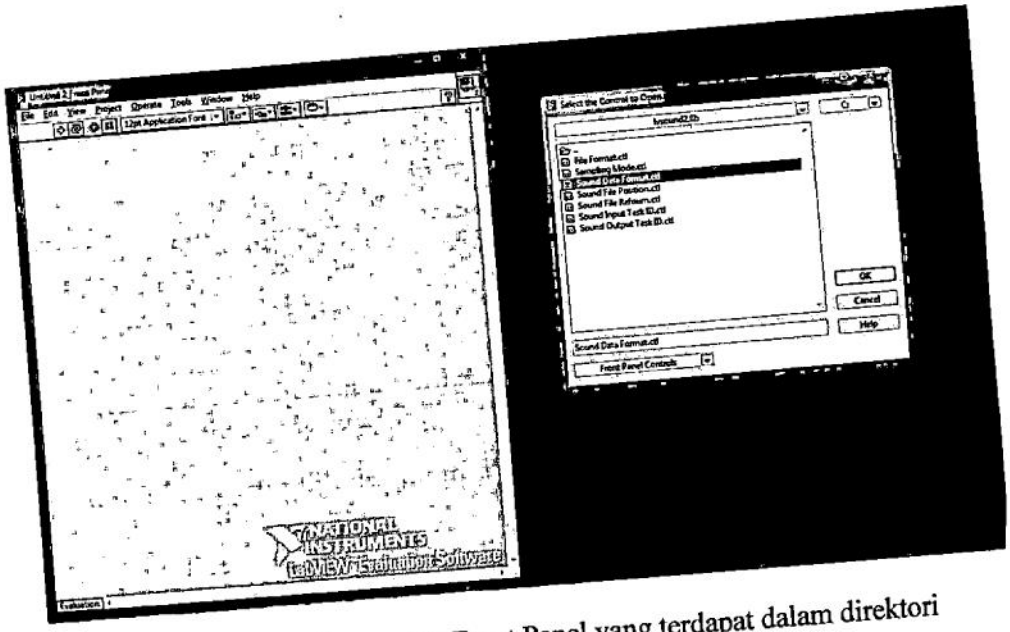
**Gambar 3.11** Library Program pada jendela Block Diagram

4. Langkah pertama adalah membuat rangkaian input, maka kita haru
komponen yang berfungsi untuk mengkonfigurasi perangkat in
untuk memperoleh data yaitu *Sound Input Configure VI* dengan
*→Programing→Graphic & Sound →Sound →Input →S... Input Convigure.vi* maka akan muncul ikon dengan beberapa terminal seperti gambar di bawah ini:




**Gambar 3.12** *Sound Input Configure.vi* pada Block Diagram

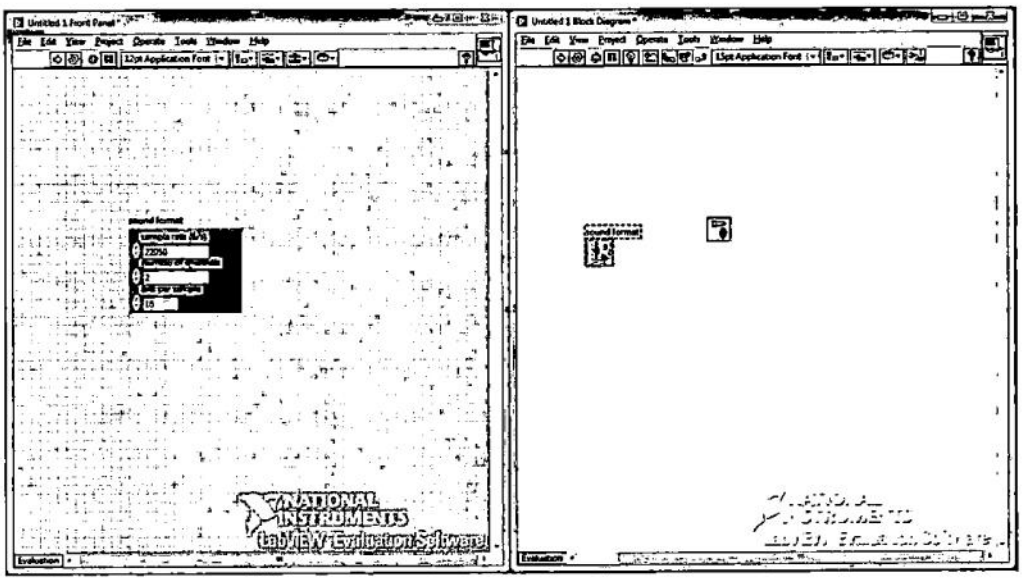
5. Ada beberapa komponen dalam rangkaian yang tidak ditampilkan langsung pada tampilan *Control* dan *Functions* saat kita klik kanan jendela Front Panel maupun jendela Block Diagram, komponen-komponen tersebut disimpan di dalam direktori library LabVIEW, oleh karena itu kita harus mengambilnya didalam direktori library yang telah disediakan Lab View. Klik kanan pada jendela front panel kemudian pilih *Select Control*, masuklah ke dalam direktori *C:/Program Files/National Instrument/LabView2009/vi.lib/sound2/lvsound.llb* maka akan muncul jendela seperti gambar di bawah ini:




**Gambar 3.13** Library pada Front Panel yang terdapat dalam direktori

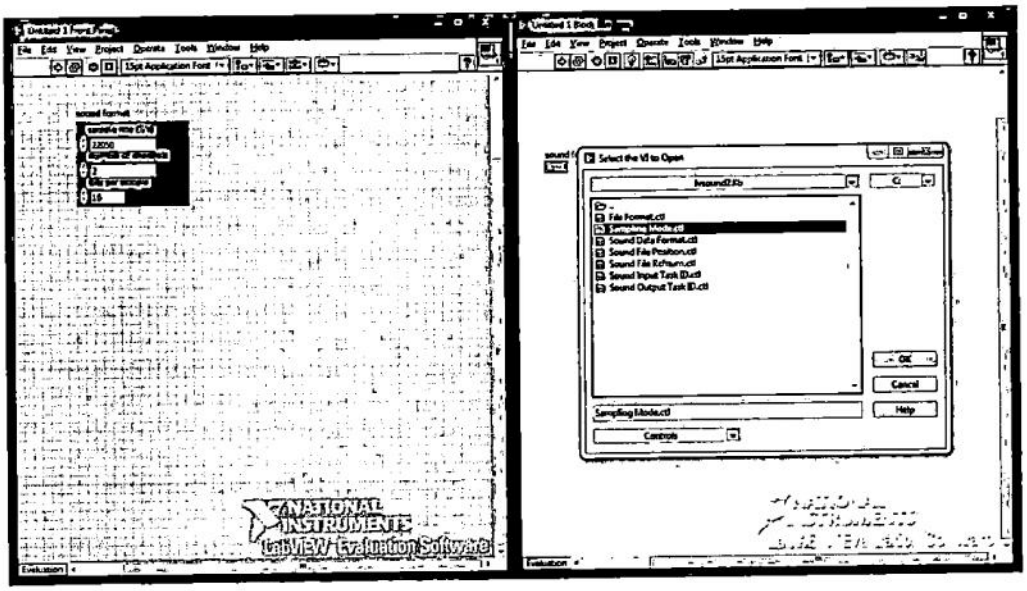
6. Setelah itu pilihlah file *Sound Data Format.ctl* lalu klik OK maka akan keluar tampilan dari file *Sound Data Format.ctl* dan pada saat yang sama jendela

Block Diagram juga akan menampilkan ikon *Sound Data Format.ctl* seperti gambar di bawah ini:

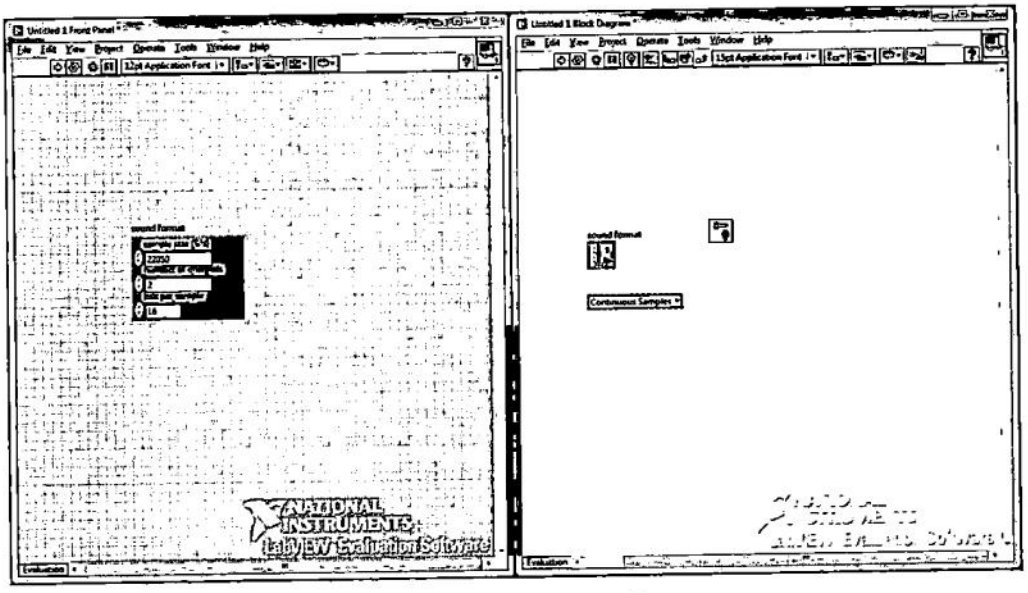


**Gambar 3.14** Tampilan *Sound Data Format.ctl* pada Front Panel dan Block Diagram

7. Setelah menampilkan *Sound Data Format.ctl* selanjutnya adalah memanggil file *Sampling Mode.ctl*, klik kanan jendela Block Diagram, lalu pilih Select a VI dan masuk pada direktori yang sama pilih file *Sampling Mode.ctl* kemudian klik OK maka akan muncul tampilan ikon *Sampling Mode.ctl* seperti pada jendela sebelah kanan gambar di bawah ini:



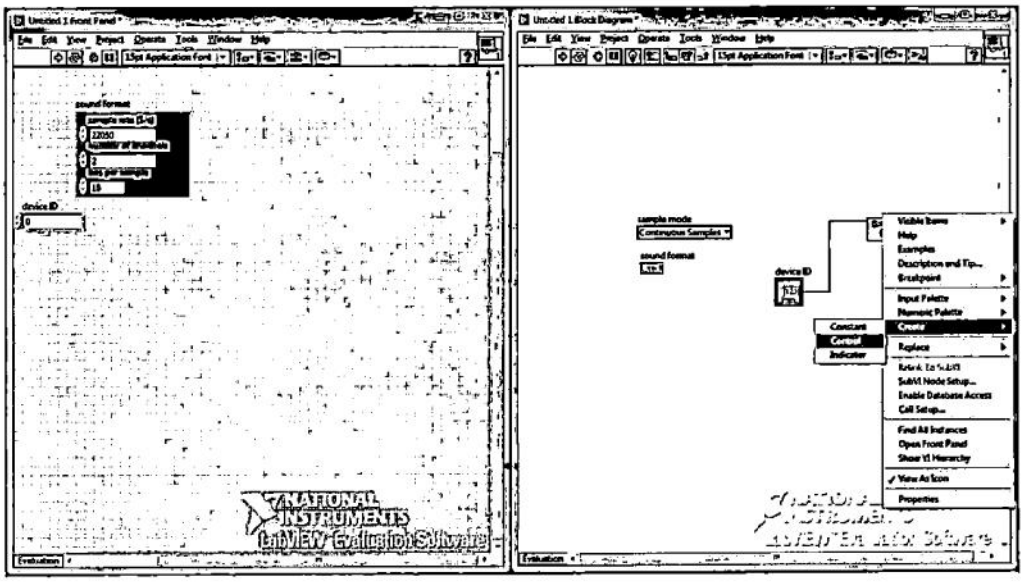
**Gambar 3.15** Library pada Block Diagram yang terdapat di dalam direktori



**Gambar 3.16** Penambahan komponen *Sampling Mode.ctl* pada Block Diagram

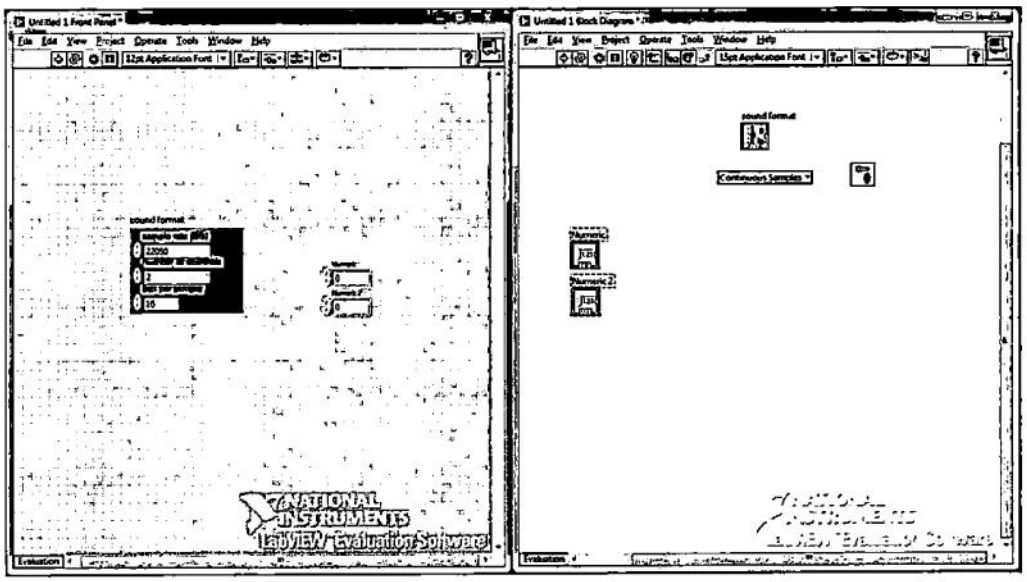
8. Setelah itu langkah selanjutnya adalah melengkapi beberapa komponen pendukung lainya yang dibutuhkan *Sound Input Configure.vi* yaitu *Number of*

*Samples/ch* dan *Device ID* dengan langkah **klik kanan** terminal *Sound Input Configure.vi* pada *Number of Samples/ch* dan *Device ID*→**klik create** →**control** seperti pada gambar di bawah ini:



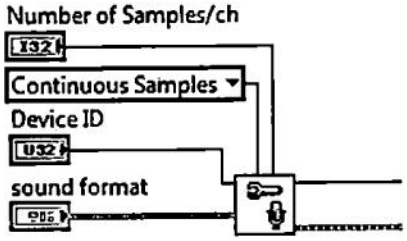
**Gambar 3.17** Membuat komponen *Device ID* berfungsi sebagai kontrol

maka *Number of Samples/ch* dan *Device ID* secara otomatis juga akan ditampilkan pada jendela Front Panel. Tipe data tersebut dapat disesuaikan dengan kebutuhan dan dapat difungsikan sebagai kontrol, konstanta dan indikator di dalam Block Diagram rangkaian seperti gambar di bawah ini:



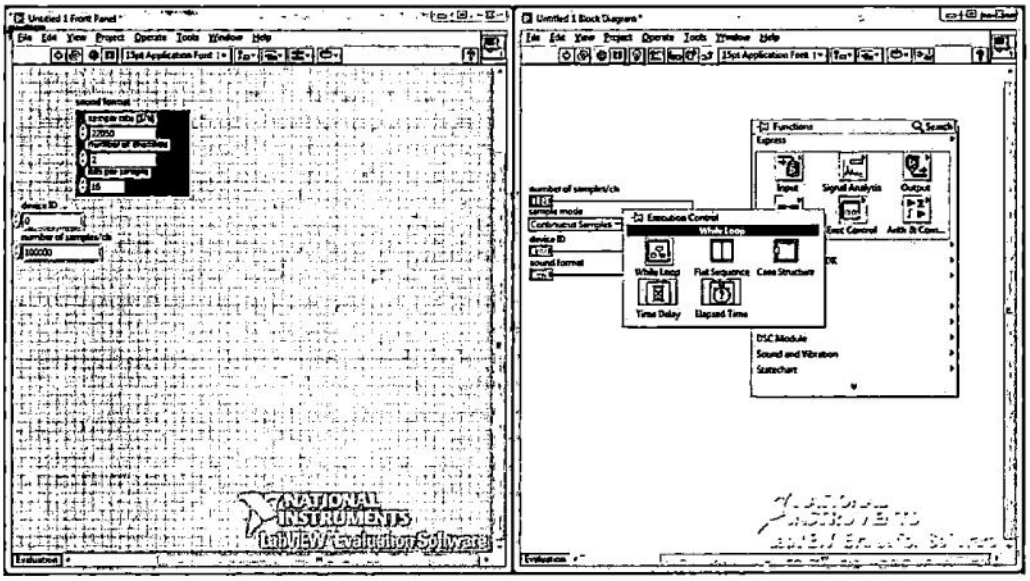

**Gambar 3.18** Penambahan komponen kontrol

9. Setelah semua komponen input disiapkan maka tahap selanjutnya adalah merangkai rangkaian input dengan cara menyambungkan komponen-komponen *Sound Data Format.ctl*, *Sampling Mode.ctl*, *Number of Sample/ch*, *Device ID* ke masing-masing terminal pada *Sound Input Configure.vi* seperti gambar di bawah ini:



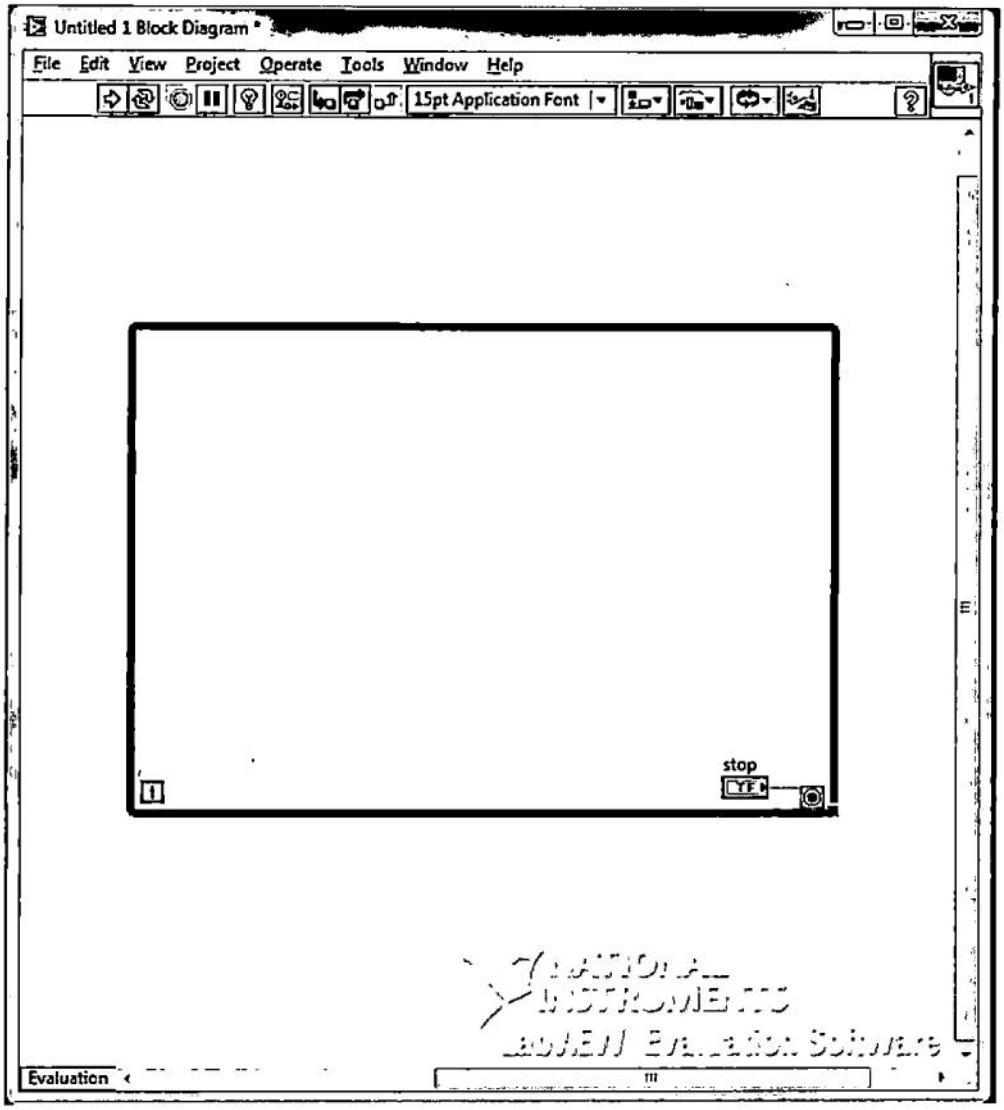

**Gambar 3.19** Rangkaian dari komponen input

10. Langkah selanjutnya adalah membuat rangkaian proses, pertama adalah membuat bingkai struktur *while loop* yang berfungsi agar rangkaian dapat menjalankan proses secara berulang-ulang pada Block Diagram seperti gambar di bawah ini:




**Gambar 3.20** Menambahkan komponen *while loop.vi* dalam Block Diagram

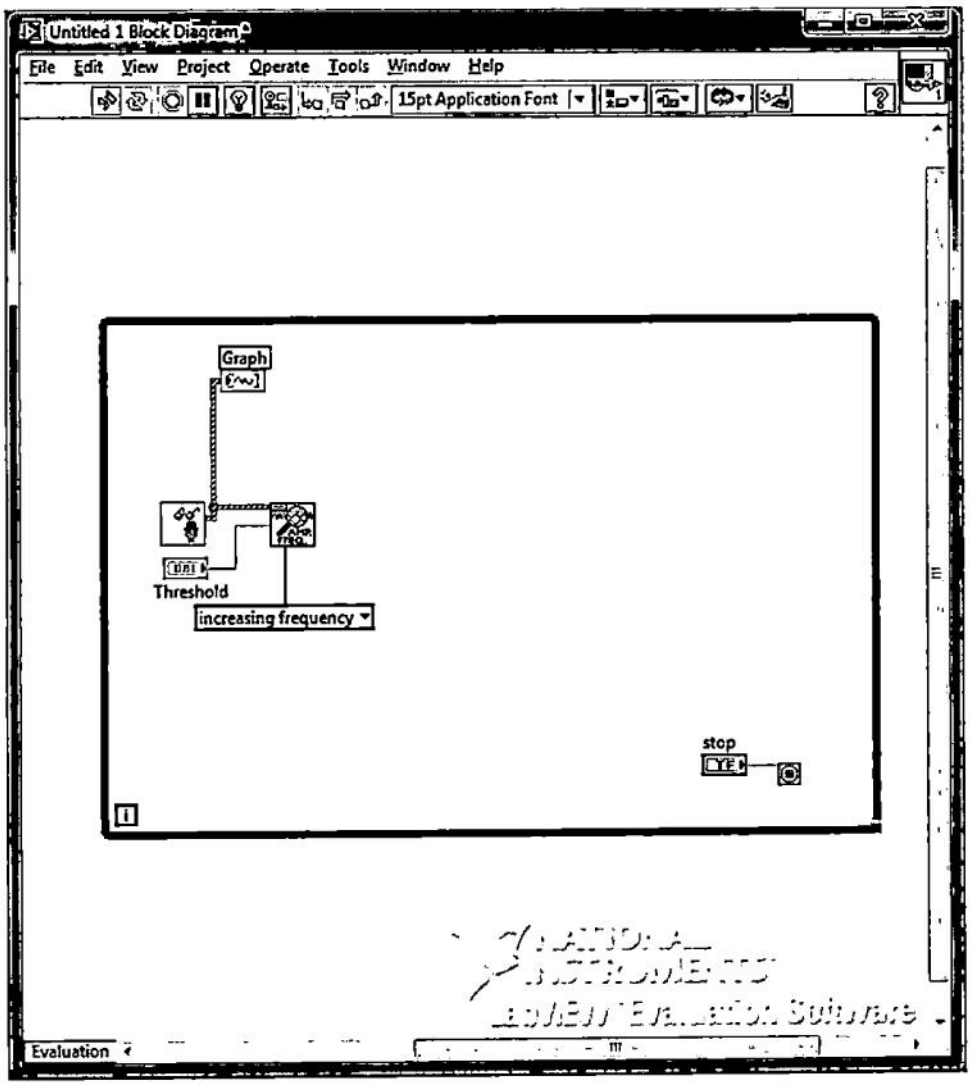
Lalu akan muncul gambar pada jendela Block Diagram seperti gambar di bawah ini :




**Gambar 3.21** Bingkai *while loop.vi*

11. Setelah itu menyiapkan dan merangkai komponen-komponen rangkaian proses didalam struktur *while loop* secara bertahap di antaranya yaitu *Sound Input Read.vi* berfungsi untuk membaca data input yang masuk, *Waveform Graph VI* berfungsi untuk memvisualisasikan gelombang suara input, *Extract Multiple Tone Information VI* berfungsi menguraikan informasi nada input

seperti frekuensi, amplitudo dan fase yang dapat diatur batas minimal (threshold) amplitudonya dan juga menaikkan frekuensi agar dapat dibaca oleh rangkaian seperti gambar di bawah ini:

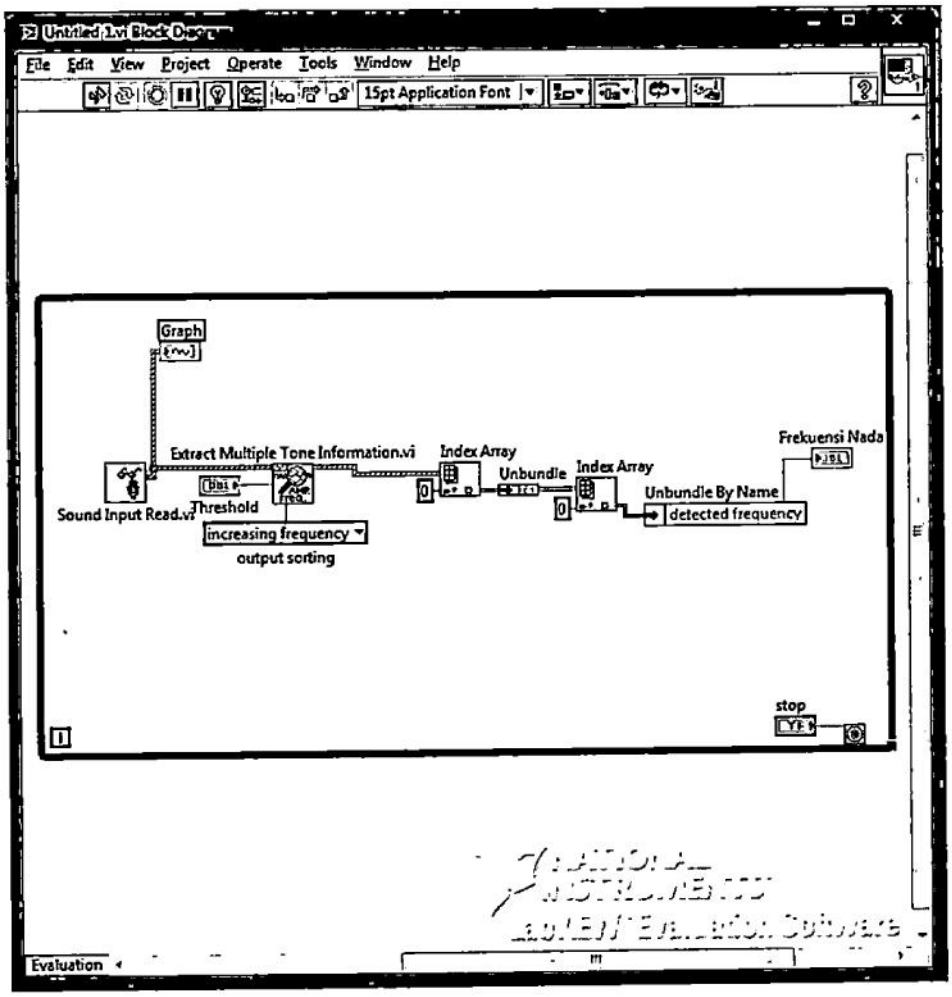



**Gambar 3.22** Membuat rangkaian proses

12. Selanjutnya *Extract Multiple Tone Information VI* akan dihubungkan dengan *index array* yang terangkai secara deret dengan *unbundle cluster* dan *index*

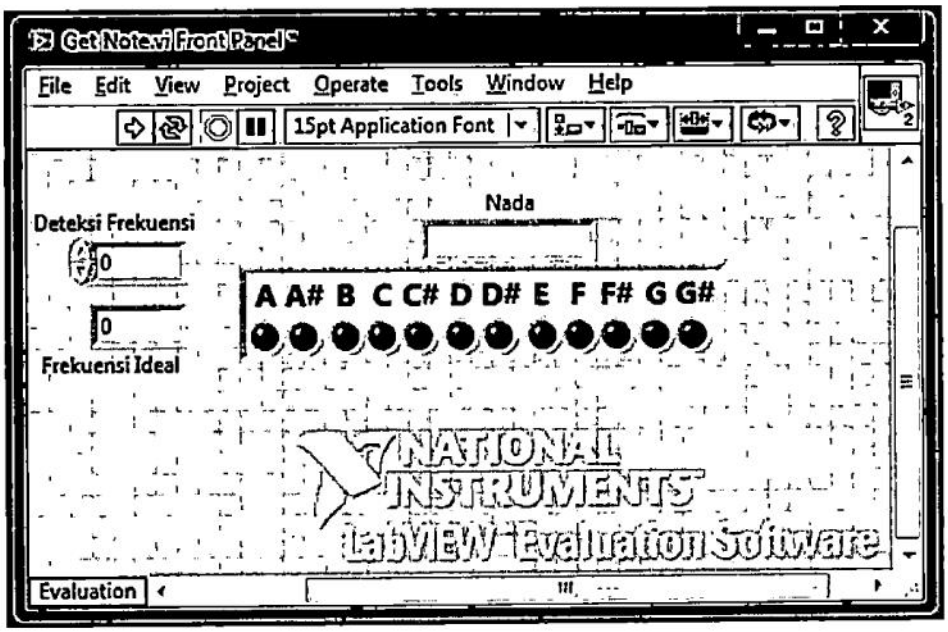
*array* serta *unbundle by name*, *index array* berfungsi untuk mengembalikan elemen n-dimensi pada indeks dan mengubah ukuran secara otomatis untuk menampilkan input indeks untuk setiap dimensi.

*Unbundle cluster* adalah pembagi sebuah rangkaian data ke setiap elemen individu, dan *unbundle by name* berfungsi untuk mengembalikan elemen rangkaian data dengan nama yang telah ditentukan. seperti gambar rangkaian di bawah ini:

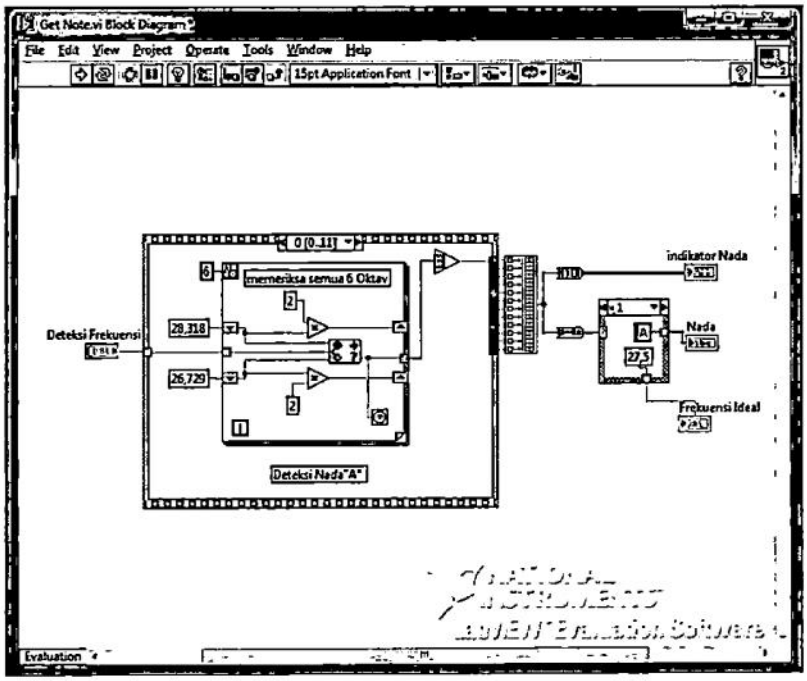



**Gambar 3.23** Rangkaian proses Penala Gitar LabVIEW

13. Setelah rangkaian proses selesai langkah selanjutnya adalah menyusun rangkaian output yaitu menampilkan sinyal suara yang telah diproses ke dalam bentuk indikator. Rangkaian output disini terdiri dari dua sub VI, sub VI yang pertama untuk mendapatkan nada (*Get Note.vi*) dan sub VI yang kedua untuk mendapatkan nada yang ideal (*Get Note State.vi*). Sebelumnya saya telah membuat dua sub VI tersebut, tetapi saya tidak dapat memunculkan terminal yang digunakan untuk menghubungkan sub VI ke komponen yang ada pada VI utama agar dapat menampilkan indikator penalaan. Setelah saya mencari di http://forums.ni.com akhirnya saya mendapatkan dua sub VI tersebut dan semua terminalnya dapat dihubungkan dengan komponen yang ada pada VI utama. Berikut ini adalah gambar tampilan *Get Note.vi* yang telah diubah:

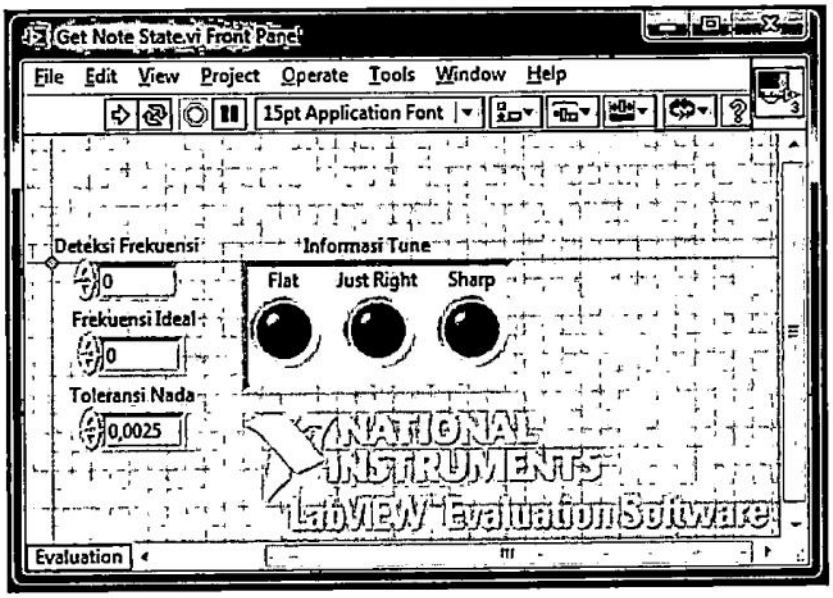



**Gambar 3.24** Tampilan *Get Note.vi* pada Front Panel

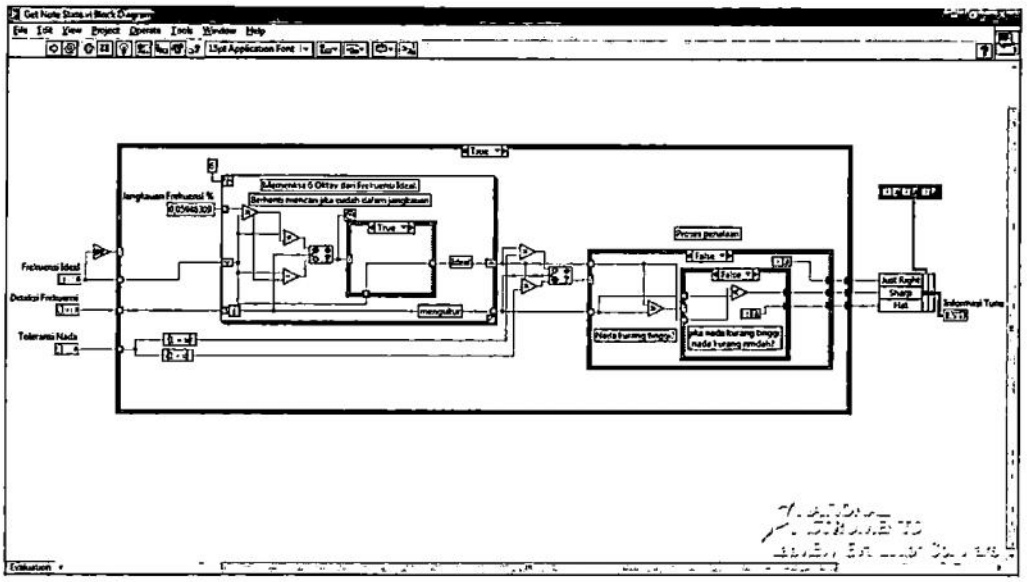



**Gambar 3.25** Tampilan *Get Note.vi* pada Block Diagram

Berikut ini adalah gambar tampilan *Get Note State.vi* yang sedikit saya ubah :

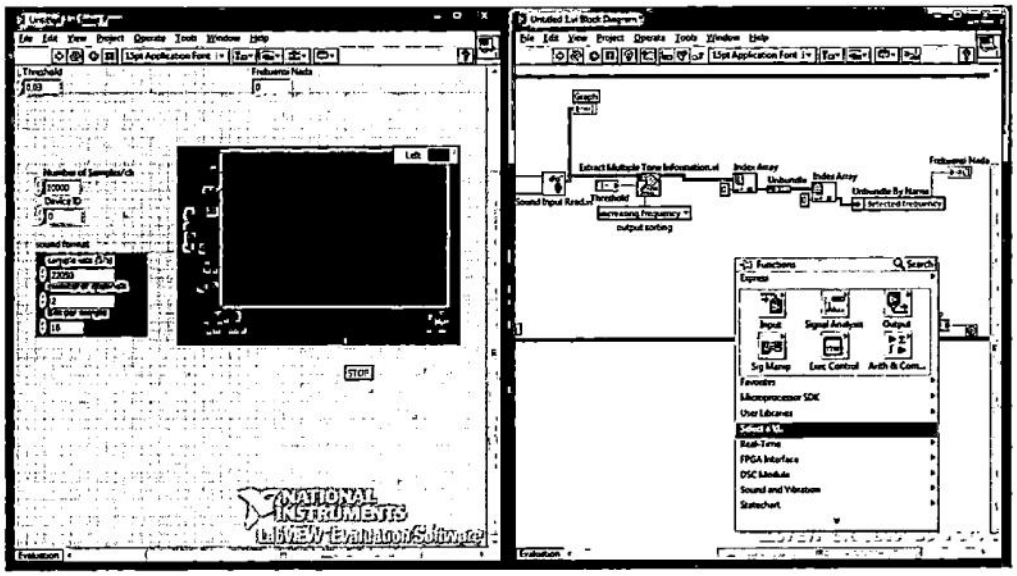



**Gambar 3.26** Tampilan *Get Note State.vi* pada Front Panel



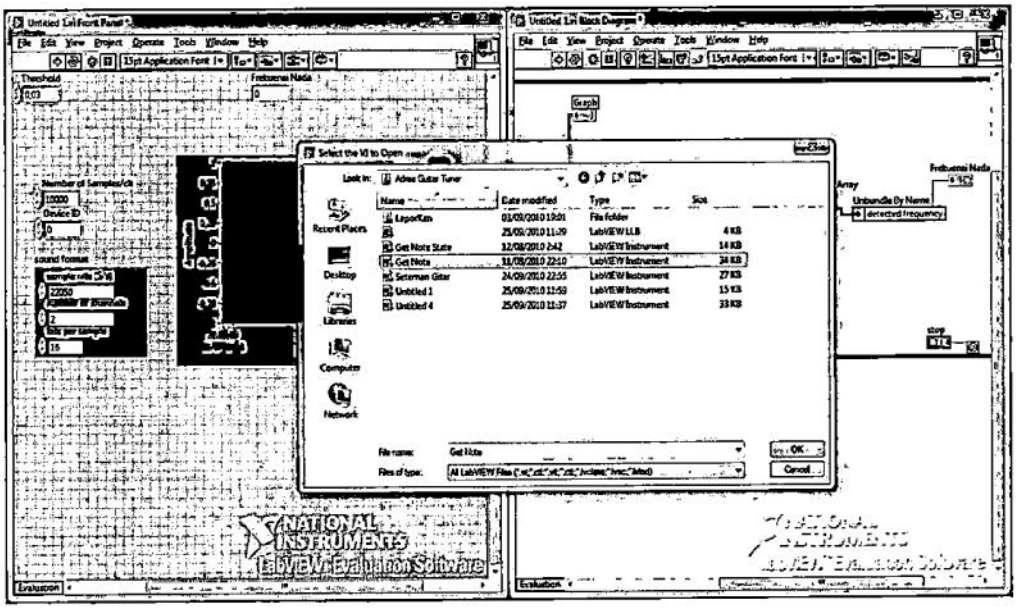
**Gambar 3.27** Tampilan *Get Note State.vi* pada Block Diagram

untuk memasukkan sub VI kedalam main VI **klik kanan pada jendela block diagram→select VI→klik kiri** seperti pada gambar di bawah ini :



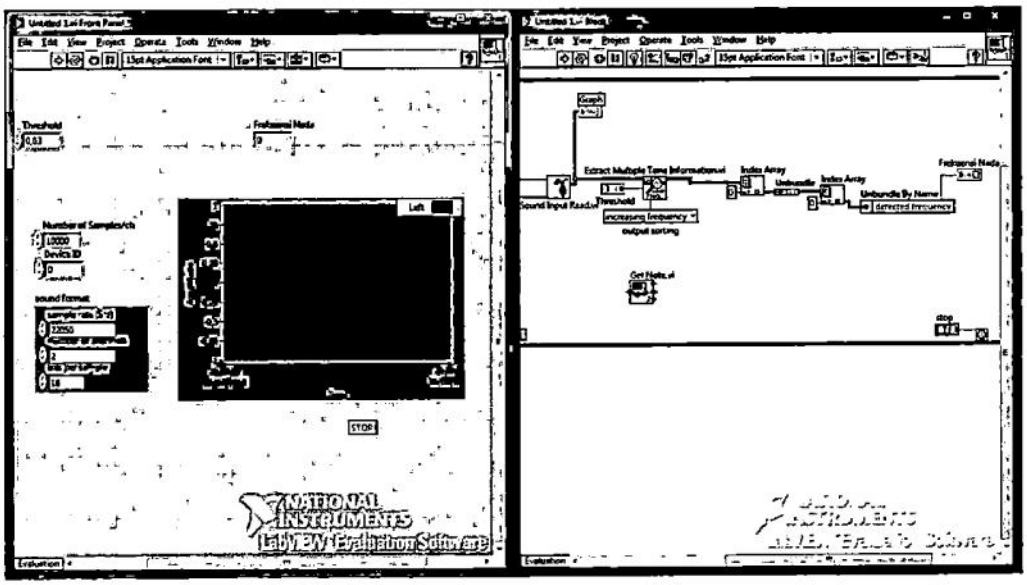
**Gambar 3.28** Memanggil sub VI *Get Note.vi* dari direktori Library

lalu akan muncul jendela untuk memanggil VI yang tersimpan dalam direktori Library.



**Gambar 3.29** *Get Note.vi* pada direktori Library LabVIEW

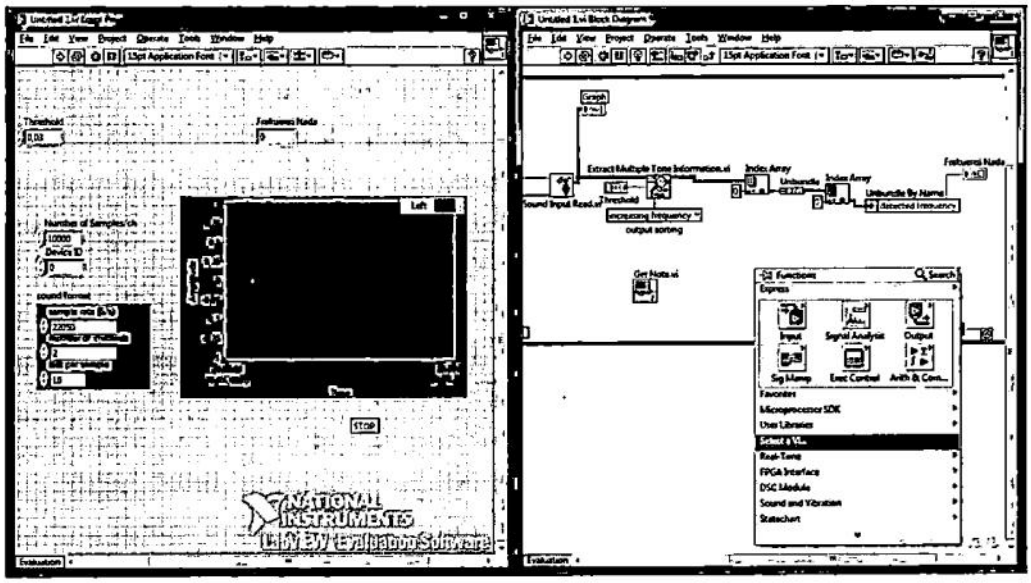
Pilih VI yang akan kita panggil, setelah itu tekanlah tombol OK, maka akan muncul tampilan seperti gambar di bawah ini:



**Gambar 3.30** Sub VI *Get Note.vi* pada Block Diagram

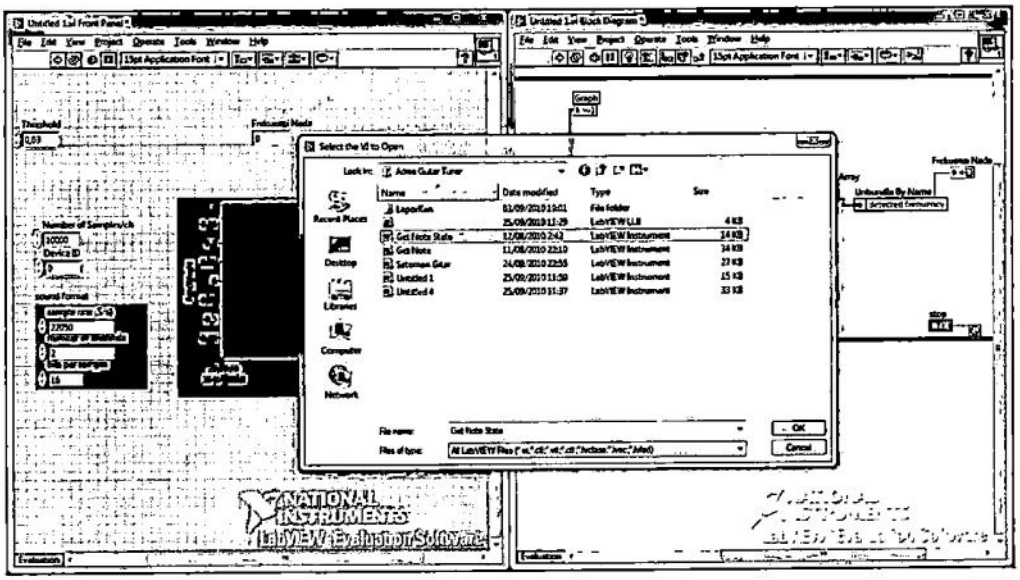
Maka ikon dari sub VI tersebut akan muncul pada jendela Block Diagram, begitu juga pada sub VI yang selanjutnya.

**klik kanan pada jendela block diagram→select VI→klik kiri** seperti pada gambar di bawah ini :

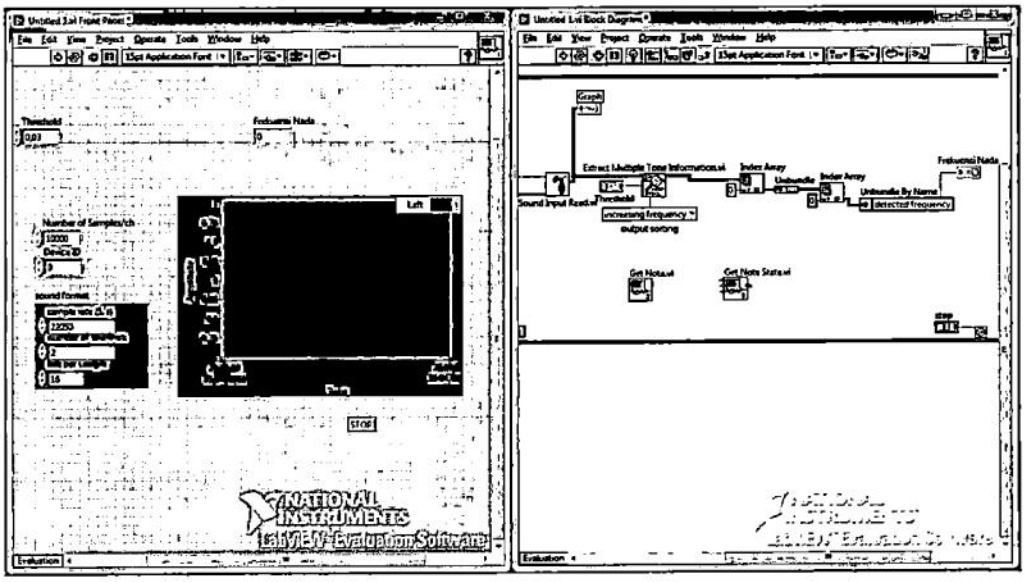


**Gambar 3.31** Memanggil sub VI *Get Note State.vi* dari direktori Library

Ulangi langkah tadi untuk memanggil VI di dalam direktori Library.

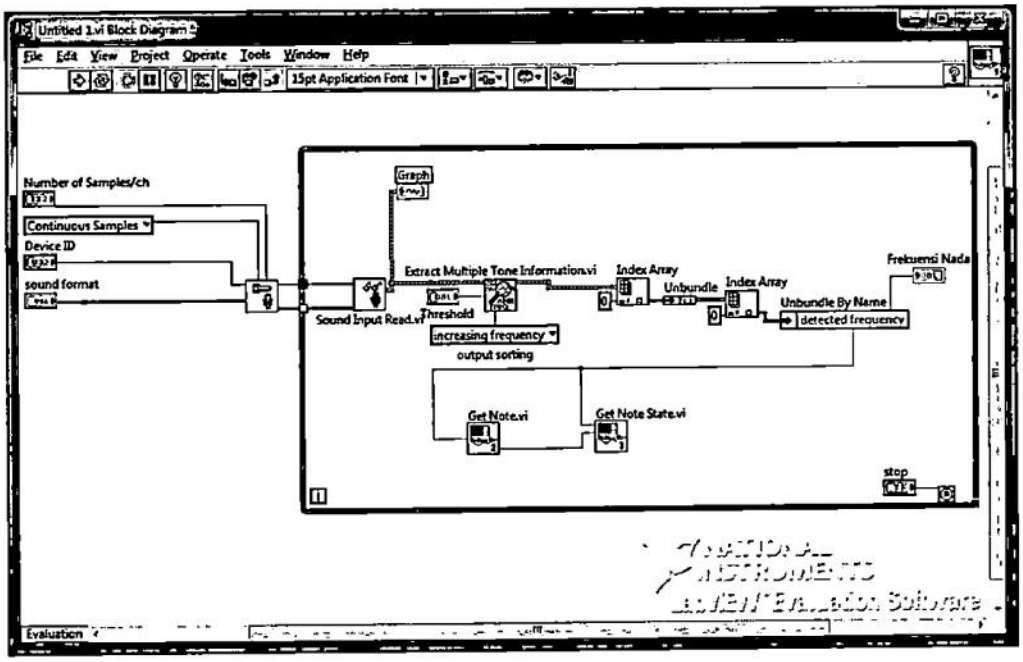


**Gambar 3.32** *Get Note State.vi* pada direktori Library LabVIEW



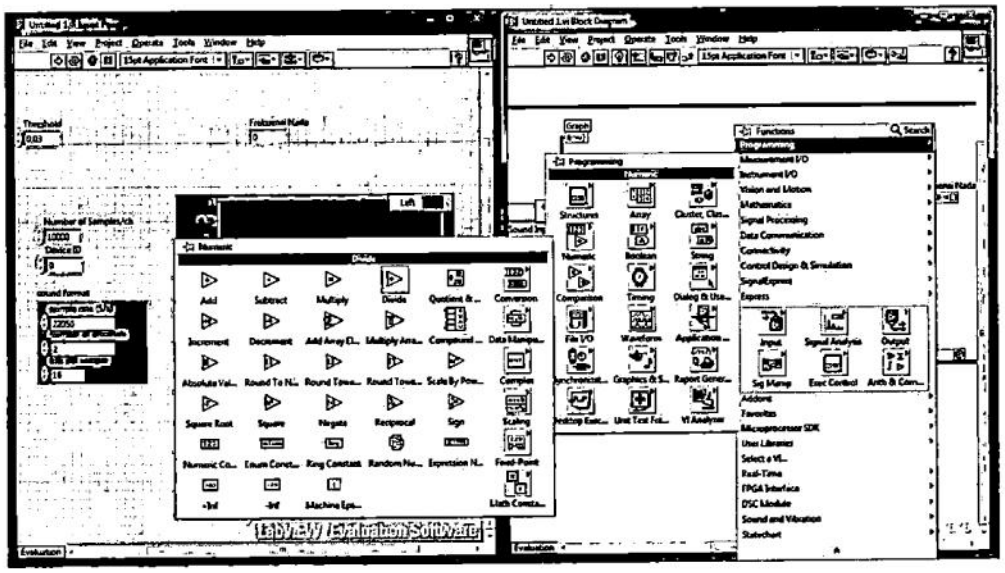
**Gambar 3.33** Sub VI *Get Note State.vi* pada Block Diagram

14. Setelah sub VI dimasukkan, langkah selanjutnya adalah menghubungkan terminal dari sub VI seperti pada gambar di bawah ini :



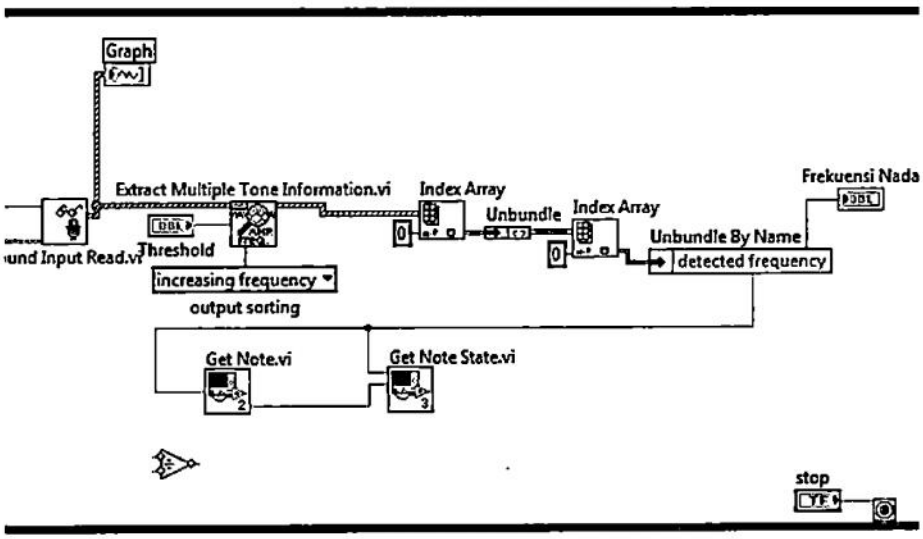

**Gambar 3.34** Menghubungkan sub VI ke dalam rangkaian

15. Pada salah satu terminal dari sub VI *Get Note State.vi* adalah toleransi nada, adapun standar toleransi yang ada pada *Get Note State.vi* yaitu sebesar 0,0025. Agar toleransi nada yang diberikan dapat diubah, maka toleransi dibuat dengan menggunakan kontrol numerik dengan tipe data double yang dibagi dengan sebuah konstanta yang bernilai 10000 dengan tipe data double. Untuk memanggil komponen pembagi **klik kanan pada block diagram → klik program → numeric → devide** seperti pada gambar di bawah ini :



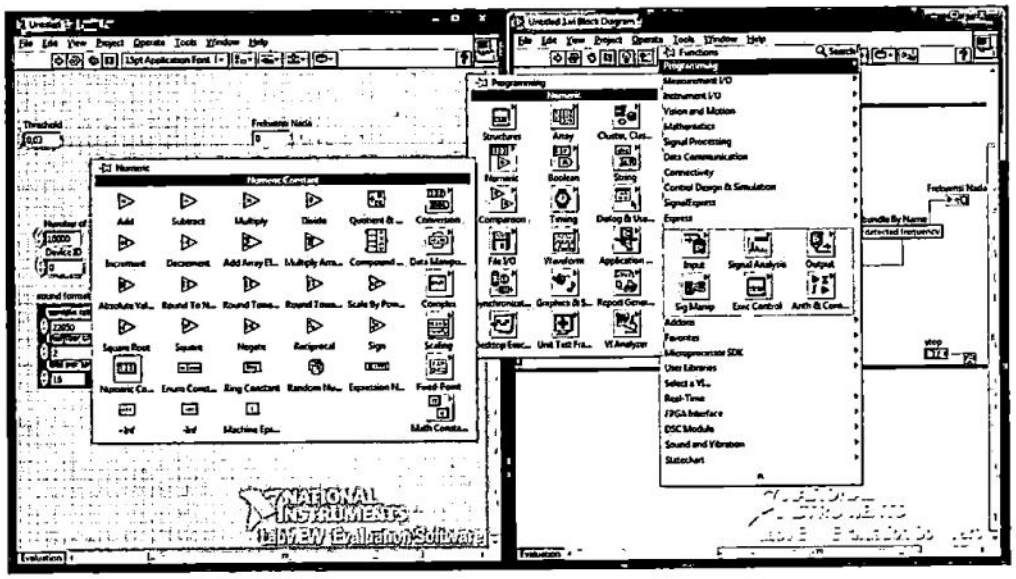
**Gambar 3.35** Memanggil komponen algoritma pada Library Block Diagram

Lalu akan muncul ikon pembagi seperti pada gambar di bawah ini:



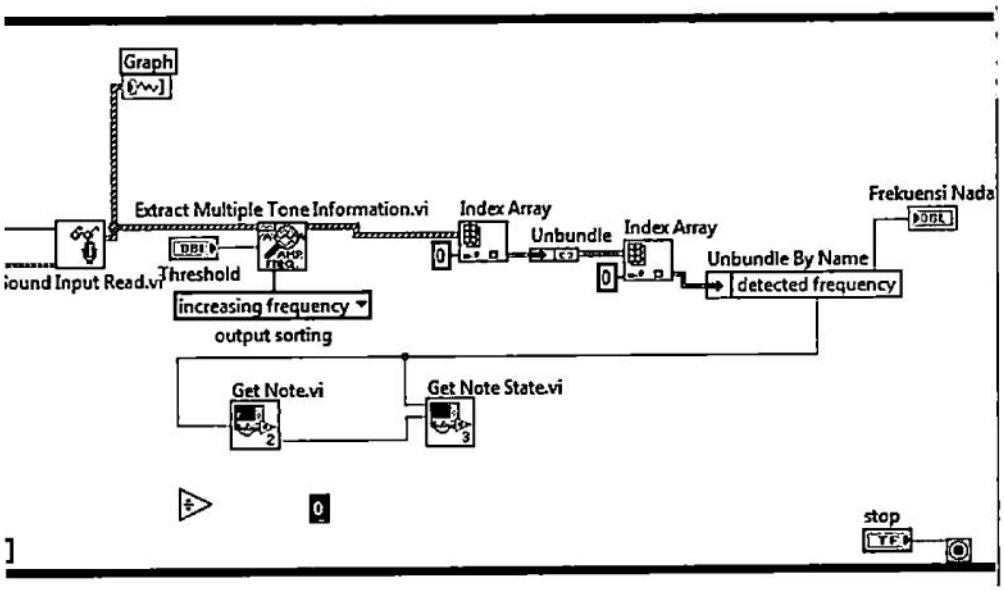

**Gambar 3.36** Tampilan komponen algoritma pembagian pada Block Diagram

16. Kemudian membuat kontrol tipe data double dengan cara **klik kanan → programing → numeric →numeric constant**



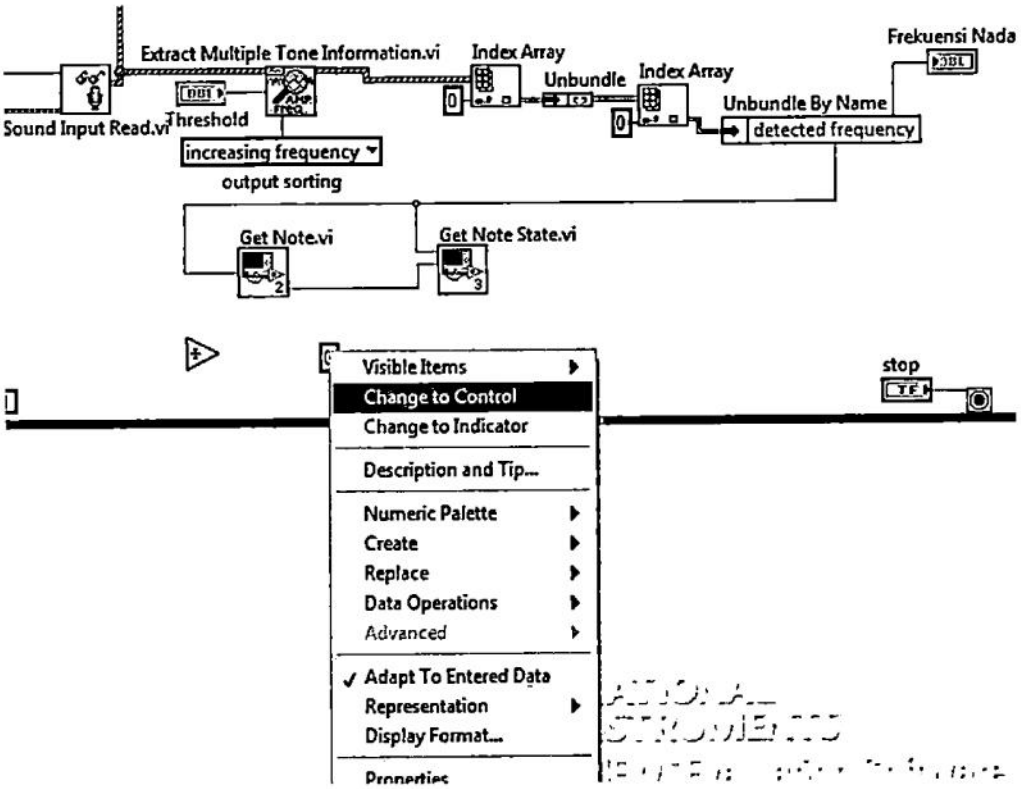
**Gambar 3.37** Memanggil komponen konstanta pada Library Block Diagram

Lalu akan muncul ikon konstanta numerik yang bernilai "0" seperti pada gambar di bawah ini:



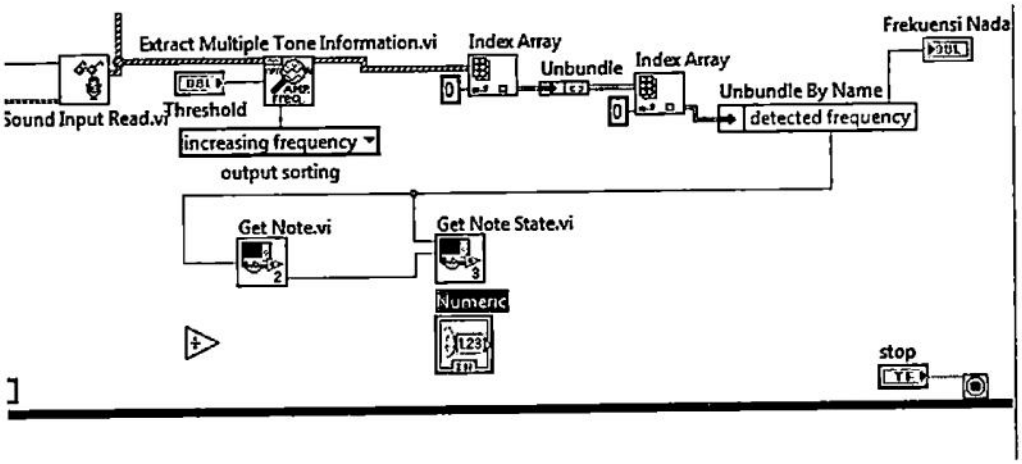

**Gambar 3.38** Tampilan komponen konstanta pada Block Diagram

Setelah itu **klik kanan pada ikon konstanta numerik → change to control** seperti pada gambar di bawah ini :



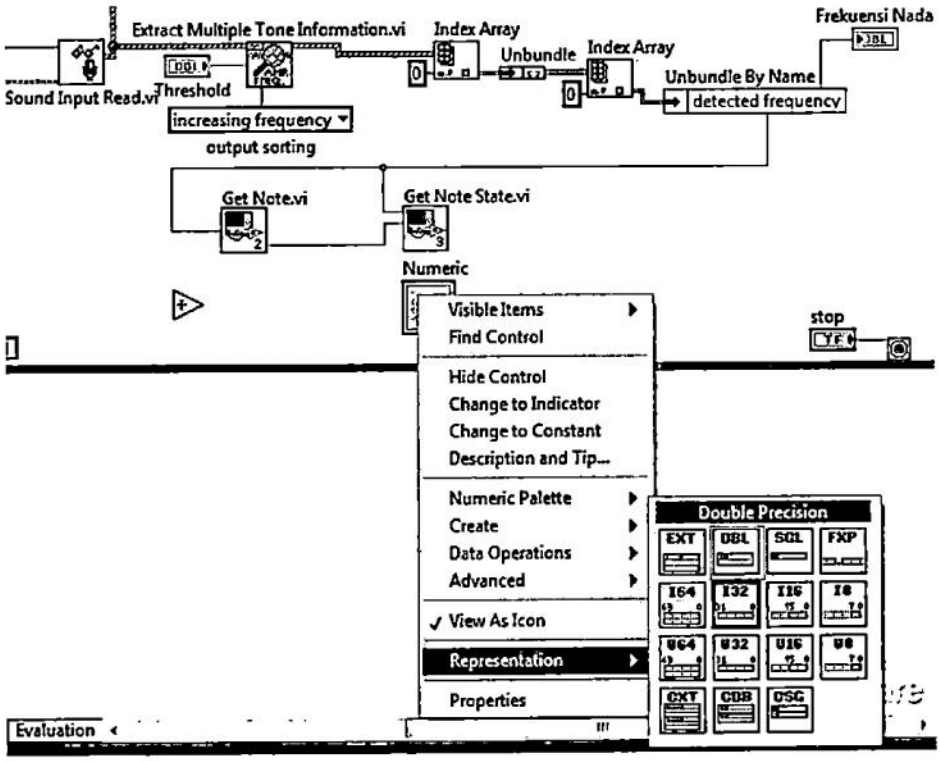

**Gambar 3.39** Mengubah fungsi konstanta menjadi kontrol pada Block Diagram

Maka ikon akan berubah seperti pada gambar di bawah ini :



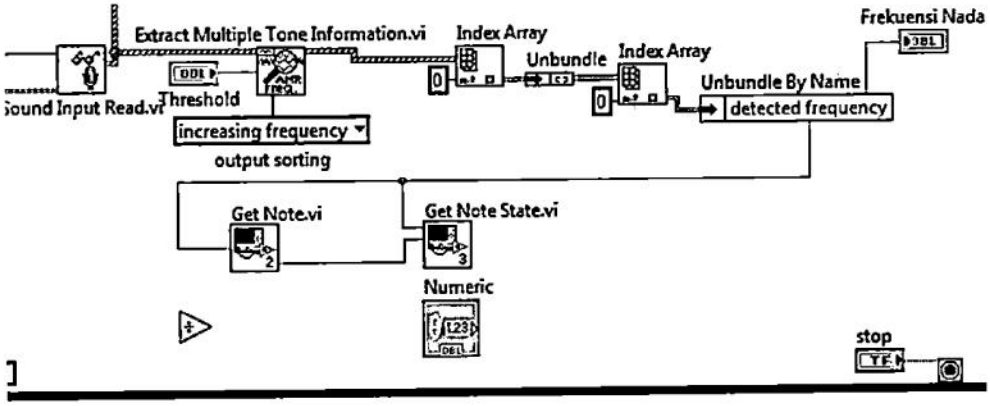

**Gambar 3.40** Tampilan ikon kontrol pada Block Diagram

Lalu ubah tipe data dengan cara **klik kanan pada ikon→klik Representation → pilih double precision**



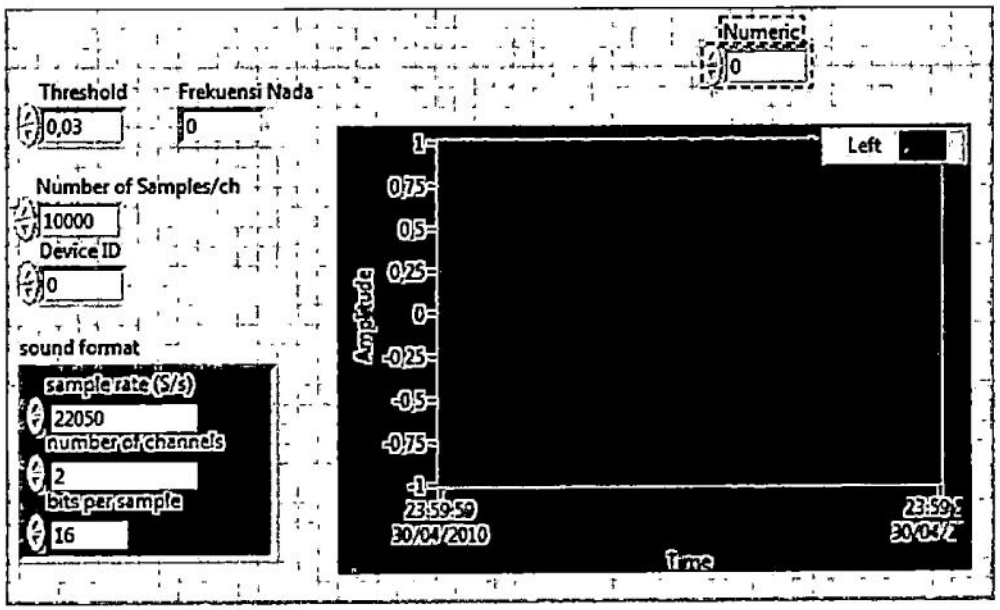

**Gambar 3.41** Mengganti tipe data komponen

Maka ikon akan berubah warna seperti pada gambar di bawah ini :




**Gambar 3.42** Tampilan tipe data *Double Precision*

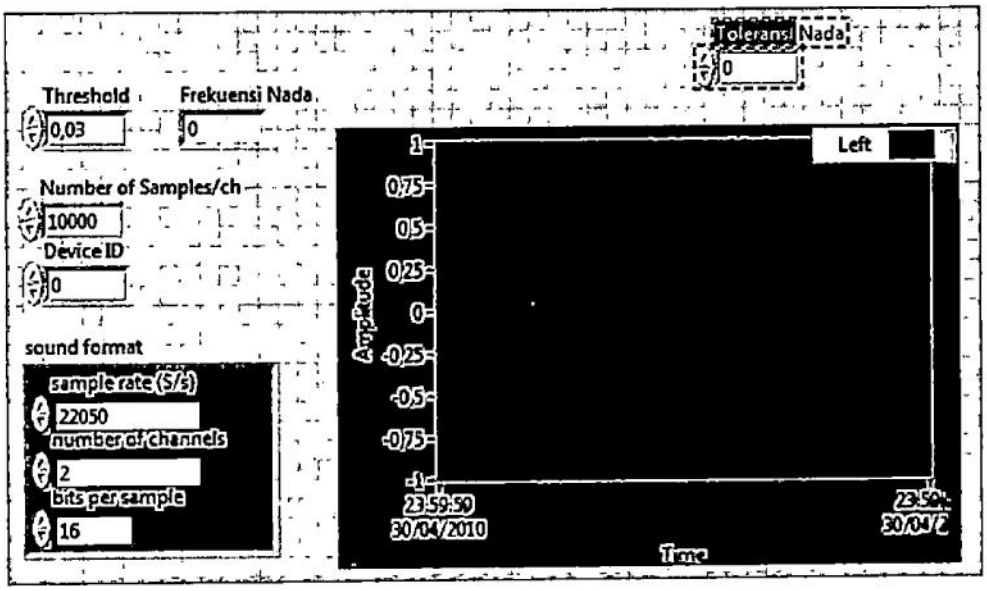
Pada jendela Front Panel akan muncul gambar seperti di bawah ini :




**Gambar 3.43** Tampilan kontrol pada Front Panel

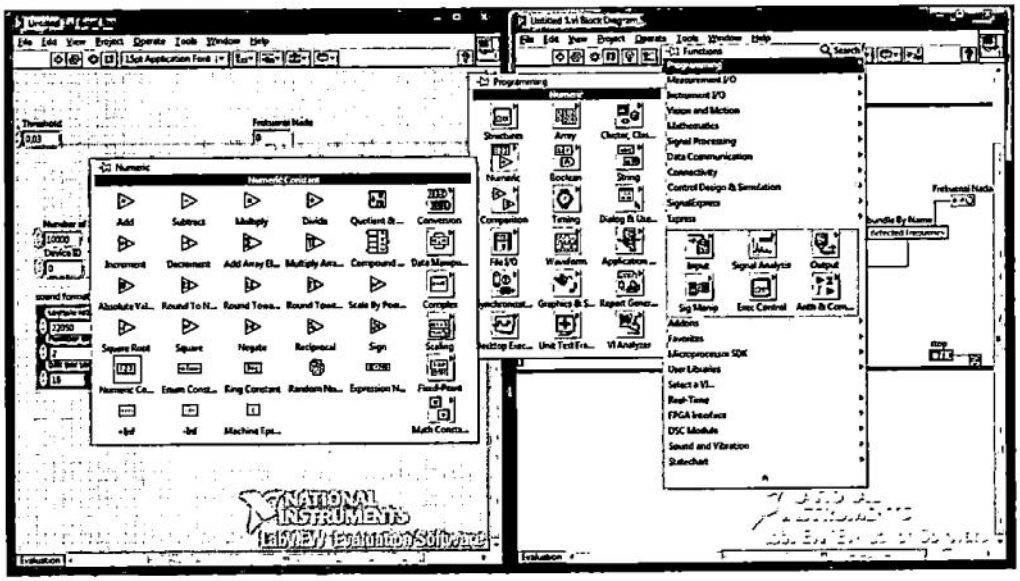
Nama dari masing-masing komponen dapat diganti dengan cara **klik dua kali**

→ **ganti dengan nama yang sesuai** seperti pada gambar di bawah ini:



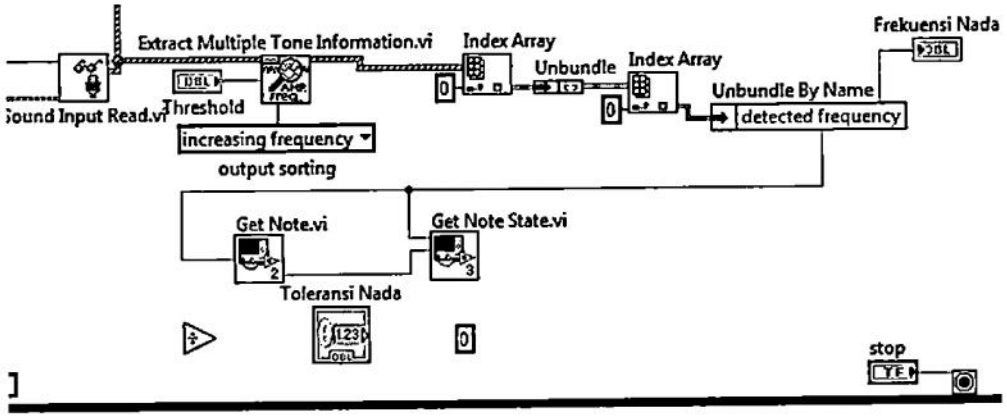

**Gambar 3.44** Mengganti nama komponen pada Front Panel

17. Setelah membuat kontrol, selanjutnya adalah membuat konstanta pembagi dengan tipe data double dengan cara **klik kanan → programing → numeric →numeric constant** seperti pada gambar di bawah ini :



**Gambar 3.45** Library pada Block Diagram

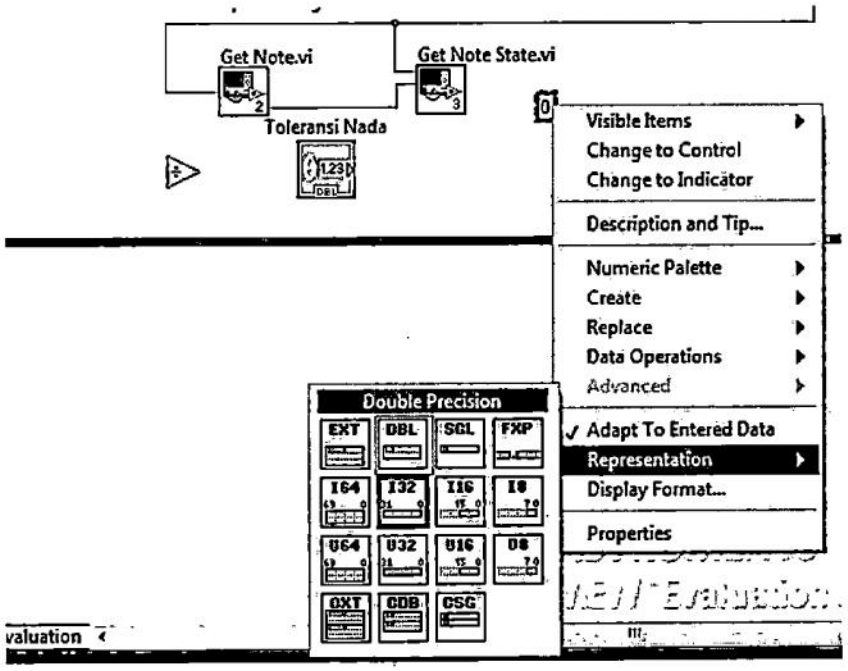
kemudian akan muncul ikon konstanta numerik yang bernilai "0" seperti pada gambar di bawah ini:




**Gambar 3.46** Tampilan ikon konstanta pada Block Diagram

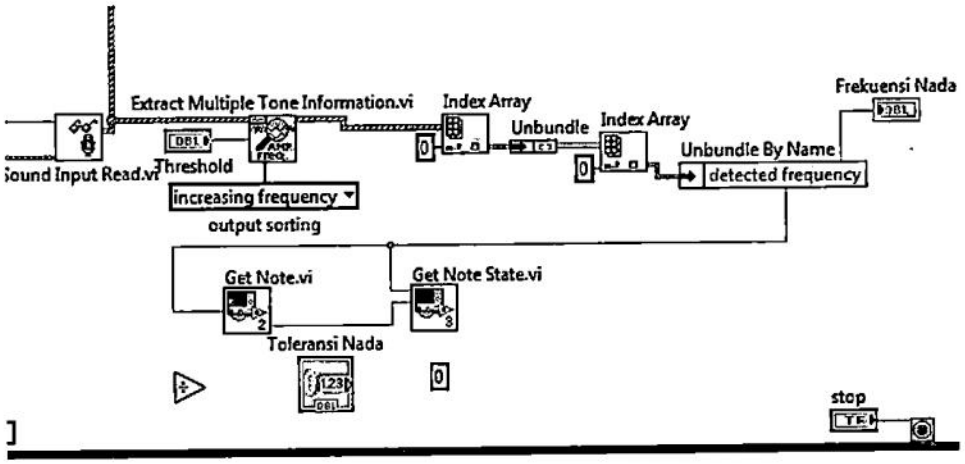
Setelah itu ubah tipe data dengan cara **klik kanan pada ikon→klik**

**Representation → pilih double precision** seperti pada gambar di bawah ini :



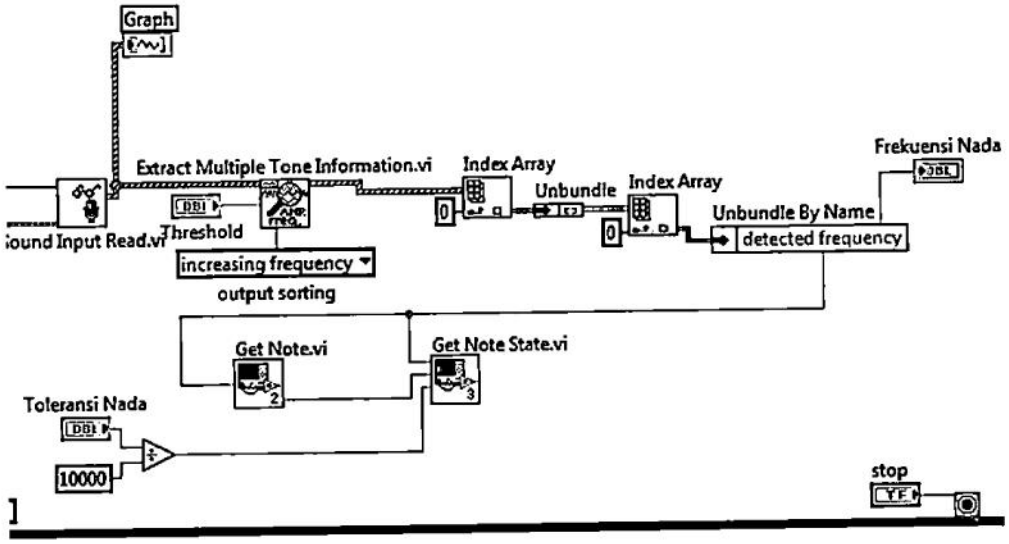

**Gambar 3.47** Mengganti tipe data konstanta menjadi *Double Presicion*

Maka warna ikon akan berubah seperti pada gambar di bawah ini :



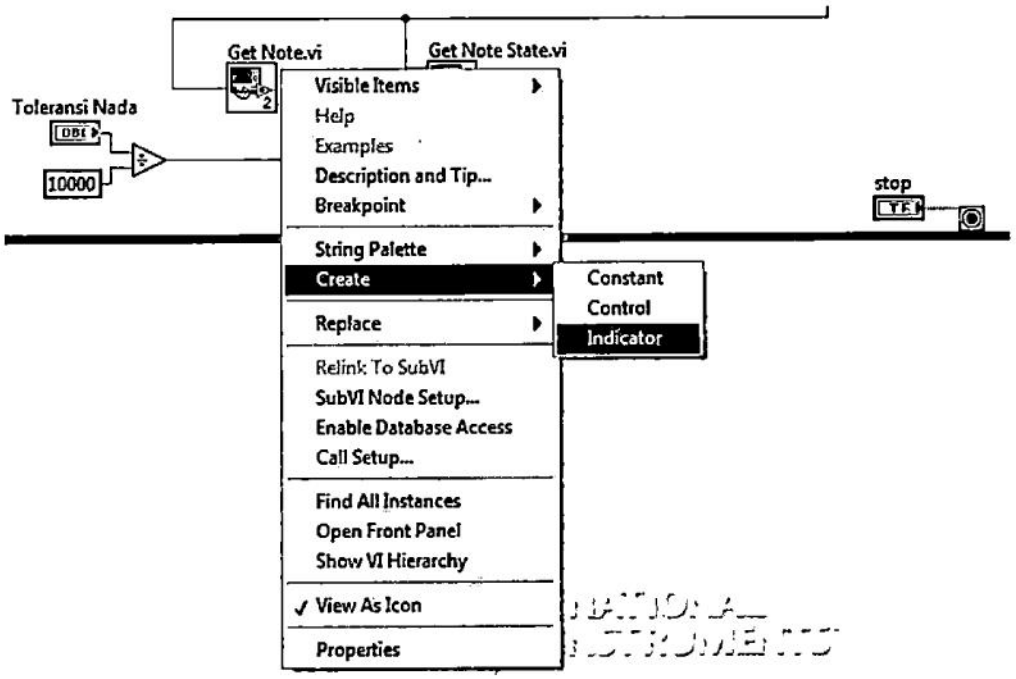

**Gambar 3.48** Tampilan ikon konstanta dengan tipe data *Double Presicion*

18. Langkah selanjutnya yaitu menghubungkan ketiga ikon komponen tersebut dengan sub VI *Get Note State.vi* seperti pada gambar di bawah ini :



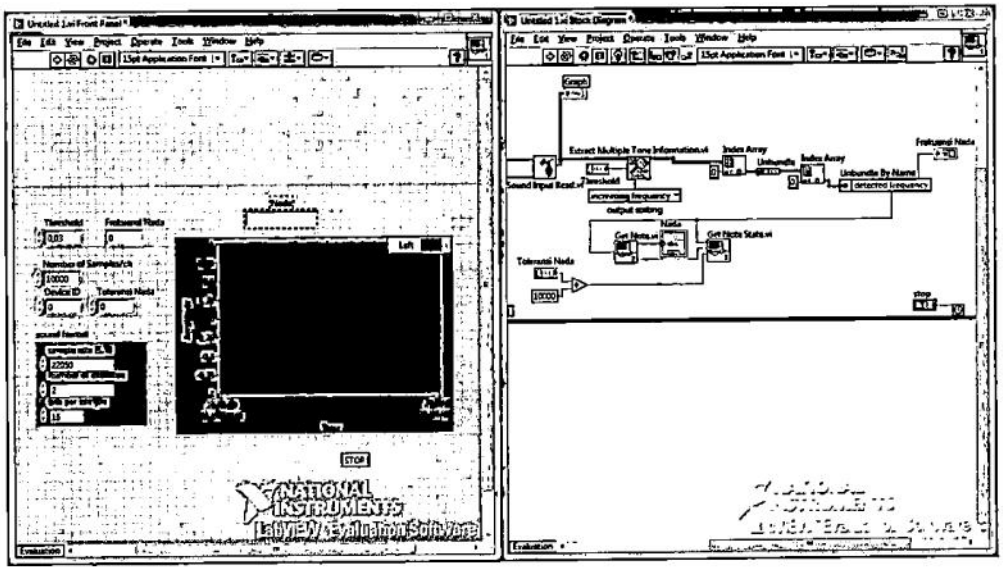

**Gambar 3.49** Menghubungkan sub VI dengan rangkaian proses

19. Selanjutnya adalah menampilkan indikator dari sub VI ke jendela Front Panel, pertama adalah menampilkan indikator dari sub VI *Get Note.vi* dengan cara **klik kanan terminal "nada" pada*Get Note.vi* → Create → Indicator** seperti pada gambar di bawah ini :



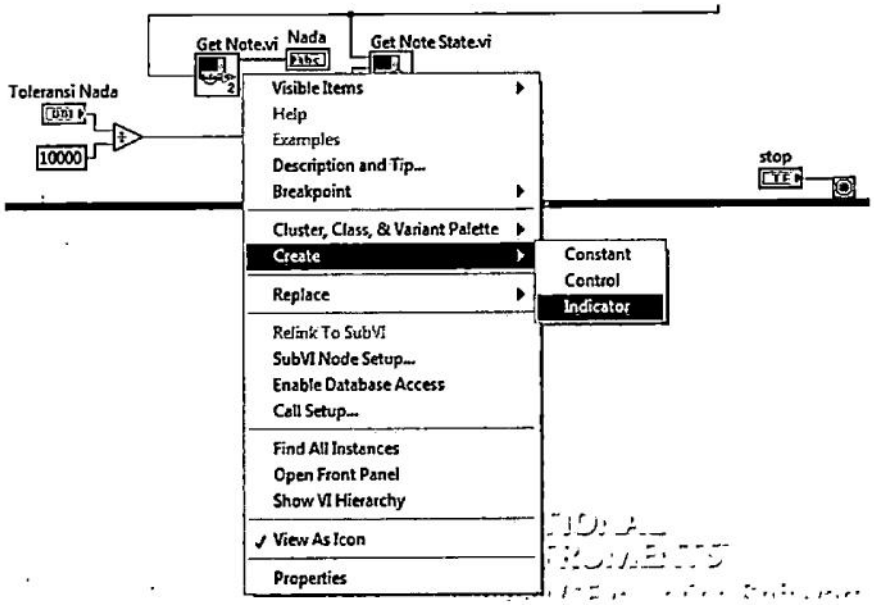

**Gambar 3.50** Menambahkan Indikator pada sub VI *Get Note.vi*

kemudian akan muncul tampilan "nada" pada jendela front panel dan ikon "nada" pada block diagram seperti pada gambar di bawah ini :



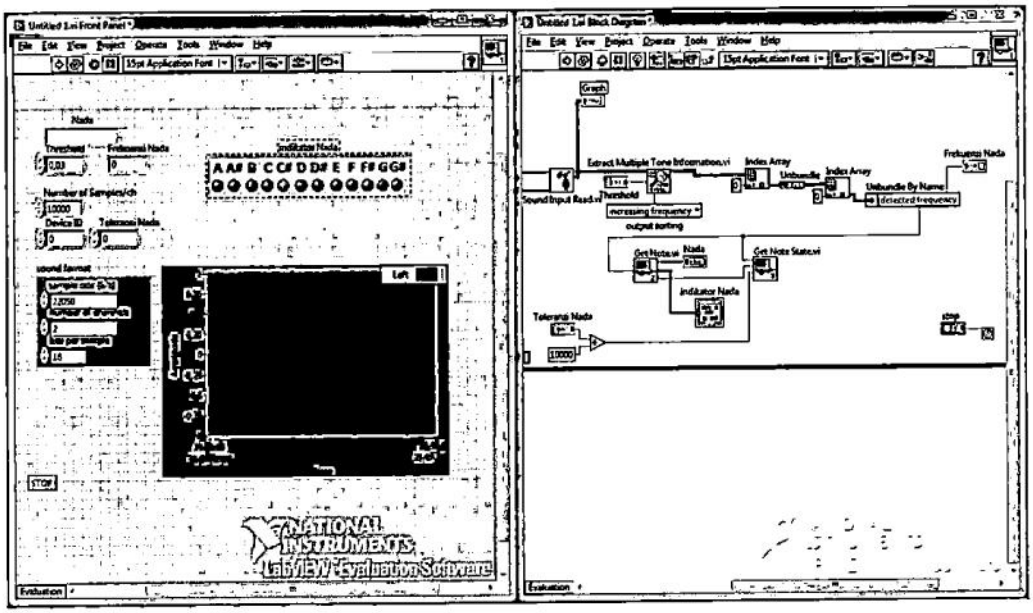
**Gambar 3.51** Tampilan Indikator pada sub VI *Get Note.vi*

Berikutnya adalah menampilkan "indikator nada" dengan cara **klik kanan terminal "indikator nada" pada*Get Note.vi* → Create → Indicator** seperti pada gambar di bawah ini :



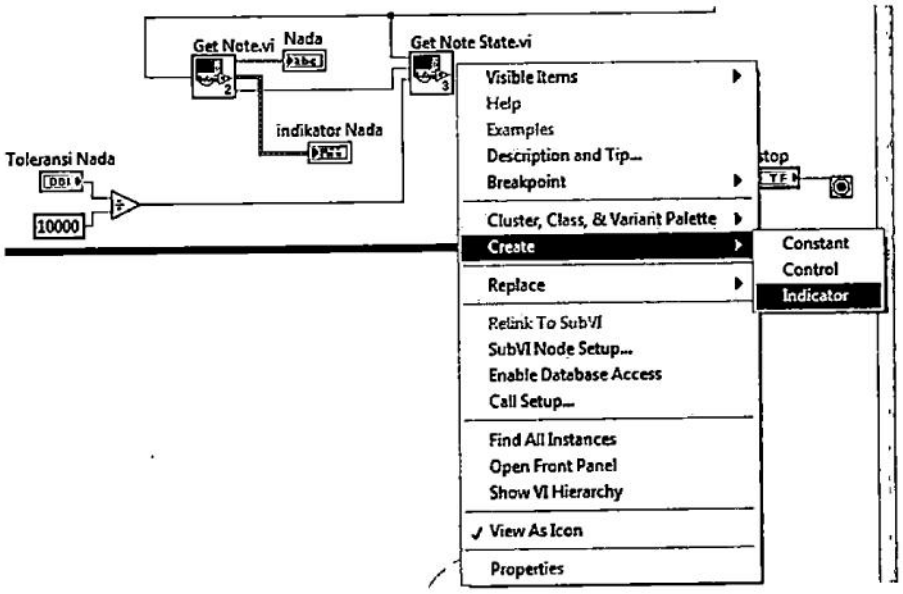

**Gambar 3.52** Menambahkan Indikator pada sub VI *Get Note.vi*

Lalu akan muncul tampilan "indikator nada" pada jendela Front Panel dan ikon "indikator nada" pada Block Diagram seperti pada gambar di bawah ini :



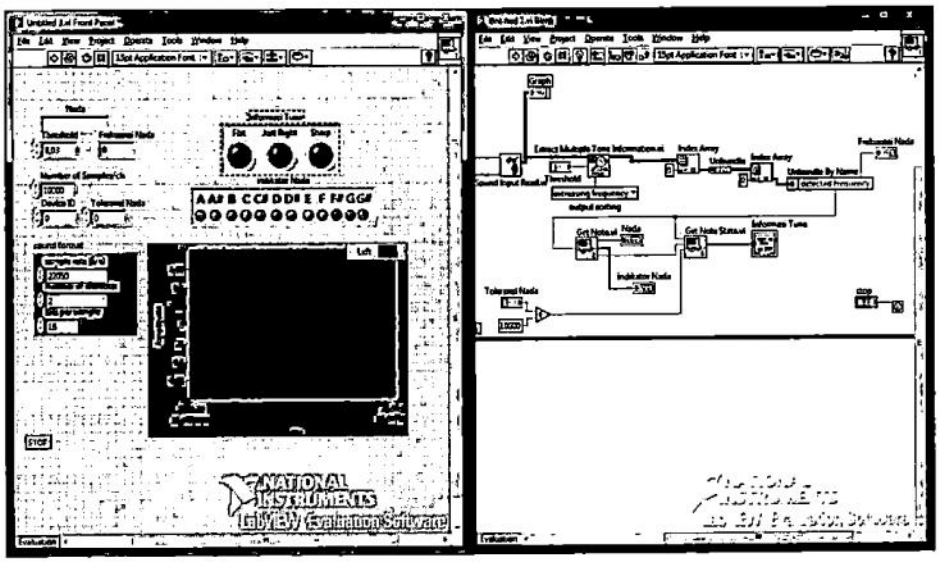
**Gambar 3.53** Tampilan indikator pada sub VI *Get Note.vi*

Selanjutnya adalah menampilkan indikator pada sub VI *Get Note State.vi*, dengan cara **klik kanan terminal "informasi tune"pada *Get Note State.vi*** → **Create→Indicator** seperti pada gambar di bawah ini :



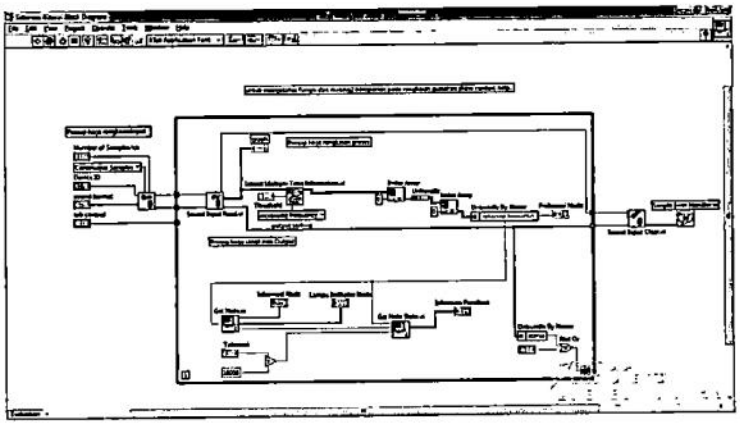

**Gambar 3.54** Menambahkan Indikator pada sub VI *Get Note State.vi*

kemudian akan muncul tampilan "Informasi Tune" pada jendela front panel dan ikon "Informasi Tune" pada Block Diagram seperti pada gambar di bawah ini :



**Gambar 3.55** Tampilan indikator pada sub VI *Get Note State.vi*

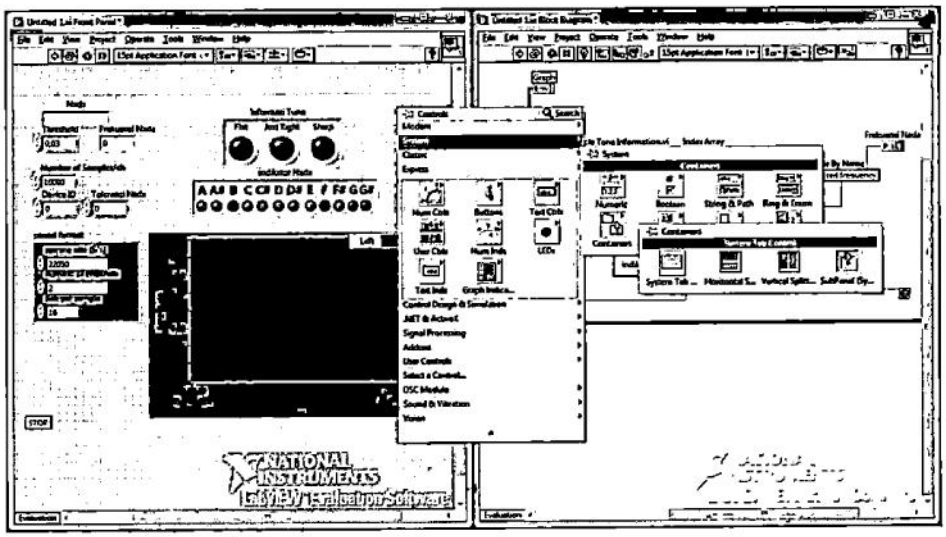
20. Setelah semua langkah tersebut dilakukan maka kita tinggal melengkapi beberapa bagian sambungan pada jendela Block Diagram dari input ke proses, sambungan error data dan sound out clear seperti pada gambar di bawah ini :



**Gambar 3.56** Tampilan rangkaian Penala Gitar LabVIEW

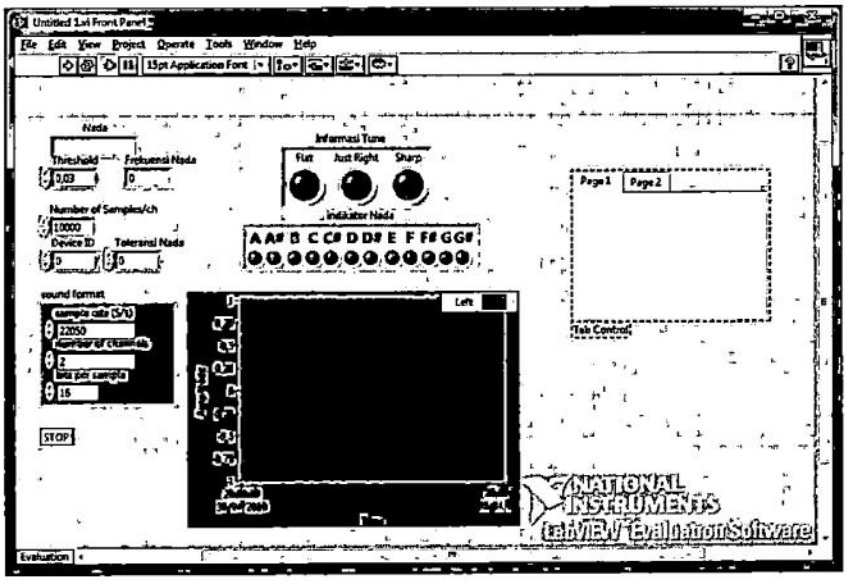
21. Untuk merapikan tampilan pada Front Panel kita gunakan "System Tab Control" agar tampilan dapat dikelompokan menjadi beberapa bagian.

**Klik kanan pada jendela front panel→System → Containers → System Tab Control** seperti pada gambar di bawah ini :



**Gambar 3.57** Library pada Front Panel

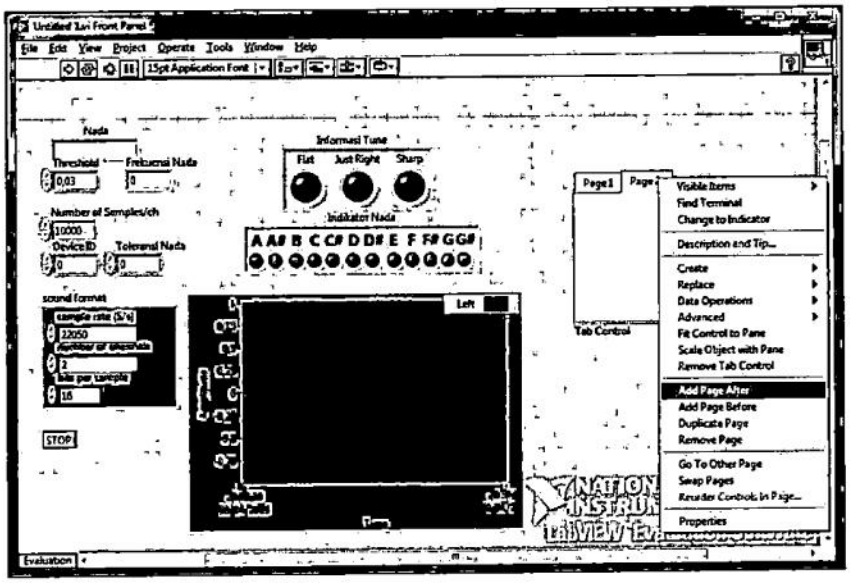
Lalu pada jendela Front Panel akan muncul tampilan Tab Control seperti pada gambar di bawah ini :




**Gambar 3.58** Tampilan Tab Control pada Front Panel

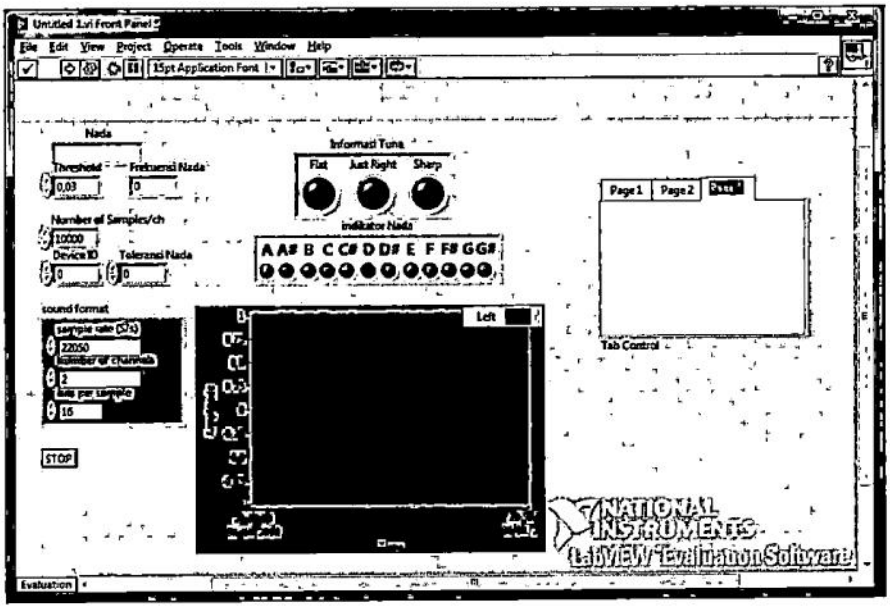
Untuk menambahkan bagian halaman pada system tab control yaitu :

**Klik kanan pada system tab controlyang bertuliskan "Page" →Add Page After / Add Page Before / Duplicate** seperti pada gambar di bawah ini :




**Gambar 3.59** Menambah jumlah halaman pada Tab Control

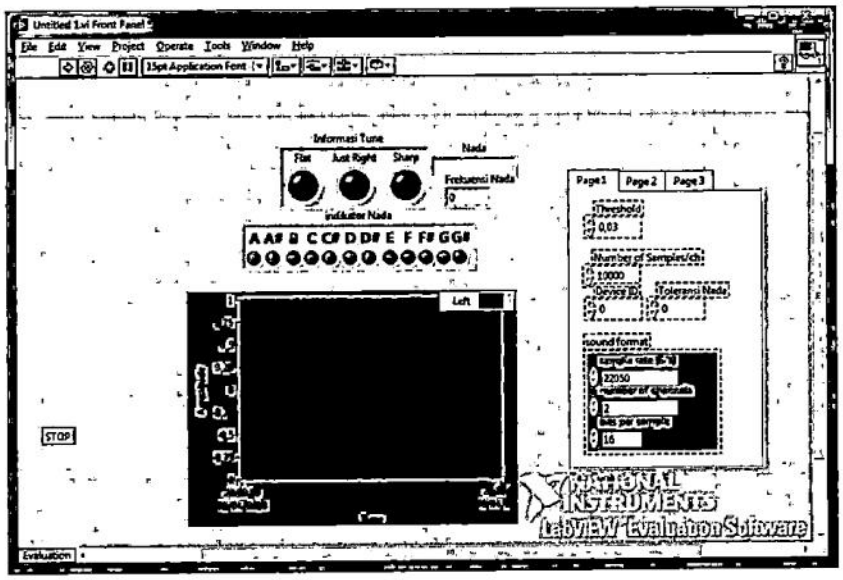
Lalu halaman pada system tab control akan bertambah seperti pada gambar di bawah ini :




**Gambar 3.60** Tampilan Tab Control pada Front Panel

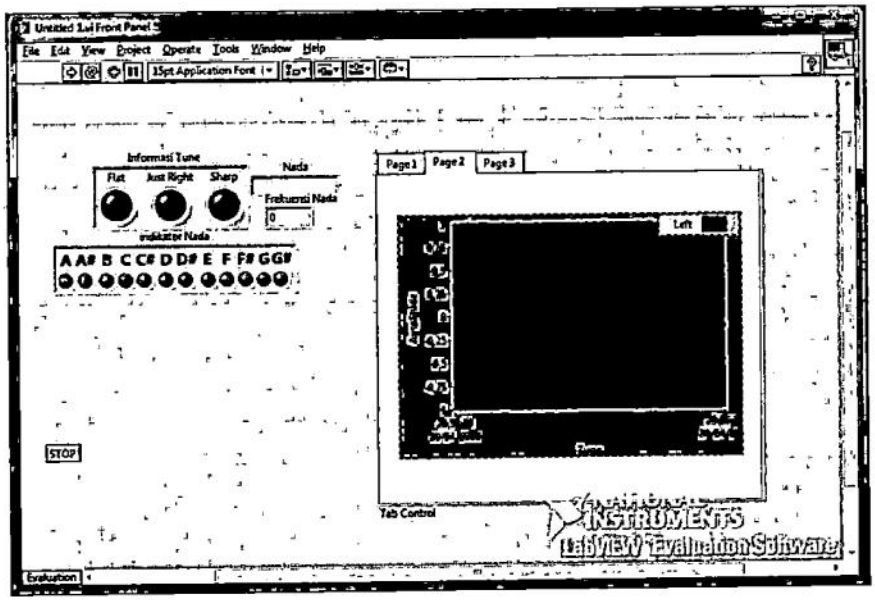
22. Selanjutnya adalah mengelompokkan tampilan sesuai dengan fungsinya masing-masing menjadi 3 bagian, yaitu bagian pertama adalah Konfigurasi Input, bagian kedua adalah Indikator Sinyal Input dan bagian ketiga adalah Indikator Penalaan.

    Untuk mengelompokanya yaitu **klik "Page 1" → klik dan seret object ke dalam system tab control** seperti pada gambar di bawah ini :



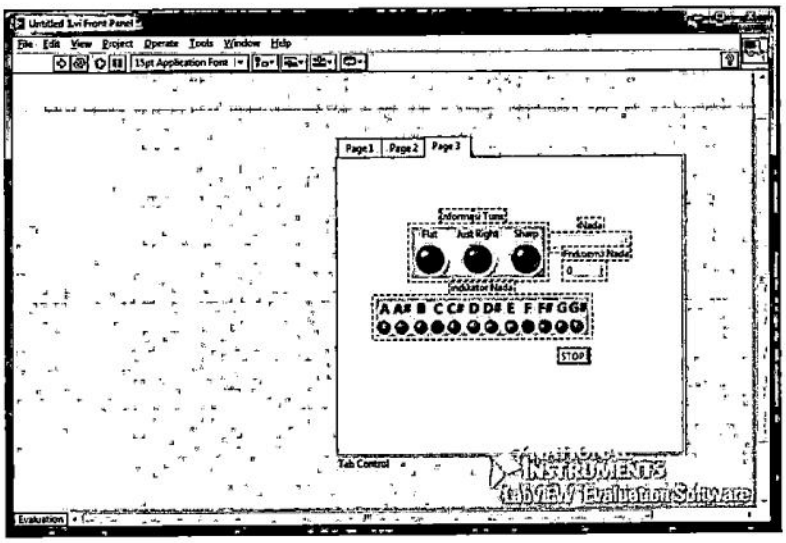

**Gambar 3.61** Memindahkan parameter Konfigurasi Input kedalam Tab Control

**klik "Page 2" → klik dan seret object ke dalam system tab control** seperti pada gambar di bawah ini :



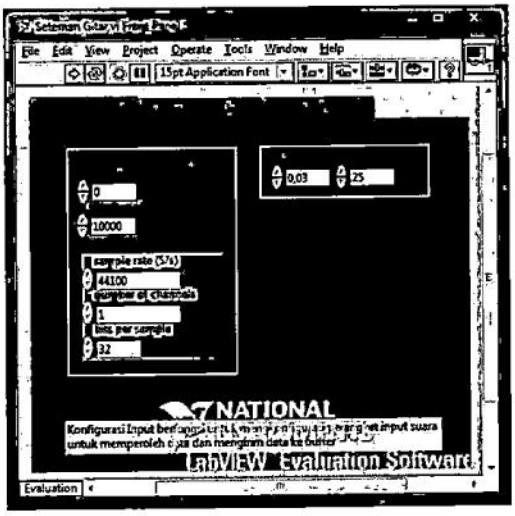

**Gambar 3.62** Memindahkan *Waveform Graph* kedalam Tab Control

**klik "Page 3" → klik dan seret object ke dalam system tab control** seperti pada gambar di bawah ini :

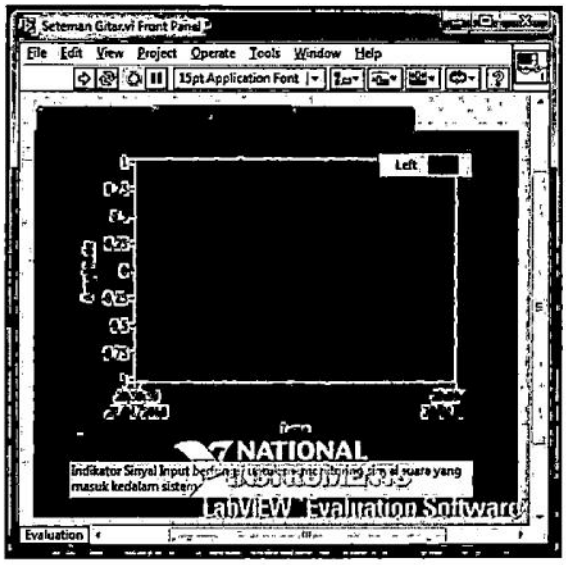



**Gambar 3.63** Memindahkan Indikator ke dalam Tab Control

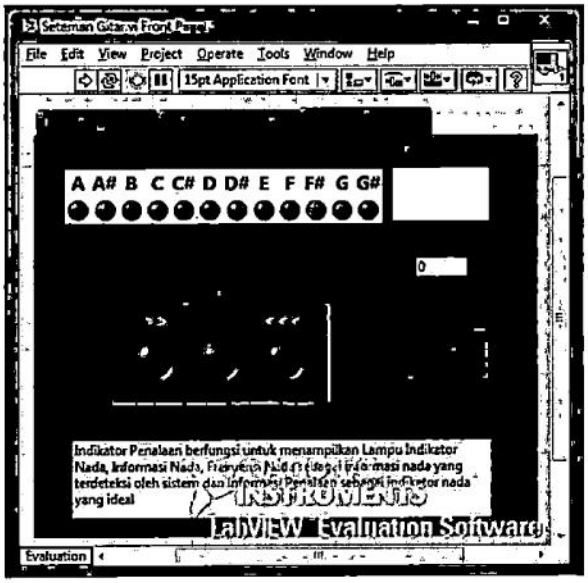
Setelah semua selesai dikelompokkan maka kita dapat mengganti nama bagian halaman (*Page*), dan menempatkan objek sesuai dengan yang kita inginkan seperti gambar di bawah ini:




**Gambar 3.64** Tampilan Tab Control pada Halaman Pertama




**Gambar 3.65** Tampilan Tab Control pada Halaman Kedua




**Gambar 3.66** Tampilan Tab Control pada Halaman Ketiga