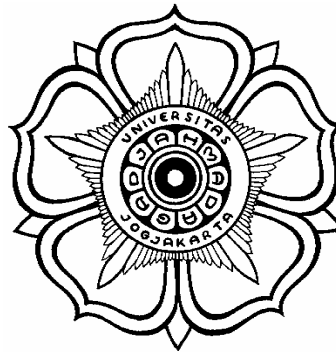


**PERANCANGAN SISTEM AKUISISI DATA
MENGUNAKAN MASUKAN *SOUNDCARD***

Skripsi

untuk memenuhi sebagian persyaratan
untuk mencapai derajat Sarjana S-1

Program Studi Fisika Teknik
Jurusan Teknik Fisika



oleh :

Hasan Murod

99/129995/TK/24473

**JURUSAN TEKNIK FISIKA
FAKULTAS TEKNIK
UNIVERSITAS GADJAH MADA
YOGYAKARTA
2005**

SKRIPSI

**PERANCANGAN SISTEM AKUISISI DATA
MENGUNAKAN MASUKAN *SOUNDCARD***

Yang dipersiapkan dan disusun oleh
Hasan Murod
99/129995/TK/24473

telah dipertahankan di depan Dewan Penguji
pada tanggal 9 Mei 2005
dan dinyatakan telah memenuhi syarat

Susunan Dewan Penguji

Penguji I,

Ir. Sunarno, M.Sc., Ph.D.
NIP. 131 281 881

Tanggal :

Penguji II,

Ir. Agus Arif, M.T.
NIP. 132 049 644

Tanggal :

Penguji III,

Ir. Balza Achmad, M.Sc.E
NIP. 132 133 730

Tanggal :

Penguji IV,

Ir. Warsun Nadjib, M.Sc.
NIP. 132 207 783

Tanggal :

**DEPARTEMEN PENDIDIKAN NASIONAL
UNIVERSITAS GADJAH MADA
FAKULTAS TEKNIK
JURUSAN TEKNIK FISIKA
PROGRAM STUDI FISIKA TEKNIK**

HALAMAN TUGAS

Nama : Hasan Murod
Nomor Mahasiswa : 99/129995/TK/24473
Pembimbing I : Ir. Sunarno, M.Sc., Ph.D.
Pembimbing II : Ir. Agus Arif, M.T.
Judul : **Perancangan Sistem Akuisisi Data Menggunakan Masukan *Soundcard***
Permasalahan : Sistem akuisisi data merupakan elemen penting dalam setiap sistem instrumentasi. Mahalnya sistem akuisisi data seringkali dipengaruhi oleh mahalnnya harga ADC (*Analog-to-Digital Converter*). Semakin tinggi resolusi ADC maka semakin mahal pula harganya. Selain harga dan resolusi, sistem akuisisi juga diharapkan mudah dipindah-pindah (*portable*). Untuk mengatasi masalah harga, portabilitas, dan resolusi dari sistem akuisisi data, maka akan dirancang sistem akuisisi data menggunakan masukan *soundcard* untuk antar muka.

Pembimbing I,

Pembimbing II,

Ir. Sunarno, M.Sc., Ph.D.
NIP 131 281 881

Ir. Agus Arif, M.T.
NIP 132 049 644

Mengetahui
Ketua Jurusan Teknik Fisika
Fakultas Teknik UGM

Dr-Ing. Kusnanto
NIP 131 695 242

Kupersembahkan Tugas Akhir ini

*Kepada orang-orang yang dengan tekun
mempelajari bermacam-macam ilmu,
mengembangkan,
mengamalkan,
dan mengajarkannya kepada orang lain.*

*Semoga mereka semua mendapat rahmat
dari Tuhan Yang Esa,
Yang memiliki sumber segala ilmu,
Yang menampakkan ayat-ayatNya
di seluruh penjuru alam semesta,
yang kepadaNya kita akan kembali.*

KATA PENGANTAR

Assalamu'alaikum Wr. Wb.

Puji syukur kehadiran Allah S.W.T. yang telah memberikan segala kenikmatan dan anugrah kesehatan, sehingga penulis dapat menyelesaikan Tugas Akhir ini sesuai rencana. Tugas Akhir ini diajukan sebagai salah satu syarat untuk mencapai gelar sarjana S1 di Jurusan Teknik Fisika Fakultas Teknik Universitas Gadjah Mada Yogyakarta.

Telah banyak tenaga dan pikiran yang penulis curahkan untuk menyelesaikan Tugas Akhir ini. Banyak kesulitan teknis maupun non teknis yang ditemui selama pengerjaan, tanpa bantuan dari orang-orang yang peduli maka penulis tidak akan mampu menyelesaikan Tugas Akhir ini. Pada kesempatan ini penulis ucapkan terimakasih setulus-tulusnya kepada Yth :

1. Bapak Dr-Ing. Kusnanto selaku ketua jurusan Teknik Fisika FT UGM.
2. Bapak Ir. Sunarno, M.Sc., Ph.D., selaku dosen pembimbing I atas bimbingannya kepada penulis.
3. Bapak Ir. Agus Arif, M.T., selaku dosen pembimbing II atas bimbingannya kepada penulis.
4. Seluruh rekan-rekan FisTek '99 atas dukungan yang diberikan selama pengerjaan tugas akhir.

Penulis hanya dapat berdoa kepada Allah S.W.T., semoga semua kebaikan yang telah diberikan kepada penulis mendapatkan balasan yang setimpal dari-Nya. Dan dengan segala kerendahan hati, penulis mengharapkan semoga tugas akhir ini dapat bermanfaat. Terimakasih.

Wassalamu'alaikum Wr. Wb.

Yogyakarta, April 2005

Penulis

DAFTAR ISI

	halaman
Halaman Judul.....	i
Halaman Pengesahan.....	ii
Halaman Tugas.....	iii
Halaman Persembahan.....	iv
Kata Pengantar.....	v
Daftar Isi.....	vii
Daftar Tabel.....	xi
Daftar Gambar.....	xii
Daftar Lampiran.....	xv
Intisari.....	xvi
<i>Abstract</i>	xvii
BAB I. PENDAHULUAN.....	1
I.1. Latar Belakang.....	1
I.2. Batasan Masalah.....	3
I.3. Tujuan.....	4
I.4. Manfaat.....	4
BAB II. TINJAUAN PUSTAKA.....	5
BAB III. DASAR TEORI.....	8
III.1. Modulasi Amplitudo.....	8
III.2. TDM (<i>Time Division Multiplexing</i>).....	8

III.3. <i>Soundcard</i>	10
III.4. Sistem Operasi Windows, <i>Multitasking</i> , <i>Multi Threading</i> , dan <i>Event-Driven</i>	12
III.5. Memori Bersama dan Objek <i>FileMapping</i>	13
BAB IV. TATA LAKSANA PENELITIAN.....	15
IV.1. Perancangan Perangkat Keras.....	15
IV.1.1. Tuntutan Perancangan.....	15
IV.1.2. Konsep Dasar Solusi Rancangan.....	16
IV.1.3. Pemilihan Jenis Multipleksi.....	16
IV.1.4. Pemilihan Jenis Modulasi.....	17
IV.1.5. Pemilihan Bentuk Gelombang Pembawa.....	17
IV.1.6. Penentuan Frekuensi Pembawa.....	18
IV.1.7. Penentuan Panjang <i>Frame</i>	19
IV.1.8. Perancangan Metode Sinkronisasi <i>Frame</i>	20
IV.1.9. Penggambaran Bentuk Sinyal.....	20
IV.1.10. Implementasi Sistem Elektronik.....	22
IV.1.11. Perancangan Subsistem Penyinkron dan Osilator	25
IV.1.12. Perancangan Subsistem TDM.....	29
IV.1.13. Perancangan Subsistem Modulator.....	30
IV.1.14. Perancangan Subsistem Pemilih Sinyal.....	33
IV.1.15. Perancangan Subsistem Catu Daya.....	34
IV.1.16. Pembuatan Prototip.....	34

IV.2. Perancangan Perangkat Lunak.....	42
IV.2.1 Tuntutan Perancangan.....	42
IV.2.2. Konsep Dasar Solusi.....	42
IV.2.3. Perancangan Aplikasi Pendekode.....	43
IV.2.4. Perancangan Kode Antarmuka dalam C++ (C++ Builder).....	48
IV.2.5. Perancangan Kode Antarmuka dalam Pascal (Delphi).....	50
IV.3. Pengujian Unjuk Kerja.....	52.
IV.3.1. Alat dan Bahan.....	52
IV.3.2. Uji Derau <i>Soundcard</i>	53
IV.3.3. Uji Akurasi dan Linearitas.....	53
IV.3.4. Uji Tanggapan Langkah.....	53
IV.3.5. Uji Tanggapan Frekuensi.....	54
IV.3.6. Uji Cakap Silang.....	54
IV.3.7. Uji Aplikasi.....	54
BAB V. HASIL PENGUJIAN DAN PEMBAHASAN.....	56
V.1. Derau <i>Soundcard</i> dan Resolusi.....	56
V.2. Akurasi dan Linearitas Sistem Akuisisi Data.....	59
V.3. Tanggapan Langkah.....	64
V.4. Tanggapan Frekuensi.....	65
V.5. Cakap Silang.....	66
V.6. Uji Aplikasi.....	67

BAB VI. KESIMPULAN DAN SARAN.....	70
VI.1. Kesimpulan.....	70
VI.2. Saran untuk Penelitian Selanjutnya.....	71
DAFTAR PUSTAKA.....	72
LAMPIRAN.....	74

DAFTAR TABEL

	halaman
Tabel V.1. Uji Akurasi dan Linearitas.....	60
Tabel V.2. Hasil Uji Akurasi dan Linearitas Mode Lambat.....	63
Tabel V.3. Hasil Uji Tanggapan Frekuensi.....	65
Tabel V.4. Hasil Uji Cakap Silang (dalam dB).....	67
Tabel V.5. Korelasi Cakap Silang.....	67

DAFTAR GAMBAR

	halaman
Gambar 3.1. Modulasi Amplitudo.....	8
Gambar 3.2. Skematik Sistem TDM Yang Disederhanakan.....	19
Gambar 3.3. Eksekusi Aplikasi dalam Sistem Operasi Windows.....	12
Gambar 3.4. Eksekusi <i>Thread</i> dalam Sistem Operasi Windows	13
Gambar 3.5. Hubungan <i>FileMapping</i> , <i>File View</i> , dan <i>File</i> dalam <i>Disk</i>	14
Gambar 4.1. Bentuk Sinyal Yang Harus Dihasilkan oleh Sistem Akuisisi Data.....	21
Gambar 4.2. Hubungan Subsistem-Subsistem dalam Sistem.....	24
Gambar 4.3. Sinyal-Sinyal Yang Harus Dihasilkan oleh Subsistem Penyinkron dan Osilator	26
Gambar 4.4. Skematik Rangkaian Elektronik Subsistem Penyinkron dan Osilator.....	27
Gambar 4.5. Skematik Rangkaian Elektronik Subsistem TDM.....	29
Gambar 4.6. Skematik Rangkaian Elektronik Penguat Terkendali Tegangan.....	30
Gambar 4.7. Skematik Rangkaian Elektronik Subsistem Modulator...	31
Gambar 4.8. Skematik Rangkaian Elektronik Subsistem Pemilih Sinyal.....	33
Gambar 4.9. Skematik Rangkaian Elektronik Subsistem Catu Daya	34

Gambar 4.10. Rangkaian Elektronik Subsistem Penyinkron	
dan Osilator.....	36
Gambar 4.11. Rangkaian Elektronik Subsistem Modulator	
dan Pemilih Sinyal.....	37
Gambar 4.12. Rangkaian Elektronik Subsistem TDM.....	37
Gambar 4.13. Rangkaian Elektronik Subsistem Catu Daya.....	37
Gambar 4.14. Bentuk Sinyal Pembawa.....	38
Gambar 4.15. Bentuk Sinyal Penyinkron.....	38
Gambar 4.16. Bentuk Sinyal Kendali Pemilih Sinyal.....	39
Gambar 4.17. Bentuk Sinyal A0.....	39
Gambar 4.18. Bentuk Sinyal A1.....	39
Gambar 4.19. Bentuk Sinyal A2.....	39
Gambar 4.20. Bentuk Sinyal TDM.....	40
Gambar 4.21. Bentuk Sinyal Keluaran Modulator	40
Gambar 4.22. Bentuk Sinyal Keluaran Pemilih Sinyal.....	41
Gambar 4.23. Konsep Dasar Solusi Rancangan Perangkat Lunak...	43
Gambar 4.24. Diagram Alir Aplikasi Pendekode.....	44
Gambar 4.25. Bentuk Sinyal dan Parameter <i>Decoding</i>	45
Gambar 4.26. Ketidakstabilan Sinyal.....	46
Gambar 5.1. Derau <i>Soundcard</i>	56
Gambar 5.2. Statistik Derau <i>Soundcard</i> dalam rms.....	57
Gambar 5.3. Statistik Derau <i>Soundcard</i>	57
Gambar 5.4. Spektrum Derau <i>Soundcard</i>	58

Gambar 5.5. Plot Rerata Hasil Bacaan Akuisisi Data	60
Gambar 5.6. Kesalahan Prediksi Regresi Linier.....	61
Gambar 5.7. Kesalahan Regresi Polinomial.....	62
Gambar 5.8. Deviasi Standar Terhadap Rerata.....	62
Gambar 5.9. Deviasi Standar Terhadap Rerata Mode Lambat.....	64
Gambar 5.10. Hasil Uji Tanggapan Langkah.....	64
Gambar 5.11. Tanggapan Frekuensi.....	66
Gambar 5.12. Kesalahan Prediksi Frekuensi.....	66
Gambar 5.13. Aplikasi Pengendali PID untuk Motor DC ($K_p=500, K_i=0,1$).....	68
Gambar 5.14. Aplikasi Pengendali PID untuk Motor DC ($K_p=800, K_i=0,05$).....	68
Gambar 5.15. Aplikasi Pengendali PID untuk Motor DC ($K_p=300, K_i=0,1$).....	69
Gambar 5.16. Aplikasi Termometer Dijalankan bersama Aplikasi Pengendali PID.....	69

DAFTAR LAMPIRAN

	halaman
Lampiran 1. Servermain.dfm.....	75
Lampiran 2. Servermain.h.....	81
Lampiran 3. Servermain.cpp.....	83
Lampiran 4. TWaveProcessor.h.....	89
Lampiran 5. TWaveProcessor.cpp.....	91
Lampiran 6. TBeep.h.....	96
Lampiran 7. TBeep.cpp.....	97
Lampiran 8. TDrawChart.h.....	98
Lampiran 9. TDrawChart.cpp.....	99
Lampiran 10. TScdaqInterface.h.....	100
Lampiran 11. TScdaqInterface.cpp.....	101
Lampiran 12. ScdaqInterface.pas	104
Lampiran 13. Rancangan Perangkat Keras Pengendali Motor DC dan Pemantau Suhu.....	108

PERANCANGAN SISTEM AKUISISI DATA MENGGUNAKAN MASUKAN *SOUNDCARD*

INTISARI

Sistem akuisisi data merupakan elemen penting dalam setiap sistem instrumentasi. Mahalnya sistem akuisisi data seringkali dipengaruhi oleh mahalnnya harga ADC (*Analog-to-Digital Converter*) di dalamnya. Semakin tinggi resolusi ADC maka semakin mahal pula harganya. Selain harga dan resolusi, sistem akuisisi juga diharapkan mudah dipindah-pindah (*portable*). Untuk mengatasi masalah harga, portabilitas, dan resolusi maka perlu diteliti rancangan sistem yang menggunakan masukan *soundcard* untuk antar muka.

Dengan menggunakan masukan *soundcard*, telah berhasil dirancang dan dibuat sebuah sistem akuisisi data banyak kanal yang bekerja dengan cara memultipleksi 16 kanal data, kemudian memodulasikan sinyal termultipleksi ke sinyal pembawa. Perangkat lunak pendekode telah dibuat agar proses dekoding berjalan efisien ketika beberapa program aplikasi mengakses sistem secara bersamaan. Untuk memudahkan pengguna mengakses sistem, skrip kode antar muka program aplikasi telah dibuat dalam bahasa C++ dan Pascal.

Hasil uji unjuk kerja menunjukkan akurasi $\pm 1,74\%$ (tingkat keyakinan 95%) pada mode cepat (100 sampel / detik) atau $\pm 0,24\%$ pada mode lambat (1 sampel / 80 mili detik); linearitas sebesar 1,5 % bacaan skala penuh dan dapat ditingkatkan menjadi 0,56% dengan koreksi polinomial; cakap silang maksimum antar kanal yang berdekatan -66,65 dB; tanggapan frekuensi 0 s/d -3dB untuk masukan 0 s/d 14,98 Hz; dan waktu mantap 30 mili detik (waktu mantap 95%) untuk masukan langkah.

Kata kunci: *Soundcard*, ADC, Sistem Akuisisi Data, Modulasi, Multipleksi, C++, Pascal.

DESIGNING A DATA ACQUISITION SYSTEM USING SOUNDCARD'S LINE-IN

ABSTRACT

Data acquisition system is an important element in every instrumentation system. The high price of data acquisition system usually caused by the high price of the ADC (Analog-to-Digital Converter) inside. The higher the ADC's resolution the higher the price. Beside the price and resolution, a data acquisition system is also preferred to be portable. To solve the price, the resolution, and the portability problem, a design of the system using soundcard's line-in as the interface needs to be searched.

Using soundcard's input, a multi channel data acquisition system has been successfully designed and built, it works with the mechanism of multiplexing 16 data channels, then modulating the multiplexed signal to a carrier signal. A decoding software has been built in order to make the decoding process runs efficiently while a number of application programs are accessing the system simultaneously. Application program interface code scripts has been built in C++ and Pascal to provide an easy way of accessing the system for the user application.

Performance test shows $\pm 1.74\%$ accuracy (95% confident level) on fast mode (100 samples per second) or $\pm 0.24\%$ accuracy on slow mode (1 sample per 80 mili second); 1.5 % full-scale reading linearity and can be improved to 0.56 % by polynomial correction; -66,65 dB maximum crosstalk between adjacent channels; 0 to -3 dB frequency response for input of 0 to 14.98 Hz; and 30 mili second settling time (95 % settling time) for step input.

Keywords: Soundcard, ADC, Data Acquisition System, Modulation, Multiplexing, C++, Pascal.

BAB I. PENDAHULUAN

I.1. Latar Belakang

Pengukuran memegang peranan yang sangat penting dalam dunia teknik. Pada tahap penelitian atau perancangan, pengukuran diperlukan untuk analisis teknik eksperimental. Pada tingkat aplikasi, misalnya pada industri proses, pengukuran diperlukan dalam pemantauan dan pengendalian suatu proses. Dengan pesatnya perkembangan teknologi komputer, saat ini hampir semua kegiatan dalam bidang teknik telah memanfaatkan komputer. Untuk dapat memanfaatkan komputer, suatu sistem pengukuran memerlukan sistem akuisisi data untuk mendapatkan data yang siap diolah secara digital.

Sistem akuisisi data yang ada sekarang ini biasanya memiliki pengubah analog ke digital (ADC, *Analog-to-Digital Converter*) yang terintegrasi di dalamnya. Beberapa sistem seperti ini terhubung ke komputer melalui port eksternal (misalnya port LPT, COM, atau USB) dan beberapa yang lain melalui *PC system bus* (misalnya bus PCI atau ISA). Keuntungan dari antar muka menggunakan port eksternal adalah sifatnya yang *portable*, sedangkan kekurangannya yaitu laju data yang rendah. Untuk antar muka menggunakan *system bus*, keuntungan yang diperoleh yaitu laju data yang tinggi, tetapi mempunyai kekurangan yaitu tidak *portable*. Bagian utama dari sistem akuisisi data adalah komponen pengubah analog ke digital (ADC). Mahalnya harga komponen ADC yang beresolusi tinggi menyebabkan mahalnya sistem akuisisi data yang baik.

Untuk mengatasi masalah harga, portabilitas, dan resolusi dari sistem akuisisi data, maka akan dirancang sistem akuisisi data yang menggunakan masukan *soundcard* untuk antar muka. *Soundcard* itu sendiri sebenarnya sudah merupakan sistem akuisisi data, tetapi khusus untuk sinyal-sinyal suara dengan pita frekuensi antara (20-20.000) Hz. Salah satu masalah yang harus dipecahkan adalah bagaimana agar *soundcard* mampu melakukan akuisisi data untuk sinyal-sinyal DC atau sinyal dengan frekuensi sangat rendah (frekuensi di bawah rentang frekuensi suara). Selanjutnya, masalah yang harus dipecahkan dalam penelitian ini adalah sulitnya mengakses *soundcard* secara langsung (*low-level*) karena dokumentasinya tidak disediakan oleh produsen *soundcard*. Solusi untuk masalah yang pertama akan diteliti dengan merancang perangkat keras tambahan, sedangkan untuk masalah kedua akan dicoba dengan merancang perangkat lunak antar muka menggunakan fungsi-fungsi API (*Application Program Interface*) dari Windows. Dari penelitian ini diharapkan dapat dihasilkan sistem akuisisi data yang murah, portabel, dan beresolusi tinggi. Dari segi harga, sistem akuisisi data diharapkan menjadi murah karena tidak memerlukan ADC tersendiri. Dari segi portabilitas, sistem ini diharapkan cukup portabel karena menggunakan antar muka melalui port eksternal (masukan *soundcard*). Dari segi resolusi, sistem ini diharapkan mampu mencapai resolusi sama dengan resolusi ADC pada *soundcard*, yaitu 16 bit (atau bahkan 24 bit pada beberapa *soundcard* tertentu).

I.2. Batasan Masalah

Dalam penelitian ini, masalah yang akan diteliti dibatasi sesuai judul yang diajukan. Judul yang diajukan adalah “Perancangan Sistem Akuisisi Data Menggunakan Masukan *Soundcard*”.

Yang dimaksud dengan perancangan di sini adalah proses mulai dari merancang di atas kertas sampai membuat prototipe yang teruji. Sistem akuisisi data yang dirancang terdiri dari perangkat keras dan perangkat lunak. Perancangan sampai pengujian prototipe perangkat keras dibatasi sampai pada fungsi kerjanya sebagai perangkat akuisisi data, tidak termasuk didalamnya faktor ergonomis, kehandalan (dalam jangka waktu lama), dan kemampurawatannya. Perancangan perangkat lunak dibatasi sampai pada penulisan kode program untuk mengakses sistem akuisisi. Penulisan kode program dibatasi menggunakan bahasa C++ dan Pascal.

Yang dimaksud dengan “Menggunakan Masukan *Soundcard*” pada judul yang diajukan adalah bahwa perangkat keras yang dirancang nantinya akan dihubungkan ke komputer melalui masukan (line-in) *soundcard*, yang sudah tersedia pada hampir semua komputer yang ada saat ini. Komputer yang dipakai dibatasi pada komputer pribadi (*PC, Personal Computer*) yang dilengkapi *soundcard* dengan sistem operasi Windows.

I.3. Tujuan

Tujuan penelitian ini adalah:

1. Menghasilkan sebuah sistem akuisisi data dengan perangkat keras yang *portable*, harga terjangkau, resolusi tinggi, yang dapat digunakan pada semua PC yang memenuhi spesifikasi tertentu.
2. Menghasilkan suatu kode program untuk memudahkan para pemakai sistem akuisisi data ini dalam merancang perangkat lunak.

I.4. Manfaat

Jika tujuan penelitian ini tercapai, maka hasil dari penelitian ini akan membawa beberapa manfaat:

1. Pada tingkat penelitian, alat yang dihasilkan akan bermanfaat dalam penelitian-penelitian yang membutuhkan pengukuran banyak variabel secara simultan untuk analisis teknik.
2. Pada tingkat aplikasi, alat yang dihasilkan bermanfaat dalam proses pengukuran untuk pemantauan dan kendali.

BAB II. TINJAUAN PUSTAKA

Soundcard pada dasarnya merupakan sistem akuisisi data untuk sinyal suara, dan telah dipakai oleh beberapa perangkat lunak untuk megemulasikan osiloskop dalam mode AC, di antaranya adalah *Softscope* (The Mathworks, 2002) dan *BIP Electronics Lab Oscilloscope* (PCplus, 2000). Penelitian yang menggunakan *soundcard* untuk berbagai macam pengukuran telah dilakukan di antaranya untuk menguji *transmission loss* akustik bahan sekat (Sumawas, 2004), getaran jembatan (Khotimah, 2004), dan kepekaan telinga (Kardianto, 2004). Pada semua penelitian tersebut, *soundcard* digunakan untuk mengukur sinyal-sinyal yang rentang frekuensinya memang dapat ditangkap oleh *soundcard*.

Penggunaan *soundcard* untuk membaca sinyal-sinyal di luar rentang tanggapan *soundcard*, yaitu sinyal DC atau sinyal-sinyal yang berfrekuensi sangat rendah tidak dijumpai dalam studi literatur yang dilakukan, tetapi dasar-dasar teori yang menunjukkan kemampuan *soundcard* untuk itu banyak dijumpai dalam literatur. Untuk dapat membaca level DC, yaitu sinyal yang rentang frekuensinya berada di luar tanggapan frekuensi *soundcard*, maka sinyal DC yang akan dibaca harus diubah sehingga dapat ditanggapi oleh *soundcard*, dan ini dilakukan dengan teknik modulasi. Analisis Fourier dari sinyal termodulasi memberikan efek penggeseran spektrum frekuensi (Carlson, 1986), sama seperti ditunjukkan oleh hasil analisis trigonometri (Comer, 1981). Penggeseran spektrum frekuensi ini penting karena *soundcard* dirancang untuk frekuensi audio. Untuk dapat membaca banyak sinyal

dilakukan dengan cara multipleksi; yang pada intinya, beberapa sinyal yang berbeda dapat dibuat berselang-seling (*interlaced*) menjadi bentuk gelombang tunggal (Carlson,1986).

Literatur untuk perancangan perangkat keras sudah banyak tersedia. Prosedur perancangan filter aktif telah tersedia untuk bermacam-macam pendekatan, antara lain Butterworth, Elliptic, dan Chebyshev (Johnson, 1980). Pengetahuan karakteristik komponen elektronik diperlukan dalam perancangan rangkaian elektronik. Salah satu referensi tentang karakteristik dari bermacam-macam komponen elektronik yang cukup baik adalah “Handbook of Solid-State Devices” (Thomason,1979).

Untuk perancangan perangkat lunak, studi pustaka difokuskan untuk mempelajari metode pengaksesan *soundcard*. Dalam bahasa C++, penggunaan fungsi-fungsi antar muka program aplikasi multimedia (*Multimedia API, Multimedia Application Program Interface*) telah digunakan dalam pengolahan sinyal digital secara *real-time* (Browning, 1997). Dokumen yang lengkap mengenai penggunaan fungsi-fungsi API untuk mengakses *soundcard* telah dipublikasikan oleh Microsoft (1996). Selain menggunakan fungsi-fungsi API, pengaksesan *soundcard* juga dapat dilakukan menggunakan antar muka DirectX (Microsoft, 1999). Jika struktur perangkat keras (register dalam *soundcard*, alamat, fungsi, dan aturannya) diketahui, maka pengaksesan *soundcard* juga dapat dilakukan pada aras rendah. Dalam literatur internet, cara ini banyak dilakukan untuk *soundcard* merek tertentu (Brodsky, 1997). Dalam bahasa Matlab, pengaksesan *soundcard* dapat dilakukan menggunakan fungsi-fungsi *data acquisition toolbox* (The Mathworks, 2002).

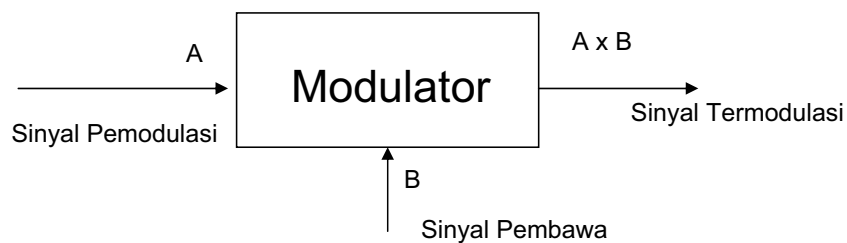
Dalam lingkungan Simulink, pengaksesan *soundcard* dapat dilakukan menggunakan blok yang ada pada *DSP Blockset* (The Mathworks, 2002).

Pada akhir perancangan, unjuk kerja sistem akan diuji untuk kemudian dianalisis. Karena sistem akuisisi data merupakan bagian dari sistem pengukuran, maka parameter-parameter yang diuji akan menggunakan terminologi sistem pengukuran. Metode pengujian karakteristik statik dan dinamik sistem pengukuran telah dibahas oleh Doebelin (1992).

BAB III. DASAR TEORI

III.1. Modulasi Amplitudo

Modulasi amplitudo adalah suatu proses untuk mengubah sinyal pembawa sehingga amplitudonya dibuat bervariasi menurut harga sesaat dari sinyal pemodulasi. Modulasi biasanya diperlukan untuk transmisi sinyal melalui medium yang tanggapan frekuensinya tidak cocok dengan frekuensi sinyal tersebut, sehingga diperlukan sinyal lain (dengan frekuensi yang cocok) untuk membawa informasi yang dikandungnya. Modulasi amplitudo dapat dilakukan dengan operasi perkalian sinyal pembawa dengan sinyal pemodulasi, seperti ditunjukkan pada Gambar 3.1.

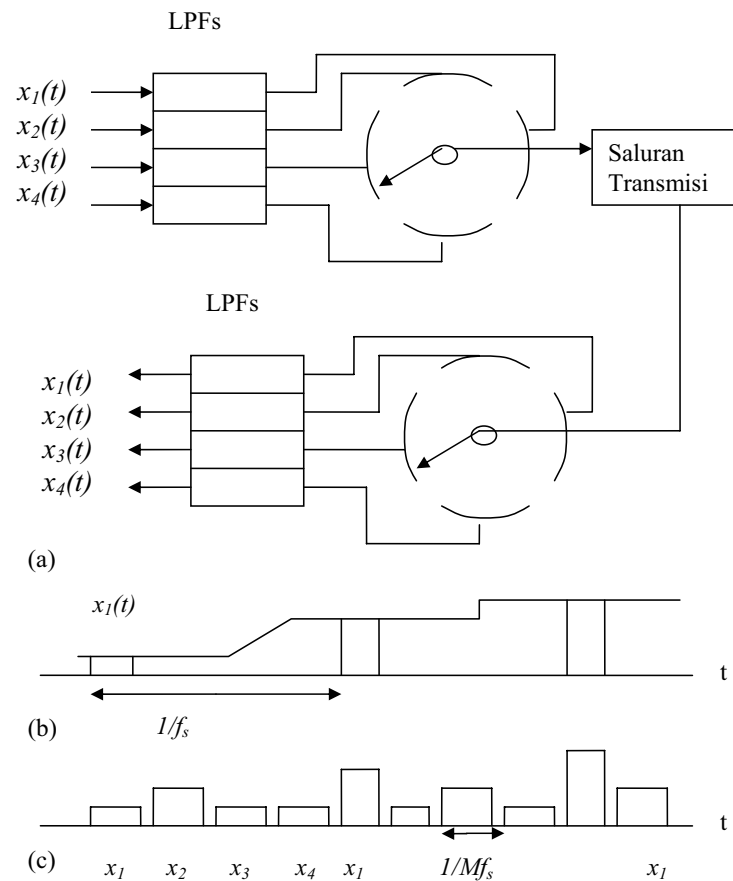


Gambar 3.1. Modulasi Amplitudo

III.2. TDM (*Time Division Multiplexing*)

TDM adalah penggabungan beberapa sinyal yang berbeda secara berselang-seling (*interlaced*) membentuk sebuah gelombang tunggal. Multipleksi ini diperlukan agar beberapa sinyal dari kanal yang berbeda dapat dikirimkan melalui

saluran tunggal untuk kemudian dipisahkan kembali sesuai dengan kanalnya



Gambar 3.2. Skematik Sistem TDM Yang Disederhanakan

masing-masing. Skematik sistem TDM yang disederhanakan ditunjukkan dalam Gambar 3.2(a). Dalam sistem tersebut, proses multipleksi dilakukan oleh saklar putar atau komutator yang mengambil satu sampel masukan tiap putaran. Sinyal tunggal $x_1(t)$ yang ditunjukkan dalam Gambar 3.2(b) digabung secara berselang-

seling dengan tiga buah sinyal yang lain membentuk sinyal termultipleksi yang ditunjukkan pada Gambar 3.2(c). Jika semua masukan mempunyai *message bandwidth* yang sama sebesar W , maka komutator harus berputar pada laju $f_s \geq 2W$ sehingga sampel yang berurutan dari tiap satu input akan mempunyai spasi sebesar $T_s = 1/f_s \leq 1/2W$. Interval waktu T_s yang berisi satu sampel dari tiap input dinamakan *frame*. Jika terdapat sejumlah M kanal masukan, maka satu pulsa ke pulsa berikutnya dalam satu *frame* akan mempunyai spasi sebesar $T_s/M = 1/Mf_s$. Oleh sebab itu, jumlah keseluruhan pulsa per detik akan sama dengan $r = Mf_s \geq 2MW$, yang merepresentasikan laju pulsa atau laju pensinyalan (*signaling rate*) dari sinyal TDM.

III.3. Soundcard

Soundcard adalah peralatan tambahan dalam sistem PC (*personal computer*) untuk memasukkan dan mengeluarkan sinyal suara. Komponen utama *soundcard* adalah ADC (*Analog-to-Digital Converter*) dan DAC (*Digital-to-Analog Converter*). *Soundcard* pada PC yang tersedia saat ini kebanyakan memenuhi spesifikasi PC multimedia level 2¹⁾:

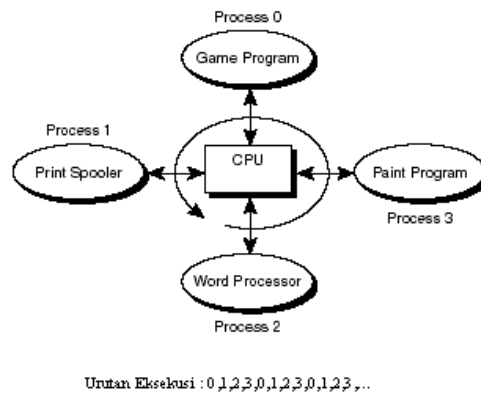
1. *Driver* CD-ROM dengan keluaran CD-DA dan kontrol volume
2. DAC 16-bit dengan karakteristik :
 - 2.a. Pencuplikan *PCM* (*Pulse Code Modulation*) yang linear.
 - 2.b. Kemampuan transfer DMA atau FIFO tersangga dengan interupsi pada kekosongan penyangga.

¹⁾) Spesifikasi ini dibuat oleh *Multimedia PC Marketing Council*, 1730 M Street NW, Suite 707 Washington, DC 20036

- 2.c. Laju pencuplikan sebesar 44,1; 22,05; dan 11,025 kHz.
 - 2.d. Kanal yang bersifat stereo.
 - 2.e. *Bandwidth* CPU yang terpakai kurang dari atau sama dengan 10% ketika mengeluarkan suara pada laju sampel 22,5 atau 11,025 kHz, atau 15% pada 44,1 kHz.
3. ADC 16-bit dengan karakteristik sebagai berikut:
 - 3.a. Pencuplikan PCM yang linear.
 - 3.b. Kemampuan transfer DMA atau FIFO tersangga dengan interupsi pada kekosongan penyangga.
 - 3.c. Laju pencuplikan sebesar 44,1; 22,05; dan 11,025 kHz.
 - 3.d. Masukan mikrofon.
 4. Kemampuan *synthesizer* internal dengan *multivoice*, *multitimbral*, yakni tujuh nada melodi simultan ditambah dengan dua nada perkusi simultan.
 5. Pencampuran internal dengan kemampuan sebagai berikut:
 - 5.a. Dapat mengkombinasikan tiga sumber suara dan menampilkannya sebagai keluaran stereo, dengan taraf *line-out* pada panel belakang.
 - 5.b. Sumber pencampuran adalah CD audio, *synthesizer*, dan DAC.
 - 5.c. Tiap sumber pencampuran mempunyai kontrol volume 3-bit dengan kenaikan logaritmik.

III.4. Sistem Operasi Windows, *Multitasking*, *Multi Threading*, dan *Event-Driven*

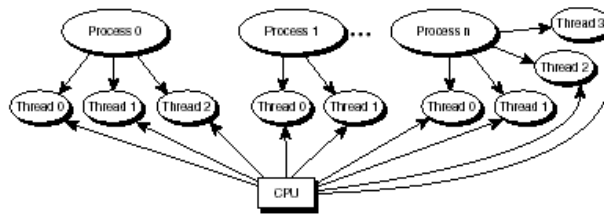
Sistem operasi Windows memungkinkan sejumlah aplikasi untuk dijalankan bersama-sama (Gambar 3.3). Dalam operasi sebenarnya, tiap aplikasi mendapatkan jatah *slot* waktu bergantian secara cepat, sehingga semua aplikasi seolah berjalan bersama-sama.



Gambar 3.3. Eksekusi Aplikasi dalam Sistem Operasi Windows

(Lamothe, 1999)

Jika dilihat lebih dalam, sistem operasi Windows tidak hanya bersifat “*multitasking*” tetapi juga bersifat “*multithreaded*”. Ini berarti program-program tersusun dari sejumlah eksekusi “*thread*” yang lebih sederhana (seperti ditunjukkan Gambar 3.4).



Gambar 3.4. Eksekusi *Thread* dalam Sistem Operasi Windows

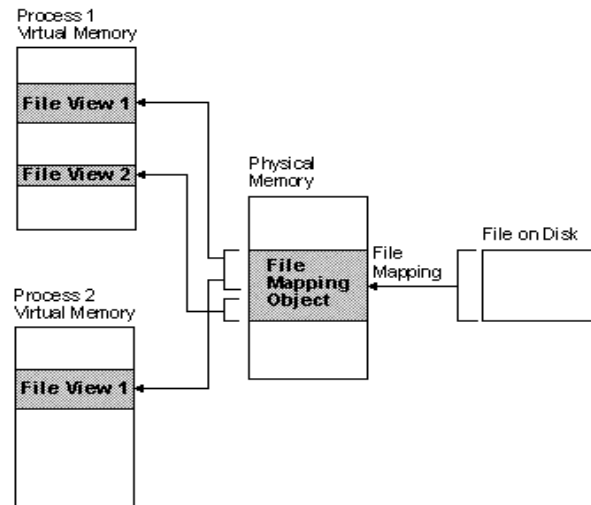
(Lamothé, 1999)

Selain bersifat *multitasking* dan *multithreading*, Windows juga bersifat “*event-driven*”. Tidak seperti dalam DOS, kebanyakan program-program dalam Windows tinggal diam dan menunggu pengguna melakukan sesuatu, yang akan memicu “*event*”, baru kemudian Windows akan menanggapi *event* tersebut.

III.5. Memori Bersama dan Objek *FileMapping*

Memori bersama digunakan untuk berbagi data antar aplikasi. *Harddisk* dan RAM (*Random Access Memory*) adalah contoh media yang dapat digunakan untuk berbagi data. *Harddisk* dapat digunakan untuk berbagi data antar aplikasi dengan objek *File*, sedangkan RAM dapat digunakan untuk berbagi data dengan menggunakan objek memori bersama (*shared memory*). Memori bersama adalah memori yang alamatnya dapat diakses oleh banyak aplikasi. Dalam Windows, Objek *FileMapping* digunakan untuk mengimplementasikan objek memori bersama. Untuk dapat mengakses objek *FileMapping*, aplikasi atau *process* dapat memetakan alamat *FileMapping* ke ruang alamat (*address space*) *process* dengan membuat *file*

view. Hubungan *FileMapping*, *File View*, dan *File* dalam *Disk* ditunjukkan dalam Gambar 3.5.



Gambar 3.5. Hubungan *FileMapping*, *File View*, dan *File* dalam *Disk*

(Microsoft Corp., 1996)

Jika *Handle* dari *file* dalam *disk* (*File On Disk*) digunakan dalam parameter penciptaan objek *FileMapping*, maka isi *file* akan dimuat ke dalam RAM dan dapat diakses oleh banyak *process*. Jika objek *FileMapping* dibuat tanpa parameter *Handle file* dalam *disk* maka objek tersebut tidak memiliki asosiasi ke *file* tertentu dalam *disk* dan hanya berfungsi sebagai penyimpanan sementara yang digunakan untuk berbagi data menggunakan RAM.

BAB IV. TATA LAKSANA PENELITIAN

Tata laksana penelitian terdiri dari perancangan perangkat keras (Bab IV.1), perancangan perangkat lunak (Bab IV.2), dan pengujian unjuk kerja sistem (Bab IV.3).

IV.1. Perancangan Perangkat Keras

Untuk dapat merancang perangkat keras, yang pertama kali harus dilakukan adalah merumuskan tuntutan perancangan (Bab IV.1.1). Setelah tuntutan perancangan dirumuskan, konsep dasar solusinya dapat ditentukan (Bab IV.1.2). Jika konsep dasar solusi telah ditentukan, maka konsep-konsep detil teknisnya kemudian dapat ditentukan (Bab IV.1.3 sampai dengan Bab IV.1.9). Setelah konsep-konsep detil teknisnya ditentukan, perangkat keras kemudian dapat dirancang (Bab IV.1.10 sampai dengan Bab IV.1.15). Untuk memastikan bahwa rancangan dapat bekerja seperti yang diharapkan maka perlu dibuat prototipnya (Bab IV.1.16).

IV.1.1. Tuntutan Perancangan

Soundcard dirancang untuk akuisisi sinyal suara dengan frekuensi antara 20 sampai dengan 20.000 Hz. Untuk dapat mengakuisisi sinyal dengan frekuensi kurang dari 20 Hz menggunakan masukan *soundcard* maka diperlukan

perangkat tambahan yang berfungsi untuk mengubah sinyal tersebut menjadi sinyal dengan frekuensi yang dapat ditangkap oleh *soundcard*.

Soundcard pada PC Multimedia setidaknya mempunyai satu buah masukan *line-in* yang bersifat stereo, sehingga mempunyai dua kanal masukan yaitu kanal kanan dan kiri. Laju *sampling soundcard* pada umumnya cukup tinggi (44100 sampel per detik). Karena sinyal yang akan diakuisisi mempunyai dinamika yang lambat, maka pembacaan banyak kanal dapat dilakukan untuk mendapatkan manfaat dari laju *sampling* yang tinggi.

IV.1.2. Konsep Dasar Solusi Rancangan

Untuk dapat membaca sinyal dari banyak kanal, solusinya adalah dengan memultipleksi sinyal-sinyal sinyal tersebut. Agar dapat ditangkap oleh *soundcard*, sinyal termultipleksi tersebut kemudian dimodulasikan ke sinyal pembawa yang frekuensinya sesuai dengan rentang tanggapan *soundcard*.

IV.1.3. Pemilihan Jenis Multipleksi

Jenis multipleksi analog yang mungkin diterapkan untuk saluran transmisi kabel tunggal adalah multipleksi pembagian waktu (TDM, *Time Division Multiplexing*) dan multipleksi pembagian frekuensi (FDM, *Frequency Division Multiplexing*). Jenis multipleksi yang dipilih dalam rancangan ini adalah multipleksi pembagian waktu (TDM). Jenis TDM dipilih karena kesederhanaan proses *encoding*

dan *decodingnya*²⁾. Dari segi komputasi, memisahkan data berdasarkan urutan waktu jauh lebih cepat dari pada memisahkan data sesuai frekuensinya karena pada pemisahan frekuensi dibutuhkan algoritma penapisan atau FFT (*Fast Fourier Transform*). Dari segi *encoding*, pembuatan modulator untuk FDM jauh lebih rumit dari pada pembuatan saklar untuk TDM (saat ini, saklar elektronik dalam bentuk rangkaian terintegrasi tersedia di pasaran dengan harga yang ekonomis).

IV.1.4. Pemilihan Jenis Modulasi

Dalam rancangan ini, jenis modulasi yang digunakan adalah modulasi amplitudo. Pemilihan jenis modulasi terutama didasarkan pada kecepatan *decoding* dalam hubungannya dengan resolusi. Jika digunakan modulasi frekuensi dan diharapkan dengan resolusi 16 bit, maka metode *decoding* secara penghitungan perioda (yang menurut hemat penulis merupakan metode tercepat) akan membutuhkan data sebanyak 2^{16} sampel, sehingga akan memakan waktu akuisisi sebesar $2^{16} \times T_s$, dengan T_s adalah periode *sampling* dari *soundcard*. Jika digunakan modulasi amplitudo maka satu sampel sudah memiliki resolusi 16 bit, sehingga hanya membutuhkan waktu satu periode *sampling*.

IV.1.5. Pemilihan Bentuk Gelombang Sinyal Pembawa

Dalam rancangan ini digunakan sinyal dengan bentuk gelombang kotak sebagai sinyal pembawa. Alasan utama dipilihnya gelombang kotak adalah karena sepanjang siklusnya hanya terdiri dari dua fasa, sehingga memudahkan proses

²⁾ Dalam tulisan ini, istilah *demultiplexing* dan *demodulation* banyak digantikan dengan istilah *decoding* karena proses keduanya dilakukan secara bersamaan di dalam perangkat lunak, sehingga dipakai istilah *decoding*, yang lazim dalam terminologi perangkat lunak.

decoding karena tidak membutuhkan deteksi fasa yang rumit. Jika dipilih gelombang sinus maka untuk mengetahui fasa simpangan maksimumnya diperlukan deteksi yang lebih rumit karena fasa tersebut belum tentu bersesuaian dengan sampel yang bernilai maksimum, mengingat sampel tersebut bersifat diskrit sementara sinyal yang dicuplik sebenarnya bersifat kontinyu.

IV.1.6. Penentuan Frekuensi Pembawa

Dalam rancangan ini digunakan frekuensi sekitar 1,8 kHz. Penentuan frekuensi pembawa terutama ditentukan oleh kemampuan *soundcard* dalam menanggapi sinyal pembawa yang berbentuk sinyal kotak. Untuk memperoleh laju data yang tinggi atau jumlah kanal yang banyak diperlukan frekuensi pembawa yang tinggi. Di sisi lain, frekuensi pembawa tidak boleh terlalu tinggi untuk menghasilkan bentuk gelombang yang baik ketika dibaca oleh *soundcard*. Kriteria ideal bentuk gelombang yang baik dalam rancangan ini adalah tidak mengalami cacat akibat tanggapan *soundcard* yang jelek. Dari pengalaman penulis menggunakan *soundcard* dalam pengukuran sinyal gelombang kotak, hasil bacaan selalu mengalami cacat frekuensi tinggi meski sinyal masukan berfrekuensi rendah. Ini disebabkan karena gelombang kotak dengan frekuensi serendah berapapun selalu mempunyai harmonik frekuensi tinggi dalam komponen frekuensinya. Untuk alasan praktis, kriteria dari gelombang kotak yang baik adalah gelombang kotak yang setidak-tidaknya yang pada separuh waktu fase (baik fase tinggi maupun fase rendah) telah mencapai keadaan mantap.

IV.1.7. Penentuan Panjang *Frame*

Akuisisi sampel suara dari *soundcard* menggunakan fungsi *Windows's API* merupakan akuisisi sampel suara dengan metode tersangga (*buffered*), yang artinya suatu aplikasi secara *real-time* menerima kumpulan sampel suara yang tersusun dalam suatu lokasi memori penyangga (*buffer*). Istilah *real-time* di sini mengacu pada waktu diterimanya sekelompok sampel, sehingga tidak benar-benar *real-time* tetapi hanya pendekatan. Semakin kecil ukuran *buffer* maka semakin mendekati *real-time*. Fungsi *Windows's API* untuk pengaksesan *soundcard* memungkinkan pengaturan ukuran *buffer* sekecil-kecilnya, tetapi pada prakteknya, tiap aplikasi yang akan memproses sampel dalam *buffer* hanya dilayani tiap 10 mili detik oleh sistem operasi, sehingga terjadi penumpukan *buffer* yang tertunda pemrosesannya. Dalam praktek, pemilihan panjang buffer kurang dari 10 mili detik akan menimbulkan efek ketidakrataan waktu (karena penumpukan *buffer*) dan kehilangan sampel (ada sampel suara yang tidak terakuisisi). Untuk menghindari hal-hal yang tidak diinginkan tersebut, panjang *buffer* untuk akuisisi ditentukan 10 mili detik.

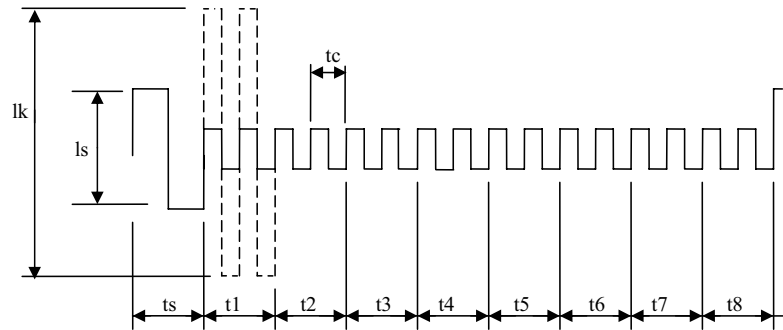
Panjang *frame* menentukan laju *sampling* sistem TDM tiap detik, yang berarti juga laju *sampling* data tiap detik untuk tiap kanal. Semakin pendek *frame* maka laju *sampling* semakin cepat. Untuk memaksimalkan laju *sampling*, maka panjang *frame* diatur agar tiap mengakses *buffer* dari *soundcard* didapatkan satu *frame* yang utuh, sehingga panjang *frame* diatur ≤ 10 mili detik. Dengan demikian, laju *sampling* 100 data per detik (tiap kanal) dapat dicapai.

IV.1.8. Perancangan Metode Sinkronisasi *Frame*

Untuk dapat memisahkan kembali sinyal yang termultipleksi, demultiplekser memerlukan sinyal sinkronisasi yang disertakan di dalam sinyal TDM sebagai penanda awal *frame*. Dalam sistem TDM pada umumnya, sinyal sinkronisasi menempati salah satu kanal tetapi memiliki nilai yang berbeda dari rentang sinyal seluruh kanal lainnya. Jika perbedaan taraf sinyal kanal data dan taraf sinyal sinkronisasi tidak cukup besar, maka akan ada kemungkinan kesalahan deteksi sinyal sinkronisasi karena derau, sehingga biasanya taraf sinyal sinkronisasi akan dibuat cukup jauh dari rentang sinyal kanal-kanal data. Dalam rancangan menggunakan *soundcard* untuk deteksi, metode seperti itu tidak menguntungkan karena akan mengurangi resolusi sinyal pada kanal yang dibutuhkan (kanal yang bukan untuk sinkronisasi). Untuk mengatasi masalah ini, dirancanglah metode sinkronisasi di luar sistem TDM dan modulator. Sinkronisasi dilakukan dengan menyisipkan sinyal sinkronisasi ke dalam sinyal pembawa termodulasi sinyal TDM (keluaran modulator) pada akhir *slot* waktu kanal 8 (akhir *frame*). Sinyal yang disisipkan ini mempunyai taraf setengah dari taraf maksimum sinyal pembawa termodulasi TDM dan mempunyai periode dua kali periode sinyal pembawa. Setelah satu periode sinyal sinkronisasi ini, sistem TDM kembali membaca kanal 1 (*frame* selanjutnya).

IV.1.9. Penggambaran Bentuk Sinyal

Bentuk sinyal yang harus dihasilkan oleh sistem akuisisi data ditunjukkan dalam Gambar 4.1.



Gambar 4.1. Bentuk Sinyal Yang Harus Dihasilkan oleh Sistem Akuisisi Data

Keterangan Gambar 4.1 :

t_s : slot waktu sinyal sinkronisasi = periode sinyal sinkronisasi

$t_1 \dots t_8$: slot waktu kanal 1 ...kanal 8

t_c : periode sinyal pembawa

l_k : taraf sinyal kanal maksimum

l_s : taraf sinyal sinkronisasi = $\frac{1}{2} l_k$

$t_s + t_1 + t_2 + t_3 + t_4 + t_5 + t_6 + t_7 + t_8 = \text{panjang frame}$

Dalam perancangan sistem elektronik, pada kasus dengan metode perancangan yang tidak diketahui karena sulit didapat dalam literatur, penggambaran bentuk sinyal dalam diagram waktu sering digunakan untuk membantu menentukan aturan-aturan yang dapat diterjemahkan ke dalam rangkaian elektronik. Dalam praktek, gambar bentuk sinyal dalam diagram waktu ini dibuat lebih dahulu secara intuitif setelah merumuskan tuntutan perancangan (IV.1.1), sebelum melakukan langkah-langkah yang diuraikan dalam bab IV.1.2 sampai dengan IV.1.8. Uraian

langkah-langkah tersebut adalah usaha penulis untuk menjelaskan secara sistematis proses penggambaran sinyal yang dilakukan secara intuitif tersebut, dengan harapan metode perancangannya dapat dipelajari oleh orang lain.

IV.1.10. Implementasi Sistem Elektronik

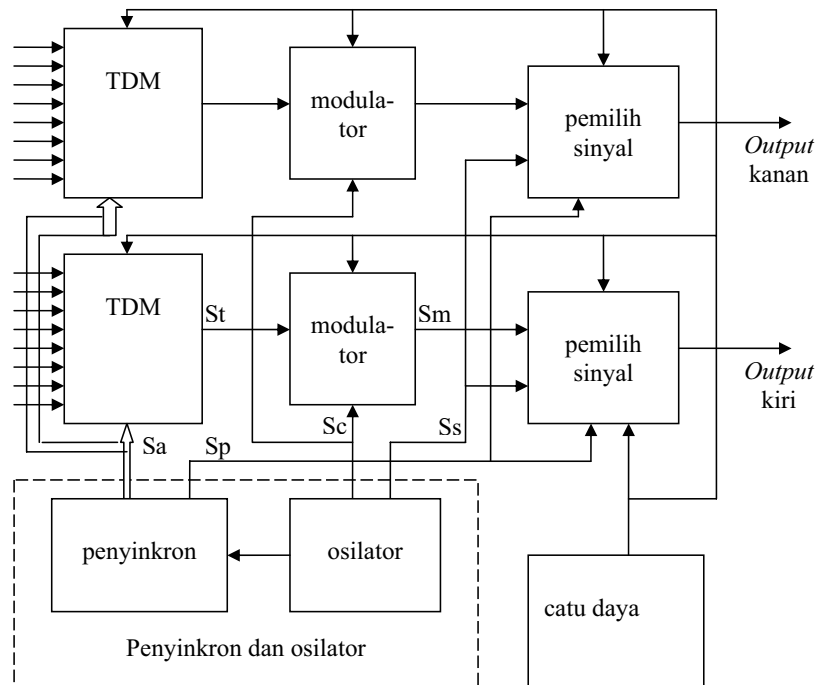
Dengan mengamati bentuk sinyal dalam diagram waktu, aturan-aturan dasar perancangan sistem elektronik dapat disusun sebagai berikut:

- a. Sinyal pembawa merupakan gelombang kotak yang kemudian akan dimodulasi oleh sinyal kanal. Ini berarti dibutuhkan sumber sinyal gelombang kotak sebagai sinyal pembawa dan modulator untuk memodulasi sinyal pembawa.
- b. Pada tiap *slot* waktu yang berbeda (t_1 sampai dengan t_8), sinyal pembawa dimodulasi oleh sinyal pemodulasi dari kanal yang berbeda. Ini berarti dibutuhkan sistem TDM untuk menyusun sinyal dari semua kanal ke dalam *slot* waktu yang berbeda-beda.
- c. Sinyal sinkronisasi merupakan gelombang kotak dengan periode dua kali sinyal pembawa, dengan titik pergantian fasa yang sinkron dengan sinyal pembawa. Berarti sinyal sinkronisasi dapat dibangkitkan dengan flip-flop T (*toggle flip-flop*) yang masukannya berasal dari sinyal pembawa.
- d. Pergantian kanal dari masukan pemodulasi adalah sinkron dengan pergantian fasa sinyal pembawa. Ini berarti dibutuhkan rangkaian penyinkron untuk menyinkronkan sistem TDM.

Dari aturan-aturan dasar perancangan sistem elektronik tersebut, subsistem penyusun sistem elektronik dapat ditentukan terdiri dari:

1. Osilator, yang berfungsi membangkitkan sinyal gelombang kotak untuk sinyal pembawa dan sinyal penyinkron.
2. Modulator, yang berfungsi untuk memodulasi sinyal pembawa, dengan sinyal TDM sebagai sinyal pemodulasinya.
3. Pemilih sinyal, berfungsi memilih apakah sinyal termodulasi atau sinyal sinkronisasi yang akan ditampilkan sebagai sinyal keluaran.
4. Penyinkron, untuk membangkitkan sinyal kendali TDM (yang berupa sinyal digital alamat kanal) agar sistem TDM berjalan sinkron dengan sinyal-sinyal pembawa dan sinyal penyinkron. Penyinkron juga berguna untuk mengendalikan pemilih sinyal agar dapat memilih sinyal yang tepat pada waktu yang tepat.
5. Rangkaian catu daya, berfungsi untuk memberi daya listrik semua subsistem.

Hubungan subsistem-subsistem dalam membentuk sistem digambarkan dalam Gambar 4.2.



Gambar 4.2. Hubungan Subsistem-Subsistem dalam Sistem

Keterangan Gambar 4.2:

- | | |
|---|--|
| St : sinyal TDM | Sp : sinyal kendali pemilih sinyal |
| Sm : sinyal hasil modulasi (sinyal pembawa termodulasi TDM) | Sc : sinyal pembawa |
| Sa : sinyal kendali alamat | Ss : sinyal penyinkron |
| | <i>Output</i> : sinyal keluaran sistem |

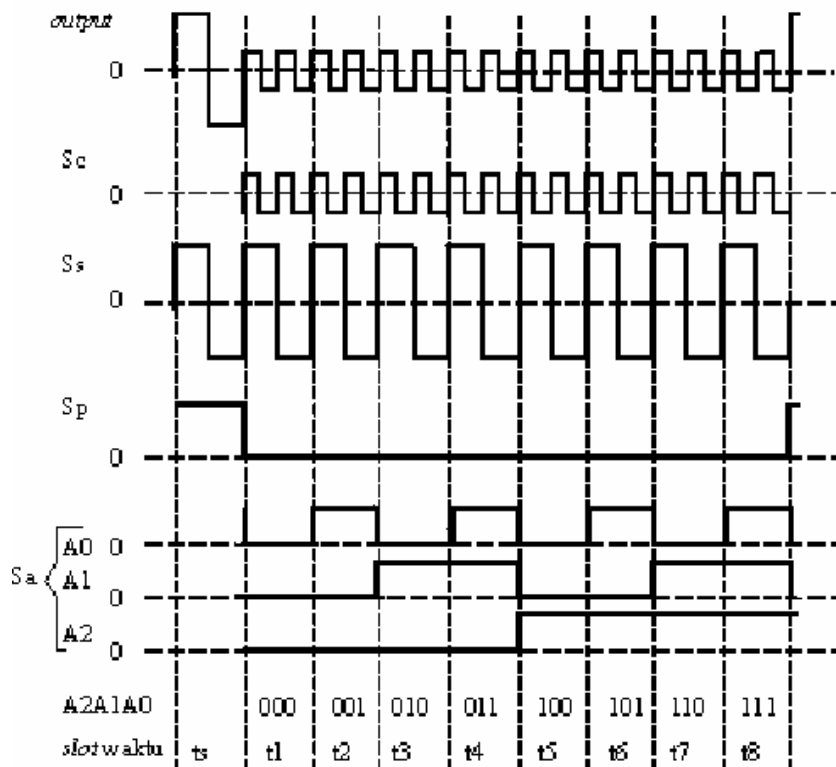
IV.1.11. Perancangan Subsistem Penyinkron dan Osilator

Subsistem penyinkron dan osilator dalam perancangan digabung menjadi satu subsistem dengan alasan kedua subsistem ini sangat terkait dalam hal pewaktuan, sehingga perancangan mekanismenya (dan juga implementasinya) menjadi kompak. Subsistem ini merupakan bagian yang sangat penting karena fungsinya mengendalikan subsistem-subsistem lainnya.

Tuntutan perancangan subsistem penyinkron dan osilator

1. Menghasilkan sinyal pembawa (S_c) gelombang kotak.
2. Menghasilkan sinyal penyinkron (S_s) gelombang kotak dengan perioda dua kali perioda sinyal pembawa.
3. Menghasilkan sinyal kendali alamat subsistem TDM (S_a).
4. Menghasilkan sinyal kendali subsistem pemilih sinyal (S_p)

Bentuk-bentuk sinyal yang harus dihasilkan ditunjukkan pada Gambar 4.3.



**Gambar 4.3. Sinyal-Sinyal Yang Harus Dihasilkan
oleh Subsistem Penyinkron dan Osilator**

Bentuk sinyal kendali pemilih sinyal (S_p) dirancang untuk mengendalikan pemilihan sinyal dalam subsistem pemilih sinyal, sedangkan bentuk sinyal kendali alamat (A_0, A_1 , dan A_2) dirancang untuk mengendalikan subsistem TDM. Jika semua sinyal yang harus dibangkitkan ini diamati, tampak bahwa semua sinyal, kecuali sinyal kendali pemilih sinyal (S_p), sangat mirip dengan sinyal yang dibangkitkan

untuk membangkitkan sinyal-sinyal dasar yang dapat diolah menjadi sinyal-sinyal keluaran.

Keluaran Q7 dari pencacah U2 mempunyai frekuensi $1/128$ dari frekuensi osilator. Sinyal ini kemudian masuk ke komparator U4A (IC TL082), yang ambangnya diatur pada $\frac{1}{2} V_{cc}$ (7,5 V). Fungsi komparator di sini adalah untuk membentuk sinyal yang simetri terhadap *ground* dengan cara mengayunkan sinyal sampai saturasi baik ke arah V_{cc} maupun ke arah $-V_{cc}$. Setelah sinyal menjadi simetri kemudian dibalik dan diatur tarafnya⁴⁾ dengan penguat membalik U4B. Sinyal penyinkron (Ss) dibentuk dengan cara yang sama tetapi diambil dari keluaran Q8 pencacah sehingga memiliki periode dua kali sinyal pembawa.

Sinyal kendali alamat TDM (Sa) dibangkitkan dari keluaran Q9, Q10, dan Q11 pencacah U2. Sinyal ini tidak diolah lebih lanjut karena sudah sesuai dengan sinyal-sinyal yang diharapkan.

Dekoder biner ke desimal U3 (IC 4028) digunakan untuk mendeteksi urutan waktu. Mekanisme dekode ini adalah salah satu keluarannya (pin Q0 sampai dengan Q9) akan “rendah” sesuai data yang masuk (pin A, B, C, dan D). Pin-pin keluaran inilah yang digunakan untuk mendeteksi urutan waktu. Keluaran Q12 pencacah U2 dimasukkan sebagai masukan dekode untuk mendeteksi *slot* waktu sinkronisasi. Pada keadaan awal, keluaran Q12 pencacah berkeadaan “rendah”, keluaran Q9, Q10, dan Q11 akan mencacah dari 0 sampai 7. Pada saat ini keluaran Q8 dekode akan “rendah” menandai saat *slot* waktu kanal (t_1 sampai dengan t_8). Pada *slot* waktu berikutnya, keluaran Q12 pencacah akan tinggi sementara keluaran

⁴⁾ Cara lain yang lebih sederhana dan ekonomis untuk menjadikan simetri, membalik, dan mengatur tarafnya adalah dengan menghilangkan rangkaian komparator dan menambahkan offset pada rangkaian membalik. Penulis tidak melakukannya karena sudah terlanjur dibuat.

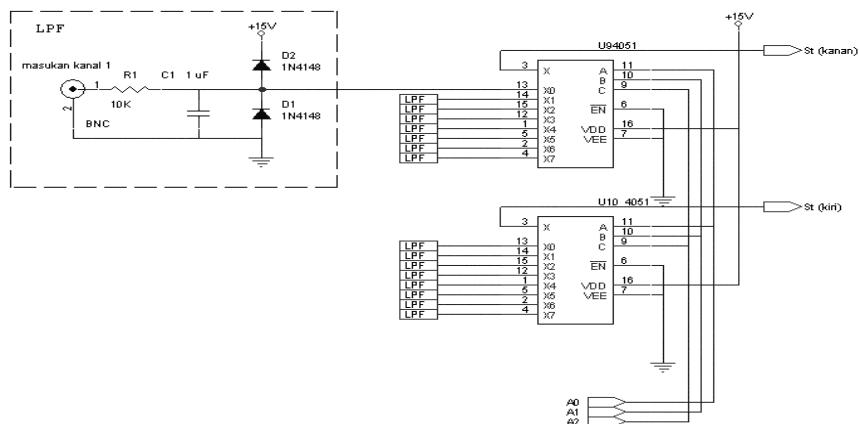
Q9, Q10, dan Q11 pencacah akan kembali ke nol (*overflow*), menandakan *slot* waktu sinkronisasi. Kondisi ini terdeteksi oleh dekoder dengan berubahnya keluaran Q8 dekoder menjadi keadan “tinggi”. Pada *slot* waktu berikutnya, masukan dekoder akan berubah sehingga keluaran Q9 dekoder akan berubah menjadi “rendah”, yang secara langsung akan me-reset pencacah ke keadaan awal (kembali ke *slot* waktu t1).

IV.1.12. Perancangan Subsistem TDM

Tuntutan perancangan :

1. terdiri dari 16 kanal, stereo (8 kanal kanan + 8 kanal kiri).
2. memiliki masukan kendali alamat kanal.

Rangkaian elektronik diimplementasikan dengan komponen rangkaian terpadu (IC, *integrated circuits*) 4051. Komponen ini dipilih karena sudah memiliki spesifikasi seperti yang diharapkan dalam rancangan. Skematik rangkaian dapat dilihat pada Gambar 4.5.

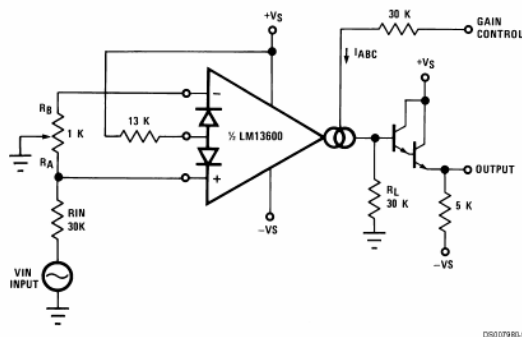


Gambar 4.5. Skematik Rangkaian Elektronik Subsistem TDM

Subsistem TDM terdiri dari 16 kanal tapis lolos-bawah (LPF, *low-pass filter*), multiplexer U10 (IC 4051), dan buffer U11 (IC TL 082). Tapis berfungsi untuk menghalangi komponen frekuensi tinggi dari sinyal masukan kanal. Komponen yang membentuk tapis (kanal 1) terdiri dari R1 dan C1. Dua buah dioda (D1 dan D2) berguna untuk membatasi sinyal yang masuk ke IC multiplexer U10. Tapis ini diperlukan untuk membatasi komponen frekuensi masukan agar tidak lebih dari setengah frekuensi *sampling* TDM.

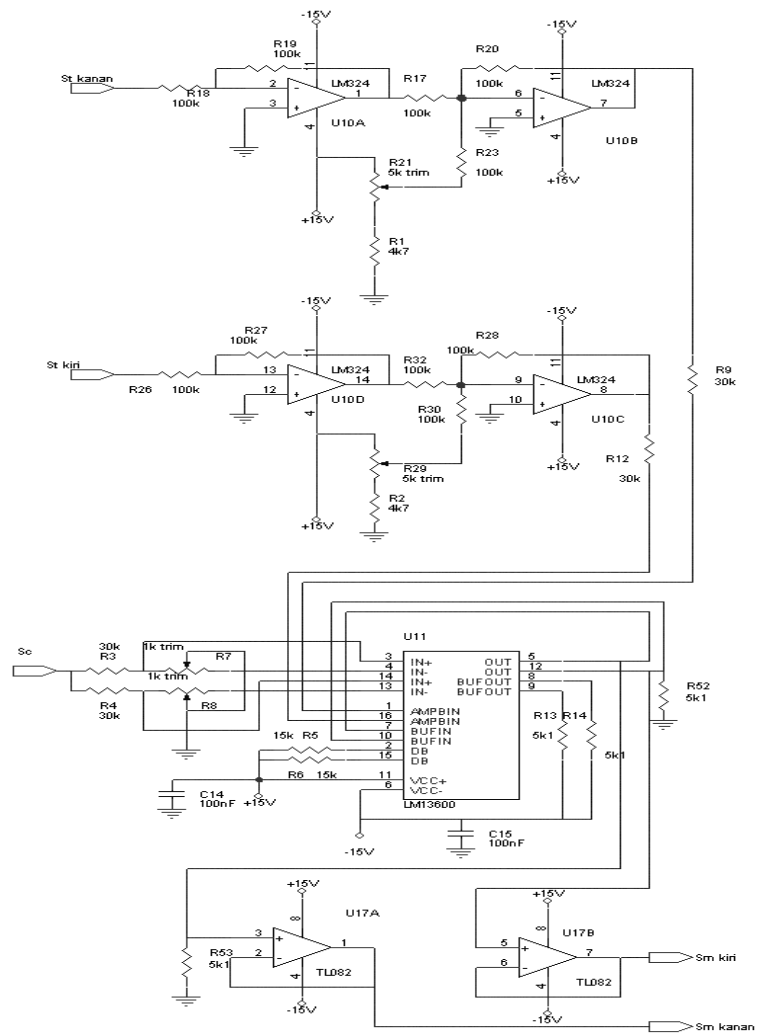
IV.1.13. Perancangan Subsistem Modulator

Modulator berfungsi untuk memodulasikan sinyal TDM (S_t) ke sinyal pembawa (S_c). Rangkaian elektronik modulator dirancang menggunakan komponen penguat transkonduktansi operasional (OTA, *Operational Transconductance Amplifier*). OTA yang digunakan adalah tipe LM13600 buatan National Semiconductor. Rancangan dasarnya mengikuti skema penguat terkendali tegangan pada Gambar 4.6, yang diambil dari lembar data komponen.



**Gambar 4.6. Skematik Rangkaian Elektronik Penguat Terkendali Tegangan
(National Semiconductor, 1998)**

Implementasi rangkaian dasar dalam subsistem modulator dapat dilihat pada Gambar 4.7. Subsistem ini tersusun dari penggeser taraf (*level shifter*) dan penguat terkendali tegangan yang dimodifikasi.



Gambar 4.7. Skematik Rangkaian Elektronik Subsistem Modulator

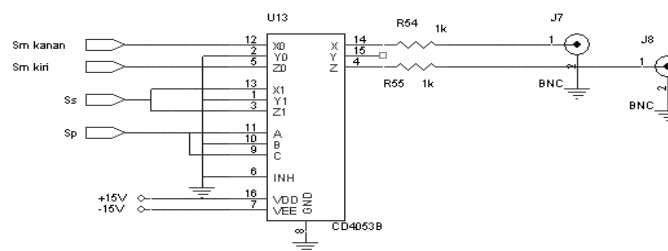
Penggeser taraf dibentuk oleh komponen aktif U10A, U10B, U10C, dan U10D yang berupa satu IC LM324. Fungsi dari penggeser taraf yaitu untuk menngeser taraf sinyal pemodulasi S_t agar sesuai dengan rentang taraf kendali penguat. Penguat terkendali tegangan sesungguhnya merupakan penguat terkendali arus, yang penguatannya sesuai dengan arus yang masuk ke pin 1 U11 (atau pin 16 U11 untuk kanal kiri). Tegangan pada pin-pin tersebut dijaga tetap kira-kira sebesar $-V_{cc} + 1,4$ volt. Resistor R9 dan R13 (masing-masing bernilai $30\text{ k}\Omega$) berfungsi membatasi arus kendali, yang berarti juga berfungsi sebagai konverter tegangan ke arus, karena arus yang mengalir sebanding dengan tegangan. Operasi yang diharapkan adalah bahwa keluaran penguat (yang masukannya berasal dari sinyal pembawa) mempunyai amplitudo nol ketika sinyal pemodulasi S_t bernilai nol, dan mempunyai amplitudo sekitar $1,41\text{ volt}^5$ ketika sinyal pemodulasi S_t bernilai 10 volt. Untuk mendapatkan amplitudo keluaran nol, tegangan yang masuk ke resistor pembatas arus R9 dan R13 harus sebesar $-V_{cc} + 1,4$ volt agar arus berhenti mengalir. Dengan menggunakan penggeser taraf, sinyal S_c yang rentangnya (0 – 10) volt digeser menjadi $(-V_{cc} + 1,4)$ volt sampai dengan $(-V_{cc} + 11,4)$ volt. Penggeseran taraf dilakukan oleh rangkaian penguat penjumlah U10B (atau U10C untuk kanal kiri). Rangkaian penguat membalik U10A (atau U10D untuk kanal kiri) berfungsi untuk membalikkan sinyal agar secara keseluruhan rangkaian penggeser tidak membalik polaritas sinyal. IC LM324 dipilih karena ayunan outputnya dapat mencapai $-V_{cc}$, sehingga cocok untuk menggeser sinyal sampai mendekati $-V_{cc}$.

⁵) Tegangan puncak maksimum yang diijinkan pada masukan *line-in* pada *soundcard* yang dipakai dalam penelitian ini (AC'97) adalah standar 1 volt rms. Ukuran dalam volt rms tidak dipakai karena dalam rancangan sistem akuisisi data ini dipakai sinyal gelombang kotak, sehingga dipakai acuan tegangan puncak sesaat (1,41 volt).

Keluaran OTA berupa arus, yang diubah menjadi tegangan oleh resistor R53 (atau R52 untuk kanal kiri). Tegangan ini kemudian disangga oleh penyangga U17A (atau U17B untuk kanal kiri). Rangkaian penyangga di sini tidak memanfaatkan transistor seperti pada rangkaian dasar (Gambar 4.6) karena ketika diuji ternyata mengalami cacat pada siklus negatifnya.

IV.1.14. Perancangan Subsistem Pemilih Sinyal

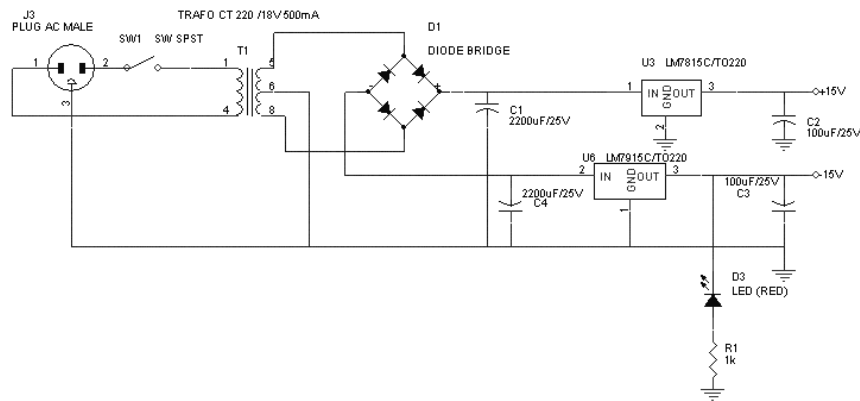
Subsistem pemilih sinyal berfungsi memilih sinyal yang harus ditampilkan sebagai keluaran sesuai *slot* waktunya. Sinyal masukannya adalah sinyal hasil modulasi (Sm) dan sinyal penyinkron (Ss). Sinyal yang harus ditampilkan (apakah Sm ataukah Ss) sudah ditentukan oleh subsistem penyinkron dan osilator, sehingga subsistem ini sebenarnya hanya berupa saklar yang dikendalikan oleh sinyal dari penyinkron (sinyal Sp). Subsistem ini diimplementasikan menggunakan IC 4053. Skematik selengkapnya dapat dilihat pada Gambar 4.8. Resistor R54 dan R55 digunakan untuk membatasi arus, misalnya ketika terjadi hubung singkat dengan *ground*.



Gambar 4.8. Skematik Rangkaian Elektronik Subsistem Pemilih Sinyal

IV.1.15. Perancangan Subsystem Catu Daya

Catu daya yang dibutuhkan adalah sumber tegangan 15 volt dan -15 volt yang stabil. Untuk kestabilan dipakai IC 7815 dan IC 7915 yang mudah didapat di pasaran. Skematik subsystem catu daya dapat dilihat pada Gambar 4.9.



Gambar 4.9. Skematik Rangkaian Elektronik Subsystem Catu Daya

IV.1.16. Pembuatan Prototip

Pembuatan prototip sangat penting dalam perancangan karena seringkali dalam perhitungan tidak dipertimbangkan semua faktor, sehingga ketika rancangan dibuat belum tentu dapat bekerja seperti dalam perhitungan. Metode coba-coba⁶⁾ (*trial and error*) diakui masih terdapat dalam proses perancangan ini. Proses pembuatan prototip dalam penelitian ini tidak dilaporkan secara detil karena antara proses pembuatan prototip dan proses perancangan (yang diuraikan dalam bab IV.1)

⁶⁾ Ini setara dengan, misalnya, metode biseksi dalam komputasi numeris.

dalam praktek merupakan proses yang seringkali bersifat timbal balik⁷⁾, sehingga terlalu panjang jika dilaporkan. Secara garis besar, pembuatan prototip terdiri dari langkah perancangan jalur PCB (*Printed Circuit Board*), pembuatan PCB, penyolderan komponen, dan pengujian tiap blok.

Perancangan jalur PCB. Jalur PCB dirancang menggunakan perangkat lunak Orcad Layout versi 9.2.3. Untuk membuat rancangan jalur PCB, skematik rangkaian dibuat terlebih dahulu menggunakan Orcad Capture. Setelah skematik dibuat, *file* netlist dapat diciptakan untuk kemudian diproses oleh Orcad Layout untuk menghasilkan rancangan jalur PCB secara otomatis. Sebelum rancangan jalur PCB dibuat secara otomatis, tata letak komponen harus diatur terlebih dahulu secara manual. Setelah jalur PCB dibuat secara otomatis oleh Orcad Layout, hasil rancangan jalur tersebut kemudian dicetak pada transparansi.

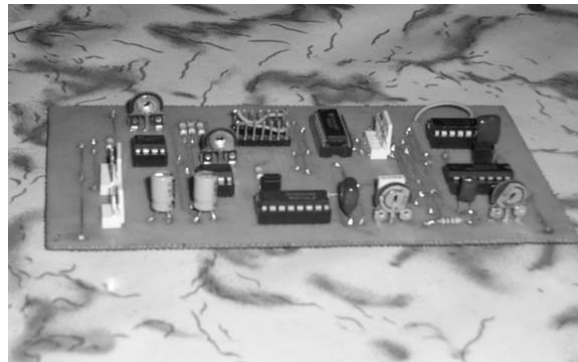
Pembuatan PCB. Pembuatan PCB terdiri dari langkah transfer tinta tahan air ke papan bahan PCB, pengetsaan (*etching*), dan pengeboran. Transfer tinta tahan air dalam penelitian ini menggunakan dua macam teknik. Yang pertama menggunakan setrika, dilakukan dengan menempelkan sisi tinta dari transparansi ke sisi tembaga dari papan bahan PCB, kemudian menggosoknya dengan setrika panas. Teknik yang kedua menggunakan sablon, dengan cara membuat cadar (*screen*) untuk mencetak pola jalur PCB.

⁷⁾ Sebagai contoh, nomor-nomor pin pada rancangan skematik dapat dipertukarkan ketika dalam perancangan jalur PCB terjadi kesulitan.

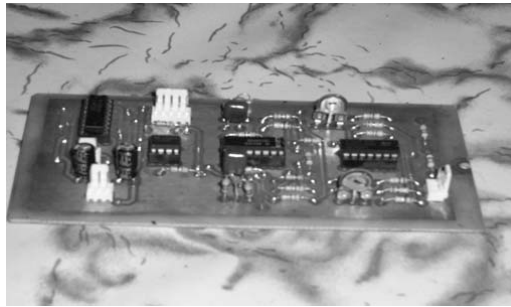
Setelah jalur PCB dicetak ke sisi tembaga dari papan bahan PCB, maka langkah selanjutnya adalah pengetsaan dengan pelarut PCB (Feri Klorida). Lapisan tembaga yang tidak tertutup tinta tidak akan larut sementara yang lainnya larut, sehingga terbentuk pola jalur konduktor sesuai pola tinta.

Setelah dietsa dan dibersihkan, jalur PBB kemudian dibor untuk menyediakan tempat masuk bagi kaki komponen elektronik. Bor yang digunakan berukuran 1 mm dan 0,8 mm tergantung jenis komponen yang akan dipasang.

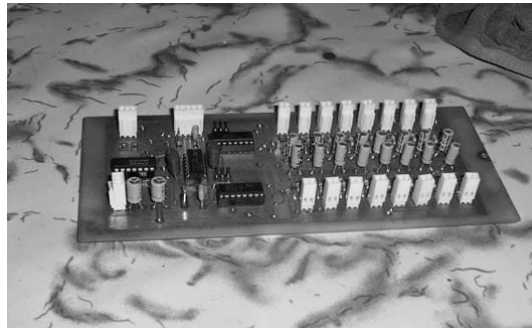
Penyolderan komponen. Komponen elektronik disolder menggunakan solder 30 Watt. Penyolderan dilakukan sesempurna mungkin untuk menghindari ketidaksempurnaan koneksi kaki komponen dengan jalur PCB, tetapi dengan lama penyolderan untuk tiap titik sesingkat mungkin untuk menghindari kerusakan komponen akibat kelebihan panas. Rangkaian yang telah disolder ditampilkan pada Gambar 4.10, Gamba 4.11, Gambar 4.12 dan Gambar 4.13.



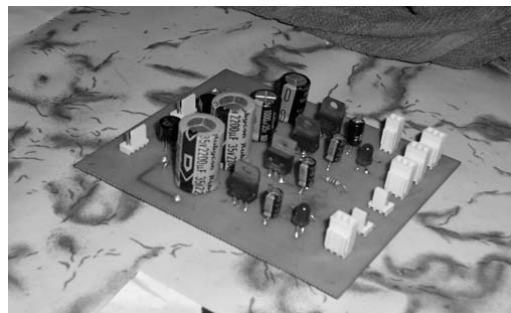
Gambar 4.10. Rangkaian Elektronik Subsistem Penyinkron dan Osilator



Gambar 4.11. Rangkaian Elektronik Subsistem Modulator dan Pemilih Sinyal



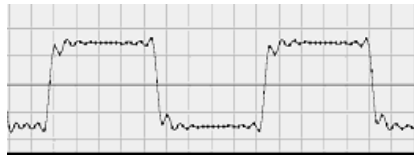
Gambar 4.12. Rangkaian Elektronik Subsistem TDM



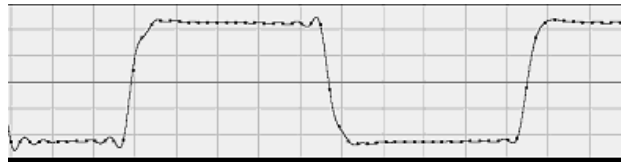
Gambar 4.13. Rangkaian Elektronik Subsistem Catu Daya

Pengujian tiap blok. Pengujian tiap blok dilakukan untuk memastikan bahwa rancangan sudah sesuai dengan yang diharapkan. Pengujian dilakukan terutama untuk melihat bentuk sinyal. Taraf sinyal tidak diuji karena hampir semua sinyal analog dapat diatur tarafnya dengan menala komponen variabel, dan semua sinyal digital mempunyai toleransi taraf yang besar. Ini menguntungkan karena pengujian dapat dilakukan dengan PC melalui masukan *soundcard* yang tidak perlu dikalibrasi. Perangkat lunak yang digunakan untuk pengujian ini adalah Cool Edit 2000.

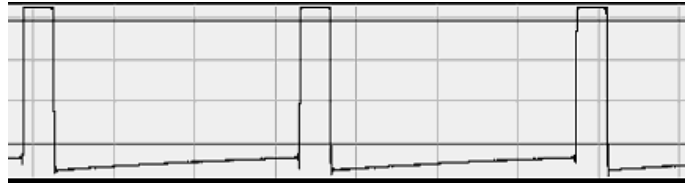
Bentuk sinyal pembawa ditunjukkan pada Gambar 4.14, sinyal penyinkron pada Gambar 4.15, sinyal kendali pemilih sinyal pada Gambar 4.16, sinyal kendali alamat A0 pada Gambar 4.17, sinyal kendali alamat A1 pada Gambar 4.18, dan sinyal kendali alamat A2 pada Gambar 4.19.



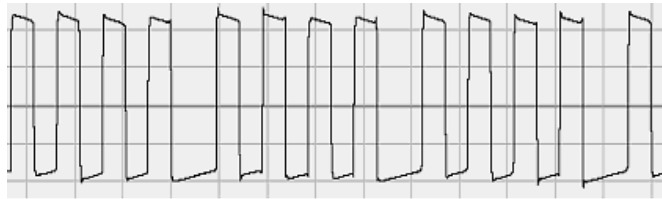
Gambar 4.14. Bentuk Sinyal Pembawa



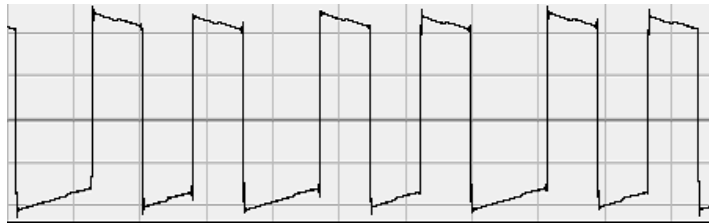
Gambar 4.15. Bentuk Sinyal Penyinkron



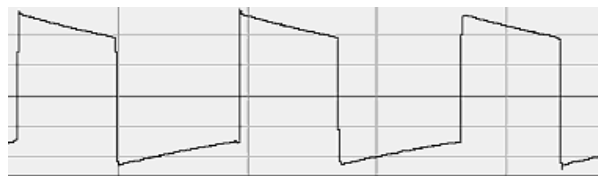
Gambar 4.16. Bentuk Sinyal Kendali Pemilih Sinyal



Gambar 4.17. Bentuk Sinyal A0



Gambar 4.18. Bentuk Sinyal A1



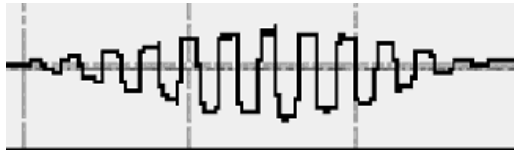
Gambar 4.19. Bentuk Sinyal A2

Keluaran sinyal TDM dengan masukan tegangan yang berubah dari 0 ke 10 volt pada salah satu kanal tampak pada Gambar 4.20.



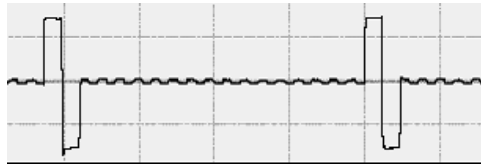
Gambar 4.20. Bentuk Sinyal TDM

Bentuk sinyal keluaran modulator dengan masukan masukan pembawa 1,8 kHz dan masukan pemodulasi setengah siklus sinyal sinus 75 Hz ditunjukkan pada Gambar 4.21.



Gambar 4.21. Bentuk Sinyal Keluaran Modulator

Bentuk sinyal keluaran blok pemilih sinyal diamati dengan merangkai seluruh blok kecuali blok TDM. Bentuk sinyal keluaran blok pemilih sinyal, tanpa masukan sinyal pada blok modulator, tampak pada Gambar 4.22.



Gambar 4.22. Bentuk Sinyal Keluaran Pemilih Sinyal

Karena blok catu daya tidak menghasilkan sinyal gelombang, maka pengujian dilakukan dengan tujuan untuk mengetahui tegangan dan arus yang dikeluarkan. Hasil pengukuran arus dan tegangan adalah sebagai berikut:

- | | | |
|-------------------------------|---|------------|
| a. Keluaran 15 volt | : | 15,01 volt |
| b. Keluaran -15 volt | : | 14,98 volt |
| c. Arus pada saluran 15 volt | : | 167 mA |
| d. Arus pada saluran -15 volt | : | 126 mA |

IV.2. Perancangan Perangkat Lunak

Perancangan perangkat lunak diawali dengan merumuskan tuntutan perancangan (Bab IV.2.1), kemudian merumuskan konsep dasar solusinya (Bab IV.2.2). Solusi yang dirumuskan yaitu merancang aplikasi pedekode sinyal (Bab IV.2.3) dan membuat kode antar muka (Bab IV.2.4 dan Bab IV.2.5).

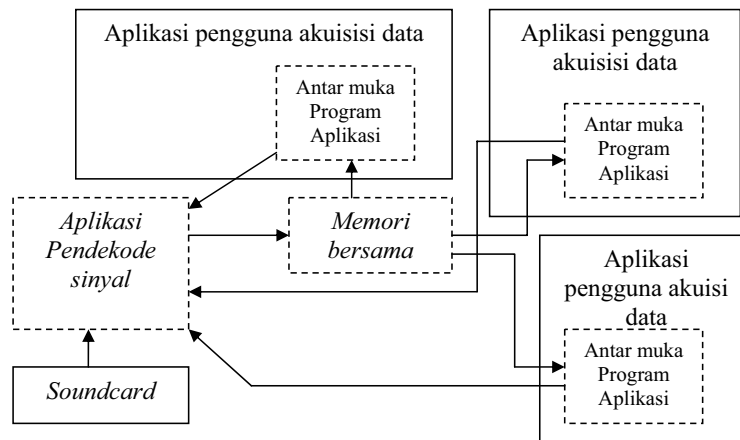
IV.2.1. Tuntutan Perancangan

Tuntutan perancangan adalah bagaimana agar pengguna dapat mengakses sistem akuisisi data dengan mudah dalam penulisan perangkat lunak. Bagi pengguna, cara kerja sistem akuisisi data tidaklah penting, sehingga pengguna tidak perlu mempelajari bagaimana mengakses *soundcard* dan menerjemahkan “suara” yang didapat dari *soundcard* tersebut menjadi data yang diperlukan (nilai sinyal masukan kanal). Beberapa program dalam satu komputer juga diharapkan dapat mengakses sistem akuisisi secara bersamaan secara cepat dan efisien.

IV.2.2. Konsep Dasar Solusi

Untuk dapat mengakses dengan mudah, maka perlu dirancang skrip program antar muka dalam bahasa pemrograman tingkat tinggi (atau menengah) yang dirancang dengan baik sehingga mudah dipakai oleh pengguna. Untuk kecepatan, skrip program yang nantinya dipakai oleh pengguna tidak boleh menghasilkan kode eksekusi yang banyak memakan waktu komputasi, mengingat dalam satu komputer terdapat kemungkinan berjalan beberapa program yang mengakses sistem akuisisi data secara bersamaan. Untuk mengatasi masalah kecepatan dirancang aplikasi yang

khusus bertugas mengakses sistem akuisisi data dengan cara membaca masukan *soundcard* dan medekodekannya. Kemudian dirancang skrip antarmuka program aplikasi untuk mengakses data yang telah didekodekan. Konsep sistem perangkat lunak yang dirancang ditunjukkan dalam Gambar 4.23.

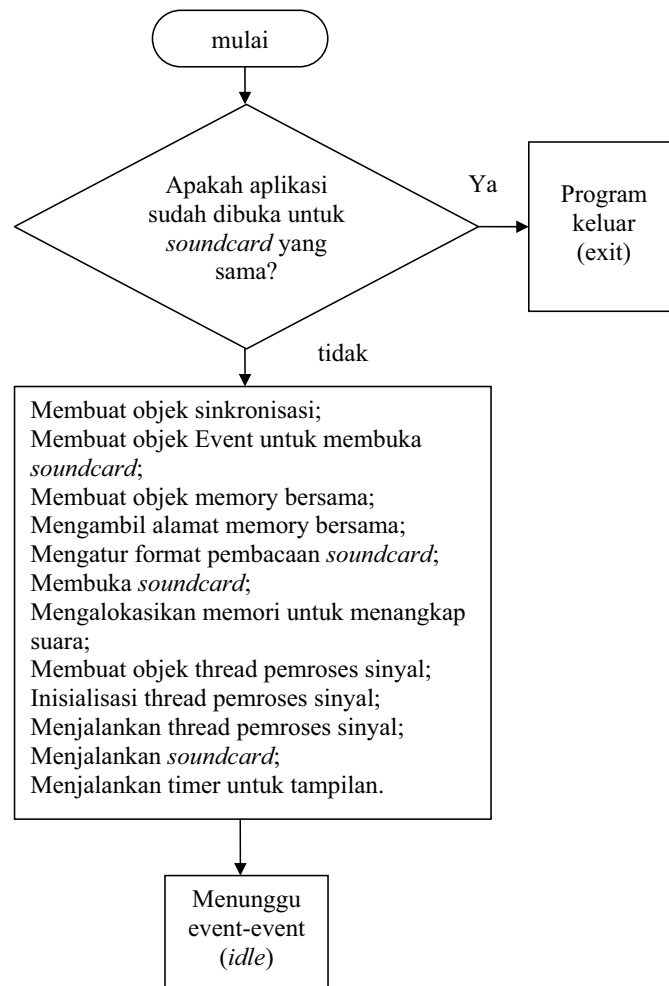


Gambar 4.23. Konsep Dasar Solusi Rancangan Perangkat Lunak

IV.2.3. Perancangan Aplikasi Pendekode

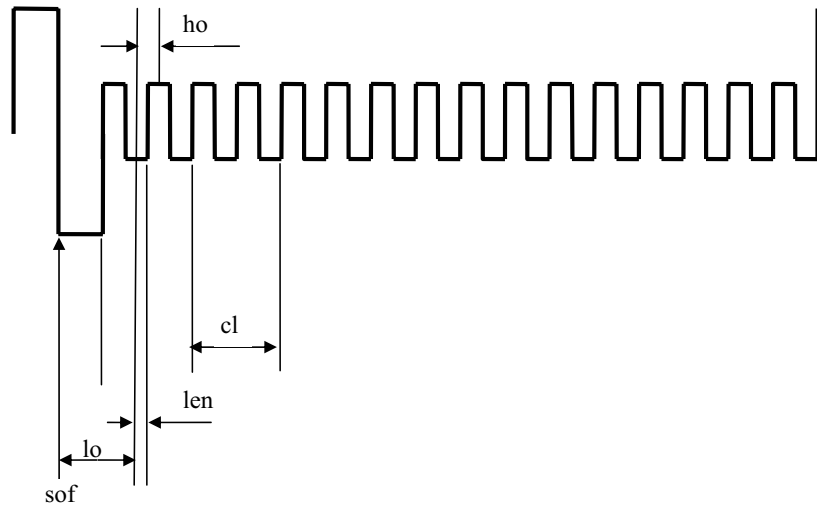
Diagram alir program dapat dilihat pada Gambar 4.24, listing program selengkapnya ada dalam Lampiran 3. Yang pertama kali harus dilakukan oleh aplikasi adalah memeriksa apakah aplikasi yang sama telah dijalankan untuk mendekode sinyal dari *soundcard* yang sama, jika sudah maka program aplikasi langsung keluar (selesai dan menutup *window*). Ini dimaksudkan agar sinyal yang

sama tidak didekode oleh dua aplikasi. Jika aplikasi pendekode dipanggil untuk *soundcard* yang berbeda maka aplikasi akan dijalankan.



Gambar 4.24. Diagram Alir Aplikasi Pendekode

Algoritma Decoding. Proses *decoding* di dalam aplikasi pendekode dilakukan oleh *thread* pemroses sinyal. Listing *thread* pemroses sinyal selengkapnya ada dalam Lampiran 5. Bentuk sinyal beserta parameter *decoding* ditampilkan pada Gambar 4.25.



Gambar 4.25. Bentuk Sinyal dan Parameter Decoding

Jika sinyal yang direpresentasikan oleh Gambar 4.25 dicuplik sehingga menjadi data diskrit yang dapat disimpan dalam suatu variabel array, maka

sof = indeks awal *frame*,

lo = offset (banyaknya data yang harus dilewati) untuk mengambil data fase “rendah”, dihitung dari sof.

ho = offset (banyaknya data yang harus dilewati) untuk mengambil data fase “tinggi”, dihitung dari sof + lo.

len = banyaknya titik data yang akan diambil nilai rata-ratanya, dan

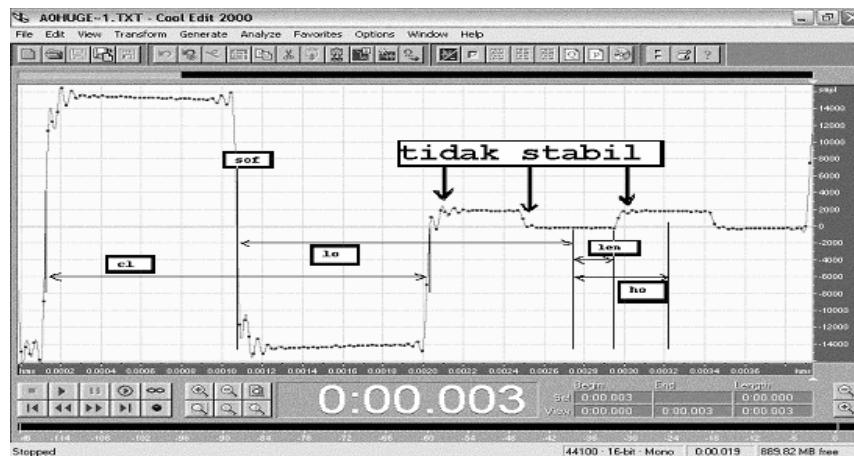
cl = banyaknya titik data tiap kanal.

Jika sampel tersebut disimpan dalam variabel array dengan nama buffer, maka besarnya sinyal pada kanal ke n (yang nilainya 0 sampai 7) adalah

$$\text{offset} + \text{Gain} \times \left(\frac{\sum_{i=0}^{\text{len}} \text{buffer}[\text{sof} + \text{lo} + \text{ho} + (\text{cl} \times n) + i] - \sum_{i=0}^{\text{len}} \text{buffer}[\text{sof} + \text{lo} + (\text{cl} \times n) + i]}{\text{len}} \right)$$

Besarnya cl (sampel per kanal) dapat dikalibrasi dalam perangkat lunak agar sesuai dengan frekuensi operasi perangkat keras, mengingat frekuensi tersebut dapat bergeser sejalan dengan waktu. Dengan mempertimbangkan ketidakstabilan bentuk sinyal aktual seperti tampak pada Gambar 4.26, nilai nilai ho, lo, dan len ditentukan sebagai berikut:

- a. ho (offset “tinggi”) = (1/4) cl
- b. lo (offset “rendah”) = (7/8) cl
- c. len (banyaknya data yang akan dirata-rata) = (1/9) cl



Gambar 4.26. Ketidakstabilan Sinyal

Nilai *sof* (indeks posisi awal *frame*) dicari dengan fungsi `findFrame()` sebagai berikut:

```
sof = findFrame(buffer, panjangBuffer);
```

Fungsi `findFrame()` diimplementasikan dalam `TWaveProcessor` sebagai berikut:

```
int __fastcall TWaveProcessor::findFrame(short* buff,int buffLen)
{
    for(int index=0;index<buffLen;index++)
    {
        if(isEdge(index,buff)==true)
        {
            if(isValidSync(index+1,buff)==true)
                return index+1;
        }
    }
    return -1; //gagal menemukan sinyal sinkronisasi
}
```

Dalam fungsi `findFrame()` terdapat fungsi untuk mendeteksi tepi turunnya sinyal, yaitu fungsi `isEdge()` sebagai berikut:

```
bool TWaveProcessor::isEdge(int index, short* buff)
{
    if(index+1>=SamplesPerBuff) //di luar buffer
        return false;
    if((buff[index]>=0)&&(buff[index+1]<0)) //sisi turun
        return true;
    return false;
}
```

Dalam fungsi `findFrame` juga terdapat fungsi untuk menguji konsistensi taraf sinyal sinkronisasi, yaitu fungsi `isValidSync()` yang diimplementasikan sebagai berikut:

```

bool TWaveProcessor::isValidSync(int index, short * buff)
{
    minSyncLen=numberOfSamplesPerChannel/2.4;
    if(index+minSyncLen>=SamplesPerBuff //jika
                                        //perpanjangannya
                                        //di luar buffer

        return false;
    for(int i=1;i<minSyncLen;i++) //cek konsistensi taraf
    {
        if(buff[index+i]>=-10000)
            return false;
        if(buff[index+i]<=-20000)
            return false;
    }
    return true;
}

```

IV.2.4. Perancangan Kode Antarmuka Dalam C++ (C++ Builder)

Kode antarmuka diciptakan untuk memudahkan pengguna dalam mengakses sistem akuisisi. Kode antarmuka dirancang sesederhana mungkin agar mudah digunakan, dan ini diimplementasikan dengan menciptakan kelas TSdaqInterface. Kesederhanaan kelas TSdaqInterface dapat dilihat pada deklarasi kelas, yang dirancang sebagai berikut:

```

class TSdaqInterface
{
private:
    STARTUPINFO SI; //struktur untuk mengkonfigurasi
                    //pemanggilan aplikasi pendekode
    PROCESS_INFORMATION PI; //struktur untuk memantau
                             //keadaan aplikasi pendekode
    SdaqData* data; //struktur untuk mengakses data
                    //pada memori bersama
    HANDLE hMapFile;

```

```

        HANDLE hMutex;
        bool opened;
        void showError(DWORD errorCode);
    public:
        TScdaqInterface();
        bool Open(unsigned int DeviceID);
        double getData(int channelNumber);
        double getSlowData(int channelNumber);
};

```

Untuk dapat menggunakan kelas TScdaqInterface, yang perlu diketahui adalah penggunaan fungsi-fungsi dalam seksi public. Listing definisi fungsi-fungsi anggota TScdaqInterface ada dalam Lampiran 11.

Fungsi Open() dirancang untuk memanggil aplikasi pendekode serta menginisialisasi objek sinkronisasi dan memori bersama. Parameter fungsi ini adalah DeviceID yang merupakan nomor *soundcard* yang terpasang dalam sistem PC, yang urutannya dimulai dari satu. Jika diperlukan untuk mengakses dua *soundcard* yang berbeda maka perlu diciptakan dua bentukan TScdaqInterface, kemudian fungsi Open() dari masing-masing bentukannya dipanggil dengan parameter yang berbeda. Berikut ini contoh penggunaan TScdaqInterface untuk membaca masukan dari dua buah sistem akuisisi data yang berbeda, dengan masukan pertama adalah kanal 15 yang dari sistem akuisisi data pertama, yang masuk ke komputer melalui masukan *soundcard* pertama, dan masukan kedua adalah kanal 15 yang dari sistem akuisisi data kedua, yang masuk ke komputer melalui masukan *soundcard* kedua :

```

TScdaqInterface sdaq1; //menciptakan objek (bentukan)-
                        // TScdaqInterface.
TScdaqInterface sdaq2; //menciptakan objek (bentukan)-
                        //TScdaqInterface.

```

```

Scdaq1.Open(1);           //menjalankan aplikasi pendekode-
                          //akuisisi soundcard pertama.
Scdaq2.Open(2);           //menjalankan aplikasi -
                          //pendekode akuisisi soundcard kedua.
double hasilBaca1=scdaq1.getData(15);    //membaca kanal 15 -
                                          //sistem akuisisi 1.
double hasilBaca2=scdaq2.getData(15);    //membaca kanal 15 -
                                          //sistem akuisisi 2.

```

Nilai pengembalian fungsi `Open()` adalah data dengan tipe `bool`, satu (*true*) jika berhasil membuka aplikasi pendekode dan nol (*false*) jika gagal.

Fungsi `getData` dan `getSlowData` digunakan untuk membaca kanal. Parameter fungsi ini adalah nomor kanal yang dimulai dari nol sampai lima belas. Perbedaan antara `getData()` dan `getSlowData` adalah bahwa `getSlowData` mengembalikan nilai yang diperbarui (oleh aplikasi pendekode) setiap 800 mili detik, sedangkan `getData` mengembalikan nilai yang diperbarui tiap 10 mili detik. Pengguna dapat memanggil fungsi `getData` (atau `getSlowData`) tiap saat dengan atau tanpa mempertimbangkan jeda waktu antara pemanggilan satu dengan pemanggilan selanjutnya dengan resiko mendapatkan data yang sama (belum diperbarui) jika jeda pemanggilan kurang dari 10 milidetik (atau 800 mili untuk fungsi `getSlowData`). Nilai pengembalian fungsi ini adalah data tipe `double` antara 0.0 sampai 1.0 sesuai tegangan masukan kanal (tegangan masukan 0 volt menghasilkan pembacaan 0.0 dan 10 volt menghasilkan 1.0)

IV.2.5. Perancangan Kode Antarmuka Dalam Pascal (Delphi)

Kode antarmuka dalam Pascal dirancang memiliki fungsi yang sama persis dengan kode dalam C++. Karena sama-sama diimplementasikan menggunakan

fungsi-fungsi *Windows API*, maka kodenya mirip sehingga hampir semua baris-baris dalam bahasa C++ dapat diterjemahkan ke dalam bahasa Pascal. Kelas dalam C++ diimplementasikan menggunakan objek Pascal. Karena struktur C++ tidak sama dengan record Delphi dalam hal susunannya di dalam memori, maka pembacaan variabel array dari memori dilakukan dengan operasi pointer. Listing lengkap implementasi antarmuka dalam Pascal dapat dilihat pada Lampiran 12.

Untuk mengakses sistem akuisisi, yang diperlukan adalah mendeklarasikan objek dari kelas `TScdaqInterface` sebagai variabel seperti contoh berikut:

```
Var
    Scdaq1:TScdaqInterface;
    Scdaq2:TScdaqInterface;
```

Kemudian, dalam implementasi, konstruktor `TScdaqInterface.Create` dipanggil untuk menciptakan bentukannya, seperti contoh berikut:

```
Scdaq1:=TScdaqInterface.Create;
Scdaq2:=TScdaqInterface.Create;
```

Setelah itu *methods* dari `Scdaq1` dan `Scdaq2` dapat dipanggil untuk mengakses sistem akuisisi, seperti contoh berikut:

```
Scdaq1.Open(1); //menjalankan aplikasi pendekode
                //sist. akuisisi pertama.
Scdaq2.Open(2); //menjalankan aplikasi pendekode
                //sist. akuisisi kedua.

Data1:=Scdaq1.GetData(15); //membaca data
Data2:=Scdaq2.GetData(15); //membaca data
```

IV.3. Pengujian Unjuk Kerja

Pengujian unjuk kerja dilakukan untuk menilai kinerja dari alat yang dihasilkan dari penelitian ini. Hasil pengujian unjuk kerja ini akan dibahas dalam Bab V. Pengujian yang dilakukan adalah uji derau *soundcard* (Bab IV.3.2), akurasi dan linearitas (Bab IV.3.3), tanggapan langkah (Bab IV.3.4), tanggapan frekuensi (Bab IV.3.5), cakap silangan (Bab IV.3.6), dan uji aplikasi (Bab IV.3.7).

IV.3.1. Alat dan Bahan

Alat dan bahan yang digunakan dalam pengujian ini adalah sebagai berikut:

1. Sistem akuisisi yang akan diuji.
2. Komputer, dengan spesifikasi:
Pentium III 500 MHz, RAM 512MB, sistem operasi Windows XP,
Soundcard AC '97.
3. Perangkat lunak CoolEdit, digunakan untuk merekam derau, serta menampilkan hasil pengukuran yang disimpan dalam berkas berformat txt.
4. Perangkat lunak Uji Coba, yang dibuat untuk tujuan uji unjuk kerja. Perangkat lunak ini ditulis dengan C++ Builder dan menggunakan TSCdaqInterface untuk mengakses sistem akuisisi. Perangkat lunak ini dilengkapi fitur untuk menyimpan hasil pembacaan sistem akuisisi ke dalam berkas komputer.
5. Voltmeter, dengan spesifikasi:
Jumlah digit display: 5 digit (multimeter Philips PM2525).

6. Sumber tegangan DC variabel, dengan spesifikasi:
Rentang : 0 Volt sampai dengan 10 Volt, Kestabilan : 0,1 %.
7. Generator Fungsi (*Function Generator*).

IV.3.2. Uji Derau *Soundcard*

Derau *soundcard* diuji untuk membantu analisis unjuk kerja alat yang dibuat. Pengujian dilakukan merekam saluran *line-in* tanpa sumber sinyal, dan mengatur taraf volumenya (melalui *mixer* pada Windows) sesuai dengan taraf ketika digunakan untuk membaca sinyal dari alat.

IV.3.3. Uji Akurasi dan Linearitas

Untuk menguji akurasi dan linearitas digunakan pembanding voltmeter yang diyakini mempunyai linearitas dan akurasi yang lebih baik dari sistem yang akan diuji. Pengujian dengan memberi masukan tegangan DC yang divariasikan dari 0 sampai 10 volt dengan langkah $\frac{1}{2}$ volt sesuai pembacaan voltmeter. Pembacaan kanal oleh perangkat lunak baik untuk mode cepat maupun lambat dilakukan 99 kali dan dicatat ke dalam *file*. Pengujian ini dilakukan pada satu kanal saja (kanal 4).

IV.3.4. Uji Tanggapan Langkah (*Step Response*)

Uji tanggapan langkah dilakukan dengan memberi masukan kanal dengan sinyal gelombang kotak frekuensi rendah. Pembacaan oleh perangkat lunak dilakukan dalam mode cepat. Pengujian ini dilakukan untuk satu kanal saja.

IV.3.5. Uji Tanggapan Frekuensi

Uji respon frekuensi dilakukan dengan memberi masukan sinyal gelombang sinus ke satu kanal. Sinyal gelombang sinus dihasilkan dari Generator Fungsi, dengan amplitudo dan offsetnya diatur ke 5 volt, sehingga rentangnya sesuai dengan sistem akuisisi yang dibuat, yaitu 0 sampai 10 volt. Frekuensi masukan divariasikan dari 0,2 Hz sampai 50 Hz. Pembacaan sampel untuk tiap frekuensi dilakukan dengan kecepatan 100 sampel per detik, dan lamanya pengambilan sampel diatur agar sampel yang didapat mencakup setidaknya empat siklus gelombang sinus. Hasil tiap bacaan kemudian dicatat dalam berkas komputer. Pengujian ini hanya dilakukan pada satu kanal saja.

IV.3.6. Uji Cakap Silang

Uji cakap silang dilakukan untuk menguji interferensi dari satu kanal ke kanal lainnya. Uji ini dilakukan dengan memberi tegangan DC pada salah satu masukan sementara masukan lainnya diatur ke nol volt. Perangkat lunak membaca data dari kanal 0 sampai 7 dan mencatatnya ke dalam berkas komputer. Pengujian cakap silang dilakukan untuk masukan ke kanal 0 sampai kanal 7.

IV.3.7. Uji Aplikasi

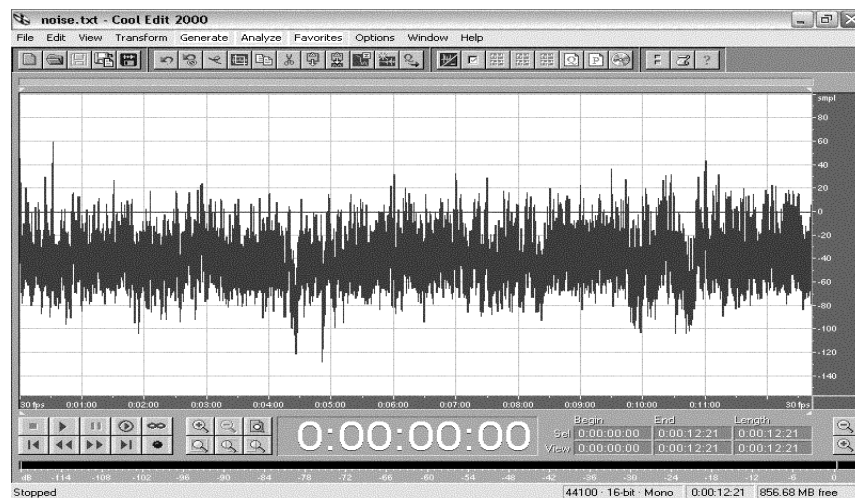
Aplikasi yang diujikan adalah pengendali PID (Proporsional Integral Diferensial) untuk mengatur kecepatan motor DC dan Termometer untuk memantau suhu udara. Pengujian hanya dilakukan untuk mengamati apakah sistem dapat bekerja untuk aplikasi praktis, sehingga tidak ada parameter yang diukur (secara

kuantitatif) dalam pengujian ini. Perangkat lunak aplikasi pengendali PID dibuat menggunakan kompilasi C++ Builder Professional 4.0, sedangkan Termometer dibuat menggunakan Delphi 7. Pembuatan aplikasi pengendali PID dan Termometer berada di luar batasan penelitian ini sehingga tidak dibahas, tetapi rancangan perangkat kerasnya dapat dilihat pada Lampiran 13.

BAB V. HASIL PENGUJIAN DAN PEMBAHASAN

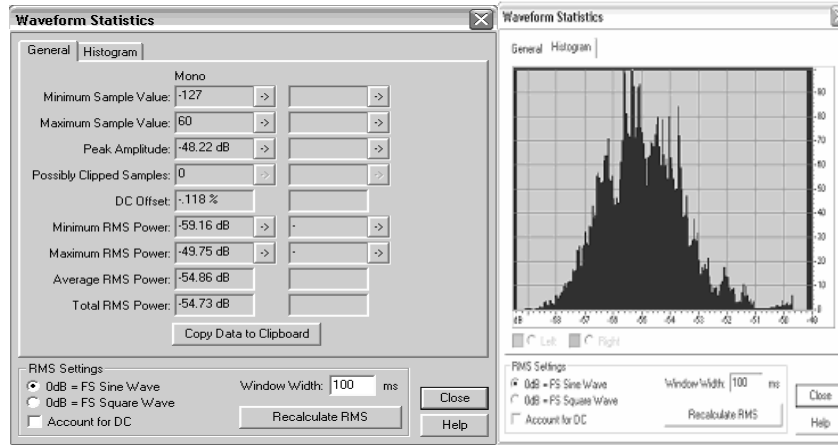
V.1. Derau *Soundcard* dan Resolusi

Hasil pengukuran derau *soundcard* dalam domain waktu ditampilkan pada Gambar 5.1. Gambar tersebut menunjukkan sinyal derau yang direkam selama 12 detik lebih 21/30 detik (560070 sampel). Statistik sampel yang telah dikonversi ke nilai rms ditampilkan pada Gambar 5.2 . Statistik sampel dalam bentuk aslinya ditampilkan dalam Gambar 5.3, yang merupakan statistik dari 65536 sampel⁸⁾. Statistik dalam nilai rms ditampilkan terutama untuk melihat distribusinya (dalam bentuk histogram), karena perangkat lunak yang mampu mengolah sampel dalam jumlah banyak yang tersedia (CoolEdit) hanya menyediakan statistik untuk nilai rms dari sampel. Spektrum derau ditampilkan pada Gambar 5.4.

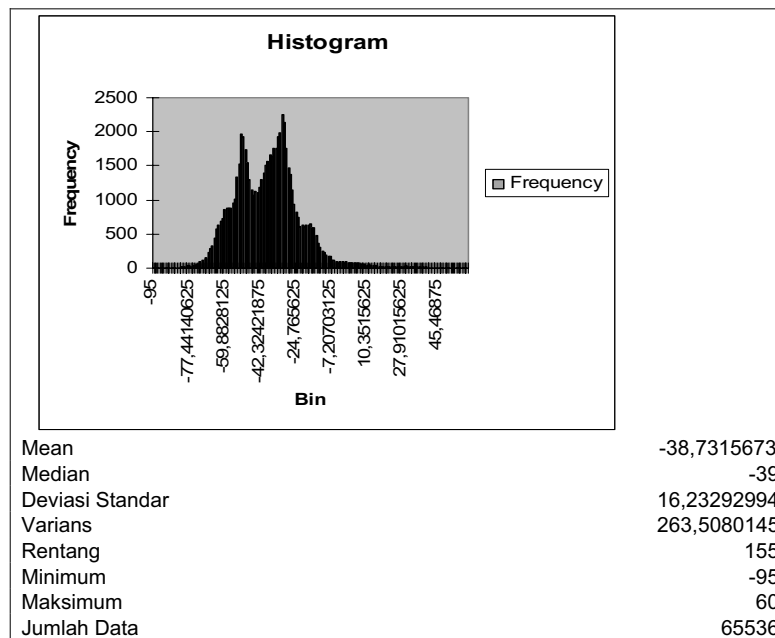


Gambar 5.1. Derau *Soundcard*

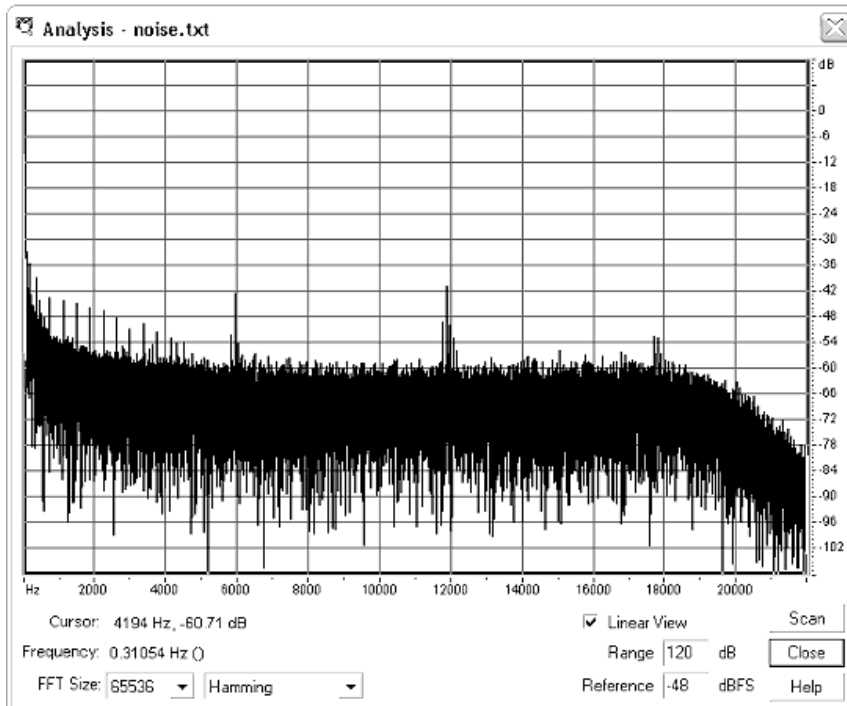
⁸⁾ Jumlah sampel dibatasi karena perangkat lunak yang digunakan (MS Excel) hanya mampu mengolah data sebanyak 65536 sampel.



Gambar 5.2. Statistik Derau *Soundcard* dalam rms



Gambar 5.3. Statistik Derau *Soundcard*



Gambar 5.4. Spektrum Derau *Soundcard*

Dalam spektrum yang ditampilkan pada gambar 5.4 tampak bahwa derau merupakan sinyal derau putih, yang memiliki komponen frekuensi hampir merata di sepanjang lebar pita tanggapan *soundcard*. Penurunan magnitud derau pada frekuensi 20 kHz menandakan bahwa derau berasal dari bagian sebelum tapis anti-aliasing dalam *soundcard*. Kemungkinan derau ini berasal dari pencampur analog dalam *soundcard* dan kabel konektor *line-in*. Karena sumber-sumber ini masih berada dalam *soundcard* maka untuk selanjutnya derau ini disebut derau *soundcard*.

Untuk analisis resolusi dan akurasi, derau *soundcard* dimodelkan sebagai derau aditif, dimana derau bersifat menambahkan kepada sinyal masukan suatu kesalahan dengan rentang tetap (tidak bergantung pada besarnya sinyal masukan). Dengan asumsi distribusi derau adalah normal, maka 95% pembacaan akan berada dalam rentang \pm dua kali standar deviasi. Ini berarti, dengan tingkat keyakinan 95%, kesalahan pengukuran akibat dari derau adalah ± 32 atau $\pm 0,049\%$ dari skala penuh 65536. Dengan mempertimbangkan derau sebagai bagian yang menyatu dari *soundcard*, resolusi keseluruhan akan lebih bermakna jika dinyatakan dalam (rentang maksimum) / (rentang kesalahan pengukuran), yaitu sebesar

$$\frac{2^{16}}{4 \times 16,2329} \approx 1009 \text{ tingkatan yang dapat dibedakan.}$$

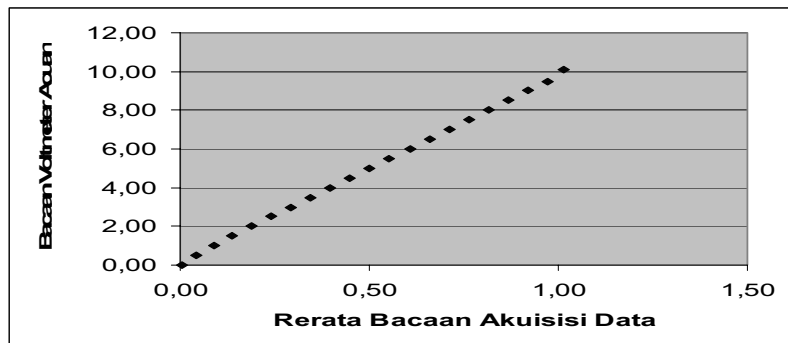
Resolusi ini adalah resolusi ideal yang dapat dicapai oleh sistem akuisi data yang dirancang dalam penelitian ini.

V.2. Akurasi dan Linearitas Sistem Akuisisi Data

Hasil uji akurasi dan linearitas ditampilkan dalam Tabel V.1. Asumsi yang dipakai adalah bahwa voltmeter yang dipakai sebagai acuan memiliki linearitas dan akurasi yang lebih baik dari sistem akuisisi data yang dibandingkan. Hasil yang diharapkan adalah pembacaan sistem akuisisi data menunjukkan nilai 0 pada masukan 0 volt dan 1 pada masukan 10 volt. Plot rerata hasil bacaan akuisisi data terhadap bacaan multimeter acuan ditampilkan pada Gambar 5.5.

Tabel V.1. Uji Akurasi dan Linearitas

hasil baca voltmeter (Vref)	Hasil baca sistem akuisisi data					
	rerata	median	Std Dev.	rentang	minimum	maks.
0	0,0028	0,0028	0,0001	0,0009	0,0023	0,0032
0,499	0,0401	0,0403	0,0007	0,0032	0,0376	0,0409
1,006	0,0889	0,0893	0,0015	0,0067	0,0838	0,0905
1,5	0,1371	0,1380	0,0025	0,0098	0,1298	0,1396
1,996	0,1874	0,1884	0,0029	0,0128	0,1778	0,1906
2,506	0,2399	0,2407	0,0029	0,0151	0,2284	0,2435
2,998	0,2896	0,2905	0,0037	0,0175	0,2769	0,2943
3,499	0,3407	0,3418	0,0044	0,0197	0,3270	0,3466
4,005	0,3918	0,3931	0,0055	0,0245	0,3742	0,3987
4,502	0,4425	0,4439	0,0059	0,0243	0,4242	0,4485
5,007	0,4970	0,4986	0,0047	0,0214	0,4803	0,5017
5,507	0,5481	0,5493	0,0052	0,0282	0,5287	0,5569
6,008	0,5983	0,6001	0,0061	0,0243	0,5810	0,6053
6,503	0,6520	0,6533	0,0053	0,0211	0,6389	0,6600
7,001	0,7023	0,7035	0,0060	0,0242	0,6909	0,7151
7,503	0,7542	0,7551	0,0068	0,0261	0,7418	0,7679
8,002	0,8062	0,8082	0,0056	0,0187	0,7958	0,8144
8,504	0,8581	0,8585	0,0072	0,0293	0,8426	0,8719
9,01	0,9100	0,9121	0,0078	0,0312	0,8947	0,9259
9,501	0,9584	0,9609	0,0078	0,0364	0,9426	0,9790
10,011	1,0036	1,0034	0,0070	0,0328	0,9929	1,0257

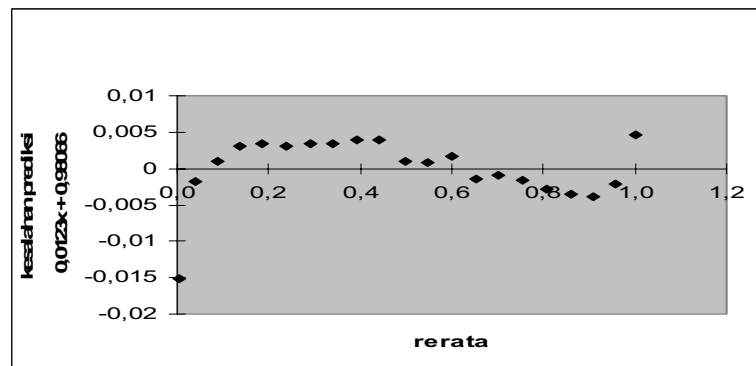


**Gambar 5.5. Plot Rerata Hasil Bacaan Akuisisi Data
Terhadap Bacaan Multimeter Acuan**

Analisis regresi linear antara rerata hasil baca sistem akuisisi (V_{rerata}) dan $1/10$ hasil baca voltmeter acuan (V_{ref}) menunjukkan hubungan

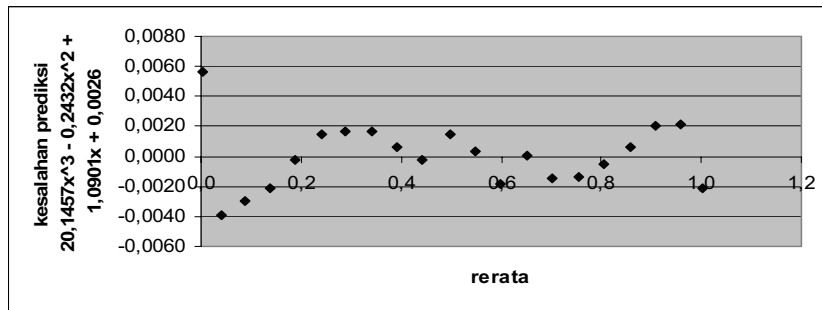
$$\frac{V_{\text{ref}}}{10} = 0,0123 + 0,9807 V_{\text{rerata}}, \text{ yang memberikan kesalahan prediksi seperti}$$

digambarkan dalam Gambar 5.6



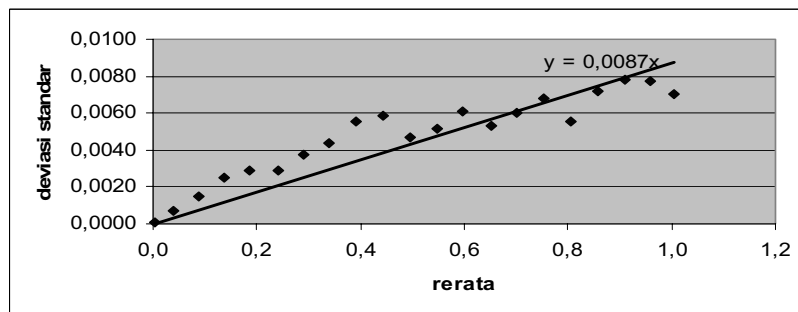
Gambar 5.6. Kesalahan Prediksi Regresi Linier

Kesalahan prediksi paling besar ada di titik nol volt sebesar 0,015. Kesalahan ini diakibatkan oleh nonlinearitas, sehingga nonlinearitas alat dapat dinyatakan sebesar 1,5 % dari bacaan skala penuh. Untuk memperbaiki kesalahan linearitas dapat digunakan linearisasi polinomial. Dengan polinomial $y = 0,1457x^3 - 0,2432x^2 + 1,0901x + 0,0026$; kesalahan linearitas dapat dikurangi menjadi 0,56 % bacaan skala penuh, seperti ditunjukkan pada Gambar 5.7.



Gambar 5.7. Kesalahan Regresi Polinomial

Deviasi dari nilai rata-rata mempunyai nilai yang kecil untuk sinyal masukan kecil, dan makin besar untuk sinyal masukan yang besar. Plot deviasi standar terhadap rerata ditampilkan dalam Gambar 5.8.



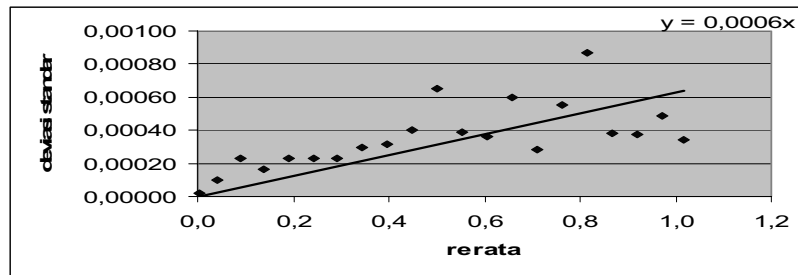
Gambar 5.8. Deviasi Standar Terhadap Rerata

Deviasi yang tergantung pada nilai reratanya menyebabkan ketidakpastian alat tidak dapat dinyatakan dalam nilai konstan, sehingga harus dinyatakan dalam prosentase dari nilai yang terbaca. Dengan regresi linear, deviasi dapat didekati dengan rumus $0,0087$ rerata. Dengan kriteria kesalahan ± 2 x deviasi standar, maka ketidakpastian

pengukurannya menjadi $\pm 1,74$ % dari hasil bacaan. Karena ketidakpastian terdistribusi secara normal, maka kesalahan yang sifatnya acak ini dapat dikurangi dengan merata-rata beberapa kali pengukuran. Mode lambat yang disediakan oleh aplikasi pendekode memberikan nilai rata-rata dari 80 kali pembacaan. Hasil pengujian dalam mode lambat disajikan pada Tabel V.2 Dan Gambar 5.9. Dengan metode analisis yang sama seperti pada analisis data hasil pengukuran mode cepat, pengukuran dalam mode lambat menunjukkan linearitas yang sama dengan mode cepat (1,5 % bacaan skala penuh) tetapi mempunyai ketidaktentuan yang lebih rendah, yaitu $\pm 0,24$ % hasil bacaan.

Tabel V.2. Hasil Uji Akurasi dan Linearitas Mode Lambat

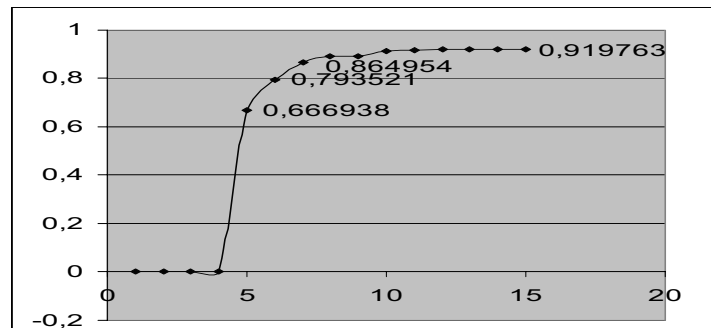
Hasil baca voltmeter	Hasil baca sistem akuisisi data	
	rerata	deviasi standar
0,000	0,00279	0,00002
0,499	0,04055	0,00010
1,006	0,08960	0,00023
1,500	0,13914	0,00016
1,996	0,18938	0,00023
2,506	0,24202	0,00023
2,996	0,29241	0,00023
3,498	0,34453	0,00030
4,004	0,39681	0,00032
4,052	0,44840	0,00040
5,007	0,50100	0,00065
5,506	0,55391	0,00039
6,006	0,60647	0,00036
6,502	0,65879	0,00060
7,002	0,71099	0,00029
7,502	0,76338	0,00055
8,001	0,81525	0,00087
8,503	0,86807	0,00038
9,007	0,92099	0,00038
9,498	0,97042	0,00049
10,080	1,01600	0,00034



Gambar 5.9. Deviasi Standar Terhadap Rerata Mode Lambat

V.3. Tanggapan Langkah

Hasil uji tanggapan langkah ditampilkan dalam Gambar 5.10. Untuk waktu mantap 5 %, waktu mantap instrumen orde satu sama dengan tiga kali konstanta waktu⁹⁾. Nilai yang stabil adalah 0,92 sehingga 5 % waktu mantap adalah waktu ketika hasil baca menunjukkan nilai 0,87. Tampak dalam Gambar 5.10, titik yang dekat dengan nilai ini adalah titik 0,86. Titik tersebut berselang tiga sampel, yang berarti mempunyai waktu mantap 30 mili detik. Ini sesuai dengan tiga kali konstanta waktu tapis lolos rendah pada sistem TDM yang dirancang sebesar 10 mili detik



Gambar 5.10. Hasil Uji Tanggapan Langkah

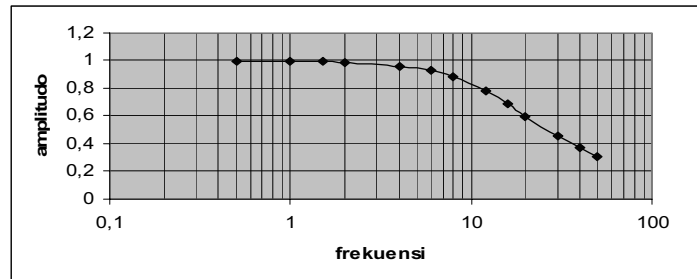
⁹⁾ Doebelin, 1983, *Sistem Pengukuran: Aplikasi dan Perancangan*, hal.102

V.4. Tanggapan Frekuensi

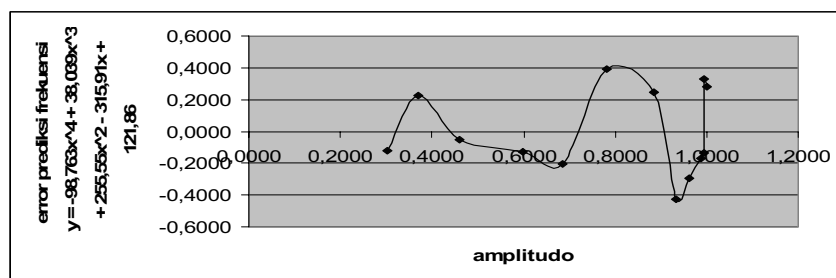
Hasil uji tanggapan frekuensi ditampilkan dalam Tabel V.3 dan Gambar 5.11. Titik sumbat -3 dB (sekitar 0,708 nilai maksimum) terletak diantara frekuensi 12 dan 16 Hz. Dengan interpolasi polinomial $y = -98,763x^4 + 38,039x^3 + 255,55x^2 - 315,91x + 121,86$, frekuensi pada amplitudo 0,708 (-3 dB) diprediksi sebesar 14,98 Hz, berbeda dengan rancangan tapis lolos rendah sistem TDM yang dirancang pada 15,92 Hz. Perbedaan ini kemungkinan disebabkan oleh kesalahan prediksi atau oleh ketidaktepatan komponen pembentuk filter pada sistem TDM. Kesalahan prediksi polinomial ditampilkan pada Gambar 5.12 menunjukkan kesalahan prediksi tidak lebih dari 0,4 Hz pada rentang amplitudo 0,6 sampai 0,8.

Tabel V.3. Hasil Uji Tanggapan Frekuensi

Frekuensi	Amplitudo	Amplitudo Yang Dinormalkan
0,5	0,8824	1,0000
1	0,8767	0,9934
1,5	0,8763	0,9930
2	0,8713	0,9873
4	0,8493	0,9624
6	0,8241	0,9338
8	0,7804	0,8843
12	0,6890	0,7808
16	0,6058	0,6864
20	0,5272	0,5974
30	0,4054	0,4594
40	0,3252	0,3685
50	0,2671	0,3027



Gambar 5.11. Tanggapan Frekuensi



Gambar 5.12. Kesalahan Prediksi Frekuensi

IV.5. Cakap Silang

Hasil uji cakap silang ditampilkan pada Tabel V.4. Cakap silang dari kanal k1 ke kanal k2 dalam dB dihitung dengan rumus

$20 \times \log \frac{\Delta S_2}{\Delta S_1}$, dengan ΔS_1 dan ΔS_2 masing masing adalah perubahan sinyal yang

terbaca pada kanal k1 dan k2 ketika kanal k1 mendapat perubahan masukan sementara k2 tidak mendapat perubahan masukan.

Tabel V.4. Hasil Uji Cakap Silang (dalam dB)

	ke kanal 0	ke kanal 1	ke kanal 2	ke kanal 3	ke kanal 4	ke kanal 5	ke kanal 6	ke kanal 7
Cakap silang dari kanal 0	0,00000	-66,72385	-89,07068	-116,69209	-119,34132	-116,99352	-113,64062	-129,07851
Cakap silang dari kanal 1	-120,64004	0,00000	-68,36003	-97,77778	-127,42294	-109,68744	-120,88135	-123,30340
Cakap silang dari kanal 2	-102,06620	-128,50163	0,00000	-66,82299	-86,68316	-116,69456	-102,03674	-126,38804
Cakap silang dari kanal 3	-95,55581	-114,72152	-103,88058	0,00000	-69,73866	-98,58588	-129,17756	-111,72284
Cakap silang dari kanal 4	-111,00775	-121,63184	-129,21041	-127,93218	0,00000	-69,32515	-88,37168	-129,14524
Cakap silang dari kanal 5	-127,66750	-115,81236	-94,75214	-118,30562	-99,92131	0,00000	-66,64935	-107,08778
Cakap silang dari kanal 6	-118,54997	-113,90157	-97,25754	-101,89898	-93,39695	-110,40075	0,00000	-69,30560
Cakap silang dari kanal 7	-102,91316	-99,58919	-110,38477	-95,67391	-94,26362	-106,75307	-121,70434	0,00000

Tabel V.5. Korelasi Cakap Silang Antar Kanal

	ke kanal 0	ke kanal 1	ke kanal 2	ke kanal 3	ke kanal 4	ke kanal 5	ke kanal 6	ke kanal 7
Korelasi dari kanal 0	1,00000	0,99732	0,65259	-0,00541	0,01132	0,05884	0,05879	-0,11027
Korelasi dari kanal 1	0,19392	1,00000	0,99859	0,45428	-0,11122	0,13762	0,11798	0,14179
Korelasi dari kanal 2	-0,08778	0,11066	1,00000	0,99767	0,78927	-0,13663	0,19965	0,04166
Korelasi dari kanal 3	-0,19297	0,15903	-0,07427	1,00000	0,99709	0,49013	-0,11821	-0,17002
Korelasi dari kanal 4	0,10543	-0,01272	0,17901	-0,09374	1,00000	0,99680	0,78794	0,14454
Korelasi dari kanal 5	0,19247	-0,05253	0,07187	-0,17188	0,03181	1,00000	0,99749	0,35569
Korelasi dari kanal 6	0,16396	-0,17383	-0,11578	-0,13169	-0,17223	-0,13719	1,00000	0,99842
Korelasi dari kanal 7	-0,09483	0,13171	0,11457	-0,08220	0,13969	-0,12179	0,15668	1,00000

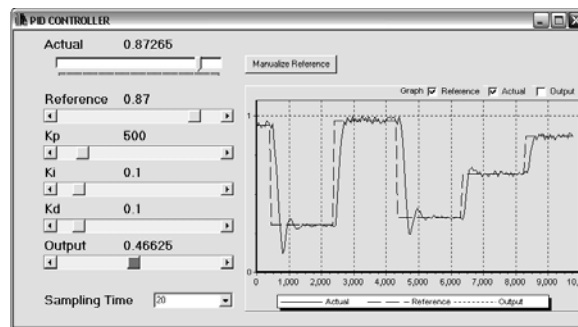
Cakap silang paling besar terjadi dari kanal 5 ke kanal 6 sebesar -66,65 dB. Nilai ini kemudian digunakan untuk menyatakan cakap silang maksimum antar kanal yang berdekatan. Koefisien korelasi dari uji cakap silang ditampilkan pada Tabel V.5., menunjukkan pengaruh yang signifikan dari kanal satu ke kanal berikutnya.

IV.6. Uji Aplikasi

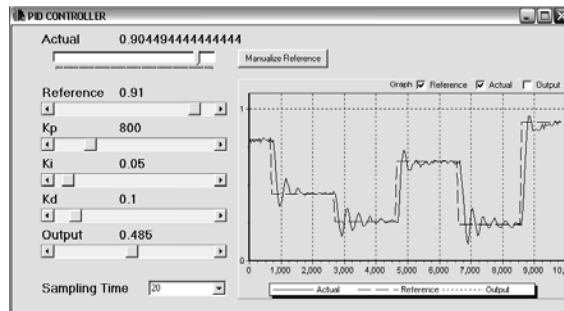
Uji aplikasi pengendali jenis PID untuk mengatur kecepatan motor DC ditampilkan pada Gambar 5.13, Gambar 5.14, dan Gambar 5.15. Ketiga gambar ditampilkan untuk melihat perbedaan tanggapan pengendali ketika parameter K_p

(Konstanta Proporsional) dan K_i (Konstanta Integral) diatur dengan kombinasi yang berbeda.

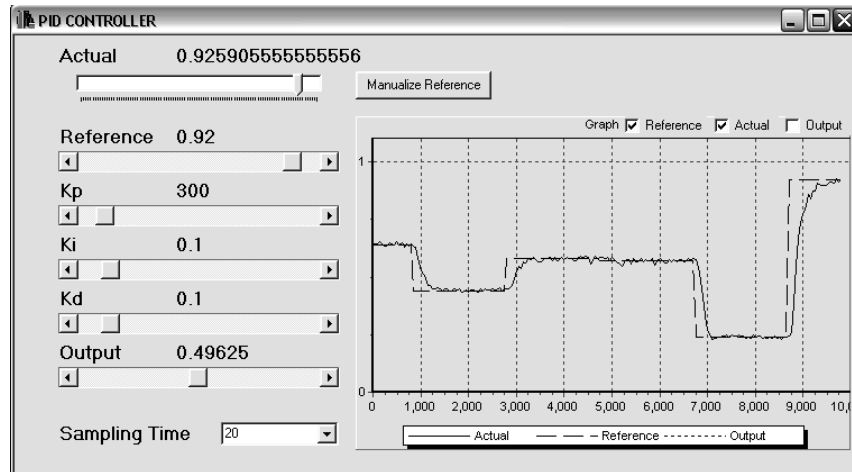
Uji aplikasi Termometer untuk memantau suhu udara ditampilkan pada Gambar 5.16. Dalam gambar tersebut tampak aplikasi pendekode sinyal (dengan window berjudul “ScdaqServer1”), pengendali PID untuk motor DC (dengan window berjudul “PID CONTROLLER”), dan Termometer (dengan window berjudul “Termometer”) berjalan bersama-sama dalam satu komputer. Ini menunjukkan keberhasilan sistem akuisisi untuk melayani lebih dari satu aplikasi.



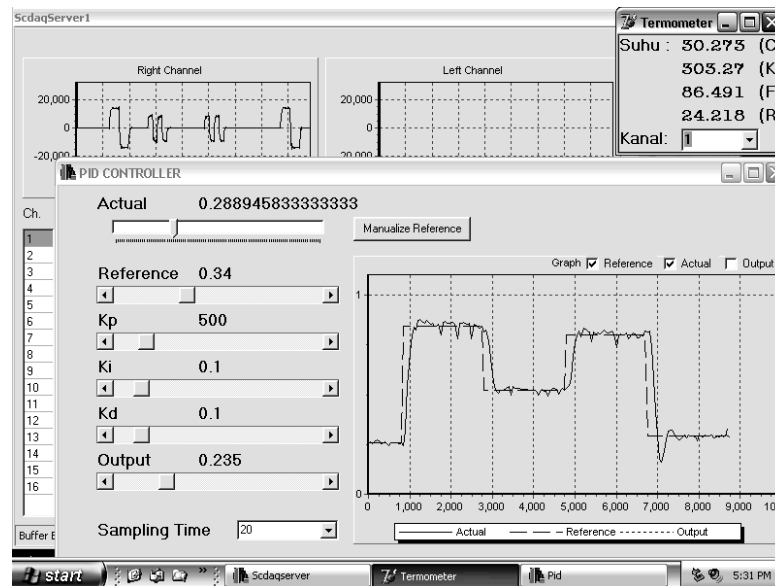
Gambar 5.13. Aplikasi Pengendali PID untuk Motor DC ($K_p=500$, $K_i=0,1$)



Gambar 5.14. Aplikasi Pengendali PID untuk Motor DC ($K_p=800$, $K_i=0,05$)



Gambar 5.15. Aplikasi Pengendali PID untuk Motor DC ($K_p=300$, $K_i=0,1$)



Gambar 5.16. Aplikasi Termometer Dijalankan bersama Aplikasi Pengendali PID

BAB VI. KESIMPULAN DAN SARAN

VI.1. Kesimpulan

1. Dengan menggunakan masukan *soundcard*, telah berhasil dirancang dan dibuat sebuah sistem akuisisi data banyak kanal yang bekerja dengan cara memultipleksi kanal data, kemudian memodulasikan sinyal termultipleksi ke sinyal pembawa. Sistem akuisisi yang dihasilkan mempunyai spesifikasi sebagai berikut:

- a. Jumlah kanal : 16 kanal
- b. Rentang masukan : 0 Volt s/d 10 Volt
- c. Laju *sampling* maksimum (tiap kanal) : 100 data /detik
- d. Akurasi (tingkat keyakinan 95%)
 - (mode cepat) : $\pm 1,74\%$ hasil bacaan
 - (mode lambat) : $\pm 0,24\%$ hasil bacaan
- e. Linearitas (tanpa koreksi polinomial) : 1,5 % skala penuh
(dengan koreksi polinomial) : 0,56 % skala penuh
- f. Cakap silang maksimum
(antar kanal yang berdekatan): -66,65 dB
- g. Tanggapan frekuensi (0 dB s/d -3dB) : 0Hz s/d 14,98 Hz
- h. Waktu mantap (5% kesalahan) : ≈ 30 mili detik

2. Untuk mempermudah pengguna dalam mengakses sistem akuisisi, telah diciptakan kelas `TSedaqInterface` (dalam C++ dan Pascal) dengan fungsi anggota:
 - a. `Open`, untuk menjalankan aplikasi pendekode.
 - b. `GetData`, untuk membaca data dengan mode cepat.
 - c. `GetSlowData`, untuk membaca data dengan mode lambat.

VI.2. Saran untuk Penelitian Selanjutnya

1. Perlu dikembangkan rancangan yang menghasilkan sistem yang lebih tahan derau (*Signa-to-Noise Ratio yang lebih tinggi*), sehingga akurasi dan resolusinya dapat ditingkatkan.
2. Perlu dibuat antar muka program dengan bahasa pemrograman yang lain, sehingga sistem akuisisi dapat dimanfaatkan oleh lebih banyak pengguna dengan penguasaan bahasa pemrograman yang beragam.
3. Perlu dikembangkan aplikasi yang menggunakan sistem akuisisi yang telah dibuat dalam penelitian ini, misalnya perangkat lunak osiloskop yang bisa bekerja pada mode AC dan DC, mengingat perangkat lunak osiloskop berbasis *soundcard* saat ini hanya bisa bekerja pada mode AC.
4. Pada kasus pengukuran untuk analisis keteknikan kadang-kadang diperlukan suatu pengukuran banyak titik secara serempak, untuk tujuan ini dapat dikembangkan sistem akuisisi dengan tambahan perangkat *sample and hold* pada rangkaian masukan.

DAFTAR PUSTAKA

- Amos, S.W.,1988, *Kamus Elektronika*, terj., PT Elex Media Komputindo (Kelompok Gramedia), Jakarta.
- Asche, Ruediger R., 1994, *The Little Device Driver Writer*, Device Driver Technical Articles, Microsoft Development Network Technology Group, -.
- Brodsky, Ethan, 1997, *Programming the SoundBlaster 16 DS*, Version 3.5 - 6/26/1997 , <http://www.harmony-central.com/>.
- Browning, Paul L., 1997, *Audio Digital Signal Processing in Real Time*, West Virginia University, Virginia.
- Carlson, A. Bruce, 1986, *Communication Systems : An Introduction to Signal and Noise in Electrical Communication*, 3rd Ed., pp. 203-204, 363, McGraw-Hill Book Company, Singapore.
- Comer, David J., 1981, *Electronic Design With Integrated Circuit*, pp. 283-284, Addison-Wesley Publishing Company, Canada.
- Doebelin, Ernest O., 1983, *Sistem Pengukuran: Aplikasi dan Perancangan*, Ed.3, Jilid 1, Terjemahan, Bab 3 hal 34-181, Penerbit Erlangga, Jakarta.
- Gibbon, James S., 1998, *More Real-Time Windows NT*, <http://home.pacifer.com/~jgibbon/>.
- Hunt, Galent C., 1997, *Creating User Mode Device Drivers with a Proxy*, Proceedings of the USENIX Windows NT Workshop, Seattle, Washington, Agustus 1997.
- Intel Corporation, 2002, *Audio Codec '97*, Revision 2.3 Revision 1.0, Intel Corporation, -.
- Ito, Hitoshi, -, *A Realtime Personal Computer: R&D Progress Report*, Mitsubishi Electric Advance, -.
- Johnson, D.E., Johnson, J.R., Moore, H.P., *A Handbook of Active Filters*, Prentice-Hall, Inc., New Jersey.
- Kardianto, 2004, *Pengembangan Audiometer Berbasis Soundcard pada Komputer Personal*, Universitas Gadjah Mada, Yogyakarta.
- Khotimah, Khusnul, 2004, *Identifikasi Kendaraan di Jembatan Menggunakan Piezoelectric*, Universitas Gadjah Mada, Yogyakarta.
- Lamothe, Andre, 1999, *Tricks of the Windows Game Programming Gurus: Fundamentals of 2D and 3D Game Programming*, Sams, Indianapolis.
- Microsoft Corporation, 1996, *Win32 Multimedia Programmer's Reference*, Online Help Reference, Microsoft Corporation.
-, 1996, *Win32 Programmer's Reference*, Online Help Reference, Microsoft Corporation.
- _____, 1999, *DirectX 7.0 Programmer's Reference*, DirectX Documentation, Mocosoft Corporation, -.
- PCplus, 2000, *Resep Praktis Memiliki Osiloskop Sendiri*, PCplus, No.02/I/24 Oktober 2000, hal.15.

- Ragoisha, Genady A., 2000, *Data Acquisition and Control in a User-Mode Real-Time System for Electrochemical Equipment*, Dedicated Systems Magazine (<http://www.dedicated-systems.com>), -.
- Sumawas, M. Utoyo, 2004, *Perancangan Perangkat Uji Transmission Loss pada Bahan Sekat*, Universitas Gadjah Mada, Yogyakarta.
- The Mathworks, Inc, 2002, *Data Acquisition Toolbox User's Guide*, Ver.2, Online Reference, The Mathworks, Inc, -.
- _____, 2002, *DSP Blockset User's Guide*, Ver.5, Sixth printing, The Mathworks, Inc., -.
- Thomason, Michael, 1979, *Handbook of Solid-State Devices*, Reston Publishing Company, Virginia.

LAMPIRAN

Lampiran 1. Servermain.dfm

```
object Form1: TForm1
  Left = 80
  Top = 12
  AutoSize = True
  BorderStyle = bsToolWindow
  Caption = 'Form1'
  ClientHeight = 535
  ClientWidth = 640
  Color = clBtnFace
  Font.Charset = DEFAULT_CHARSET
  Font.Color = clWindowText
  Font.Height = -11
  Font.Name = 'MS Sans Serif'
  Font.Style = []
  OldCreateOrder = False
  Position = poDesktopCenter
  OnClose = FormClose
  OnCloseQuery = FormCloseQuery
  OnCreate = FormCreate
  PixelsPerInch = 96
  TextHeight = 13
  object Label2: TLabel
    Left = 8
    Top = 184
    Width = 19
    Height = 16
    Caption = 'Ch.'
    Font.Charset = DEFAULT_CHARSET
    Font.Color = clWindowText
    Font.Height = -13
    Font.Name = 'MS Sans Serif'
    Font.Style = []
    ParentFont = False
  end
  object Label3: TLabel
    Left = 208
    Top = 184
    Width = 38
    Height = 16
    Caption = 'Result'
    Font.Charset = DEFAULT_CHARSET
    Font.Color = clWindowText
    Font.Height = -13
    Font.Name = 'MS Sans Serif'
    Font.Style = []
    ParentFont = False
  end
  object Label4: TLabel
    Left = 48
    Top = 184
    Width = 34
    Height = 16
    Caption = 'Offset'
    Font.Charset = DEFAULT_CHARSET
```

```
    Font.Color = clWindowText
    Font.Height = -13
    Font.Name = 'MS Sans Serif'
    Font.Style = []
    ParentFont = False
end
object Label5: TLabel
    Left = 128
    Top = 184
    Width = 28
    Height = 16
    Caption = 'Gain'
    Font.Charset = DEFAULT_CHARSET
    Font.Color = clWindowText
    Font.Height = -13
    Font.Name = 'MS Sans Serif'
    Font.Style = []
    ParentFont = False
end
object Label6: TLabel
    Left = 440
    Top = 376
    Width = 182
    Height = 13
    Caption = 'Buffer Overstack limit /10 ms to Beep :'
end
object Label7: TLabel
    Left = 440
    Top = 424
    Width = 145
    Height = 13
    Caption = 'Buffer Overstack Count / sec :'
end
object Label8: TLabel
    Left = 592
    Top = 424
    Width = 6
    Height = 13
    Caption = '0'
end
object StatusBar1: TStatusBar
    Left = 0
    Top = 511
    Width = 640
    Height = 24
    Panels = <>
    SimplePanel = False
end
object Chart1: TChart
    Left = 8
    Top = 32
    Width = 305
    Height = 145
    AllowPanning = pmNone
    BackWall.Brush.Style = bsClear
    Title.Text.Strings = (
```

```

    'Right Channel')
BottomAxis.LabelStyle = talValue
LeftAxis.LabelStyle = talValue
Legend.Visible = False
RightAxis.LabelStyle = talNone
TopAxis.LabelStyle = talNone
View3D = False
View3DWalls = False
TabOrder = 0
object Series1: TFastLineSeries
    Marks.ArrowLength = 8
    Marks.Visible = False
    LinePen.Color = clRed
    SeriesColor = clRed
    XValues.DateTime = False
    XValues.Name = 'X'
    XValues.Multiplier = 1
    XValues.Order = loAscending
    YValues.DateTime = False
    YValues.Name = 'Y'
    YValues.Multiplier = 1
    YValues.Order = loNone
end
end
object Chart2: TChart
    Left = 320
    Top = 32
    Width = 305
    Height = 145
    BackWall.Brush.Style = bsClear
    Title.Text.Strings = (
        'Left Channel')
    BottomAxis.LabelStyle = talValue
    LeftAxis.LabelStyle = talValue
    Legend.Visible = False
    View3D = False
    View3DWalls = False
    TabOrder = 1
    object Series2: TFastLineSeries
        Marks.ArrowLength = 8
        Marks.Visible = False
        LinePen.Color = clRed
        SeriesColor = clRed
        XValues.DateTime = False
        XValues.Name = 'X'
        XValues.Multiplier = 1
        XValues.Order = loAscending
        YValues.DateTime = False
        YValues.Name = 'Y'
        YValues.Multiplier = 1
        YValues.Order = loNone
    end
end
object StringGrid1: TStringGrid
    Left = 208
    Top = 208

```

```
Width = 137
Height = 297
ColCount = 1
DefaultColWidth = 130
DefaultRowHeight = 16
Enabled = False
FixedCols = 0
RowCount = 16
FixedRows = 0
ScrollBars = ssNone
TabOrder = 2
end
object StringGrid2: TStringGrid
Left = 8
Top = 208
Width = 33
Height = 297
ColCount = 1
DefaultColWidth = 30
DefaultRowHeight = 16
Enabled = False
FixedCols = 0
RowCount = 16
FixedRows = 0
ScrollBars = ssNone
TabOrder = 3
end
object Panell1: TPanel
Left = 472
Top = 184
Width = 153
Height = 161
TabOrder = 5
object Label1: TLabel
Left = 8
Top = 9
Width = 82
Height = 16
Caption = 'Frame Length'
Font.Charset = DEFAULT_CHARSET
Font.Color = clWindowText
Font.Height = -13
Font.Name = 'MS Sans Serif'
Font.Style = []
ParentFont = False
end
object Button1: TButton
Left = 96
Top = 8
Width = 43
Height = 45
Caption = 'Apply'
TabOrder = 0
OnClick = Button1Click
end
object Edit1: TEdit
```

```
    Left = 8
    Top = 32
    Width = 81
    Height = 21
    TabOrder = 1
    Text = '0.00748'
end
object Button2: TButton
    Left = 8
    Top = 64
    Width = 129
    Height = 41
    Caption = 'Flip Input'
    TabOrder = 2
    OnClick = Button2Click
end
object Button3: TButton
    Left = 8
    Top = 112
    Width = 129
    Height = 41
    Caption = 'Pause'
    TabOrder = 3
    OnClick = Button3Click
end
end
object ToolBar1: TToolBar
    Left = 0
    Top = 0
    Width = 640
    Height = 29
    Caption = 'ToolBar1'
    TabOrder = 6
end
object StringGrid3: TStringGrid
    Left = 48
    Top = 208
    Width = 73
    Height = 297
    ColCount = 1
    DefaultColWidth = 80
    DefaultRowHeight = 16
    FixedCols = 0
    RowCount = 16
    FixedRows = 0
    Options = [goFixedVertLine, goFixedHorzLine, goVertLine,
        goHorzLine, goRangeSelect, goEditing]
    ScrollBars = ssNone
    TabOrder = 7
    OnSetEditText = StringGrid3SetEditText
end
object StringGrid4: TStringGrid
    Left = 128
    Top = 208
    Width = 73
    Height = 297
```

```

ColCount = 1
DefaultColWidth = 80
DefaultRowHeight = 16
FixedCols = 0
RowCount = 16
FixedRows = 0
Options = [goFixedVertLine, goFixedHorzLine, goVertLine,
           goHorzLine, goRangeSelect, goEditing]
ScrollBars = ssNone
TabOrder = 8
OnSetEditText = StringGrid4SetEditText
end
object ComboBox1: TComboBox
  Left = 440
  Top = 392
  Width = 177
  Height = 21
  ItemHeight = 13
  Items.Strings = (
    '0'
    '1'
    '2'
    '3'
    '4'
    '5'
    '6'
    '7'
    '8')
  TabOrder = 9
  Text = '4'
  OnChange = ComboBox1Change
end
object Timer1: TTimer
  OnTimer = Timer1Timer
  Left = 192
  Top = 104
end
object Timer2: TTimer
  Enabled = False
  OnTimer = Timer2Timer
  Left = 224
  Top = 104
end
object Timer3: TTimer
  Enabled = False
  OnTimer = Timer3Timer
  Left = 136
  Top = 104
end
end
end

```

Lampiran 2. Servermain.h

```

/*=====
File      : servermain.h
Compiler  : Borlan C++ Builder Professional V.4.0
Deskripsi : Header file servermain.cpp, berisi deklarasi TForm1
           yang merupakan form utama aplikasi pendekode scdaq
           (sistem akuisisi data menggunakan masukan soundcard)
=====*/

#ifndef servermainH
#define servermainH
#define MAXNUMOFBLOCK 8
//-----
#include <Classes.hpp>
#include <syncobjs.hpp>
#include <Controls.hpp>
#include <StdCtrls.hpp>
#include <Forms.hpp>
#include <ExtCtrls.hpp>
#include "TWaveProcessor.h"
#include <Chart.hpp>
#include <Series.hpp>
#include <TeEngine.hpp>
#include <TeeProcs.hpp>
#include <Grids.hpp>
#include <ComCtrls.hpp>
#include <ToolWin.hpp>
#include <dos.h>

//-----
class TForm1 : public TForm
{
    __published:      // IDE-managed Components
        TTimer *Timer1;
        TChart *Chart1;
        TFastLineSeries *Series1;
        TChart *Chart2;
        TFastLineSeries *Series2;
        TTimer *Timer2;
        TStringGrid *StringGrid1;
        TStringGrid *StringGrid2;
        TStatusBar *StatusBar1;
        TPanel *Panel1;
        TButton *Button1;
        TLabel *Label1;
        TEdit *Edit1;
        TToolBar *ToolBar1;
        TButton *Button2;
        TButton *Button3;
        TLabel *Label2;
        TLabel *Label3;
        TStringGrid *StringGrid3;
        TLabel *Label4;
        TStringGrid *StringGrid4;
        TLabel *Label5;

```

```

TComboBox *ComboBox1;
TLabel *Label6;
TTimer *Timer3;
TLabel *Label7;
TLabel *Label8;
void __fastcall FormCreate(TObject *Sender);
void __fastcall Button3Click(TObject *Sender);
void __fastcall Timer1Timer(TObject *Sender);
void __fastcall Timer2Timer(TObject *Sender);
void __fastcall Button1Click(TObject *Sender);
void __fastcall Button2Click(TObject *Sender);
void __fastcall FormClose(TObject *Sender, TCloseAction
&Action);
void __fastcall StringGrid3SetEditText(TObject *Sender, int
ACol,
int ARow, const AnsiString Value);
void __fastcall StringGrid4SetEditText(TObject *Sender, int
ACol,
int ARow, const AnsiString Value);
void __fastcall FormCloseQuery(TObject *Sender, bool &CanClose);
void __fastcall Timer3Timer(TObject *Sender);
void __fastcall ComboBox1Change(TObject *Sender);

private: // User declarations
TWaveProcessor* WP;
WAVEFORMATEX waveFormat;
WAVEHDR header[10];
BYTE* pBuffer[10];
HWAVEIN hDevice;
TEvent* event;
double bufferLength;
int samplingRate;
void checkError(DWORD mmresult);
int count;
int overStackCount;
int overStackLimit;
HANDLE hMapFile;
HANDLE hMutex;
LPVOID lpSharedMemory;
char eventName[50];
char mutexName[50];
char offsetFileName[50];
char gainFileName[50];

public: // User declarations
__fastcall TForm1(TComponent* Owner);
};
//-----
extern PACKAGE TForm1 *Form1;
//-----
#endif

```

Lampiran 3. Servermain.cpp

```

/*=====
File      : servermain.cpp
Compiler  : Borland C++ Builder Professional V.4.0
Deskripsi : Implementasi TForm1, Form utama aplikasi pendekode
           sdaq ( sistem akuisisi data menggunakan masukan
           soundcard)
=====*/
//-----
#include <vcl.h>
#pragma hdrstop

#include "servermain.h"
//-----
#pragma package(smart_init)
#pragma resource "*.dfm"

TForm1 *Form1;
//-----
__fastcall TForm1::TForm1(TComponent* Owner)
    : TForm(Owner)
{
}
//-----

void __fastcall TForm1::FormCreate(TObject *Sender)
{
    char buff[256];

    //menghalangi aplikasi ketika dipanggil untuk memproses sinyal
    //dari soundcard yg sama
    strcpy(buff,"ScdaqServer"); //buff="ScdaqServer"
    if(_argc<2)
    {
        MessageBox(Handle,"You must run this server with
        Soundcard ID as parameter, exit now.,"Warning!!",MB_OK);
        exit(0);
    }
    else
        strcat(buff,_argv[1]); //buff="ScdaqServerX",
        X=parameter pemanggilan aplikasi server
    Form1->Caption=buff;
    HANDLE hIsRunningMutex=CreateMutex(0,false,buff);
    if(hIsRunningMutex==0)
    {
        MessageBox(Form1->Handle,"Cannot create
        mutex","ERROR",MB_OK);
        ExitProcess(1);
    }
    if(GetLastError()==ERROR_ALREADY_EXISTS)
        ExitProcess(0);

    //membuat mutex untuk sinkronisasi akses shared-memory antar
    //process
}

```

```

strcpy(mutexName, buff); //mutexName=ScdagServerX, X=parameter
    pemanggilan aplikasi server
strcat(mutexName, "Mutex"); //mutexName=ScdagserverXMutex
hMutex=CreateMutex(0, false, mutexName);
if (hMutex==0)
{
    MessageBox(Form1->Handle, "Cannot create
    mutex", "ERROR", MB_OK);
    ExitProcess(1);
}

//membuat object event untuk membuka soundcard
strcpy(eventName, buff); //eventName="ScdagServerX"
strcat(eventName, "Event"); //eventName="ScdagServer1Event"
event=new TEvent(0, false, false, eventName);

//membuat fileMapping object sebagai shared-memory
strcat(buff, "FileMap"); //buff="ScdagServerXFileMap"
hMapFile = CreateFileMapping((HANDLE)0xFFFFFFFF,
NULL, // Default security.
PAGE_READWRITE, // Read/write permission.
0, // hi object size.
sizeof(ScdagData), // Size of
    hFile.
buff); // Name of mapping object.
if (hMapFile == NULL)
{
    MessageBox(Handle, "Could not create file-mapping
    object.", 0, MB_OK);
    exit(1);
}

//Membuat fileView untuk mendapatkan alamat shared memory
lpSharedMemory = MapViewOfFile(hMapFile, // Handle to mapping
    object.
FILE_MAP_ALL_ACCESS, // Read/write permission
0, // Max. object size.
0, // Size of hFile.
0); // Map entire file.

if (lpSharedMemory == NULL)
{
    MessageBox(Handle, "Could not map view of file.", 0, MB_OK);
    exit(1);
}

//inisialisasi struktur waveFormat
samplingRate=44100;
bufferLength=0.01; //panjang buffer dalam detik
waveFormat.wFormatTag=WAVE_FORMAT_PCM;
waveFormat.nChannels=2;
waveFormat.nSamplesPerSec=samplingRate;
waveFormat.nAvgBytesPerSec=4*samplingRate;
waveFormat.nBlockAlign=4;
waveFormat.wBitsPerSample=16;

```

```

//membuka soundcard
checkError(waveInOpen(&hDevice,atoi(_argv[1])-
    1,&waveFormat,(DWORD)event->Handle,0,CALLBACK_EVENT));
if(hDevice==0)
{
    MessageBox(0,"Cannot open recording device, now will
        close.",0,MB_OK);
    ExitProcess(1);
}

//alokasi memory buffer
for(int i=0;i<MAXNUMOFBLOCK;i++)
{
    header[i].dwBufferLength=bufferLength*samplingRate*4;
    delete[] pBuffer[i];
    pBuffer[i]=new BYTE[header[i].dwBufferLength];
    header[i].lpData=(char*)pBuffer[i] ;
    header[i].dwFlags=0;

    checkError(waveInPrepareHeader(hDevice,&header[i],sizeof(WA
        VEHDR)));

    checkError(waveInAddBuffer(hDevice,&header[i],sizeof(WAVEHD
        R)));
}

for(int i=0;i<16;i++)
{
    StringGrid3->Cells[0][i]= 0; //offset;
    StringGrid4->Cells[0][i]= 1.0; //Gain
}

//membuat nama file offset dan gain
strcpy(offsetFileName,"sdaqOffset");
strcat(offsetFileName,_argv[1]);
strcpy(gainFileName,"sdaqGain");
strcat(gainFileName,_argv[1]);

//loading offset dan gain
StringGrid3->Cols[0]->LoadFromFile(offsetFileName);
StringGrid4->Cols[0]->LoadFromFile(gainFileName);

//membentuk, menginisialisasi, dan menjalankan thread pemroses
sinyal
WP=new TWaveProcessor(true);
WP->Event=event;
WP->hMutex=hMutex;
WP->SeriesRight=Series1;
WP->SeriesLeft=Series2;
WP->numberOfBuffer=8;
WP->pHeader=header;
WP->Priority=tpHighest;
WP->Count=&count;
WP->OverStackCount=&overStackCount;
overStackLimit=4;
WP->OverStackLimit= &overStackLimit;

```

```

WP->pSharedData=(ScdaqData*) lpSharedMemory;
WP->SamplesPerBuff=samplingRate*bufferLength;
WP->FrameLength=0.00748;
WP->SamplingRate=44100;
WP->hDevice=hDevice;
//inisialisasi offset dan gain
for(int i=0;i<16;i++)
{
    WP->pData->Gain[i]=StringGrid4->Cells[0][i].ToDouble();
    WP->pData->Offset[i]=StringGrid3->Cells[0][i].ToDouble();
}
WP->Resume();    //menjalankan thread pemroses sinyal

//menjalankan soundcard
checkError(waveInStart(hDevice));

//menjalankan timer untuk me-refresh tampilan
Timer2->Enabled=true;
Timer3->Enabled=true;

for(int i=0;i<16;i++)
    StringGrid2->Cells[0][i]=i+1;
}
//-----

void TForm1::checkError(DWORD result)
{
    if(result==0)
        return;
    char buff[256];
    waveInGetErrorText(result, buff, sizeof(buff));
    MessageBox(0, buff, "Wave In Error", MB_OK);
}

void __fastcall TForm1::Button3Click(TObject *Sender)
{
    if(Button3->Tag==0)
    {
        WP->Stop();
        Button3->Tag=1;
        Button3->Caption="Continue";
    }
    else
    {
        WP->Resume();
        Button3->Tag=0;
        Button3->Caption="Pause";
    }
}
//-----

void __fastcall TForm1::Timer1Timer(TObject *Sender)
{
    char text[50];
    strcpy(text, " Buffer Blocks / second : ");
    char buff[5];

```

```

        itoa(count, buff, 10);
        strcat(text, buff);
        StatusBar1->SimpleText=text;
        count=0;
    }
    //-----

void __fastcall TForm1::Timer2Timer(TObject *Sender)
{
    WaitForSingleObject(hMutex, INFINITE);
    for(int i=0; i<16; i++)
        StringGrid1->Cells[0][i]=WP->pSharedData-
            >ChannelSlow[i];
    ReleaseMutex(hMutex);
}

//-----

void __fastcall TForm1::Button1Click(TObject *Sender)
{
    WP->FrameLength=Edit1->Text.ToDouble();
}

//-----

void __fastcall TForm1::Button2Click(TObject *Sender)
{
    if(WP->Flipped)
        WP->Flipped=false;
    else WP->Flipped=true;
}

//-----

void __fastcall TForm1::FormClose(TObject *Sender, TCloseAction
    &Action)
{
    WP->Terminate();
    UnmapViewOfFile(lpSharedMemory);
    CloseHandle(hMapFile);
    StringGrid3->Cols[0]->SaveToFile(offsetFileName);
    StringGrid4->Cols[0]->SaveToFile(gainFileName);
}

//-----

void __fastcall TForm1::StringGrid3SetEditText(TObject *Sender, int
    ACol,
    int ARow, const AnsiString Value)
{
    double temp=atof(Value.c_str());
    WP->pData->Offset[ARow]=temp;
}

//-----

void __fastcall TForm1::StringGrid4SetEditText(TObject *Sender, int
    ACol,
    int ARow, const AnsiString Value)

```

```
{
    double temp=atof(Value.c_str());
    WP->pData->Gain[ARow]=temp;
}

//-----
void __fastcall TForm1::FormCloseQuery(TObject *Sender, bool
    &CanClose)
{
    if(MessageBox(Handle,"Make sure there's no other application
        using scdaq","Warning",MB_OKCANCEL)==IDOK)
        CanClose=true;
    else CanClose=false;
}

//-----
void __fastcall TForm1::Timer3Timer(TObject *Sender)
{
    Label18->Caption=overStackCount;
    overStackCount=0;
}

//-----
void __fastcall TForm1::ComboBox1Change(TObject *Sender)
{
    overStackLimit=ComboBox1->Text.ToIntDef(4);
}
//-----
```

Lampiran 4. TWaveProcessor.h

```

/*=====
File      :TWaveProcessor.h
Compiler  :Borland C++ Builder Professional V.4.0
Deskripsi :Header file TWaveProcessor.cpp, abstraksi kelas thread
           pemroses sinyal sdaq (data akuisisi menggunakan masukan
           soundcard)
=====*/
#ifndef TWaveProcessorH
#define TWaveProcessorH
//-----
#include <Classes.hpp>
#include <mmsystem.h>
#include <syncobjs.hpp>
#include <series.hpp>
#include "TDrawChart.h"
#include "TBeep.h"
#include <math.h>
#include "TScdaqInterface.h"

//-----
class TWaveProcessor : public TThread
{
private:
    short *pLeft, *pRight;
    void checkError(DWORD mmresult);
    void splitBuffer(WAVEHDR pHdr);
    void __fastcall paintChart();
    double numberOfSamplesPerFrame;
    double numberOfSamplesPerChannel;
    int loOffset;
    int hiOffset;
    int len; //panjang sampel dari channel yg dianggap stabil
    int minSyncLen; //panjang 1/2 periode sinyal sinkronisasi
                minimum agar terdeteksi
    TDrawChart* DrawChartLeft;
    TDrawChart* DrawChartRight;
    TBeep* Beep;
    int __fastcall findFrame(short* buff, int buffLen);
    bool decode(void);
    bool isEdge(int index, short* buff);
    bool isValidSync(int index, short * buff);
    int stackCount;
    int drawCount;
    double slowTemp[16];
    int slowCount;

protected:
    void __fastcall Execute();

public:
    __fastcall TWaveProcessor(bool CreateSuspended);
    int numberOfBuffer;
    int SamplesPerBuff;

```

```
int SamplingRate;
int* OverStackLimit; //batas penumpukan buffer untuk membunyikan
    beep
int* OverStackCount; //jumlah penumpukan buffer
int* Count; //jumlah buffer yg telah diisi oleh soundcard
char* lpctrEventName;
void Stop(void);
bool Flipped;
double FrameLength; //panjang frame dalam detik
HWAVEIN hDevice;
HANDLE hMutex;
TEvent* Event;
TFastLineSeries* SeriesRight;
TFastLineSeries* SeriesLeft;
WAVEHDR* pHeader;
ScdaqData* pSharedData;
ScdaqData* pData;
};
//-----
#endif
```

Lampiran 5. TWaveProcessor.cpp

```

/*=====
File      :TWaveProcessor.cpp
Compiler  :Borland C++ Builder Professional V.4.0
Deskripsi :Abstraksi kelas thread pemroses sinyal scdaq (data
           akuisisi menggunakan masukan soundcard)
=====*/
#include <vcl.h>
#pragma hdrstop

#include "TWaveProcessor.h"
#include <winuser.h>
#pragma package (smart_init)

//-----
// Important: Methods and properties of objects in VCL can only be
// used in a method called using Synchronize, for example:
//
//     Synchronize(UpdateCaption);
//
// where UpdateCaption could look like:
//
//     void __fastcall TWaveProcessor::UpdateCaption()
//     {
//         Form1->Caption = "Updated in a thread";
//     }
//-----
__fastcall TWaveProcessor::TWaveProcessor(bool CreateSuspended)
    : TThread(CreateSuspended)
{
    SeriesRight=0;    //inisialisasi pointer dengan 0
    SeriesLeft=0;    //inisialisasi pointer dengan 0
    DrawChartRight=0;
    DrawChartLeft=0;
    Beep=new TBeep(false);
    stackCount=0;
    drawCount=0;
    slowCount=0;
    SamplingRate=44100;
    OverStackCount=0;
    pData=new ScdaqData;
}
//-----
void __fastcall TWaveProcessor::Execute()
{
    pLeft=new short[SamplesPerBuff];
    pRight=new short[SamplesPerBuff];

    while(!Terminated)
    {
        if (wrTimeout==Event->WaitFor(100))
        {
            Beep->Play();
            continue;
        }
    }
}

```

```

stackCount=0;
for(int i=0;i<numberOfBuffer;i++)
{
    if(pHeader[i].dwFlags&WHDR_DONE)
    {
        splitBuffer(pHeader[i]);
        decode();
        if(drawCount>12)
        {
            paintChart();
            drawCount=0;
        }else drawCount++;

        pHeader[i].dwFlags=0;

checkError(waveInUnprepareHeader(hDevice,&pHeader[i],sizeof(WAVEHDR)
));

checkError(waveInPrepareHeader(hDevice,&pHeader[i],sizeof(WAVEHDR)))
;

checkError(waveInAddBuffer(hDevice,&pHeader[i],sizeof(WAVEHDR)));
        stackCount++;
        if(Count!=0) //jika pointer berisi alamat yg
valid
            (*Count)++;
    }
}
if(stackCount>*OverStackLimit)
    Beep->Play();
if(stackCount>1)
    (*OverStackCount)+=stackCount-1;

}
}
//-----

void TWaveProcessor::checkError(DWORD result)
{
    if(result==0)
        return;
    char buff[256];
    waveInGetErrorText(result, buff, sizeof(buff));
    MessageBox(0, buff, "Wave In Error", MB_OK);
}
//-----

void TWaveProcessor::splitBuffer(WAVEHDR pHdr)
{
    short* pBuffer;
    pBuffer=(short*)pHdr.lpData; //konversi ke pointer 16 bit
    for(int i=0,j=0;i<(2*SamplesPerBuff);i=i+2,j++)
    {
        if(Flipped)
        {
            pLeft[j]=-pBuffer[i];
            pRight[j]=-pBuffer[i+1];

```

```

        }else
        {
            pLeft[j]=pBuffer[i];
            pRight[j]=pBuffer[i+1];
        }
    }
}

//-----
void __fastcall TWaveProcessor::paintChart()
{
    if(SeriesLeft!=0)
    {
        DrawChartLeft=new TDrawChart(true);
        DrawChartLeft->hCallerThread=Handle;
        DrawChartLeft->NumberOfData=SamplesPerBuff;
        DrawChartLeft->pData=pLeft;
        DrawChartLeft->Series=SeriesLeft;
        DrawChartLeft->Priority=tpHighest;
        DrawChartLeft->FreeOnTerminate=true;
        DrawChartLeft->Resume();
    }
    if(SeriesRight!=0)
    {
        DrawChartRight=new TDrawChart(true);
        DrawChartRight->hCallerThread=Handle;
        DrawChartRight->NumberOfData=SamplesPerBuff;
        DrawChartRight->pData=pRight;
        DrawChartRight->Series=SeriesRight;
        DrawChartRight->Priority=tpHighest;
        DrawChartRight->FreeOnTerminate=true;
        DrawChartRight->Resume();
        Suspend();
    }
}

//-----
int __fastcall TWaveProcessor::findFrame(short* buff,int buffLen)
{
    for(int index=0;index<buffLen;index++)
    {
        if(isEdge(index,buff)==true)
        {
            if(isValidSync(index+1,buff)==true)
                return index+1;
        }
    }
    return -1; //gagal menemukan sinyal sinkronisasi
}

//-----

bool TWaveProcessor::decode(void)
{
    numberOfSamplesPerFrame=FrameLength*(double)SamplingRate;
}

```

```

numberOfSamplesPerChannel=numberOfSamplesPerFrame/9;
//8 channel + 1 sync

loOffset=0.875*numberOfSamplesPerChannel;
hiOffset=0.25*numberOfSamplesPerChannel;
len=numberOfSamplesPerChannel/9;
int sof; //start of frame
int loIndex,hiIndex; //indeks posisi puncak pulsa
double loTempRight,loTempLeft,hiTempRight,hiTempLeft;

sof=findFrame(pRight,SamplesPerBuff);
if(sof<0)
    sof=findFrame(pLeft,SamplesPerBuff);
if(sof<0)
    return false; //gagal menemukan sinyal sinkronisasi;

for(int i=0;i<8;i++)
{
    loIndex=sof+(i*numberOfSamplesPerChannel)+loOffset;
    hiIndex=loIndex+hiOffset;

    if(loIndex+len>=SamplesPerBuff)
        loIndex=loIndex-numberOfSamplesPerFrame;

    if(hiIndex+len>=SamplesPerBuff)
        hiIndex=hiIndex-numberOfSamplesPerFrame;

    loTempRight=0;
    loTempLeft=0;
    hiTempRight=0;
    hiTempLeft=0;
    for(int j=0;j<len;j++)
    {
        loTempRight+=pRight[loIndex+j];
        hiTempRight+=pRight[hiIndex+j];
        loTempLeft+=pLeft[loIndex+j];
        hiTempLeft+=pLeft[hiIndex+j];
    }
    pData->Channel[i]=pData->Offset[i]
        +pData->Gain[i]*( fabs(hiTempRight-
            loTempRight)/(double)(len*60000));
    pData->Channel[i+8]=pData->Offset[i+8]
        +pData->Gain[i+8]*( fabs(hiTempLeft-
            loTempLeft)/(double)(len*60000));
    slowTemp[i]+=pData->Channel[i];
    slowTemp[i+8]+=pData->Channel[i+8];
    if(slowCount>=80)
    {
        pData->ChannelSlow[i]=slowTemp[i]/80.0;
        pData->ChannelSlow[i+8]=slowTemp[i+8]/80.0;
        slowCount=0;
        slowTemp[i]=0;
        slowTemp[i+8]=0;
    }else slowCount++;
}

```

```

//sinkronisasi
WaitForSingleObject(hMutex,INFINITE);
    *pSharedData=*pData; //menyalin data yg terdekode
                                //ke shared Memory
ReleaseMutex(hMutex);
return true;
}

//-----
bool TWaveProcessor::isEdge(int index, short* buff)
{
    if(index+1>=SamplesPerBuff) //di luar buffer
        return false;
    if((buff[index]>=0)&&(buff[index+1]<0)) //sisi turun
        return true;
    return false;
}

bool TWaveProcessor::isValidSync(int index, short * buff)
{
    minSyncLen=numberOfSamplesPerChannel/2.4;
    if(index+minSyncLen>=SamplesPerBuff) //jika perpanjangannya
        // di luar buffer
        return false;
    for(int i=1;i<minSyncLen;i++) //cek konsistensi taraf
    {
        if(buff[index+i]>=-10000)
            return false;
        if(buff[index+i]<-20000)
            return false;
    }
    return true;
}

void TWaveProcessor::Stop(void)
{
    DrawChartRight->Terminate();
    DrawChartLeft->Terminate();
    Suspend();
}

```

Lampiran 6. TBeep.h

```
/*=====
File      :TBeep.h
Compiler  :Borland C++ Builder Professional V.4.0
Deskripsi :Header file TBeep.cpp, abstraksi kelas TBeep,
           kelas untuk membunyikan Beep secara asinkron.
=====
#ifndef TBeepH
#define TBeepH
//-----
#include <Classes.hpp>
//-----
class TBeep : public TThread
{
private:
protected:
    void __fastcall Execute();
public:
    __fastcall TBeep(bool CreateSuspended);
    void Play(void);
};
//-----
#endif
```

Lampiran 7. TBeep.cpp

```

/*=====
File      :TBeep.cpp
Compiler  :Borland C++ Builder Professional V.4.0
Deskripsi :Abstraksi kelas TBeep, kelas untuk
           membunyikan Beep secara asinkron.
=====*/
#include <vcl.h>
#pragma hdrstop

#include "TBeep.h"
#pragma package(smart_init)
//-----
// Important: Methods and properties of objects in VCL can only be
// used in a method called using Synchronize, for example:
//
//     Synchronize(UpdateCaption);
//
// where UpdateCaption could look like:
//
//     void __fastcall TBeep::UpdateCaption()
//     {
//         Form1->Caption = "Updated in a thread";
//     }
//-----
__fastcall TBeep::TBeep(bool CreateSuspended)
    : TThread(CreateSuspended)
{
    Resume();
}
//-----
void __fastcall TBeep::Execute()
{
    //---- Place thread code here ----
    while(!Terminated)
    {
        Suspend();
        Beep(1000,100);
    }
}
//-----

void TBeep::Play(void)
{
    Resume();
}

```

Lampiran 8. TDrawChart.h

```
/*=====
File      :TDrawChart.h
Compiler  :Borland C++ Builder Professional V.4.0
Deskripsi :Header file TDrawChart.cpp, abstraksi kelas TDrawChart,
           kelas untuk menggambar grafik secara asinkron.
=====*/
#ifndef TDrawChartH
#define TDrawChartH
//-----
#include <Classes.hpp>
#include <series.hpp>
//-----
class TDrawChart : public TThread
{
private:
    short* pBuffer;
    void __fastcall draw();
protected:
    void __fastcall Execute();
public:
    __fastcall TDrawChart(bool CreateSuspended);
    DWORD hCallerThread;
    int NumberOfData;
    short* pData;
    TFastLineSeries* Series;
};
//-----
#endif
```

Lampiran 9. TDrawChart.cpp

```

/*=====
File      :TDrawChart.cpp
Compiler  :Borland C++ Builder Professional V.4.0
Deskripsi :Abstraksi kelas TDrawChart,
           kelas untuk menggambar grafik secara asinkron.
=====
#include <vcl.h>
#pragma hdrstop

#include "TDrawChart.h"
#pragma package(smart_init)
//-----
// Important: Methods and properties of objects in VCL can only be
// used in a method called using Synchronize, for example:
//
//     Synchronize(UpdateCaption);
//
// where UpdateCaption could look like:
//
//     void __fastcall TDrawChart::UpdateCaption()
//     {
//         Form1->Caption = "Updated in a thread";
//     }
//-----
__fastcall TDrawChart::TDrawChart(bool CreateSuspended)
    : TThread(CreateSuspended)
{
}
//-----
void __fastcall TDrawChart::Execute()
{
    pBuffer=new short[NumberOfData];

    for(int i=0;i<NumberOfData;i++)
        pBuffer[i]=pData[i];
    ResumeThread( (HANDLE)hCallerThread);
    Synchronize(draw);
}
//-----

void __fastcall TDrawChart::draw()
{
    Series->Clear();
    Series->Add(-32800," ",clWhite);
    Series->Add(32800," ",clWhite);
    for(int i=0;i<NumberOfData;i++)
        Series->Add(pBuffer[i]," ",clRed);
}

```

Lampiran 10. TScdaqInterface.h

```

/*=====
File      :TScdaqInterface.h
Compiler  :Borland C++ Builder Professional V.4.0
Deskripsi :Header file TScdaqInterface.cpp, abstraksi kelas
           untuk mengakses scdaq (sistem akuisisi data menggunakan
           masukan soundcard).
=====
#ifndef TScdaqInterfaceH
#define TScdaqInterfaceH
#include <stdlib.h>
//-----
struct ScdaqData
{
    double Channel[16];
    double ChannelSlow[16];
    double Gain[16];
    double Offset[16];
};

class TScdaqInterface
{
private:
    STARTUPINFO SI;          //struktur untuk mengkonfigurasi
                            //pemanggilan aplikasi pendekode
    PROCESS_INFORMATION PI; //struktur untuk memantau keadaan
                            //aplikasi pendekode
    ScdaqData* data;        //struktur untuk menangani akses memori
bersama
    HANDLE hMapFile;
    HANDLE hMutex;
    bool opened;
    void showError(DWORD errorCode);
public:
    TScdaqInterface();
    bool Open(unsigned int DeviceID);
    double getData(int channelNumber);
    double getSlowData(int channelNumber);
    ~TScdaqInterface();
};
#endif

```

Lampiran 11. TScdaqInterface.cpp

```

/*=====
File      :TScdaqInterface.cpp
Compiler  :Borland C++ Builder Professional V.4.0
Deskripsi :Abstraksi kelas
           untuk mengakses scdag (sistem akuisisi data menggunakan
           masukan soundcard).
=====
#include <vcl.h>
#pragma hdrstop

#include "TScdaqInterface.h"

//-----
#pragma package(smart_init)

TScdaqInterface::TScdaqInterface()
{
    opened=false;
}

//-----
bool TScdaqInterface::Open(unsigned int DeviceID)
{
    if(opened)
        return false;
    char cmdLine[50];
    char idString[5];
    itoa(DeviceID,idString,10);
    strcpy(cmdLine,"ScdaqServer ");
    strcat(cmdLine,idString); //cmdLine="ScdaqServer X", X adalah
DeviceID

    DWORD creationFlag=HIGH_PRIORITY_CLASS;
    SI.wShowWindow=SW_MAXIMIZE;
    SI.dwFlags=STARTF_USESHOWWINDOW;
    if(!CreateProcess(0,cmdLine,0,0,true,creationFlag,0,0,&SI,&PI))
//menjalankan aplikasi server
    {
        MessageBox(0,"Cannot run ScdaqServer",0,MB_OK);
        return false;
    }

    WaitForInputIdle(PI.hProcess ,10000);
    unsigned long code;
    GetExitCodeProcess(PI.hProcess, &code);
    if(code!=STILL_ACTIVE)
    {
        if(code==1)
        {
            MessageBox(0,"Cannot Start Server",0,MB_OK);
            return false;
        }
    }
}

```

```

    //membuka mutex untuk sinkronisasi akses shared-memory antar
process
    char mutexName[50];
    strcpy(mutexName, "ScdaqServer"); //mutexName=ScdaqServer
    strcat(mutexName, idString); //mutexName=ScdaqServerX, X=id
    strcat(mutexName, "Mutex"); //mutexName=ScdaqServerXMutex
    hMutex=OpenMutex(MUTEX_ALL_ACCESS, false, mutexName);
    if (hMutex==0)
    {
        MessageBox(0, "Cannot open synchronization
object", "ERROR", MB_OK);
        showError(GetLastError());
        return false;
    }

    //membuka fileMapping object sebagai shared-memory
    char fileMapName[50];
    strcpy(fileMapName, "ScdaqServer"); //mutexName=ScdaqServer
    strcat(fileMapName, idString); //mutexName=ScdaqServerX, X=id
    strcat(fileMapName, "FileMap");
//mutexName=ScdaqServerXFileMap
    hMapFile =OpenFileMapping(FILE_MAP_READ , false, fileMapName);

    if (hMapFile == NULL)
    {
        MessageBox(0, "Cannot open shared memory
object", "ERROR", MB_OK);
        showError(GetLastError());
        return false;
    }

    //Membuat fileView untuk mendapatkan alamat shared memory
data=(ScdaqData*) MapViewOfFile(hMapFile, // Handle to mapping
object.
    FILE_MAP_READ, // Read/write permission
    0, // Max. object size.
    0, // Size of hFile.
    0); // Map entire file.

    if (data == NULL)
    {
        MessageBox(0, "Cannot map shared memory.", 0, MB_OK);
        showError(GetLastError());
        return false;
    }

    opened=true;
    return true;
}

//-----
void TScdaqInterface::showError(DWORD errorCode)
{
    LPVOID lpMsgBuf;
    FormatMessage(
        FORMAT_MESSAGE_ALLOCATE_BUFFER | FORMAT_MESSAGE_FROM_SYSTEM,

```

```

    NULL,
    errorCode,
    LANG_ENGLISH, // Default language
    (LPTSTR) &lpMsgBuf,
    0,
    NULL);

    // Display the string.
    MessageBox( NULL,
(char*)lpMsgBuf, "GetLastError", MB_OK|MB_ICONINFORMATION);

    // Free the buffer.
    LocalFree( lpMsgBuf );
}

//-----
double TScdaqInterface::getData(int channelNumber)
{
    if(opened==false)
        return -3;
    WaitForSingleObject(hMutex, INFINITE);
    double temp=data->Channel[channelNumber];
    ReleaseMutex(hMutex);
    return temp;
}

//-----
double TScdaqInterface::getSlowData(int channelNumber)
{
    if(opened==false)
        return -3;
    WaitForSingleObject(hMutex, INFINITE);
    double temp=data->ChannelSlow[channelNumber];
    ReleaseMutex(hMutex);
    return temp;
}

TScdaqInterface::~TScdaqInterface()
{
    UnmapViewOfFile(data);
    CloseHandle(hMapFile);
    CloseHandle(hMutex);
}

```

Lampiran 12. ScdqInterface.pas

```

=====
File      :ScdaqInterface.pas
Compiler  :Delphi 7
Deskripsi :Abstraksi kelas TScdaqInterface, kelas untuk mengakses
           scdaq (sistem akuisisi data menggunakan masukan
           soundcard).
=====
unit ScdaqInterface;

interface

uses
  SysUtils,Windows;

type
  TScdaqInterface=class
  private
    SI:_STARTUPINFOA; //struktur untuk mengkonfigurasi pemanggilan
  ap. pendekode
    PI:_PROCESS_INFORMATION; //struktur untuk memantau keadaan
  aplikasi pendekode
    hMapFile:cardinal;
    hMutex:cardinal;
    opened:boolean;
    pByte:PChar;
    procedure showError(errorCode:cardinal);
  public
    constructor TScdaqInterface();
    function Open(DeviceID:integer):boolean;
    function getData(channelNumber:integer):double;
    function getSlowData(channelNumber:integer):double;

  end;

implementation

function TScdaqInterface.Open(DeviceID:integer):boolean;
var
  cmdLine:array[0..49]of char;
  idString:array[0..4]of char;
  creationFlag:cardinal;
  exitCode:cardinal;
  mutexName:array[0..49]of char;
  fileMapName:array[0..49]of char;
  p:pointer;
begin
  //copy pascal string to null-terminated string
  strPCopy(idString,intToStr(DeviceID));

  //cmdLine="ScdaqServer "
  strcpy(cmdLine,'ScdaqServer ');

  //cmdLine="ScdaqServer X", denga X adalah DeviceID

```

```

strcat(cmdLine,idString);

creationFlag:=HIGH_PRIORITY_CLASS;
SI.wShowWindow:=SW_MAXIMIZE;
SI.dwFlags:=STARTF_USESHOWWINDOW;

//menjalankan aplikasi server

if(CreateProcess(0,cmdLine,0,0,false,creationFlag,0,0,SI,PI)=false)
then
begin
  MessageBox(0,'Cannot run server',0,MB_OK);
  Open:=false;
end;

WaitForInputIdle(PI.hProcess,10000);
GetExitCodeProcess(PI.hProcess,exitCode);
if exitCode<>STILL_ACTIVE then
begin
  if exitCode=1 then
  begin
    MessageBox(0,'Cannot Start Server',0,MB_OK);
    Open:=false;
  end;
end;

//membuka mutex untuk sinkronisasi
strcpy(mutexName,'ScdaqServer'); //mutexName=ScdaqServer
strcat(mutexName,idString); //mutexName=ScdaqServerX, X=id
strcat(mutexName,'Mutex'); //mutexName=ScdaqServerXMutex
hMutex:=OpenMutex(MUTEX_ALL_ACCESS,false,mutexName);
if(hMutex=0) then
begin
  MessageBox(0,'Cannot open synchronization
object','ERROR',MB_OK);
  showError(GetLastError());
  Open:= false;
end;

//membuka fileMapping untuk shared memory
strcpy(fileMapName,'ScdaqServer'); //mutexName=ScdaqServer
strcat(fileMapName,idString); //mutexName=ScdaqServerX, X=id
strcat(fileMapName,'FileMap');
//mutexName=ScdaqServerXFileMap
hMapFile:=OpenFileMapping(FILE_MAP_READ ,false,fileMapName);

if hMapFile=0 then
begin
  MessageBox(0,'Cannot open shared memory
object','ERROR',MB_OK);
  showError(GetLastError());
  Open:= false;
end;

//Membuat fileView untuk mendapatkan alamat shared memory

```

```

p:=MapViewOfFile(hMapFile, // Handle to mapping object.
FILE_MAP_READ,           // Read/write permission
0,                       // Max. object size.
0,                       // Size of hFile.
0);                      // Map entire file.

if p=nil then
begin
  MessageBox(0,'Cannot map shared memory',0,MB_OK);
  showError(GetLastError());
  Open:= false;
end;

pByte:=p;
opened:=true;
Open:=true;
end;

procedure TScdaqInterface.showError(errorCode:cardinal);
var
  lpMsgBuff:array[0..256]of char;
begin
  FormatMessage(
    FORMAT_MESSAGE_FROM_SYSTEM,
    nil,
    errorCode,
    LANG_ENGLISH, // Default language
    lpMsgBuff,
    0,
    nil);
  // Display the string.
  MessageBox( 0,lpMsgBuff, 'GetLastError', MB_OK);
end;

//-----
constructor TScdaqInterface.TScdaqInterface();
begin
  opened:=false;
end;

//-----
function TScdaqInterface.getData(channelNumber:integer):double;
var
  temp:double;
  pData:^double;
begin
  if(opened=false) then
  begin
    getData:= -3;
  end;
  WaitForSingleObject(hMutex, INFINITE);
  pData:=pointer(PChar(pByte)+(channelNumber*8));
  temp:=pData^;
  ReleaseMutex(hMutex);
  getData:= temp;
end;

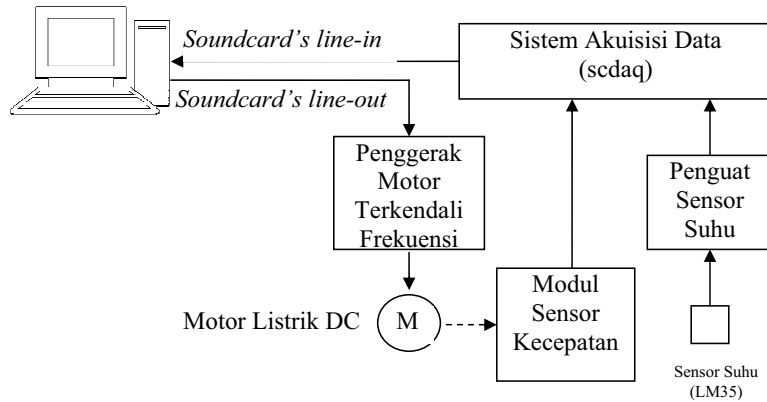
```

```
//-----  
function TSdaqInterface.getSlowData(channelNumber:integer):double;  
var  
    temp:double;  
    pData:^double;  
begin  
    if(opened=false) then  
        begin  
            getSlowData:= -3;  
        end;  
        WaitForSingleObject(hMutex,INFINITE);  
        pData:=pointer(PChar(pByte)+(channelNumber*8)+128);  
        temp:=pData^;  
        ReleaseMutex(hMutex);  
        getSlowData:= temp;  
    end;  
//-----  
  
end.
```

Lampiran 13

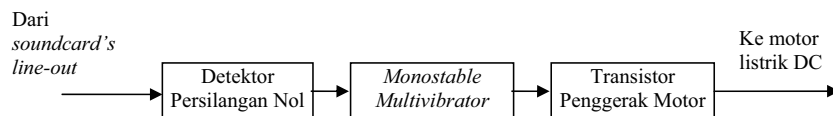
Rancangan Perangkat Keras Pengendali Motor DC dan Pemantau Suhu

Diagram blok perangkat keras sistem pengendali motor DC dan pemantau suhu disusun sebagai berikut:

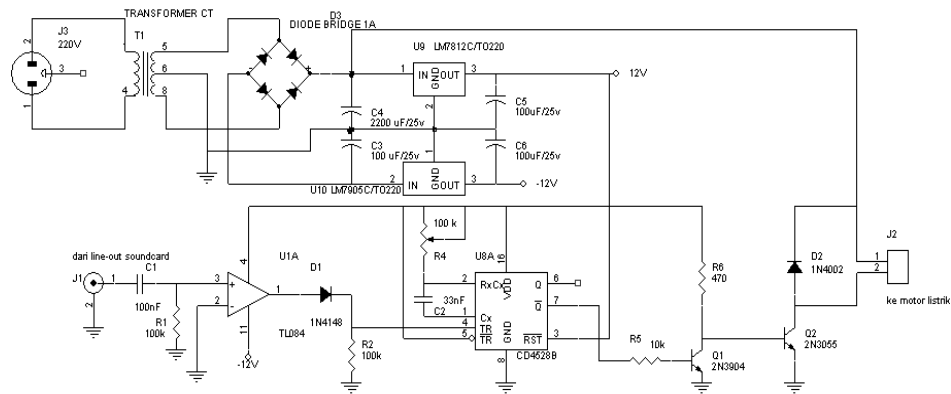


Sistem akuisisi data terhubung dengan komputer melalui masukan *line-in* dari *soundcard*. Sistem akuisisi data digunakan untuk membaca sinyal kecepatan motor DC dan sinyal sensor suhu.

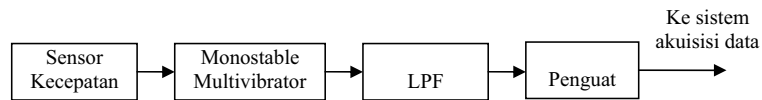
Penggerak motor DC digunakan untuk memberikan daya listrik kepada Motor DC, yang berupa pulsa-pulsa dengan lebar tetap tetapi frekuensinya dapat diatur melalui keluaran *line-out* dari *soundcard*. Berikut ini diagram blok penggerak motor DC:



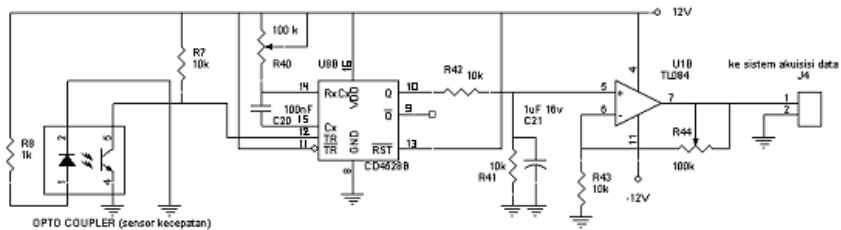
Berikut ini skematik rangkaian elektronik penggerak motor DC:



Modul sensor kecepatan digunakan untuk mengindera kecepatan motor DC. Modul ini terdiri dari sensor kecepatan dan pengkondisi sinyal. Sensor terdiri dari cakram pembangkit pulsa (yang dipasang pada as motor) dan *optocoupler*. Cakram pembangkit pulsa mempunyai celah-celah tembus pandang untuk membuka dan menutup celah *optocoupler*. Pengkondisi sinyal terdiri dari *monostable multivibrator*, *low-pass filter* (LPF), dan penguat. Berikut ini diagram blok modul sensor kecepatan:



Berikut ini skematik rangkaian elektronik modul sensor kecepatan:



Penguat sensor suhu digunakan untuk memperkuat sinyal dari sensor suhu sehingga rentangnya sesuai dengan rentang masukan sistem akuisisi data. Sensor suhu yang dipakai adalah LM35 yang mempunyai keluaran 0 volt pada 0°C dan 1 volt pada 100°C. Penguat diperlukan agar keluaran 1 volt dari LM35 menjadi 10 volt ketika memasuki sistem akuisisi data, sehingga penguatannya diatur ke 10 kali. Berikut ini skematik rangkaian elektronik penguat sensor suhu:

