UNIVERSIDADE REGIONAL DE BLUMENAU

CENTRO DE CIÊNCIAS EXATAS E NATURAIS

CURSO DE CIÊNCIAS DA COMPUTAÇÃO – BACHARELADO

PROTÓTIPO DE UM HARDWARE PERIFÉRICO PARA MIXAGEM DE MÚSICAS MP3 UTILIZANDO A PORTA PARALELA DE UM PC PADRÃO IBM

ERNANI LOPES ISENSEE

BLUMENAU

2003

2003/2-11

ERNANI LOPES ISENSEE

PROTÓTIPO DE UM HARDWARE PERIFÉRICO PARA MIXAGEM DE MÚSICAS MP3 UTILIZANDO A PORTA PARALELA DE UM PC PADRÃO IBM

Trabalho de Conclusão de Curso submetido à Universidade Regional de Blumenau para a obtenção dos créditos na disciplina Trabalho de Conclusão de Curso II do curso de Ciência da Computação — Bacharelado.

Prof. Miguel Alexandre Wisintainer

BLUMENAU
2003

2003/2-11

PROTÓTIPO DE UM HARDWARE PERIFÉRICO PARA MIXAGEM DE MÚSICAS MP3 UTILIZANDO A PORTA PARALELA DE UM PC PADRÃO IBM

Por

ERNANI LOPES ISENSEE

Trabalho aprovado para obtenção dos créditos na disciplina de Trabalho de Conclusão de Curso II, pela banca examinadora formada por:

Presidente:	Professor Miguel Alexandre Wisintainer – Orientador, FURB
Membro:	Professor Antônio Carlos Tavares, FURB
Membro:	Professor Paulo César Rodacki Gomes, FURB

Blumenau, 04 de dezembro de 2003

Dedico este trabalho a todos os amigos e familiares, especialmente aqueles que me ajudaram diretamente na realização deste.

AGRADECIMENTOS

Muitas pessoas contribuíram direta e indiretamente para a elaboração deste trabalho às quais gostaria de agradecer em especial.

Em primeiro lugar aos meus pais, Deodato e Nazir, que sempre me apoiaram em todos os sentidos nas conquistas e nas fases mais importantes da minha vida, me dando muito incentivo, amor e carinho.

Especialmente ao meu pai por inúmeras vezes me auxiliar tecnicamente, no que diz respeito a conceitos na área da eletrônica básica, devido ao conhecimento e experiência que ele têm nesta área.

Ao professor Miguel, meu orientador e que, durante este período criamos laços de amizade, sendo que ele sempre me apoiou e demonstrou disposição a esclarecer minhas dúvidas.

À minha namorada Cristiane pela compreensão e pelo incentivo no desenvolvimento desta pesquisa.

RESUMO

Este trabalho demonstra a construção de um protótipo de *hardware* e *software*, os quais têm por objetivo manipular um *software* simulador de mixagens denominado Virtual Turntables. Para isto, é realizado um estudo sobre o funcionamento da porta paralela utilizada no PC padrão IBM, a utilização da API (*Application Protocol Interface*) do Windows, utilizando o envio de mensagens para o *software* simulador, bem como o estudo de alguns circuitos integrados.

Palavras chaves: Protótipo, Entradas Analógicas e Digitais, Aquisição do *handle* de Objetos, Mixagem, Simulador.

ABSTRACT

This work demonstrates the construction of a hardware and software prototype, which have for objective to manipulate a software simulator of mixer denominated Virtual Turntables. For this, a study is accomplished on the operation of the parallel port used in the standard PC IBM, the use of API (Application Protocol Interface) of Windows, using the shipping of messages for the software simulator, as well as the study of some integrated circuits.

Key words: Prototype, Analog and Digital Inputs, Captures the handle of Objects, Mixing, Simulator.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE SIGLAS

A/D – Analógico/Digital

ADC – Analogic/Digital Converter

API – *Application Protocol Interface*

BPM – Batidas por Minuto

CD – *Compact Disc*

CI – Circuito Integrado

CMOS – *Complementary Metal Oxide Semiconductor*

CPU – *Central Process Unit*

DJ – *Disc Jokey*

Hz – Hertz

Khz – KiloHertz

MP3 – *Mpeg audio layer 3*

PC – *Personal Computer*

PCB – *Printed Circuit Board*

TTL – *Transistor Transistor Logic*

USB – *Universal Serial Bus*

V – Volts

SUMÁRIO

1 INTRODUÇÃO

Nos últimos anos surgiram diversos formatos de compressão de áudio digital, os quais foram impulsionados pela Internet e as gravadoras de CD que equipam muitos dos microcomputadores hoje em dia, sendo o MP3 o formato mais difundido, o qual foi desenvolvido pela Fraunhofen Studio situada na Alemanha. A qualidade do áudio obtido em um arquivo utilizando o formato MP3 é similar à qualidade do áudio de um CD, porém com um nível de compactação de 10 a 12 vezes maior.

A princípio, os arquivos de áudio utilizando a compressão MP3 estavam restritos a serem reproduzidos em microcomputadores, porém grandes empresas como a Sony, Creative Labs, Phillips entre outras, investiram, e até então investem muito em equipamentos dotados de microcontroladores capazes de compreender o algoritmo de compressão MP3. Muitos fabricantes fornecem uma gama diversificada destes produtos, sendo eles portáteis, semelhantes a *walkmans* e *diskmans*, DVD *Players* e MP3 *Players* automotivos.

Após invadir os microcomputadores, DVD *Players* e automóveis, o MP3 começa a ganhar espaço em uma área no qual o CD atualmente domina, que são as *pick-ups* dos DJ's. A grande vantagem da utilização da música digital no formato MP3 é o grande nível de compactação, o que reduz muito a quantidade de CD's que o DJ precisaria manusear em sua discoteca.

A Visiosonic Ltda é pioneira neste tipo de equipamento, a qual oferece produtos que integram hardware e software, que simulam os equipamentos tradicionais de mixagem utilizando CD's, porém neste caso são utilizadas músicas no formato MP3. Neste equipamento, a interface utilizada pelo *hardware* é USB. Por ser um equipamento importado e que necessita de um microcomputador dotado de conexão USB, o custo para a aquisição é muito elevado.

A Carrot Innovations comercializa o Virtual Turntables, que é um *software* simulador para mixagens e equalização com um custo mais acessível. A grande diferença é que o Virtual Turntables não possui um *hardware* específico para interface com o usuário, o que dificulta a agilidade em sua utilização, sendo a mesma fundamental ao fazer mixagens em tempo real.

Este trabalho implementa um protótipo de uma mesa de mixagem, utilizando a porta paralela como interface com o microcomputador. Por meio desta mesa o usuário poderá comandar as funções principais do Virtual Turntables através de botões (*hardware*), com isto a interação entre o usuário e o simulador que era através do *mouse* e do teclado do microcomputador, passará a ser realizada através do protótipo.

1.1 ORIGEM DO TRABALHO

Este trabalho teve início na disciplina de Projeto de Pesquisa, onde foi elaborada a proposta para o trabalho de conclusão de curso. Como o *software* simulador Virtual Turntables é muito utilizado por DJ's com pouca experiência, os quais utilizam um microcomputador para fazer mixagens, foi constatado por estes DJ's, uma certa dificuldade ao manusear o mesmo, devido a sua interface com o usuário limitada ao teclado e ao *mouse* do microcomputador. Devido a isto, foi iniciada uma pesquisa de produtos que fazem a integração entre *hardware* e *software*, com a mesma finalidade do *Virtual Turntables*, que é fazer mixagem de músicas no formato MP3 em tempo real. Durante esta pesquisa, foi localizado um produto que faz a integração entre *hardware* e *software* e que muito se assemelha aos equipamentos tradicionais dos DJ's, denominado PCDJ, o qual é desenvolvido e comercializado pela Visiosonic Ltda.

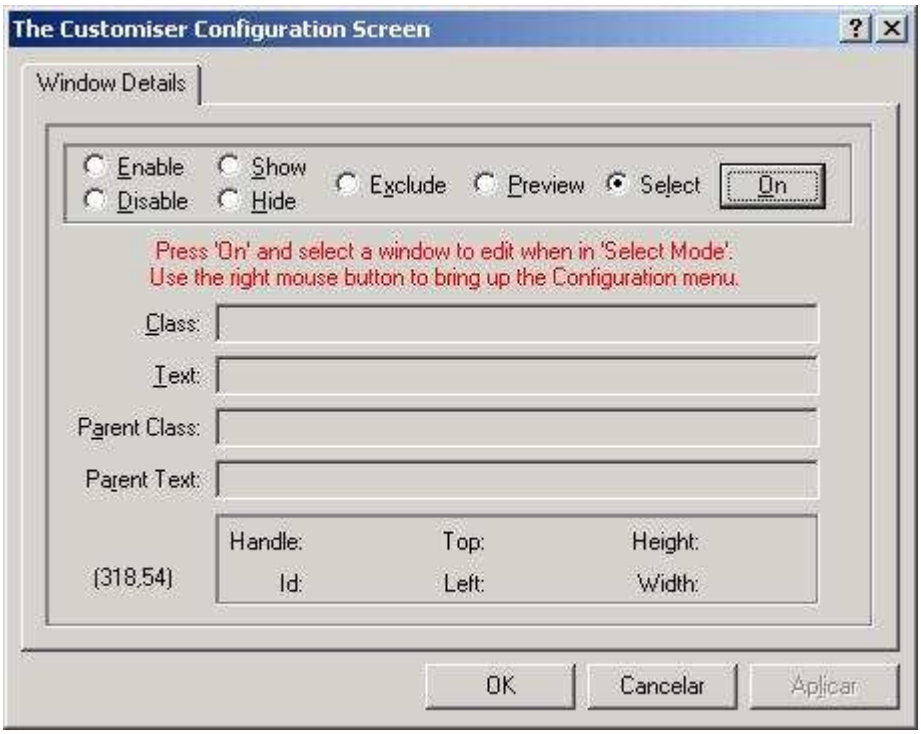
O PCDJ pode ser utilizado da mesma forma que o Virtual Turntables, tendo como interface com o usuário o *mouse* e o teclado do microcomputador, mas o grande diferencial é a utilização do *hardware* denominado PCDJ DAC-2 Controller que manipula as funções principais do PCDJ (figura 1).

Figura 1 – PCDJ DAC-2 Controller

Fonte: Visiosonic (2003)

Além da pesquisa de produtos disponíveis no mercado que façam a integração entre *hardware* e *software* para mixagem de músicas em tempo real, foi necessária a pesquisa da possibilidade de se comandar um software proprietário utilizando a API do Windows. Devido ao Virtual Turntables ser um software proprietário, não possuímos acesso ao seu código fonte. Durante esta pesquisa, foi localizado o software denominado The Customiser (figura 2) desenvolvido pela Wanga International situada na Austrália, o qual possibilita que o usuário faça a aquisição do *handle* de um objeto graficamente e, após a aquisição do *handle*, é possível mandar mensagens para o objeto que foi capturado. Este *software* facilitou o estudo da API do Windows e mostrou ser possível controlar um software proprietário utilizando a API.

Figura 2 – The Customiser



A integração entre *hardware* e *software* desenvolvida pela Visiosonic Ltda., foi a grande fonte de inspiração para o desenvolvimento deste trabalho. A incerteza que havia referente à possibilidade de controlar o Virtual Turntables a partir de outra aplicação, foi sanada após testes efetuados com o The Costumiser.

1.2 ÁREA

Este protótipo abrange a área da arquitetura de computadores, da eletrônica básica e digital para a construção do protótipo, bem como a utilização da porta paralela de um PC padrão IBM para o envio e recebimento de dados entre o *software* e o *hardware* e a utilização da API do Windows para que o software *host* desenvolvido se comunique com o Virtual Turntables.

1.3 PROBLEMA

Através de experiências pessoais como DJ e interagindo com profissionais deste ramo do entretenimento, em um primeiro momento estes profissionais rejeitavam a substituição dos equipamentos de mixagem tradicionais pelo uso de um microcomputador com *softwares* que fizessem um trabalho equivalente, alegando uma baixa confiabilidade, devido ao micro poder "travar" durante sua utilização. Mesmo após demonstrar a estes profissionais que o uso de um microcomputador com o uso de um *software* simulador de mixagens é confiável, e que não há o risco de "travar" como alegado anteriormente, estes profissionais não se sentiam bem e não conseguiam ter a mesma agilidade e precisão durante uma mixagem utilizando o *mouse* e o teclado do microcomputador.

É mais fácil efetuar a transição entre uma música e outra (mixagem), alterar o *pitch* da música (velocidade), controlar a intensidade das freqüências graves, médias e agudas (equalizador), através de um equipamento específico para esta finalidade (mesa de mixagem), do que usar o teclado e o *mouse* do microcomputador. Embora o Virtual Turntables possua suas funcionalidades básicas com teclas de atalho, para que o usuário possa usar o teclado do microcomputador de forma semelhante à mesa de mixagem, o teclado do microcomputador não possui um *layout* amigável para isto e não é tão preciso quanto uma mesa de mixagem real.

Com isto, o presente trabalho propõe a melhoria do aplicativo Virtual Turntables, fazendo com que profissionais do ramo de entretenimento tenham a opção de utilizar um PC com a mesma agilidade e precisão que teriam ao utilizar equipamentos tradicionais de mixagem.

1.4 JUSTIFICATIVA

Com o objetivo de proporcionar ao usuário do Virtual Turntables um controle mais preciso e semelhante ao de uma mesa de mixagem tradicional, o presente trabalho propõe uma solução ao problema acima citado, utilizando-se tecnologias já disponíveis.

Além disso, deseja-se demonstrar a utilização de mensagens da API do Windows para controlar outras aplicações, e os fundamentos da tradicional comunicação paralela, a qual se demonstra eficaz para atender o problema acima citado.

1.5 OBJETIVOS

Esta proposta de trabalho tem como objetivo a especificação e implementação de um periférico (*hardware*), baseado em circuitos integrados que conectados a porta paralela de um PC padrão IBM, que funcionará como interface com o Virtual Turntables.

Os objetivos específicos do trabalho são:

a) projeto e construção de um *hardware* baseado em Circuitos Integrados TTL (*transistor transistor logic*) e CMOS (*complementary metal oxide semiconductor*) para aquisição de dados analógicos gerados pelos *knobs* (potenciômetros), bem como os sinais digitais gerados por chaves;
b) elaborar um *software* utilizando o Delphi 5 (Borland), que controle o *hardware* proposto e manipule o Virtual Turntables através da API do Windows.

1.6 ORGANIZAÇÃO DO TRABALHO

Este trabalho apresenta o estudo da tecnologia de transmissão e recepção de dados através da porta paralela de um PC padrão IBM, a utilização da API do Windows para controlar um *software* de terceiro e o estudo de alguns componentes TTL e CMOS, tendo como resultado o desenvolvimento de uma mesa de mixagem com um software já existente (Virtual Turntables).

O trabalho está organizado em 8 capítulos, conforme descrito abaixo:

a) no capítulo 1 é feita a introdução ao projeto;
b) no capítulo 2 é demonstrado a utilização do software de mixagem Virtual Turntables sem a utilização do *hardware* periférico;
c) o capítulo 3 aborda conceitos básicos referentes à tecnologia TTL e características de alguns componentes que utilizam esta tecnologia, os quais serão utilizados no protótipo do *hardware*;
d) o capítulo 4 aborda conceitos básicos referentes à tecnologia CMOS e características de um componente que será utilizado no protótipo do *hardware* o qual utiliza esta respectiva tecnologia;
e) o capítulo 5 aborda conceitos básicos referentes à porta paralela de um PC padrão IBM;
f) o capítulo 6 aborda conceitos básicos referentes a API do Windows;
g) no capítulo 7 é demonstrada a especificação do protótipo;
h) no capítulo 8 são descritas as conclusões obtidas.

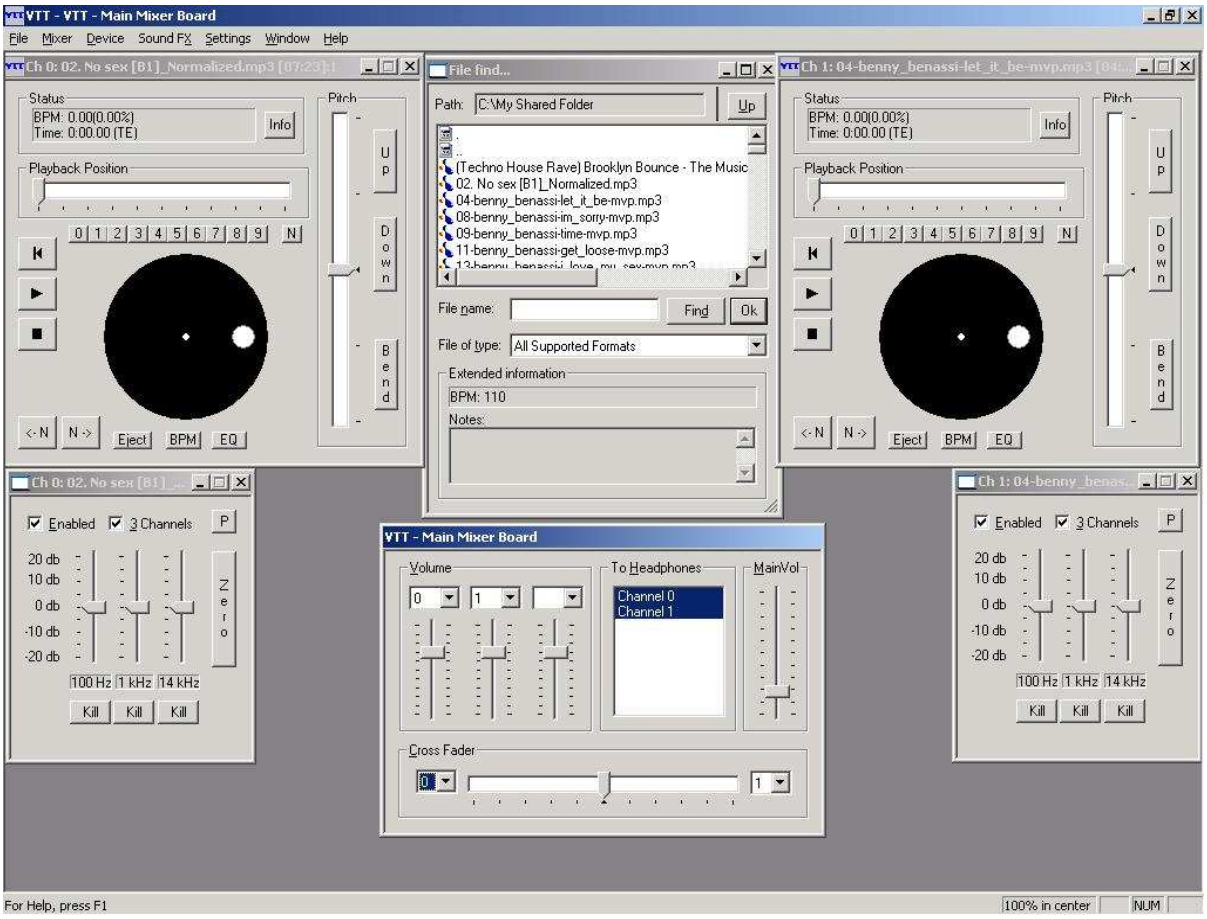
2 O SOFTWARE VIRTUAL TURNTABLES

Neste capítulo é descrita a utilização do Virtual Turntables da forma tradicional, através da utilização do teclado e do *mouse* do microcomputador. Este capítulo demonstra as funções principais do Virtual Turntables, sendo que, alguns detalhes deste simulador não são abordados.

2.1 A INTERFACE COM O USUÁRIO

O Virtual Turntables é um software simulador dos equipamentos tradicionais de mixagem. Ele é composto basicamente por dois *players*, um equalizador para cada *player*, um *mixer* e um mecanismo de busca de arquivos a serem executados em cada um dos *players* (figura 3).

Figura 3 – Virtual Turntables



Por ser um ambiente virtual para mixagens, o Virtual Turntables permite que o usuário execute simultaneamente 255 *players* com 255 equalizadores. Além disso, para cada equalizador o usuário pode optar por usá-lo no modo tradicional de três canais (figura 4), sendo um canal individual para as freqüências baixas (100Hz), médias (1 Khz) e altas (14Khz), ou utilizar um equalizador mais completo, no caso com dez canais (figura 5), sendo três canais individuais para as freqüências baixas (60Hz, 100Hz e 250Hz), quatro canais individuais para as freqüências médias (500Hz, 1Khz, 2Khz e 6,5Khz) e três canais para as freqüências altas (10Khz, 14Khz e 18Khz).

Figura 4 – Equalizador com três canais

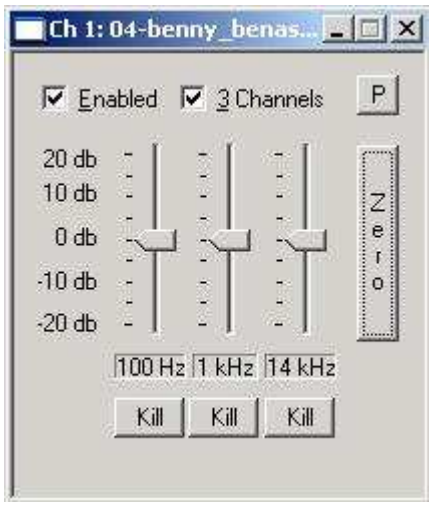
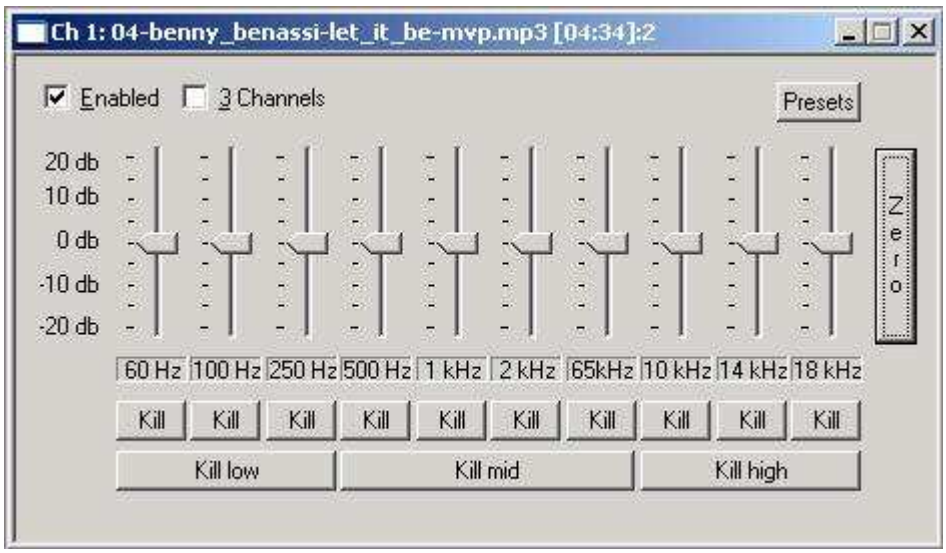


Figura 5 – Equalizador com 10 canais



Como este trabalho prevê a construção de um *hardware* periférico que irá controlar as funções básicas do Virtual Turntables, optou-se por utilizar o simulador Virtual Turntables da forma mais semelhante possível da realidade, que consistem em utilizar dois *players* com dois equalizadores de três canais e o *mixer* para controlar a intensidade do volume e efetuar a transição do volume entre um *player* e outro.

2.2 CONHECENDO AS FUNCIONALIDADES BÁSICAS DO SOFTWARE

A utilização do Virtual Turntables é simples e intuitiva, conforme exposto anteriormente ele é formado basicamente por quatro elementos, os quais são brevemente demonstrados a seguir.

2.2.1 O MECANISMO DE BUSCA DE ARQUIVOS (*FILE FIND*)

Este mecanismo de busca tem como objetivo facilitar a localização de arquivos compatíveis com o Virtual Turntables, dentre estes arquivos compatíveis o mais utilizado é o MP3. Com ele o usuário poderá navegar nas unidades presentes em seu microcomputador (C:\, D:\, etc), poderá fazer pesquisas informando parte do nome do arquivo no campo denominado "*File name*:", poderá informar o tipo do arquivo que se deseja localizar no campo denominado "*File of type*:" (Figura 6).

Figura 6 – Mecanismo de busca de arquivos

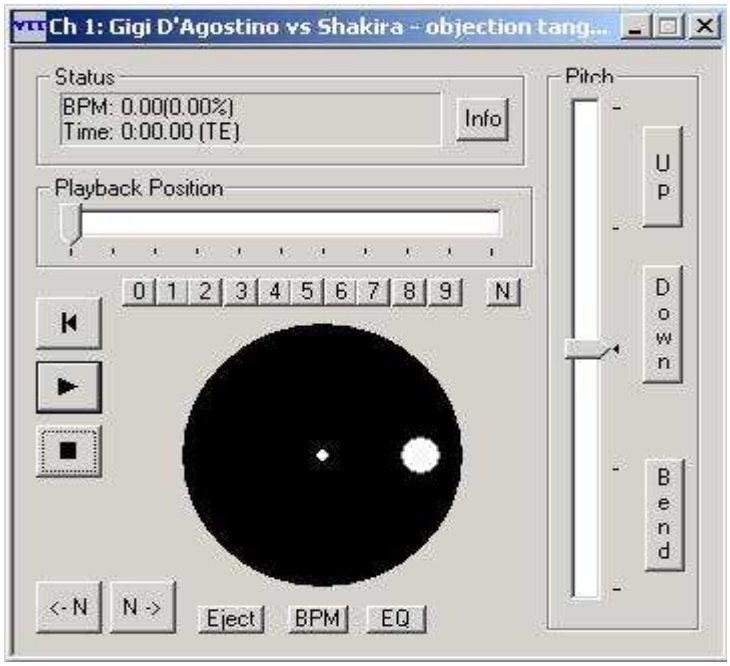
Após o usuário ter localizado o arquivo que ele quer tocar, para executar o mesmo, basta o usuário selecionar o arquivo em questão e arrastá-lo sobre o *player* desejado. Caso o *player* em que o usuário esta querendo executar o arquivo já esteja executando uma outra música, será questionado ao usuário se ele deseja parar a execução do arquivo atual para executar o novo arquivo, este é um detalhe interessante, pois se o usuário estivesse desatento e não fosse efetuado este questionamento, seria interrompida a execução da música bruscamente, que no caso de uma mixagem em tempo real estragaria o andamento da mixagem.

Conforme visto na figura 6, há uma área no mecanismo de busca denominada "*Extend information*", no qual é exibido o BPM da música e observações efetuadas pelo usuário referente à música selecionada. A inserção de informações adicionais em uma determinada música é feita através do *player* quando o arquivo estiver pronto para ser executado ou estando em execução. Estas informações são muito úteis a um DJ durante a mixagem, pois a partir delas o DJ poderá decidir qual a melhor música a ser executada após a música que já está em execução.

2.2.2 O *PLAYER*

O *player* é idêntico aos *CD-Players* utilizados por DJ's, que também se assemelha aos aparelhos de CD doméstico, porém com algumas funcionalidades a mais (Figura 7).

Figura 7 – *Player*

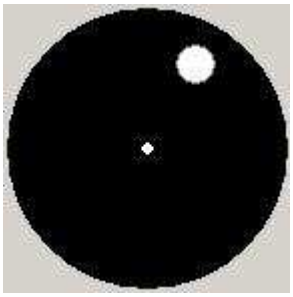


O botão denominado *Playback Position* (figura 8), tem a funcionalidade semelhante ao de uma barra de progressão, com o decorrer da execução da música este controle transitará de sua posição inicial (controle totalmente à esquerda) para a posição final (controle totalmente à direita). O usuário pode alterar a posição deste controle, com a finalidade de, por exemplo, iniciar a execução de uma música de um ponto determinado pelo usuário.

Figura 8 – Botão *Playback Position*

O botão *Jog Wheel* (figura 9), possui duas finalidades: enquanto o *player* não estiver executando a música, ao girar este botão no sentido horário o usuário adianta o ponto do início da música com precisão de um centésimo de segundo, ao girar este botão no sentido anti-horário o usuário irá atrasar o início da música também com a precisão de um centésimo de segundo. Caso a música já esteja em execução, ao girar este botão no sentido horário será aumentada a velocidade da música e ao girar o botão no sentido anti-horário será diminuída a velocidade da música, sendo que, quanto mais rápido o usuário girar este botão maior será o aumento ou diminuição na velocidade da música, e ao parar de girar o botão a música voltará à velocidade em que estava sendo executada anteriormente. Normalmente este botão é utilizado durante a execução de uma música, para corrigir uma pequena falta de sincronismo entre a música que esta em execução no *player* em questão e a música que está sendo executada no outro *player*.

Figura 9 – *Jog Wheel*



O botão *play* (figura 10), tem por objetivo iniciar a execução da música previamente atribuída ao *player*, ao iniciar a execução da música o botão *play* muda de ícone e de funcionalidade, este botão passa a exercer a função *pause* (figura 11). Quando a execução de uma música entra em pausa o *playback position* permanece parado no ponto em que a música entrou em pausa, caso seja clicado no botão *play/pause* novamente, a reprodução da faixa continuará do ponto em que entrou em pausa.

Figura 10 – Botão *Play*

Figura 11 – Botão *Pause*

O botão *stop* (figura 12), tem por objetivo interromper a execução da música, ao pressionar este botão o *playback position* voltará ao seu estado inicial.

Figura 12 – Botão *Stop*

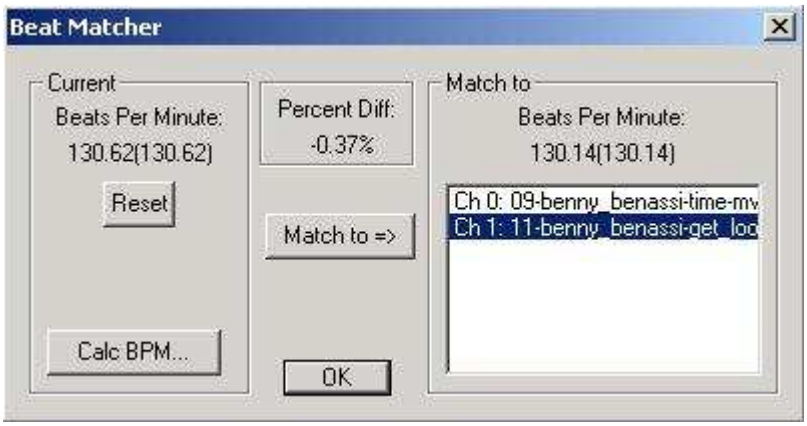
O botão *reload* (figura 13), tem por objetivo retornar a posição do *playback position* ao seu estado inicial, ao pressionar este botão durante a execução da música ela voltará a ser executada do seu início.

Figura 13– Botão *Reload*

O botão *eject* tem por objetivo abrir um arquivo sem a utilização do mecanismo de busca de arquivos (*File find*).

O botão *BPM*, disponibiliza ao usuário um mecanismo para se obter de forma manual o BPM da música que esta no *player*. Ao clicar no botão *BPM* é exibida uma janela denominada *Beat Matcher* (figura 13). Este é um recurso muito interessante, pois facilita muito a sincronização da velocidade de uma música com outra, afinal o BPM é o compasso da música.

Figura 14 – *Beat Matcher*

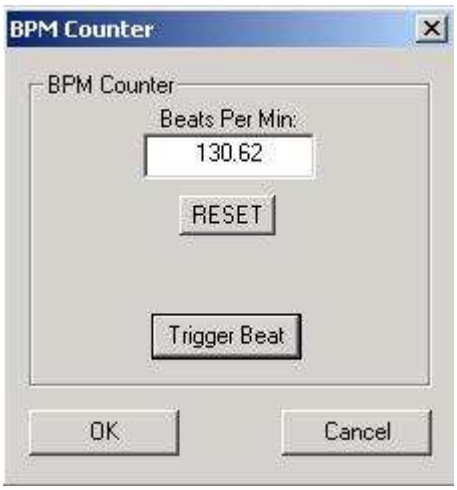


O botão *Reset* tem a finalidade de zerar o valor do BPM previamente calculado.

O botão *Match to* tem a finalidade de igualar o BPM da música do *player* em questão com a música que esta no outro *player*, para que este recurso funcione, é necessário ter calculado o BPM dos arquivos que estão nos dois *players.* Uma vez o usuário tendo calculado o BPM de uma música, a próxima vez que o usuário for tocar esta mesma música, o usuário não precisará recalcular o BPM da música, pois o Virtual Turntables armazena esta e outras informações em arquivos em uma subpasta da pasta onde o Virtual Turntables está instalado.

Para calcular o BPM de uma música, há o botão denominado *Calc BPM*. Ao clicar neste botão, será exibida uma janela denominada *BPM Counter* (figura 15).

Figura 15 – *BPM Counter*

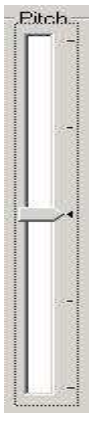


Para que o usuário possa calcular o BPM de uma música, é necessário que a mesma esteja em execução no canal principal ou no fone de ouvido, pois o usuário precisa escutar a música e acompanhar o compasso da mesma clicando no botão *Trigger Beat*, ou simplesmente pressionando a barra de espaço do teclado. Ao acompanhar o compasso da música, em aproximadamente vinte ou trinta segundos, o campo *Beats per Min* já conterá um valor que não terá uma grande oscilação, quando esta situação ocorrer, quer dizer que o valor exibido neste campo esta bem próximo do BPM exato da música.

O botão *EQ* tem a finalidade de exibir o equalizador do *player*.

O botão deslizante denominado *Pitch* (figura 16), possibilita ao usuário controlar a velocidade da música. Ao deslizar este botão para cima ocorrerá um aumento na velocidade da música e ao deslizar para baixo, a velocidade da música será diminuída.

Figura 16 – *Pitch*



O botão *Bend* ao permanecer pressionado, faz com que a posição em que o *pitch* está fique “travado”, se o usuário aumentar ou diminuir a velocidade da música através do controle deslizante do *pitch* ou clicando nos botões *up* e *down*, ocorrerá a alteração na velocidade, mas ao soltar o botão do *pitch*, *up* ou *down* a velocidade voltará a ser a mesma quando o botão *bend* foi acionado.

O botão *Up* e *Down* tem a finalidade de aumentar e diminuir a velocidade da música, estes botões normalmente são usados quando o usuário deseja um ajuste preciso na velocidade da música.

No quadro denominado *Status* (figura 17), é demonstrado o BPM da música em execução, o percentual acrescido ou diminuído no BPM da música e o tempo da música, sendo que, o tempo demonstrado pode ser o decorrido (TE – *time elapsed*) ou o tempo restante para o término do arquivo (TR – *time remaining*), para alternar entre um cronômetro e outro, basta clicar no quadro onde é exibido o *status*. O botão *Info* ao ser pressionado abre uma janela em que o usuário pode inserir um comentário sobre a música, sendo este comentário exibido ao selecionar a música no *File find.*

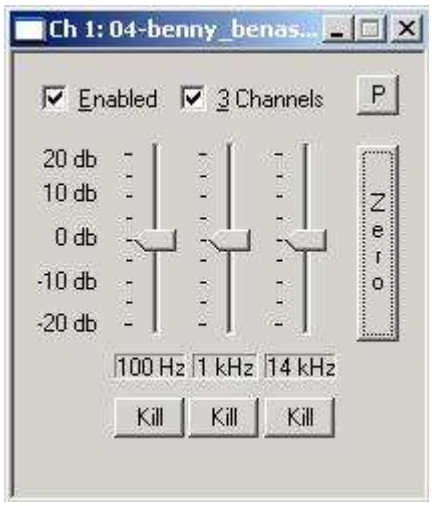
Figura 17 – *Status*

2.2.3 O EQUALIZADOR

Como o *hardware* periférico proposto será construído para a utilização do equalizador com três faixas de freqüências (figura 18), não será demonstrado o funcionamento do equalizador com dez faixas de freqüência.

O equalizador tem a finalidade de fornecer ao usuário a possibilidade de ajuste dos sons graves, médios e agudos, de acordo com o gosto e necessidade do usuário. Muitas vezes quando uma música possui o volume muito alto, ou durante a transição entre uma música e outra, pode ser notado distorções no som, o equalizador é utilizado para evitar estas distorções. Normalmente são as baixas freqüências que tendem a distorcer.

Figura 18 – Equalizador



Cada uma das freqüências, sendo as baixas (100Hz), médias (1Khz) e altas (14Khz), possuem um controle de volume individual para realçar ou reduzir cada uma das freqüências e um botão denominado *Kill*, o qual é utilizado para eliminar a reprodução da freqüência em questão.

O botão *Zero* faz com que os controles de volume de todas as freqüências voltem a posição 0db, na qual a equalização é neutra. O item "*Enabled*" faz com que o equalizador seja habilitado ou desabilitado. O item "*3 Channels*" faz com que o equalizador seja exibido na forma tradicional com 3 faixas de freqüência ou no modo avançado com 10 faixas de freqüência.

O botão "*P*" (*preset*), disponibiliza ao usuário a possibilidade de salvar a configuração da equalização atual, ou ainda abrir uma equalização salva anteriormente.

2.2.4 O *MIXER*

Um detalhe importante a ser observado é o fato do Virtual Turntables poder gerenciar mais de uma placa de som. No caso do microcomputador em que o Virtual Turntables está instalado possuir duas placas de som, é possível configurar o simulador para utilizar uma das placas como saída principal para um amplificador ou caixas de som e a outra placa será utilizada como retorno para o usuário através de um fone de ouvido. Utilizando o Virtual Turntables configurado desta forma, o simulador fica com sua funcionalidade bem próxima dos equipamentos tradicionais de mixagem, por isso vamos analisar o funcionamento do

mixer considerando que o microcomputador em que o micro está instalado possui duas placas de som.

É no mixer (figura 19), onde realmente é executada a mixagem das músicas. O mixer possui um volume individual para cada *player* e um controle de volume geral denominado *MainVol*.

Figura 19 – *Mixer*

Para identificar o volume de cada um dos *players* há o número que identifica o *player* acima do controle de volume.

O *MainVol* como já foi mencionado, é o volume geral, ao aumentar ou diminuir este volume o mesmo controlará a intensidade do volume da placa de som que está definida como saída principal, o volume da placa de som definida como retorno não será alterado.

A principal função do *mixer* é o *CrossFader*, é a partir deste controle que é feita a mixagem da música. Este controle funciona como balanço entre o *player0* e o *player1*, quando o controle esta totalmente à esquerda, o *player0* estará com seu volume integral na placa de som definida como saída principal e o *player1* estará com o volume "zerado", ao começar a deslizar este controle para a direita o volume do *player1* irá aumentando gradativamente enquanto o volume do *player0* permanecerá o mesmo, quando o botão deslizante chegar a posição central tanto o *player0* quanto o *player1* estarão com a mesma intensidade de volume na placa de som definida como saída principal. O inverso ocorrerá se o usuário posicionar o controle deslizante totalmente à direita. Este controle não interfere no volume da placa de som definida como retorno.

No quadro denominado *To Headphones*, o usuário poderá selecionar qual dos *players* terá o volume liberado para o fone de ouvido. Este recurso é muito importante para que o usuário selecione e prepare a próxima música a ser executada, pois ele pode definir de forma que a música em execução em um determinado *player* possa ser ouvida somente através do fone de ouvido.

3 FAMÍLIA DE CIRCUITOS INTEGRADOS TTL

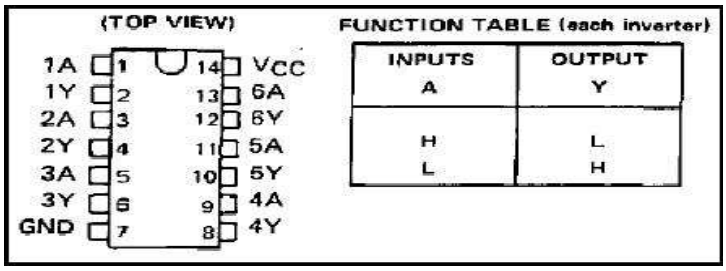
Durante este capítulo, é abordado o surgimento desta tecnologia e também é feito uma breve explanação de alguns componentes TTL, como exemplo.

Em 1964, a Texas Instruments Corporation introduziu a primeira linha de CI's TTL padrão. A série 54/74, como é chamada, tem sido uma das famílias lógicas de CI's mais amplamente utilizada, segundo Tocci et al (2003). Ao decorrer deste capítulo, esta série será referenciada como simplesmente série 74, afinal a principal diferença entre as versões 54 e 74 é que, os CI's da série 54 podem operar em faixas de temperatura e de tensão de alimentação maiores que a série 74, segundo os *datasheets* da Texas Instruments. Atualmente muitos fabricantes de semicondutores produzem CI's TTL. Felizmente, todos eles utilizam o mesmo sistema de numeração, com isto, o número básico do CI é o mesmo entre diversos fabricantes. Porém, cada fabricante normalmente acrescenta seu próprio prefixo especial ao número do CI. Por exemplo, a Texas Instruments utiliza o prefixo SN, a National Semiconductor utiliza DM, e a Signetics utiliza S. Desta forma, dependendo do fabricante, pode-se ver um chip de seis portas NOT identificado como DM7404, SN7404, S7404 ou alguma outra identificação similar. A parte principal da identificação do componente é o número 7404, que é o mesmo para a grande maioria dos fabricantes, segundo Tocci et al (2003).

3.1 O CIRCUITO INTEGRADO TTL 7404

Segundo o *datasheet* do SN74LS04 (figura 20) produzido pela Texas Instruments, este CI tem por objetivo inverter a tensão que lhe foi aplicada em sua entrada, caso uma de suas entradas esteja em nível lógico alto (+5V aproximadamente), a respectiva saída estará em nível lógico baixo (0V aproximadamente) e vice versa. Este CI é composto por seis inversores independentes.

Figura 20 – SN7404LS04

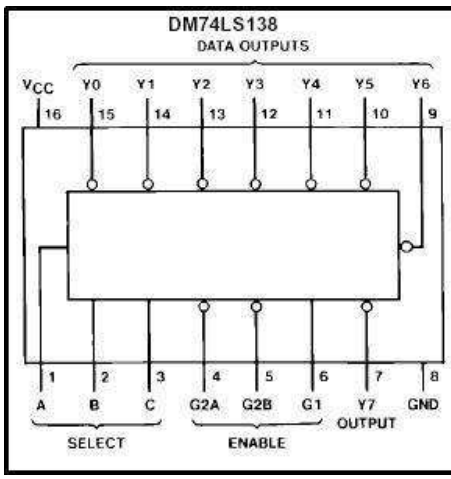


Fonte: SN74LS04 *Datasheet* (1996)

3.2 O CIRCUITO INTEGRADO TTL 74138

Segundo o *datasheet* do DM74LS138 (figura 21) produzido pela Fairchild Semiconductor, este CI é um decodificador/demultiplexador. Ele possui três entradas e oito saídas além de três pinos que habilita e desabilita o CI. O valor das três entradas equivale a um valor binário que, ao habilitar o CI, é selecionada a saída correspondente ao valor binário inserido na entrada. Um detalhe importante é que, quando uma saída é selecionada, a mesma fica em nível lógico baixo e as demais saídas em nível lógico alto.

Figura 21 – DM74LS138



DM74LS138

Inputs					Outputs							
Enable		Select										
G1	G2 (Note 1)	C	B	A	YO	Y1	Y2	Y3	Y4	Y5	Y6	Y7
X	H	X	X	X	H	H	H	H	H	H	H	H
L	X	X	X	X	H	H	H	H	H	H	H	H
H	L	L	L	L	L	H	H	H	H	H	H	H
H	L	L	L	H	H	L	H	H	H	H	H	H
H	L	L	H	L	H	H	L	H	H	H	H	H
H	L	L	H	H	H	H	H	L	H	H	H	H
H	L	H	L	L	H	H	H	H	L	H	H	H
H	L	H	L	H	H	H	H	H	H	L	H	H
H	L	H	H	L	H	H	H	H	H	H	L	H
H	L	H	H	H	H	H	H	H	H	H	H	L

H = HIGH Level
L = LOW Level
X = Don't Care

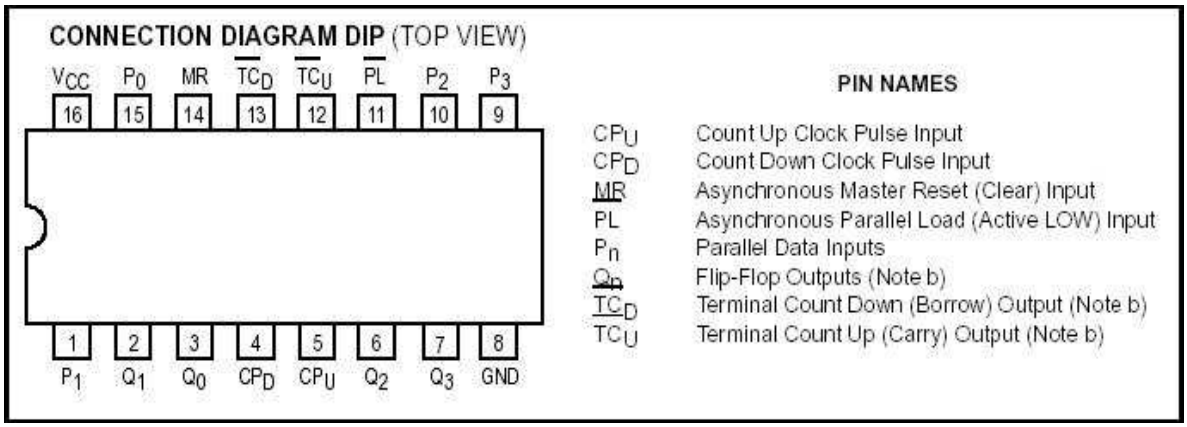
Note 1: G2 = G2A + G2B

Fonte: DM74LS138 *Datasheet* (2003)

3.3 O CIRCUITO INTEGRADO TTL 74192

Segundo o *datasheet* do SN74LS192 (figura 22) produzido pela Motorola, este CI é um contador crescente/decrescente de quatro *bits*, ele é composto por duas entradas de *clock*, um *reset*, quatro entradas paralelas, quatro saídas e um pino que habilita e desabilita as entradas paralelas.

Figura 22 – SN74LS192



Fonte: SN74LS192 *Datasheet* (2003)

As duas entradas de *clock* são responsáveis pela contagem, uma das entradas faz com que a contagem seja incrementada e a outra faz com que a contagem seja decrementada.

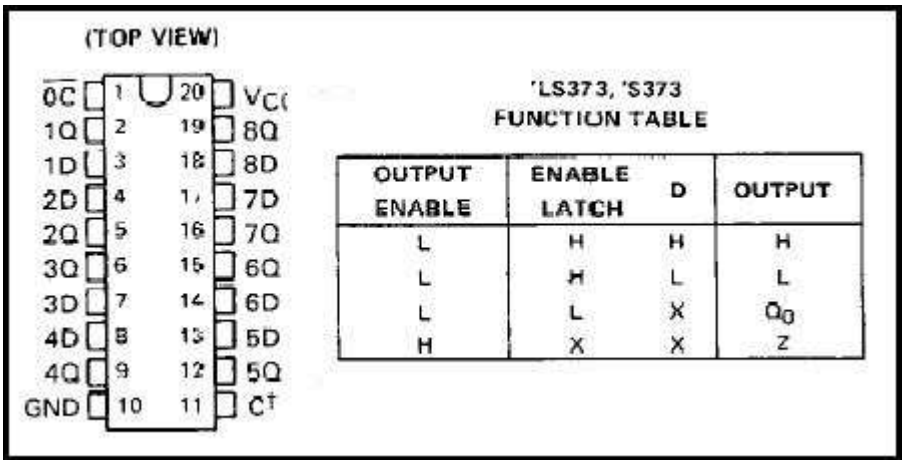
O valor binário de quatro *bits* inserido nas quatro entradas, será disponibilizado nas saídas quando o pino que habilita/desabilita estas entradas estiver em nível lógico baixo (0V aproximadamente). Quando a entrada *reset* estiver em nível lógico alto a contagem é "zerada".

Os demais pinos presentes neste circuito integrado não serão abordados, pois neste protótipo eles não serão utilizados.

3.4 O CIRCUITO INTEGRADO TTL 74373

Segundo o *datasheet* do SN74LS373 (figura 23) produzido pela Texas Instruments, este CI é composto por oito entradas, oito saídas e dois pinos responsáveis por habilitar/desabilitar as saídas do CI, estes pinos são chamados de *output enable* e *enable latch*. Quando o *output enable* estiver em nível lógico baixo e o *enable latch* estiver em nível lógico alto, o valor contido nas entradas é disponibilizado nas saídas. Quando o *output enable* esta em nível lógico alto as saídas permanecem em alta impedância.

Figura 23 – SN74LS373

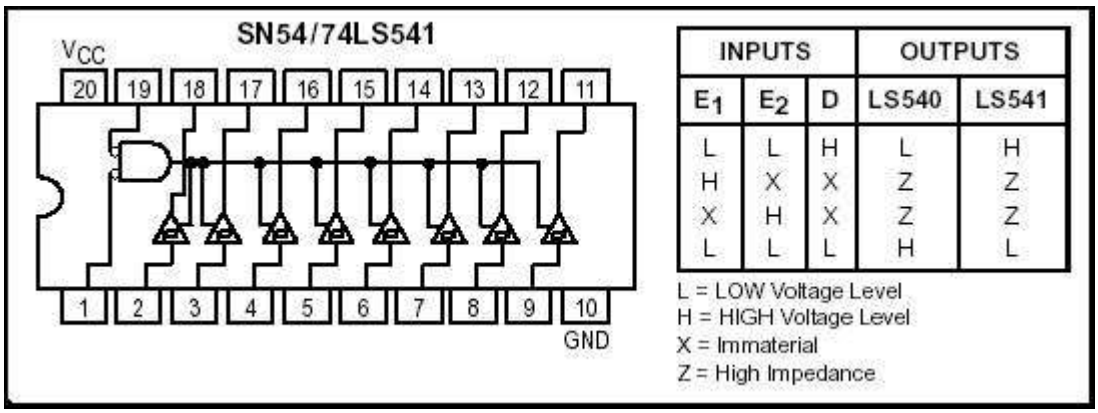


Fonte: SN74LS373 *Datasheet* (1996)

3.5 O CIRCUITO INTEGRADO 74541

Segundo o *datasheet* do SN74LS541 (figura 24) produzido pela Motorola, este CI funciona como um buffer octal. Ele é composto por oito entradas, oito saídas e dois pinos responsáveis por habilitar/desabilitar as saídas, denominadas E1 e E2. Quando E1 e E2 estão em nível lógico baixo o valor presente na entrada é disponibilizado na saída. No *hardware* proposto, este CI sempre se encontrará nesta situação (E1 e E2 com nível lógico baixo), por isso as demais combinações possíveis para estes dois pinos não serão abordadas.

Figura 24 – SN74LS541



Fonte: SN74LS541 *Datasheet* (2003)

4 A FAMÍLIA DE CIRCUITOS INTEGRADOS CMOS

Neste capítulo é realizada uma breve comparação entre circuitos integrados desenvolvidos utilizando a técnica TTL e CMOS, também será abordado o funcionamento básico do conversor analógico/digital ADC0808, o qual será utilizado no *hardware* proposto.

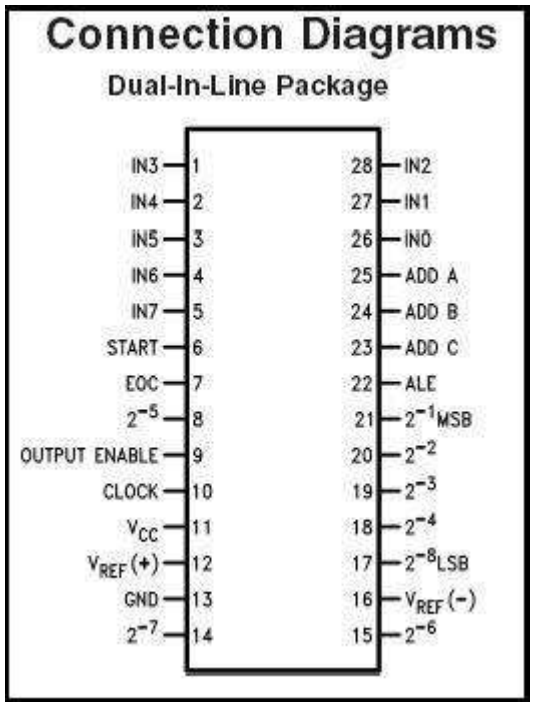
Segundo Tocci et al (2003), a família de circuitos integrados CMOS compete diretamente com o TTL nas áreas de integração de pequena e média escala. Como a tecnologia CMOS tem produzido uma *performance* cada vez maior, a tecnologia CMOS vem gradativamente ganhando sua fatia de mercado a qual tem sido dominada pela tecnologia TTL. Componentes TTL estarão presentes ainda por muito tempo, porém, equipamentos cada vez mais recentes estão utilizando circuitos lógicos predominantes CMOS.

4.1 O CONVERSOR ANALÓGICO/DIGITAL ADC0808

Os conversores A/D são disponibilizados por diversos fabricantes de CI's com uma ampla faixa de características de operação e vantagens (TOCCI, 2003, p. 536).

Segundo Bergsman (1994), o ADC0808 (figura 25) é um conversor analógico digital que contém oito linhas de entrada analógicas. Cada entrada de linha converte o sinal analógico recebido na respectiva entrada em um número binário entre 0 e 255. Para que esta conversão seja realizada é necessário informar duas tensões de referência, para isto há dois pinos chamados de VRef+ (voltagem de referência máxima) e VRef- (voltagem de referência mínima). No *hardware* proposto foi disponibilizado no pino Vref+, 5V, e ao pino Vref- foi disponibilizado 0V, diante desta situação se o valor analógico na entrada for 0V, nas saída do ADC0808 conterá 0 em binário, caso o valor analógico na entrada seja 5V, nas saídas do ADC0808 conterá 255 em binário. Ao utilizar o ADC0808 com esta configuração podemos afirmar que a precisão na entrada do AD é de 0,0196V (5V dividido por 255).

Figura 25 – ADC0808



Fonte: ADC0808 *Datasheet* (2003)

Para selecionar a entrada a ser lida, utilizam-se três linhas de endereçamento (semelhante ao funcionamento do 74138). Após selecionar a linha de entrada que se deseja converter, pode ser dado início a conversão, para iniciá-la é necessário disponibilizar ao pino denominado *start* e *ale* uma tensão de aproximadamente 5V, ou seja, nível lógico alto. Após o início da conversão, o que definirá a velocidade da conversão do valor analógico em um valor digital, é o *clock* fornecido, que no caso do ADC0808, o fabricante aconselha a utilização de um *clock* de 500Khz, segundo o *datasheet* do ADC0808 produzido pela National Semiconductor.

O pino chamado EOC (*end of conversion* – fim da conversão) mantém-se em nível lógico alto, mas ao término da conversão o nível lógico presente neste pino se torna baixo, até que seja inserido um sinal de nível lógico alto no pino OE (*output enable* – habilitar saída), então o nível lógico no pino EOC voltará a ser alto e será disponibilizado o valor obtido na conversão nas saídas do ADC0808. Enquanto não for enviado um sinal para o pino OE as saídas do ADC0808 permanecerão em alta impedância, segundo o *datasheet* do ADC0808 produzido pela National Semiconductor.

5 PORTA PARALELA

A porta paralela é uma interface de comunicação entre o computador e um periférico qualquer. Quando a IBM criou seu primeiro computador pessoal, a idéia era conectar a essa porta uma impressora, mas atualmente, são vários os dispositivos que se utilizam desta porta para enviar e receber dados para o computador, pode-se citar como exemplos de dispositivos que utilizam esta porta, scanners, câmeras de vídeo, unidades de disco removível entre outros (MESSIAS, 2003).

A grande vantagem na utilização da porta paralela é que ela transfere múltiplos *bits* de uma só vez, enquanto as portas seriais transferem um *bit* por vez, além disso, a porta paralela é encontrada em todo PC. Nos PC's mais novos são disponibilizadas outras portas como a SCSI, USB e IrDA, porém a porta paralela é mais popular e todo PC tem uma (AXELSON, 2000).

5.1 TIPOS DE PORTA

Com a evolução do PC, vários fabricantes introduziram versões melhoradas da porta paralela no PC. Os novos tipos de porta são compatíveis com a estrutura da porta original, estas novas portas ganharam novos recursos e principalmente, o aumento na velocidade de transmissão (AXELSON, 2000).

O primeiro tipo de porta paralela, que equipava os primeiros PC's foi denominado de SPP (*standard parallel port*), durante a evolução da porta paralela surgiram três versões aprimoradas da porta paralela que merecem destaque, sendo a PS/2 (bidirecional simples), EPP (*enhanced parallel port*) e ECP (*extended capabilities port*), segundo Axelson (2000).

Como o protótipo proposto precisará enviar sinais de controle através da porta paralela e receber dados gerados pelos conversores A/D's e por chaves, optou-se por utilizar a porta paralela no modo EPP.

.

5.1.1 O modo EPP

O EPP foi desenvolvido originalmente pelo fabricante de *chips* Intel, o fabricante de PC's Zenith, e Xircom que é um fabricante de produtos que utilizam a porta paralela. Como no modo PS/2, as linhas de dados são bidirecionais. No modo EPP a porta paralela pode ler ou escrever um *byte* em aproximadamente um microssegundo, o que o torna aproximadamente quatro vezes mais rápido que o modo SPP e PS/2. Além disso, no modo EPP, pode-se trocar a direção do dado rapidamente (de leitura para escrita e vice versa), com isto o modo EPP se torna muito eficiente quando utilizado para dispositivos que transferem dados em ambas as direções, como por exemplo: unidades de disco e de fita (AXELSON, 2000).

5.2 ENDEREÇAMENTO E REGISTRADORES

Segundo Axelson (2000), por padrão a porta paralela utiliza três endereços, no qual o primeiro endereço é o endereço base da porta, que é conhecido como registro de dados ou simplesmente o endereço da porta. O segundo endereço é o registro de *status* e o terceiro é o registro de controle (Tabela 1).

Tabela 1 – Endereçamento

Nome da Porta	**Registro de Dados / Endereço Base**	**Registro de Status**	**Registro de Controle**
LPT1	3BCh	3BDh	3BEh
LPT2	378h	379h	37Ah
LPT3	278h	279h	27Ah

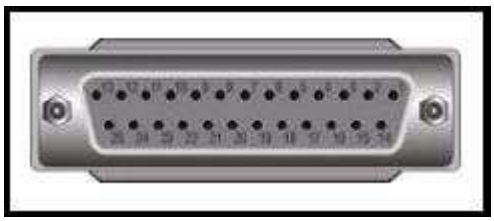
A porta paralela no modo EPP reserva endereços adicionais, sendo adicionado cinco registradores ao endereço base + 3 e endereço base + 7. Se o endereço base for 378h, os registradores serão 37Bh e 37Fh.

Nos primeiros PC's o endereço base da porta LPT1 era 3BCh, nos sistemas mais novos (atualmente), o endereço base da porta LPT1 é 378h, mas todos os três endereços que foram expostos continuam reservados para a porta paralela e se o *hardware* permitir, o usuário pode configurar uma porta paralela em qualquer um desses três endereços. Porém não é possível configurar uma porta paralela no modo EPP no endereço base 3BCh, pois os registradores somados a este endereço podem ser usados pela placa de vídeo.

5.3 A CONEXÃO DB25

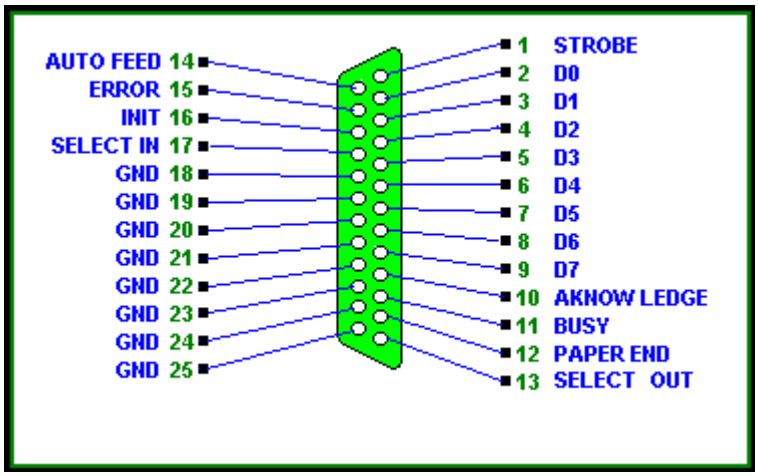
Segundo Messias (2003), o DB25 (figura 26) é um conector que está localizado na parte traseira do gabinete do microcomputador, e é através deste, que o cabo paralelo se conecta ao computador para viabilizar o envio e recebimento de dados entre o PC e um periférico qualquer. No DB25, um pino está em nível lógico baixo quando a tensão elétrica no mesmo está entre 0V e 0,4V, um pino se encontra em nível lógico alto quando a tensão elétrica no mesmo está acima de 3,1V e até 5V.

Figura 26 – Conector DB25



Cada pino do DB25 possui uma finalidade específica, na figura 27 será demonstrado o significado de cada um deles.

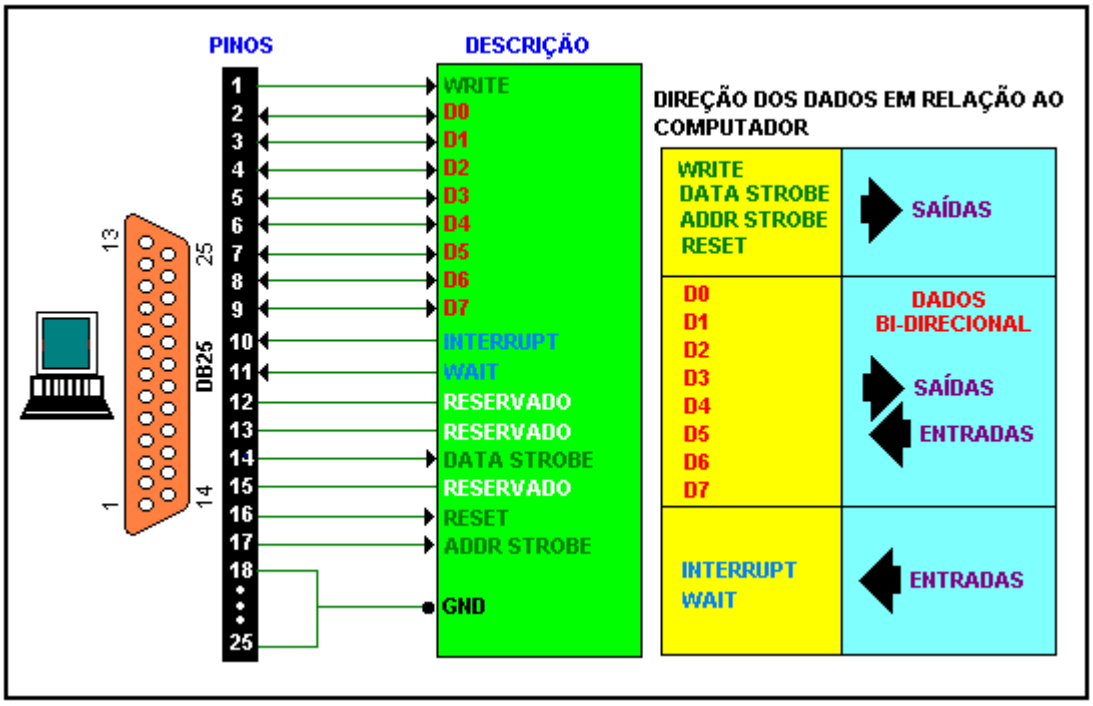
Figura 27 – Significado dos pinos do conector DB25



Fonte: Rogercom (2003)

Na figura 28, é demonstrado o sentido em que os dados podem trafegar nos pinos do DB25 quando a porta paralela se encontra configurada no modo EPP.

Figura 28 – Sentido do tráfego de dados no modo EPP



Fonte: Rogercom (2003)

No modo EPP, alguns pinos do conector DB25 receberam novos nomes, conforme relação da figura 29.

Figura 29 – Novos nomes para os pinos do DB25

Pinos	Descrição no Modo SPP	Descrição no Modo EPP
1	Strobe	Write
14	Auto Feed	Data Strobe
16	Init	Reset
17	Slct In	Address Strobe
10	Ack	Interrupt
11	Busy	Wait

Fonte: Rogercom (2003)

6 ENVIO DE MENSAGENS ENTRE APLICAÇÕES UTILIZANDO A API DO WINDOWS

A API Win32 permite que as aplicações troquem mensagens entre si e com a família de sistemas operacionais Microsoft Windows. As funções, estruturas, mensagens, macros e interfaces formam uma API uniforme e consistente para todas as plataformas de 32 *bits* da Microsoft. Utilizando a API Win32, o programador pode desenvolver aplicações que são executados com sucesso em todas as plataformas.

Diferenças na implementação dos elementos de programação podem oferecer incompatibilidade entre plataformas. A diferença mais notável é que algumas funções só podem ser executadas nas plataformas mais poderosas, por exemplo, funções de segurança só estão disponíveis no sistema operacional Windows com tecnologia NT. A maioria das outras diferenças estão em limitações do sistema, como restrições no alcance de valores (*range*) ou o número de itens que uma determinada função pode gerenciar.

6.1 O CONCEITO DE MENSAGENS DO WINDOWS

O Windows utiliza mensagens para gerenciar e sincronizar múltiplas aplicações. Eventos externos como o movimento do *mouse*, o pressionamento de uma tecla no teclado, podem ser convertidos em mensagens do Windows (Wilken, 1991).

Para o envio de mensagens para um determinado objeto, deve ser seguido o seguinte formato:

```
sendmessage(hWnd,Msg,wParam,lParam);
```

Sendmessage é uma função que possibilita o envio de uma mensagem a um determinado objeto, para isto é necessário que sejam informados os parâmetros *hWnd*, *Msg*, *wParam* e *lParam*. Estes parâmetros devem ser informados nesta ordem para serem enviados a um determinado objeto.

Uma mensagem não existe por si só, a mensagem sempre é associada a um objeto. O parâmetro *hWnd* da mensagem é o *handle* para o objeto. O parâmetro *Msg* define o tipo da mensagem a ser enviada para o objeto. O parâmetro *wParam* é uma mensagem-dependente de 16 *bits* e o *lParam* é uma mensagem-dependente de 32*bits*. Para alguns tipos de mensagens há outros dois parâmetros que devem ser informados após os parâmetros anteriormente

citados, estes parâmetros são *Time* e *PT*. O parâmetro *time* determina o tempo de espera para o envio da mensagem, este parâmetro é definido em milisegundos. O parâmetro *pt* define a posição atual, em coordenadas da tela, do cursor do *mouse.* A maioria das mensagens não utilizam estes dois parâmetros sendo que os principais são o *hWnd*, *Msg*, *wParam* e *lParam*, que devem ser informados respeitando esta ordem.

A seguir há um exemplo do envio de uma mensagem a um botão qualquer, a mensagem enviada é para simular um *click* do *mouse* sobre o botão:

```
sendmessage(handle_do_objeto,bm_click,0,0);
```

6.2 *HANDLE*

Segundo Wilken (1991), *handle* é o termo usado na programação objeto-orientada em geral, como também na programação Windows. O *handle* habilita ao programador o acesso a um objeto do sistema, mesmo que o programador não saiba muito a respeito do objeto. Muitos programadores pensam que é mais fácil chamar objetos diretamente utilizando o endereço do objeto. Porém, há uma razão para chamadas que utilizam o *handle.* O Windows vê objetos como recursos os quais podem ser janelas, objetos gráficos, contextos de dispositivos, etc. Estes recursos são armazenados na memória do computador. Para fazer o uso mais efetivo da memória do computador, o gerenciador de memória do Windows às vezes moverá objetos para outras localizações de memória ou até mesmo apagá-los. Se uma aplicação tivesse acesso a um recurso por seu endereço, a aplicação teria dificuldades em localizar um recurso que tenha sido movido pelo gerenciador de memória.

Para evitar isto, a aplicação utiliza rotinas especiais do Windows para obter o *handle* de um determinado recurso desejado. Internamente no Windows, o *handle* é representado por um número de 16 *bits*. O Windows assegura que um determinado *handle* sempre apontará para o objeto correto, mesmo que o gerenciador de memória altere o endereço de memória do objeto. A utilização do *handle* é muito diferente dos métodos tradicionais de gerenciamento de memória e endereçamento de memória.

6.2.1 Como obter o *handle* de um objeto?

Há várias formas e funções para a obtenção do *handle* de um determinado objeto, a seguir é demonstrado duas funções que têm esta finalidade, estas funções são: *findwindow* e *windowfrompoint*.

A função *findwindow* retorna o *handle* de uma janela cujo nome da classe ou o nome da janela coincida com um valor *string* informado na função. Esta função não procura janelas filhas de uma outra janela.

```
var_hwnd := findwindow(lpClassName,lpWindowName);
```

Conforme o exemplo acima, *var_hwnd* é a variável que irá receber o *handle* da janela, o parâmetro *lpClassName* conterá o nome da classe da janela que se deseja obter o *handle* e o parâmetro *lpWindowName* conterá o nome da janela. A seguir há dois exemplos práticos para a aquisição do *handle* da janela do Borland Delphi, caso o mesmo esteja em execução:

Exemplo 1:

```
hnd_delphi := findwindow('tappbuilder ',nil);
```

Exemplo 2:

```
hnd_delphi := findwindow(nil, 'Delphi 5');
```

No exemplo 1, é passado como parâmetro para a função *findwindow* o nome da classe da janela que se deseja obter o *handle*, já no exemplo 2, é passado como parâmetro para a função o nome da janela.

A função *windowfrompoint* retorna o *handle* do objeto que esteja nas coordenadas gráficas (x e y) que é passada para a função.

```
var
  var_hwnd : THandle;
  point : Tpoint;
begin
  point.x := 130;
  point.y := 325;
  var_hwnd := windowfrompoint(point);
end;
```

Conforme o exemplo acima, *var_hwnd* é a variável que irá receber o *handle* do objeto que está nas coordenadas armazenadas na variável *point.*

7 DESENVOLVIMENTO DO PROTÓTIPO

Neste capítulo é descrita a especificação do protótipo, utilizando as técnicas e tecnologias explanadas anteriormente. Serão apresentadas as ferramentas que foram utilizadas, a lógica de funcionamento do *hardware* e os componentes utilizados.

Após a especificação, será demonstrada a implementação, na qual é descrita a montagem do protótipo (*hardware*), e o desenvolvimento do *software* de integração entre o Virtual Turntables e o *hardware*.

7.1 ESPECIFICAÇÃO DO HARDWARE

Há dois pontos que devem ser analisados no desenvolvimento deste protótipo. A interface entre o *hardware* e o *software* de integração e, a comunicação entre o *software* de integração e o simulador Virtual Turntables.

7.1.1 Ferramentas Utilizadas

A ferramenta utilizada para a elaboração de fluxogramas foi o Visio da Microsoft (http://www.microsoft.com). Adotou-se a utilização de fluxogramas para especificação e documentação, por ser um método bastante difundido, principalmente quando se trata de demonstrar processos que tenham sua execução de forma seqüencial.

O esquema eletrônico foi desenvolvido utilizando o software Eagle 4.11 desenvolvido e distribuído pela CadSoft, que é situada na Flórida/EUA (http://www.cadsoftusa.com). Para simular partes do circuito eletrônico antes do desenvolvimento do esquema eletrônico propriamente dito, foi utilizado o Circuit Maker 2000 desenvolvido pela Altium, que é situada na Austrália (http://www.altium.com).

7.1.1.1 CadSoft Eagle

O Eagle é uma ferramenta para desenvolvimento de esquemas de circuitos eletrônicos, no qual é disponibilizado ao usuário uma grande diversidade de componentes eletrônicos, soquetes, conectores, entre outros, sendo que os componentes mais utilizados em diversos tipos de projetos se encontram na biblioteca de componentes do Eagle.

O Eagle disponibiliza um recurso muito útil durante o desenvolvimento e até mesmo ao término da construção de um esquema, o qual é denominado ERC (*Electrical Rule Check)*, que traduzindo significaria “verificação de regra elétrica”. Este recurso tem a finalidade de apontar erros nas conexões dos componentes e possíveis problemas que possam ocorrer (*warnings*).

Um outro recurso que poupa muito tempo do desenvolvedor é que, ao concluir o esquema, é possível migrar o mesmo para um PCB, onde os componentes podem ser posicionados conforme necessidade do usuário, e a criação das trilhas que interligaram os componentes pode ser feita de forma automática.

Foi a partir destas características apresentadas pelo Eagle, que se optou pela utilização do mesmo, principalmente pela grande gama de componentes eletrônicos entre outros, disponíveis em sua biblioteca, e a facilidade de migração de um esquema desenvolvido para um PCB.

7.1.1.2 Circuit Maker 2000

O Circuit Maker 2000, é uma ferramenta que permite a criação de esquemas e PCB’s, mas a principal finalidade dele no desenvolvimento deste protótipo, foi para simular partes do circuito a ser desenvolvido.

Embora este software também apresente uma grande gama de componentes em sua biblioteca, ele não possui o ADC0808, o qual foi utilizado no desenvolvimento deste protótipo, porém o Circuit Maker 2000 possui em sua biblioteca o ADC0804 que tem apenas uma entrada analógica. De qualquer forma ele foi muito útil, pois os demais componentes utilizados no desenvolvimento do protótipo, estavam presentes em sua biblioteca.

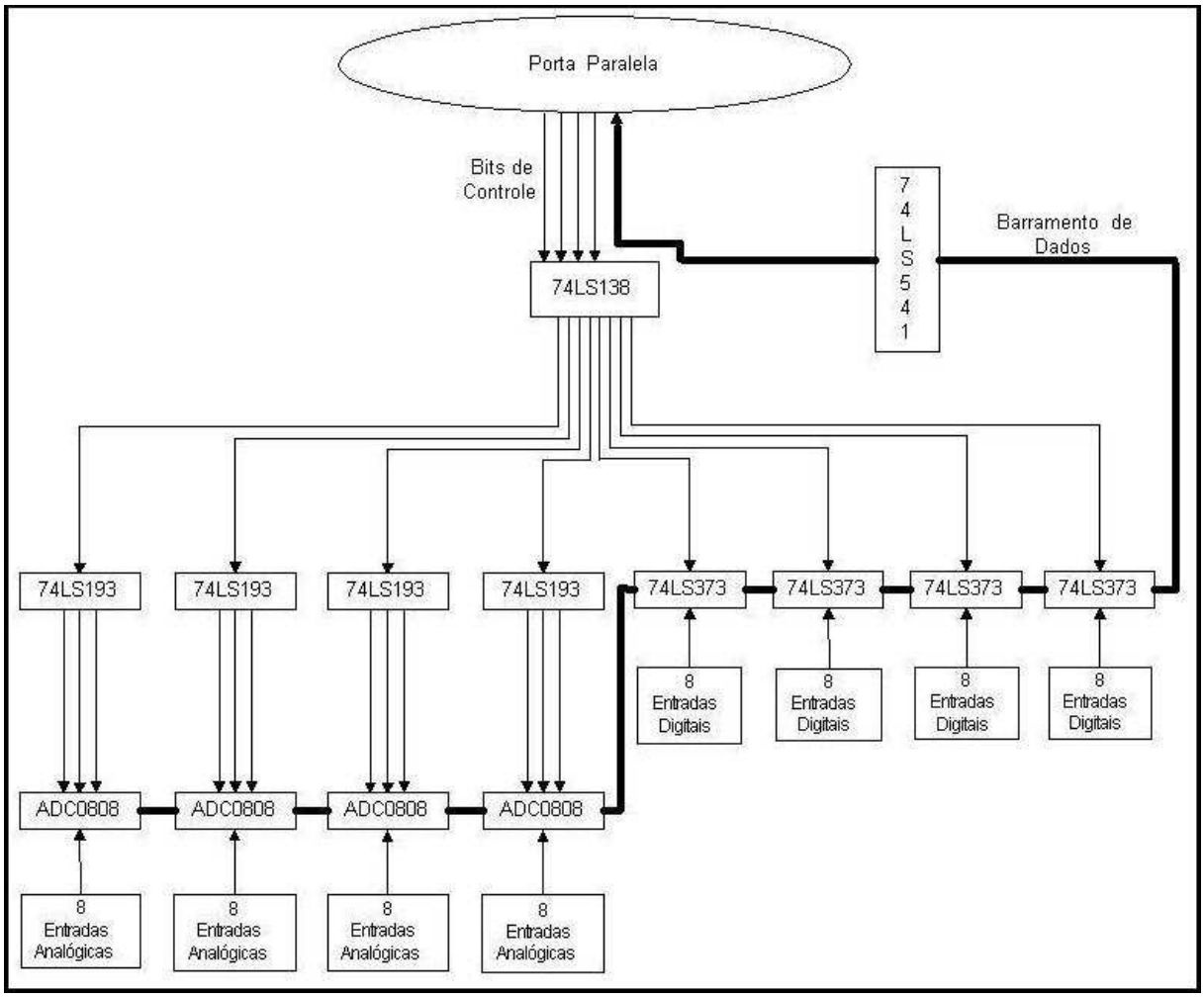
7.1.2 Estrutura do *Hardware* (Placa Principal)

Após o estudo dos componentes TTL mencionados anteriormente e da conexão da porta paralela, foi desenvolvido o esquema do *hardware* utilizando o Eagle, sendo que, através da combinação entre os CI’s TTL e a porta paralela, tornou-se possível a conexão de até 32 entradas digitais e 32 entradas analógicas utilizando a porta paralela no modo EPP. Com a possibilidade de monitoramento de 64 entradas distintas, é possível controlar as principais funções do Virtual Turntables sem utilizar todas as entradas digitais e analógicas do

hardware, desta forma, possibilitando a expansão das funções controladas pelo *hardware* sem precisar alterar ou aprimorar o esquema do mesmo.

A seguir é demonstrado o sentido em relação à porta paralela, em que os *bits* de controle e de dados trafegam no *hardware.* Para isto foi adotado o uso de um fluxograma com o intuito de facilitar a compreensão, conforme figura 30.

Figura 30 – Sentido do fluxo de dados do *hardware*



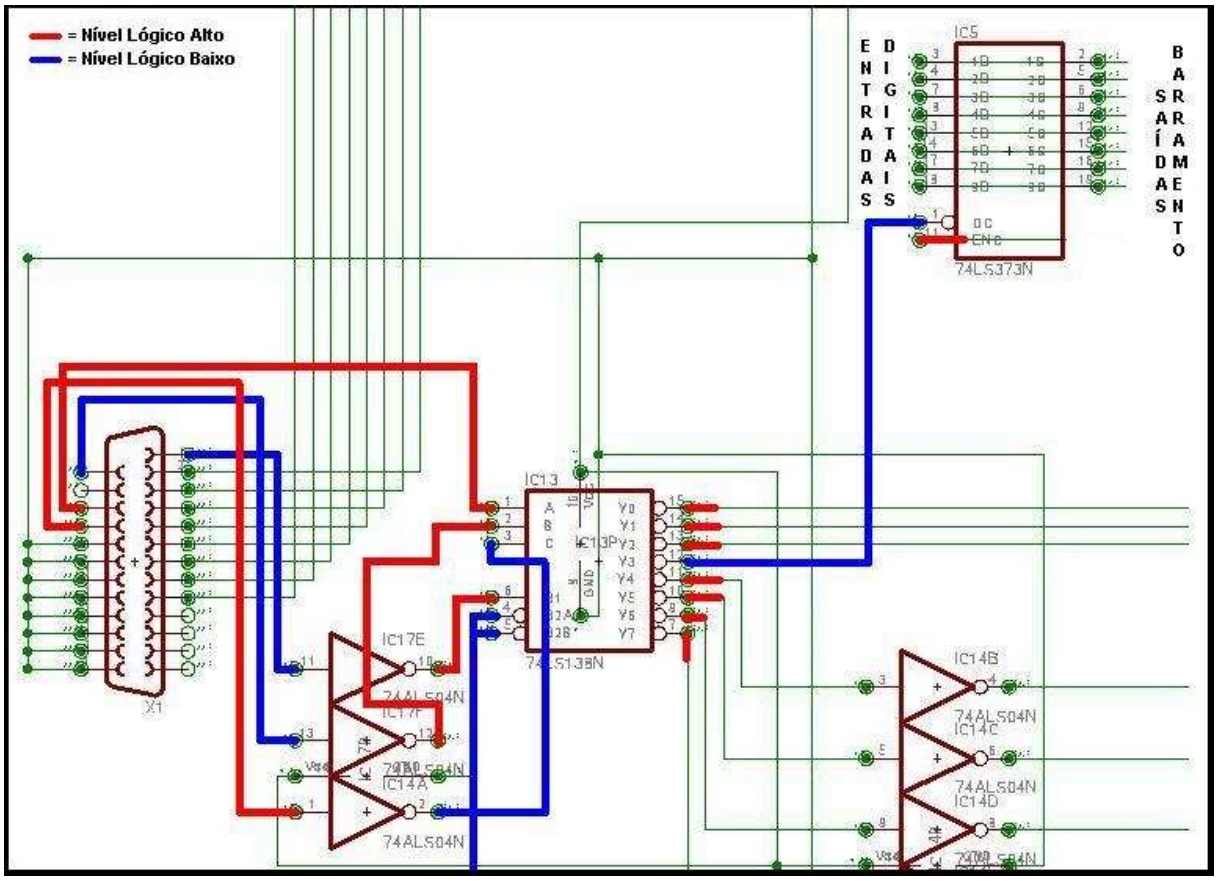
O funcionamento do *hardware* tem seu início no 74LS138 o qual fará a seleção de um dos oito CI's responsáveis pela aquisição dos dados gerados por potenciômetros e chaves (ADC0808 e 74LS373). Para controlar o 74LS138 foram necessários quatro pinos da porta paralela, sendo eles: pino 1 (*strobe*), pino 14 (*auto_feed*), pino 16 (*init*) e o pino 17 (*select_in*). O *strobe* será utilizado para habilitar a seleção do 74LS138, porém o mesmo possui lógica invertida, por este motivo, o *strobe* foi conectado a um 74LS04 para inverter o sinal disponibilizado pelo *strobe.* O *auto_feed*, *init* e *select_in* serão utilizados para

determinar qual das saídas será habilitado, porém o *auto_feed* e *select_in* possuem lógica invertida, por este motivo ambos foram conectados juntamente com o *strobe* a um 74LS04 para fazer a inversão do sinal, já o *init* possui lógica positiva, desta forma o mesmo foi conectado diretamente ao 74LS138. Estas conexões estão ilustradas no exemplo exposto na figura 31.

7.1.2.1 Aquisição das entradas digitais

Quando o 74LS138 selecionar uma de suas saídas e, se a saída selecionada for para fazer aquisição de uma entrada digital (saída do 74LS138 conectada a um 74LS373), o que estiver na entrada do 74LS373 será disponibilizado na saída, ou seja, será disponibilizado no barramento de dados para ser lido em seguida pelo *software* integrador, onde cada *bit* do *byte* disponibilizado no barramento de dados corresponde ao pressionamento de uma chave, ou seja, o software integrador irá ler oito entradas digitais simultaneamente (figura 31).

Figura 31 – Esquema de ligação do 74LS138, tendo como exemplo a seleção de um 74LS373 para aquisição de entradas digitais



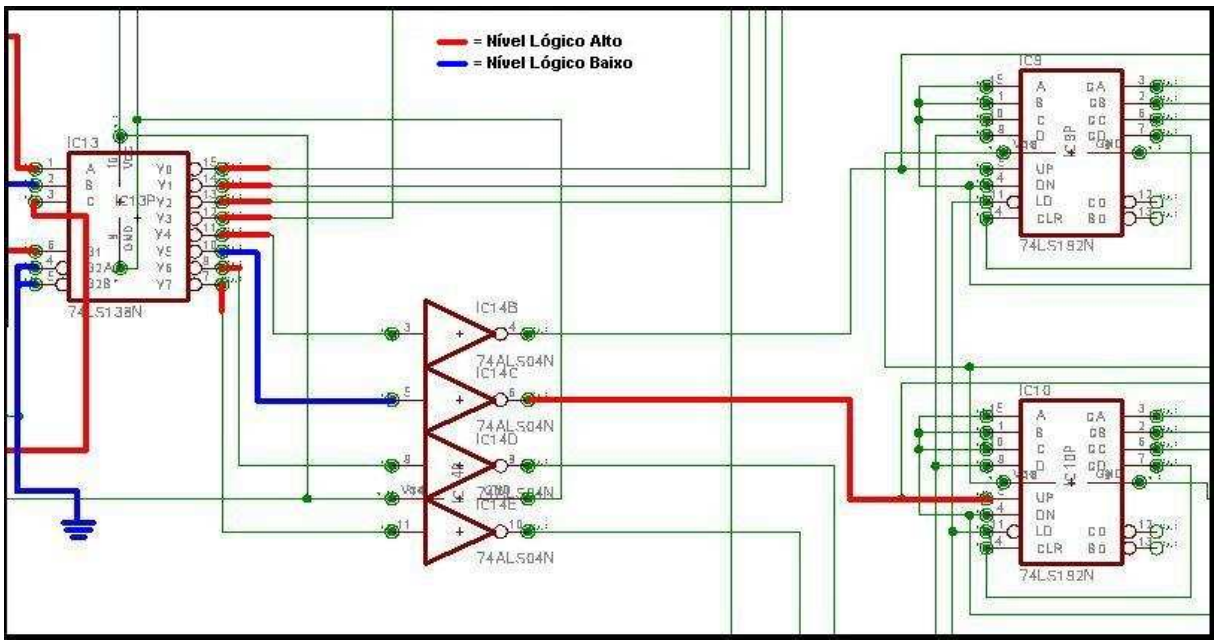
7.1.2.2 Aquisição das entradas analógicas

Quando o 74LS138 selecionar uma de suas saídas e, a saída selecionada for utilizada para aquisição de uma entrada analógica, é necessária uma combinação de componentes para que isto seja possível. Um dos motivos para isto é a falta de mais vias de controle na porta paralela e outro motivo é que, o *byte* disponibilizado no barramento refere-se a uma entrada analógica, diferente do que ocorre na aquisição das entradas digitais.

Os componentes combinados para viabilizar esta aquisição são o ADC0808, o 74LS192 e um dos conversores do 74LS04, para a aquisição de oito entradas analógicas. Para a aquisição das trinta e duas entradas analógicas serão necessários quatro ADC0808 e quatro 74LS192.

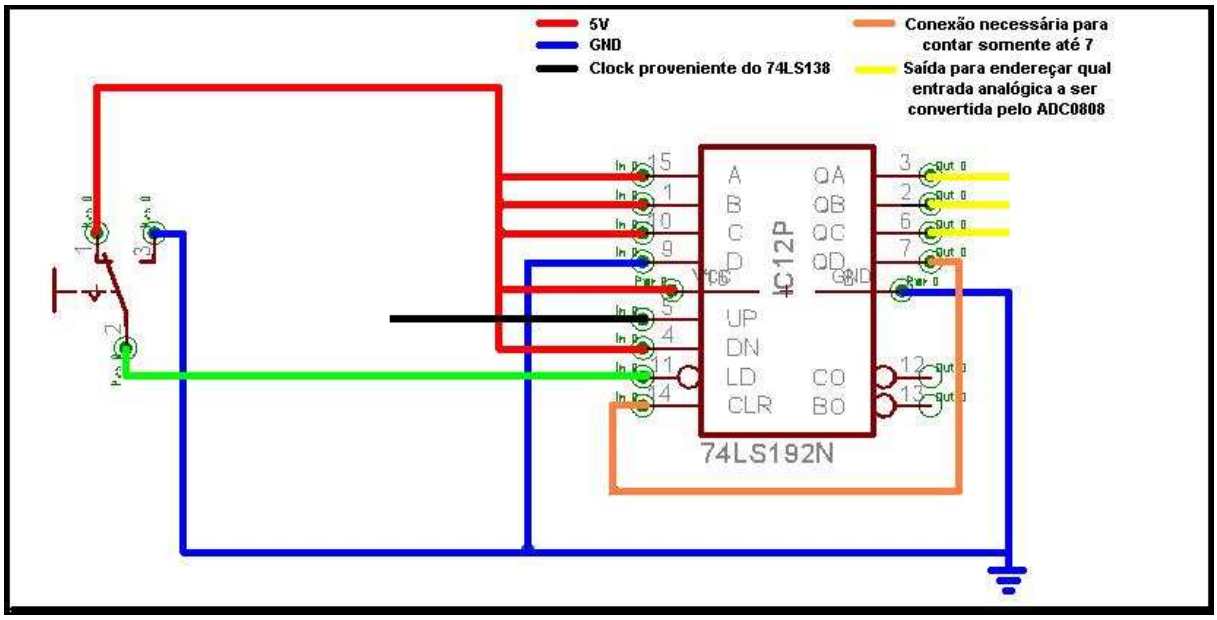
Ao selecionar uma saída do 74LS138 para aquisição de uma das oito entradas analógicas de um conversor A/D, o sinal que indica a seleção no pino em questão transitará do nível lógico alto para o nível lógico baixo, devido a este comportamento, este sinal será inserido no 74LS04 para a inversão do mesmo. A respectiva saída do inversor será utilizada como *clock* para o 74LS192 (o *clock* é na subida de borda), com isso cada vez que o 74LS138 selecionar este conversor A/D, a saída do 74LS192 será incrementada, isto é feito pois o ADC0808 necessita que seja informada qual das entradas analógicas deseja-se converter, e as saídas do 74LS192 tem por objetivo fornecer esta informação ao ADC0808 (figura 32).

Figura 32 – Esquema de ligação a partir da saída do 74LS138 que servirá de *clock* para o 74LS192



Como o ADC0808 necessita de três *bits* para selecionar uma das oito entradas analógicas e o 74LS192 é um contador de quatro *bits* e não de três *bits* (como seria o ideal), foi necessário fazer com que ele contasse de 0 a 7 e não de 0 a 15 como é o funcionamento padrão dele. No 74LS192 o pino que emite o *bit* mais significativo (o qual possibilita a contagem de 8 a 15 e é denominado de Q3), foi conectado ao pino MR (*master reset*), desta forma o 74LS192 passou a contar até 7, e da forma em que o mesmo foi ligado (união do pino Q3 ao pino MR), ao invés de incrementar sua saída para 8, a mesma retornará ao valor 0. Um outro detalhe que precisou ser observado é que se o 74LS192 estivesse ligado simplesmente desta forma, ao receber o primeiro pulso de *clock* ele informaria ao ADC0808 o endereço da segunda entrada analógica e não a primeira. Como o 74LS192 tem a possibilidade de ser inicializado de forma assíncrona utilizando o pino denominado PL, foi colocado um botão que inicializa os quatro CI's 74LS192 com o valor 7, com isso ao receber o primeiro pulso de *clock*, a saída do 74LS192 passará a conter 0 o que fará com que seja selecionada a primeira entrada analógica do ADC0808 conectada a ele. O esquema de ligação do 74LS192 é demonstrado na figura 33

Figura 33 – Esquema de ligação do 74LS192



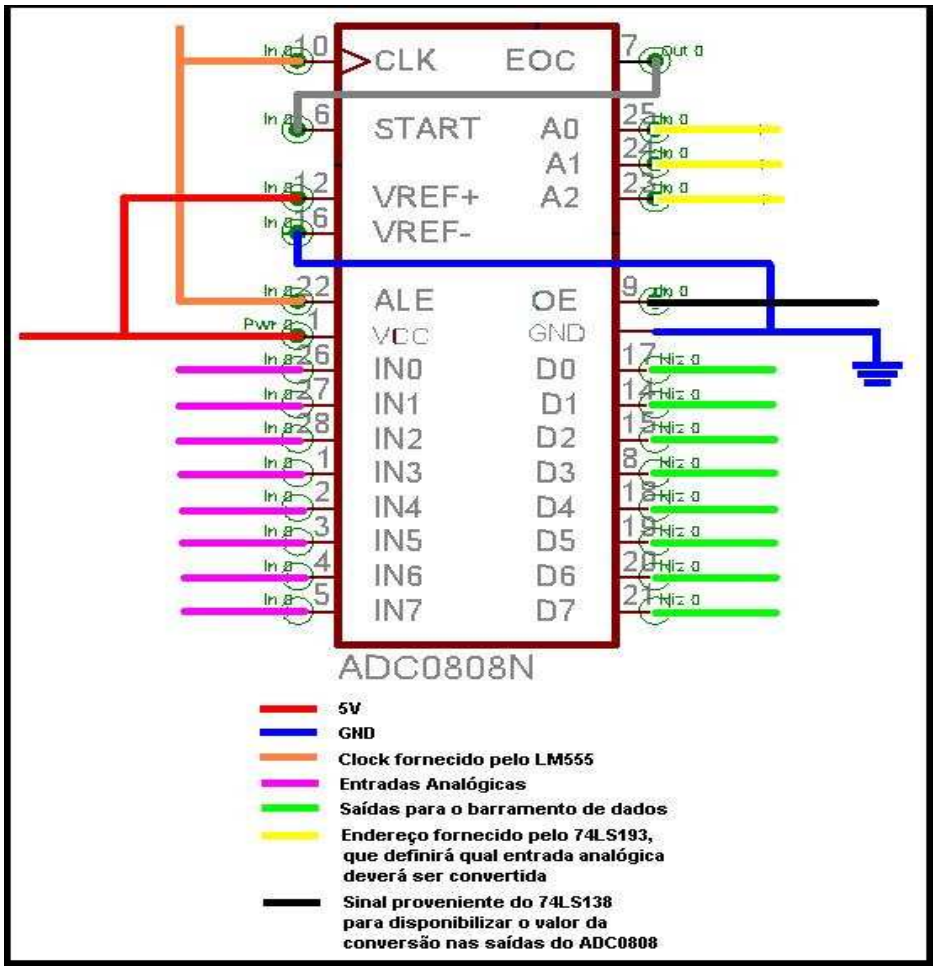
O mesmo sinal que é utilizado como *clock* para o 74LS192, é utilizado no ADC0808 conectado ao pino denominado *output enable*, para que o valor convertido conforme a entrada analógica selecionada seja disponibilizado no barramento de dados.

O ADC0808 possui um pino denominado EOC (*end of conversion*) o qual está sempre em nível lógico alto e que ao término da conversão passa a ter o nível lógico baixo, e há também o pino START, que ao receber um nível lógico alto dá início à conversão do sinal analógico. Diante destas características do ADC0808, o pino EOC foi conectado ao pino START, fazendo com que o ADC0808 fique convertendo constantemente.

Para que o ADC0808 faça a conversão é necessário que o mesmo receba um sinal de *clock* de no máximo 500Khz conforme seu *datasheet*. Para geração deste sinal de clock, foi utilizado o LM555 produzido pela National Semiconductor, o qual possibilita a geração de uma oscilação de 500Khz. Este sinal de *clock* também é conectado ao pino ALE do ADC0808 para que ele fique constantemente selecionando o endereço de entrada proveniente do 74LS192.

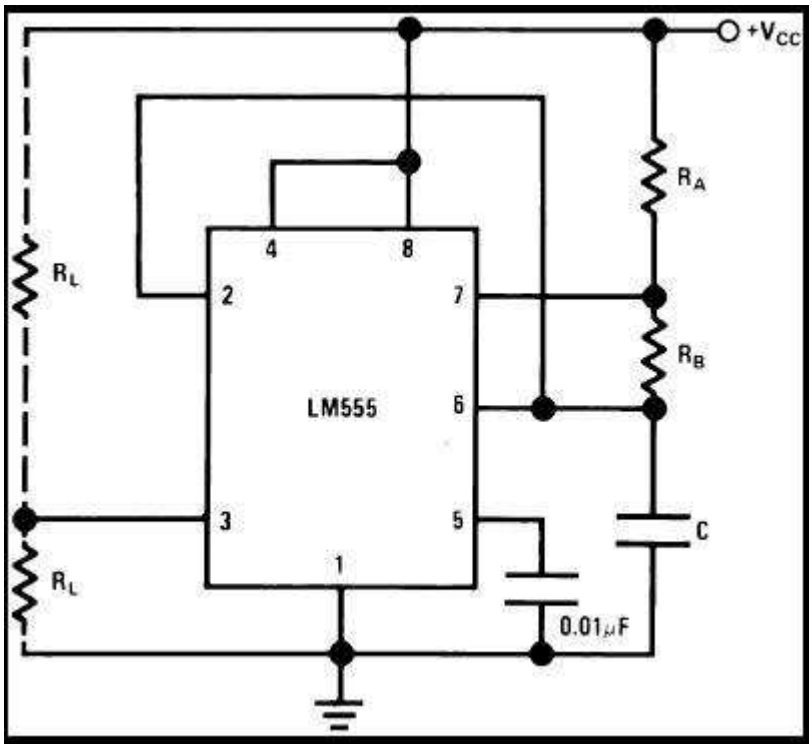
Para uma melhor compreensão do funcionamento do ADC0808, a figura 34 demonstra o esquema de ligação deste CI.

Figura 34 – Esquema de ligação do ADC0808



O LM555 foi condicionado ao modo de operação *astable* (figura 35), conforme recomendado pelo fabricante em seu *datasheet* quando a finalidade da utilização deste CI for para a geração de *clock*. O que determina a freqüência do *clock* são os valores de RA, RB e C, quanto menor o valor desses componentes, maior será a quantidade de oscilações por segundo gerados pelo LM555.

Figura 35 – LM555 configurado para operar no modo *astable*



Fonte: *Datasheet* LM555

O sinal de clock gerado pelo LM555 é compartilhado para todos os ADC0808 presentes no circuito.

O 74LS541 funciona como um buffer, a princípio este componente não seria incorporado ao circuito, mas por precaução optou-se pela inclusão do mesmo, o principal papel dele neste circuito é preservar a integridade da porta paralela, pois caso ocorra uma sobrecarga no *hardware* a tendência é que este componente queime, caso esteja ligado o barramento de dados diretamente na porta paralela, há o risco diante de uma situação dessa de danificar a porta paralela do PC. O 74LS541 está ligado de forma que os dados do barramento estejam sempre disponíveis para a porta paralela. Este CI poderia estar ligado de forma que, os dados do barramento só fossem liberados para a porta paralela através de um

sinal de controle, mas como visto anteriormente, não dispomos de mais vias de controle para este gerenciamento.

7.1.3 Estrutura do *Hardware* (Placas adicionais)

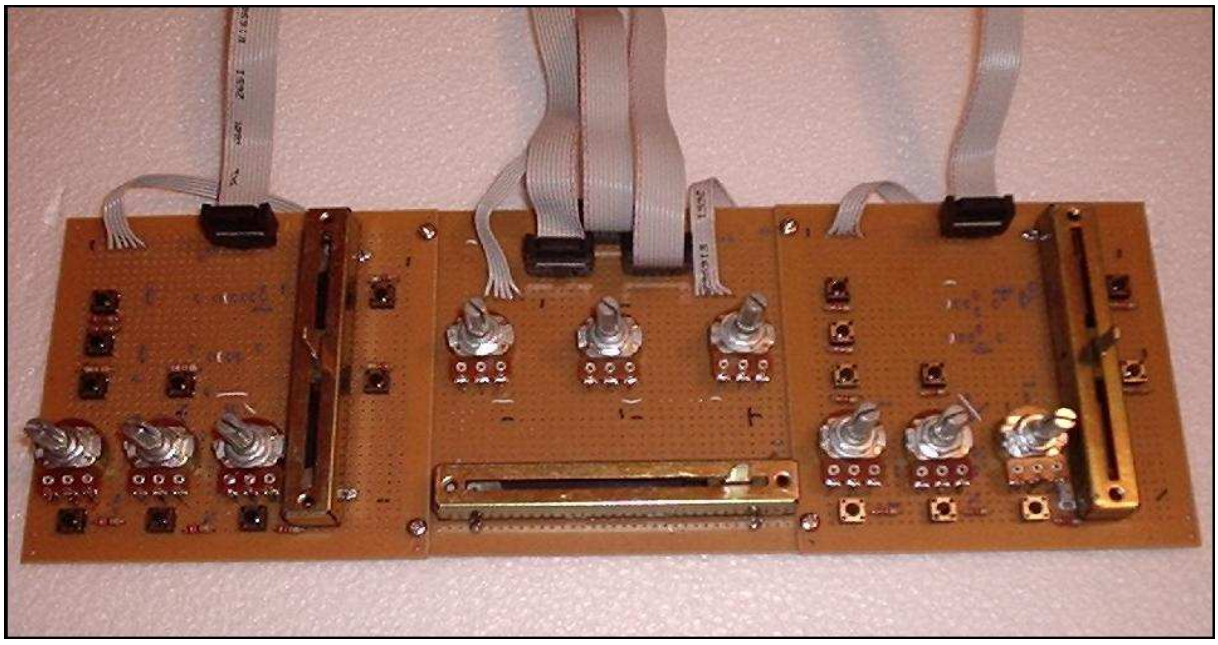
A maior ênfase no desenvolvimento deste *hardware* protótipo está no desenvolvimento da placa principal, sendo que, nela estão todos os CI's necessários para o funcionamento do *hardware*. As placas adicionais são compostas apenas por chaves, potenciômetros e soquetes, e o nível de complexidade na construção das mesmas é pequeno.

As chaves recebem uma tensão de 5V. Ao pressionar a chave, esta tensão será conduzida para os respectivos circuitos integrados. Da mesma forma funcionam os potenciômetros, que também são alimentados com 5V, a diferença é que o valor enviado para os circuitos integrados ADC é uma tensão que varia entre 0V e 5V, já que um potenciômetro é uma resistência variável.

Os soquetes têm a finalidade de interligar as placas adicionais à placa principal através de cabos *flat.*

Na figura 36 é demonstrado as placas adicionais que foram confeccionadas para a colocação das chaves, potenciômetros e soquetes, que servirão como entrada de dados para a placa principal.

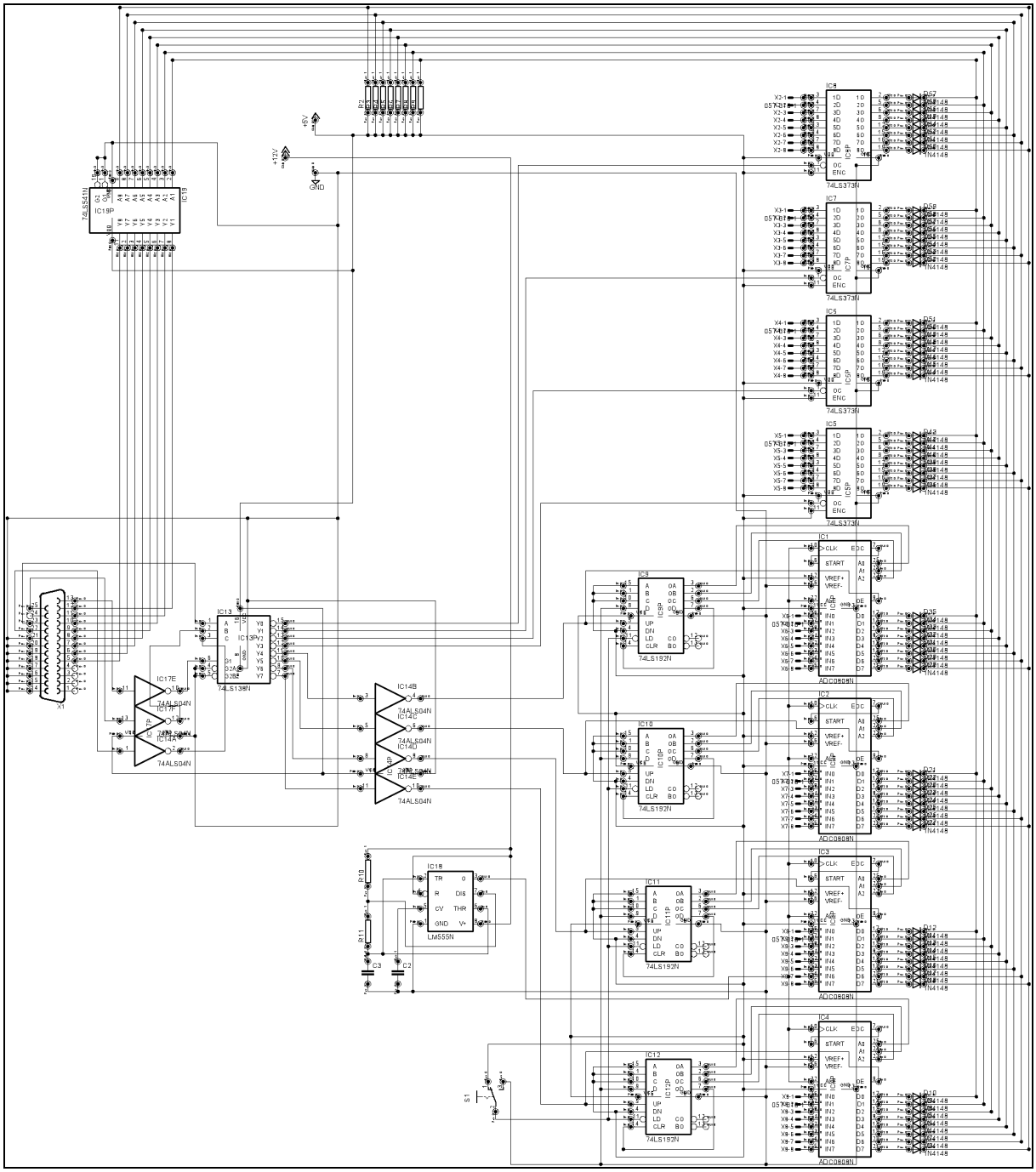
Figura 36 – Placas Adicionais



7.1.4 Protótipo (*hardware*)

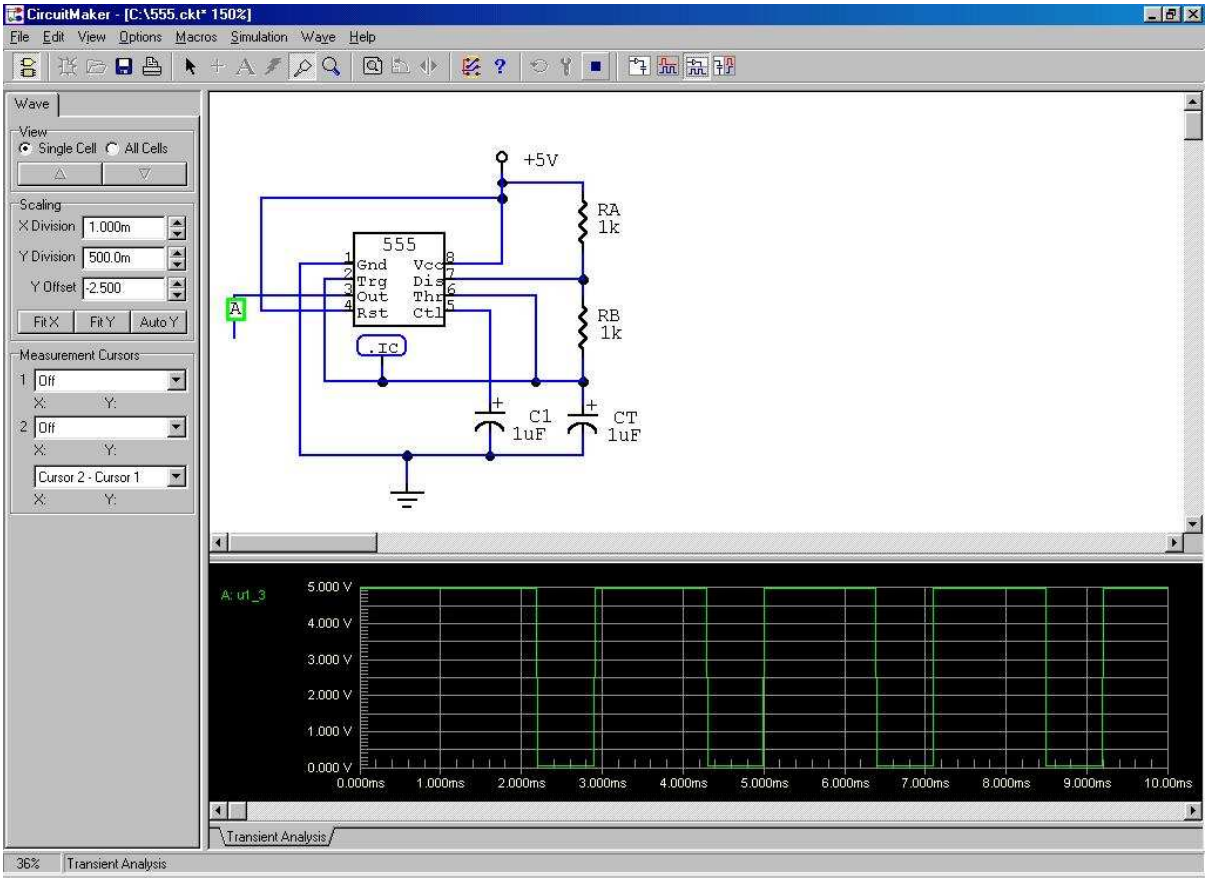
Conforme exposto no tópico 7.1.2 (estrutura do *hardware*), que trata especificamente da placa principal, foi elaborado o esquema eletrônico do protótipo utilizando o Eagle 4.11. O qual é demonstrado na figura 37.

Figura 37 – Esquema eletrônico do protótipo desenvolvido através do Eagle 4.11



Durante a elaboração do esquema foram efetuadas diversas simulações de pequenas partes do circuito proposto. Para a realização destas simulações foi utilizado o Circuit Maker 2000. Uma das simulações feitas pode ser observada na figura 38, a qual se refere ao gerador de *clock* utilizando o CI LM555.

Figura 38 – Simulação do gerador de *clock* utilizando o Circuit Maker 2000

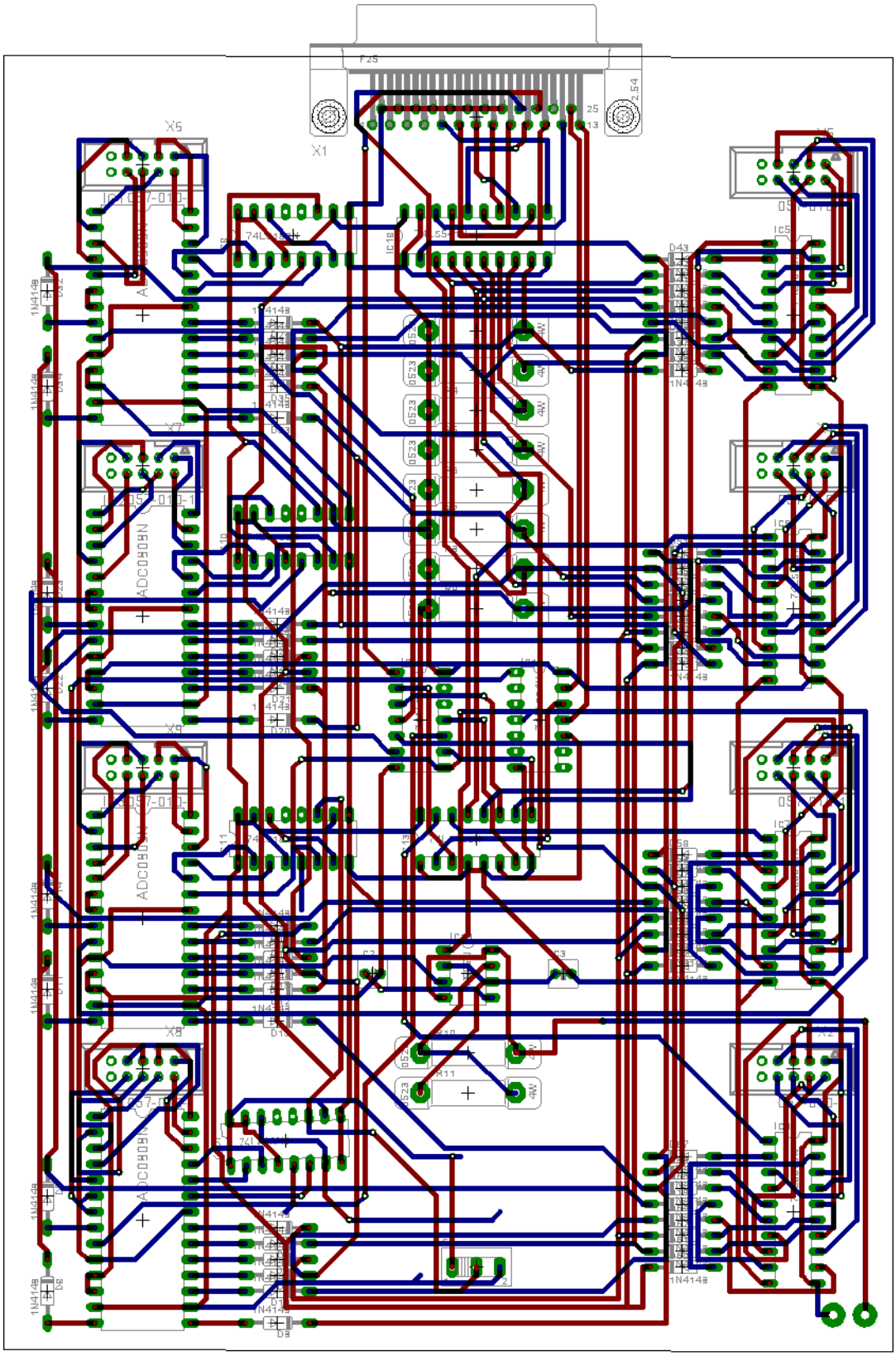


Ao término da elaboração do esquema foi utilizado o recurso ERC do Eagle, o qual já foi demonstrado anteriormente, para checar a integridade elétrica do circuito. Em um primeiro momento foram detectadas aproximadamente 100 ocorrências de irregularidades (*errors* e *warnings*), os quais foram sanados. Uma das ocorrências que pode ser utilizada como exemplo, foi a não conexão do pino GND do 74LS138 ao terra comum entre os demais CI's.

Após a correção dos erros e irregularidades constatadas através do ERC, foi efetuada a exportação do esquema para o PCB, como esta exportação é um processo automatizado pelo Eagle, a disposição dos componentes na placa não ficaram bem posicionados, levando em consideração a estética da placa a ser confeccionada. Então se optou por reposicionar os componentes na placa através do rotacionamento de alguns componentes e movendo os componentes para outros lugares, até que fosse obtido um *layout* com boa estética e que fosse de certa forma legível, tudo isto com o intuito de facilitar a compreensão do funcionamento do circuito.

Após a conclusão do *layout* da placa (organização dos componentes), foi utilizado o recurso de auto-roteamento das trilhas que interligam os componentes. Para isto foi necessário definir a espessura da broca a ser utilizada para perfurar a placa, a espessura mínima das trilhas e a distância entre as trilhas. O resultado obtido é demonstrado na figura 39.

Figura 39 – PCB do protótipo desenvolvido através do Eagle 4.11



Ao alcançar este ponto, tendo concluído o processo de elaboração do esquema eletrônico e o processo de criação do *layout* do PCB, o próximo passo foi confeccionar a placa de circuito impresso.

7.1.4.1 O PCB (Placa de circuito impresso)

Tendo o *layout* do PCB em mãos, foi iniciada uma pesquisa de preços para terceirizar a confecção da placa de circuito impresso. Por se tratar de uma única placa a ser confeccionada, o custo tornou-se muito elevado, pois o orçamento mais acessível foi o da Meyer Placas, situada em Blumenau/SC, em que o custo de confecção de um exemplar desta placa de circuito impresso seria de U$ 40,00, sendo que neste valor não estava incluso os fotolitos que teriam que ser elaborados para realizar a confecção da placa de circuito impresso.

Diante desta situação, optou-se por confeccionar a placa de circuito impresso de forma caseira, com o intuito da redução de custos na elaboração do protótipo.

Após estudar diversas técnicas para criação de placas de circuito impresso, foi adotada uma técnica simples, mas que exige paciência e que demanda bastante tempo para a criação da placa de circuito impresso, a qual será demonstrada a seguir.

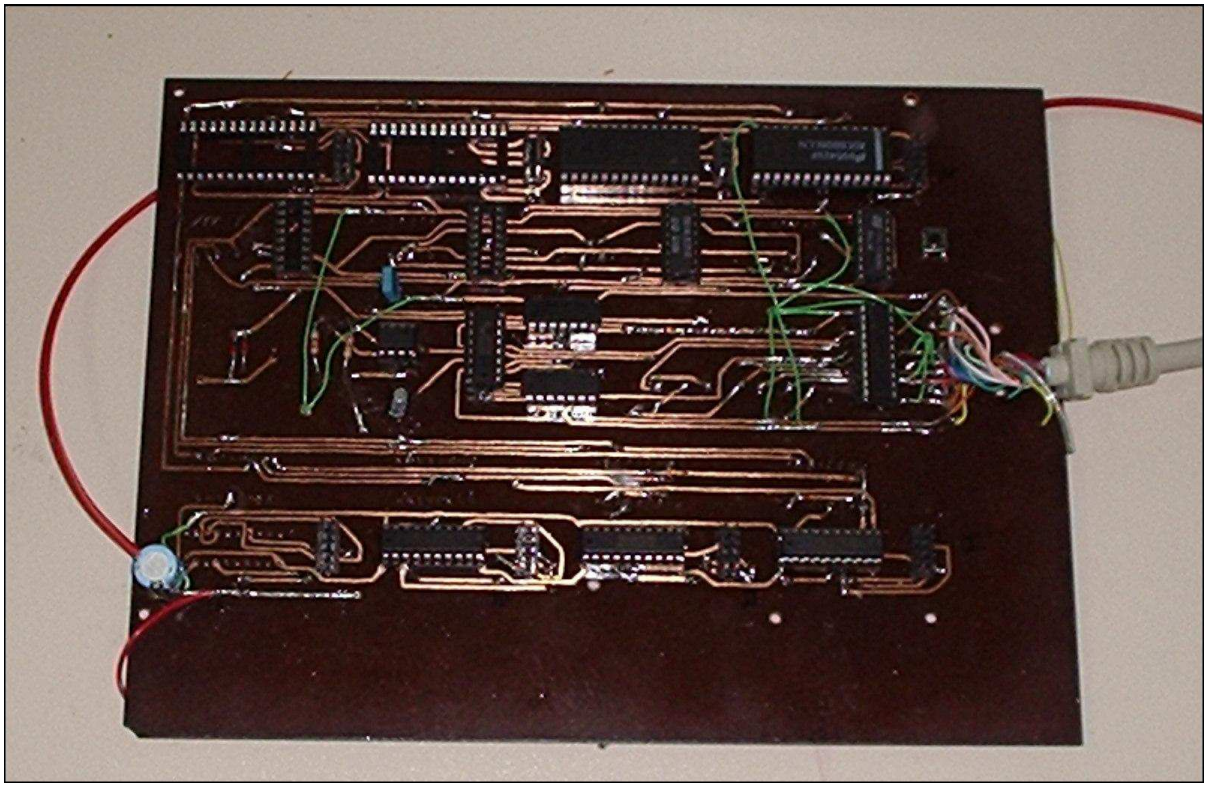
Esta técnica consiste em transferir de forma manual o *layout* do PCB para a placa de fenolite. Por se tratar de uma placa de circuito impresso de face dupla, foi impresso o PCB em duas partes e na escala original, a parte de cima do PCB foi impresso no modo normal e contendo somente as trilhas que pertenciam a parte de cima da placa de circuito impresso, e a parte de baixo do PCB foi impresso no modo *mirror* (espelho) e contendo somente as trilhas da parte de baixo da placa de circuito impresso. Para a transferência do PCB para a placa de fenolite, foi fixado o impresso da face de cima sobre uma folha de papel carbono e ambas as folhas foram fixadas através de fita adesiva à placa de fenolite, então foram feitos dois furos que serviram de referência para a fixação do impresso juntamente com o papel carbono correspondente a parte de baixo do circuito impresso. Após fixar os PCB's parciais às duas faces da placa, foram marcados os furos dos CI's e demais componentes com o uso de um furador manual, em seguida foi desenhado com a lapiseira sobre o PCB impresso, para que o mesmo seja transpassado para a placa de fenolite pelo papel carbono. Concluído este processo o *layout* da placa ficou marcado na placa de fenolite devido ao papel carbono, então foi

desenhado com uma caneta permanente de 0.1mm para retroprojetor com o auxílio de uma régua sobre o que já havia previamente sido marcado com a utilização do papel carbono. Este processo foi executado para ambos os lados da placa de circuito impresso.

Ao concluir a marcação das trilhas com o uso da caneta permanente, foi efetuada uma verificação para certificar de que não havia trilhas em curto circuito ou algo do gênero e em seguida foi submetida a placa de fenolite a uma solução de água com percloreto de ferro para efetuar a corrosão do cobre excedente da placa. Ao término da corrosão a placa de circuito impresso foi lavada com água em abundância e retirado a tinta permanente das trilhas com o auxílio de uma palha de aço, em seguida foi aplicada uma camada de verniz para evitar a oxidação prematura das trilhas.

Com isso a elaboração do *hardware* protótipo está concluída. Na figura 40 é demonstrada a placa principal após a corrosão e colocação dos componentes na mesma.

Figura 40 – Placa Principal



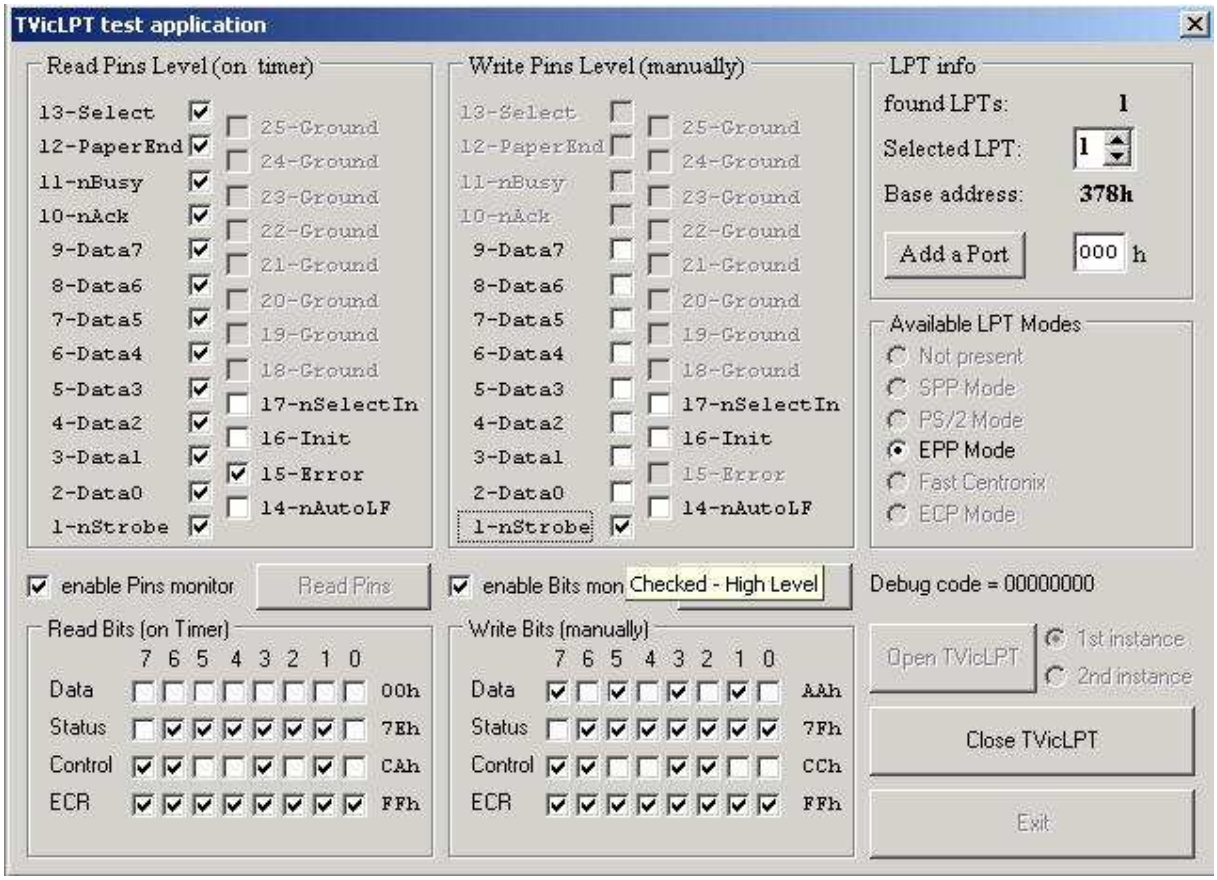
7.1.5 Protótipo (*software*)

A finalidade do *software* desenvolvido é fazer a integração entre o *hardware* proposto e o simulador Virtual Turntables, para que isto ocorra, o *software* proposto, o qual denominaremos de Integrador, precisará enviar *bits* de controle e ler os dados gerados pelo *hardware* utilizando a porta paralela. A partir destes *bits* de controle e dos dados fornecidos pelo *hardware* o Integrador enviará mensagens utilizando-se da API do Windows para o Virtual Turntables.

Um problema que foi constatado durante a fase de implementação é que a ferramenta adotada para o desenvolvimento do *software* protótipo, que no caso é o Delphi 5. Esta versão de 32 *bits* do compilador Borland Delphi Object Pascal não possui uma função para acesso à porta paralela, para o programador poder acessar e controlar a porta paralela, tem que fazer utilizando *assembly*.

Como o foco deste trabalho não é o estudo da programação *assembly*, optou-se por utilizar um componente de terceiros para o Delphi 5 que faça a interface com a porta paralela. Após estudar e analisar diversos componentes, foi escolhido o componente TVicLPT versão 1.3, o qual é desenvolvido pela EnTech Taiwan. Este componente é uma versão *shareware*, mas que funciona em sua totalidade. O TVicLPT facilita bastante a manipulação da porta paralela, pois podemos definir de forma intuitiva se um *bit* de um determinado pino da porta paralela deve estar em nível lógico alto ou baixo. Este componente vem acompanhado de um exemplo que demonstra a facilidade de manipulação dos *bits* na porta paralela, como pode ser visto na figura 41.

Figura 41 – Exemplo utilizando o componente TVicLPT

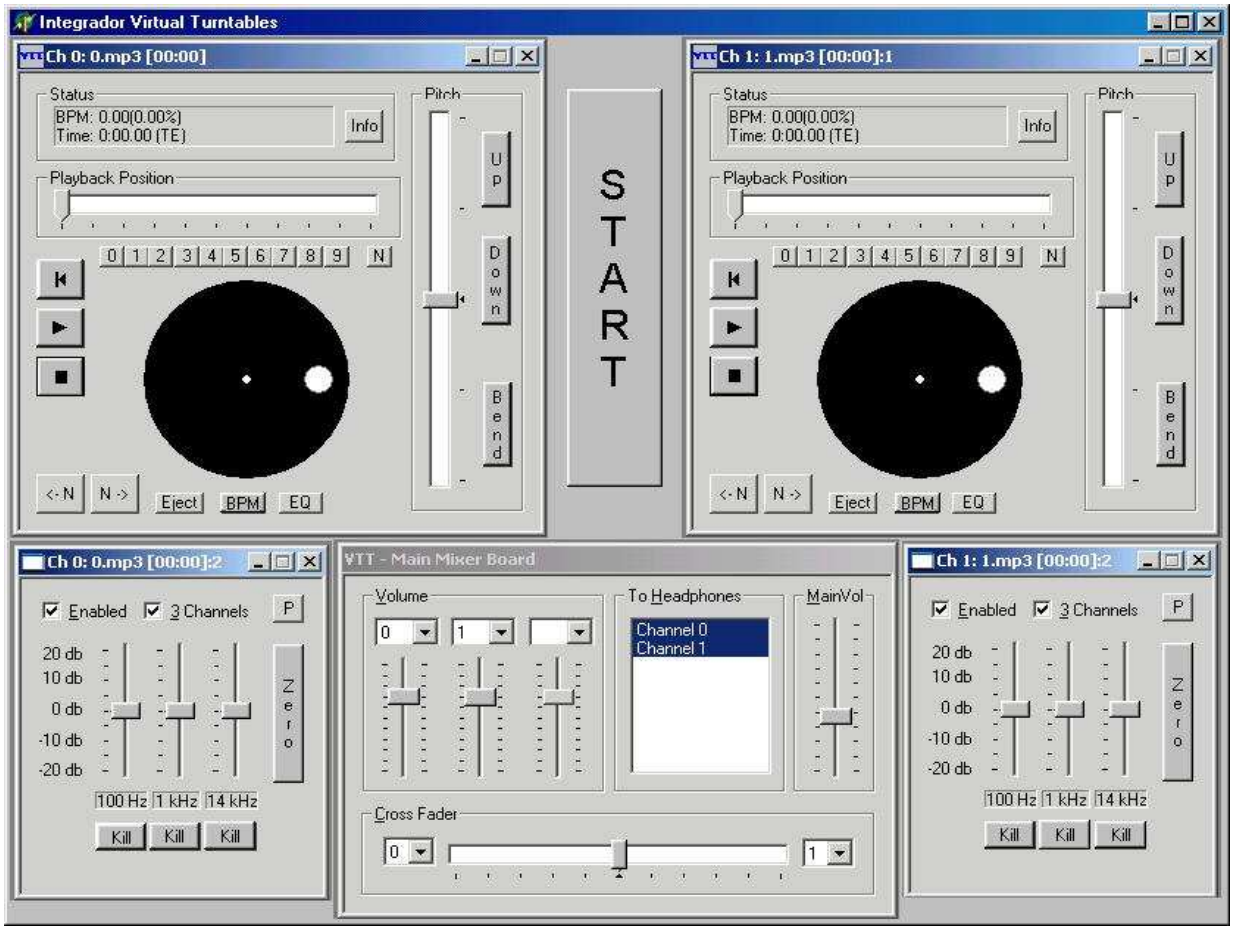


O *software* proposto é constituído basicamente por duas partes, a primeira parte é onde o usuário deverá capturar de forma manual e muito simples os botões que se deseja controlar através do *hardware*. A segunda etapa é quando o *software* realmente começa a interagir com o *hardware* desenvolvido.

Para capturar os botões que deverão ser controlados pelo *hardware*, o *software* integrador possui o mesmo *layout* do Virtual Turntables (figura 42), sendo que os botões que possuem um tom de cinza diferente dos demais, são os que o usuário deverá capturar. Para que a captura seja feita, basta clicar no botão do *software* integrador e em seguida posicionar o *mouse* sobre o respectivo botão do Virtual Turntables. O usuário tem três segundos para posicionar o mouse sobre o respectivo botão, é através da posição do *mouse* que o *software* integrador obtém o *handle* do objeto, para que o *software* integrador possa enviar mensagens para ele futuramente. A seguir é demonstrada a parte principal da rotina responsável pela captura do *handle* dos objetos a serem controlados:

```
procedure TFPrincipal.Inicio_P1Click(Sender: TObject);
var
   mouse_position : tpoint;   //Armazena as coordenadas X e Y do mouse
begin
   //Aguarda 3segundos para o usuário posicionar o mouse sobre o botão
   sleep(3000);
   // Obtém as coordenadas X e Y do mouse
   getcursorpos(mouse_position);
   // Obtém o endereço de memória do objeto que esta nas coordenadas
   //obtidas anteriormente
   hnd_inicio_p1 := windowfrompoint(mouse_position);
   // Oculta e em seguida exibe o objeto, para que o usuário certifique-se
   //de que o mesmo foi obtido com sucesso
   showwindow(hnd_inicio_p1,sw_hide);
   sleep(500);
   showwindow(hnd_inicio_p1,sw_restore);
end;
```

Figura 42 – *Software* Integrador



A segunda etapa do *software* é iniciada ao clicar no botão START, ao clicar neste botão o *software* integrador irá inicializar a porta paralela, e começará a enviar os *bits* de controle para o *hardware*, bem como, ler os *bits* que são disponibilizados no barramento de dados.

Quando for selecionada a leitura de entradas digitais, o *byte* disponibilizado no barramento será referente a oito entradas digitais, sendo que cada entrada digital corresponde a um *bit*. Se o *bit* estiver em nível lógico alto, será enviada uma mensagem para o respectivo botão, os quais já foram capturados anteriormente.

A seguir há uma demonstração do código responsável por inicializar a porta paralela, endereçar a aquisição de entradas digitais, e o tratamento feito após ler o *byte* disponibilizado no barramento de dados.

```
procedure TFPrincipal.btStartClick(Sender: TObject);
  var
    // Variáveis que armazenam o nível lógico de cada pino no barramento de
    //dados
     data0,
     data1,
     data2,
     data3,
     data4,
     data5,
     data6,
     data7 : boolean;
     // Variável para permanecer no loop
     identificador : integer;

  begin
     identificador := 174373;
     // Inicialização da porta paralela
     viclpt1.Active := 1;
     viclpt1.CurrentLptMode := lpt_epp_mode;
     viclpt1.ReadMode := true;
     viclpt1.Pin[1]  := true;

     // Laço permanente
     while identificador <> 0 do
        begin
        // Selecção do primeiro CI 74373
        if identificador = 174373 then
           begin
              // Envia endereço para o 74138 selecionar o primeiro 74373
              viclpt1.pin[14] := false;
              viclpt1.pin[16] := true;
              viclpt1.pin[17] := true;
              viclpt1.Pin[1]  := false;
```

```
            // Lê os dados disponibilizados no barramento
            data0 := viclpt1.pin[2];
            data1 := viclpt1.pin[3];
            data2 := viclpt1.pin[4];
            data3 := viclpt1.pin[5];
            data4 := viclpt1.pin[6];
            data5 := viclpt1.pin[7];
            data6 := viclpt1.pin[8];
            data7 := viclpt1.pin[9];
            viclpt1.pin[1] := true;
            identificador := 274373;

            if data0 = true then
                sendmessage(hnd_kh_p1,bm_click,0,0);

            if data1 = true then
                sendmessage(hnd_km_p1,bm_click,0,0);

            if data2 = true then
                sendmessage(hnd_kl_p1,bm_click,0,0);

            if data3 = true then
                sendmessage(hnd_down_p1,bm_click,0,0);

            if data4 = true then
                sendmessage(hnd_up_p1,bm_click,0,0);

            if data5 = true then
                sendmessage(hnd_stop_p1,bm_click,0,0);

            if data6 = true then
                sendmessage(hnd_play_p1,bm_click,0,0);

            if data7 = true then
                sendmessage(hnd_inicio_p1,bm_click,0,0);

        end;
```

A leitura das entradas analógicas e digitais é feita de forma seqüencial, começando pelas entradas digitais e em seguida, lendo as entradas analógicas. Esta rotina de leitura estará em um *loop* infinito, até que o *software* integrador seja finalizado.

Quando for selecionada a leitura de entradas analógicas, o *byte* disponibilizado no barramento será referente a uma entrada analógica, esta entrada irá variar entre 0 e 255 (decimal), e conforme o valor lido será definida a posição que o botão deverá estar no componente *trackbar* do Virtual Turntables.

A seguir há uma demonstração do código responsável por inicializar a porta paralela, endereçar a aquisição de entradas analógicas, e o tratamento feito após ler o *byte* disponibilizado no barramento de dados.

```
while loop < 9 do
  begin
  //Seleção do primeiro CI ADC0808
  if identificador = 10808 then
    begin
      // Envia endereço para o 74138 selecionar o primeiro ADC0808
      viclpt1.pin[14]  := true;
      viclpt1.pin[16]  := false;
      viclpt1.pin[17]  := false;
      viclpt1.pin[1]   := false;
      sleep(10);

      // Lê os dados disponibilizados no barramento
      decimal := 0;
      viclpt1.ReadMode := true;
      data0   := viclpt1.pin[2];
      data1   := viclpt1.pin[3];
      data2   := viclpt1.pin[4];
      data3   := viclpt1.pin[5];
      data4   := viclpt1.pin[6];
      data5   := viclpt1.pin[7];
      data6   := viclpt1.pin[8];
      data7   := viclpt1.pin[9];
      viclpt1.ReadMode := false;
      viclpt1.pin[1] := true;

      if data7 = true then
        decimal := decimal + 1;
      if data6 = true then
        decimal := decimal + 2;
      if data5 = true then
        decimal := decimal + 4;
      if data4 = true then
        decimal := decimal + 8;
      if data3 = true then
        decimal := decimal + 16;
      if data2 = true then
        decimal := decimal + 32;
      if data1 = true then
        decimal := decimal + 64;
      if data0 = true then
        decimal := decimal + 128;

      // Controla os agudos do Player2
      if loop = 1 then
        if decimal <> dec_hp2 then
          begin
            dec_hp2 := decimal;
            x_hp2 := last_pos_high_p2.x;
            y_hp2 := last_pos_high_p2.y;
            move_volumes(x_hp2,y_hp2,decimal);
            getcursorpos(last_pos_high_p2);
          end;
```

```
      // Controla os médios do Player2
      if loop = 2 then
        if decimal <> dec_mp2 then
          begin
            dec_mp2 := decimal;
            x_mp2 := last_pos_mid_p2.x;
            y_mp2 := last_pos_mid_p2.y;
            move_volumes(x_mp2,y_mp2,decimal);
            getcursorpos(last_pos_mid_p2);
          end;

      // Controla os graves do Player2
      if loop = 3 then
        if decimal <> dec_lp2 then
          begin
            dec_lp2 := decimal;
            x_lp2 := last_pos_low_p2.x;
            y_lp2 := last_pos_low_p2.y;
            move_volumes(x_lp2,y_lp2,decimal);
            getcursorpos(last_pos_low_p2);
          end;

      // Controla os agudos do Player1
      if loop = 4 then
        if decimal <> dec_hp1 then
          begin
            dec_hp1 := decimal;
            x_hp1 := last_pos_high_p1.x;
            y_hp1 := last_pos_high_p1.y;
            move_volumes(x_hp1,y_hp1,decimal);
            getcursorpos(last_pos_high_p1);
          end;

      // Controla os médios do Player1
      if loop = 5 then
        if decimal <> dec_mp1 then
          begin
            dec_mp1 := decimal;
            x_mp1 := last_pos_mid_p1.x;
            y_mp1 := last_pos_mid_p1.y;
            move_volumes(x_mp1,y_mp1,decimal);
            getcursorpos(last_pos_mid_p1);
          end;

      // Controla os graves do Player1
      if loop = 6 then
        if decimal <> dec_lp1 then
          begin
            dec_lp1 := decimal;
            x_lp1 := last_pos_low_p1.x;
            y_lp1 := last_pos_low_p1.y;
            move_volumes(x_lp1,y_lp1,decimal);
            getcursorpos(last_pos_low_p1);
          end;
```

```
      // Controla o Pitch do Player1
      if loop = 7 then
        if decimal <> dec_pp1 then
          begin
            dec_pp1 := decimal;
            x_pp1 := last_pos_pitch_p1.x;
            y_pp1 := last_pos_pitch_p1.y;
            move_pitchs(x_pp1,y_pp1,decimal);
            getcursorpos(last_pos_pitch_p1);
          end;

      // Controla o Pitch do Player2
      if loop = 8 then
        if decimal <> dec_pp2 then
          begin
            dec_pp2 := decimal;
            x_pp2 := last_pos_pitch_p2.x;
            y_pp2 := last_pos_pitch_p2.y;
            move_pitchs(x_pp2,y_pp2,decimal);
            getcursorpos(last_pos_pitch_p2);
          end;
    loop := loop + 1;
    end;
  identificador := 20808;
end;
```

Conforme o valor da variável *loop*, é disparada uma *procedure* na qual são passados os parâmetros *coord_x*, *coord_y* e *decimal*, onde *coord_x* e *coord_y* são as coordenadas x e y onde se localiza o *trackbar* (controle deslizante) no Virtual Turntables que se deseja mover e, *decimal* é o último resultado da conversão feita pelo ADC0808. Esta *procedure* assim como as demais *procedures* responsáveis por movimentar os *trackbars* do Virtual Turntables (volumes do mixer, volumes dos equalizadores e *crossfade*) têm o objetivo de simular o *click* do *mouse* sobre o *slider* do *trackbar* e posicionar o *slider* em uma nova posição conforme o valor da variável *decimal*. A seguir há uma parte do código fonte da *procedure* responsável por mover os controles do *pitch* presentes nos *players*:

```
procedure move_pitchs(coord_x,coord_y,decimal:integer);
begin
  if decimal < 65 then
    if decimal = 0 then
    begin
      setcursorpos(coord_x,coord_y);
      mouse_event(mouseeventf_leftdown,coord_x,coord_y,0,0);
      setcursorpos(coord_x,coord_y);
      mouse_event(mouseeventf_leftup,coord_x,coord_y,0,0);
    end;
```

```
if decimal = 1 then
begin
  setcursorpos(coord_x,coord_y);
  mouse_event(mouseeventf_leftdown,coord_x,coord_y,0,0);
  setcursorpos(coord_x,353);
  mouse_event(mouseeventf_leftup,coord_x,353,0,0);
end;

if decimal = 2 then
begin
  setcursorpos(coord_x,coord_y);
  mouse_event(mouseeventf_leftdown,coord_x,coord_y,0,0);
  setcursorpos(coord_x,352);
  mouse_event(mouseeventf_leftup,coord_x,352,0,0);
end;

if decimal = 3 then
begin
  setcursorpos(coord_x,coord_y);
  mouse_event(mouseeventf_leftdown,coord_x,coord_y,0,0);
  setcursorpos(coord_x,351);
  mouse_event(mouseeventf_leftup,coord_x,351,0,0);
end;

if decimal = 4 then
begin
  setcursorpos(coord_x,coord_y);
  mouse_event(mouseeventf_leftdown,coord_x,coord_y,0,0);
  setcursorpos(coord_x,350);
  mouse_event(mouseeventf_leftup,coord_x,350,0,0);
end;
.
.
.
```

O que difere esta *procedure* das *procedures* responsáveis pelo movimento do *crossfade*, dos controles de volume do *mixer* e dos equalizadores são os valores das coordenadas x e y.

8 CONCLUSÕES

A porta paralela está presente nos PC's padrão IBM desde o início da década de 80, ou seja, ela está no mercado a mais de 20 anos, e durante este período houve aprimoramentos da porta paralela, o que ocasionou um aumento de desempenho significativo na taxa de transferência e a possibilidade da comunicação bidirecional através da mesma.

Muitas pessoas e autores, principalmente os que estudam e apóiam novos meios de comunicação entre o PC e dispositivos externos, como por exemplo, a tecnologia USB, afirmam que a porta paralela está com seus dias contados. Para comprovar que esta afirmação não reflete a realidade, vamos fazer a seguinte analogia: quando as impressoras a jato de tinta se difundiram no mercado, houve rumores de que as impressoras matriciais seriam extintas, porém elas têm sua fatia de mercado até hoje, e mais, na grande maioria das impressoras matriciais, o modo de comunicação entre a impressora e o PC é através da porta paralela.

O que já vem ocorrendo é uma redução de dispositivos lançados no mercado que utilizam a porta paralela, principalmente os dispositivos que necessitam de alta taxa de transferência, exemplos típicos seriam as impressoras, *scanners* e unidades de armazenamento (unidades de disco externo, cartões de memória, etc.).

Um problema encontrado no desenvolvimento do protótipo do *software* é o fato do Delphi 5 não possuir uma função para acessar a porta paralela do PC, para a realização desta tarefa é necessário utilizar linhas de código em *assembly*, o que considero uma grande desvantagem quando é necessário desenvolver aplicações que acessem dispositivos através da porta paralela. Comparando o Borland Delphi 5 com outras linguagens de programação, como por exemplo, a linguagem C e Visual Basic, as quais são amplamente difundidas, possuem funções para acessar a porta paralela através da programação em alto nível. Inclusive o Borland Pascal também possui uma função para facilitar o acesso à porta paralela, esta função deveria ter sido mantida e até mesmo aprimorada nas versões mais recentes das ferramentas para desenvolvimento da Borland.

Outra questão referente ao desenvolvimento do protótipo do *software*, ao que se refere à comunicação com a API do Windows, é a falta de livros atuais que tratem especificamente sobre este assunto, pois o livro utilizado como principal fonte para este assunto foi publicado em 1991. Os exemplos encontrados na internet para o envio de mensagens para componentes que são utilizados pelo Virtual Turntables, mais especificamente para o *trackbar* são muito simplórios, o que gerou muita dificuldade neste ponto, sendo que a principal referência utilizada foi o arquivo de ajuda disponibilizado juntamente com o Delphi 5, denominado *Win32 Developer's References.*

Ao se tratar da construção do *hardware*, mais precisamente da placa principal, devido à confecção da mesma ter sido de forma artesanal, no momento de soldar os soquetes dos circuitos integrados, não foi possível soldar a face de cima da placa de circuito impresso, pois não era possível alcançar o ponto em que deveria ser feito a solda na face de cima com o ferro de solda devido à presença do soquete, para contornar esta situação, foi necessário refazer as ligações da face de cima da placa de circuito impresso na parte de baixo, utilizando-se *jumpers* para isto.

Uma conclusão que pude tirar referente ao software utilizado para simulações do circuito a ser desenvolvido, o Circuit Maker 2000, é que não podemos confiar cegamente nos resultados obtidos em simulações feitas através dele, pois no momento em que é passado da simulação para a realidade, não é sempre que o resultado obtido na simulação é alcançado na construção do circuito, pois durante a simulação não são considerados fatores como: quedas de tensão, ruídos, consumo dos componentes envolvidos, oscilação na tensão fornecida pela fonte, entre outros. Estes são fatores que costumam ocorrer na construção de um circuito e que normalmente são difíceis de contornar, principalmente a questão de ruídos.

O fato do ADC0808 ter resolução de oito *bits*, considero ser uma resolução baixa na qual não há a mesma precisão que há nos equipamentos tradicionais de mixagem. Este fator aliado à utilização de potenciômetros com baixa precisão, podem causar oscilações na leitura do ADC, e tende a agravar ainda mais a precisão do equipamento desenvolvido.

A comunicação através da porta paralela atende muito bem a este tipo de integração entre *hardware* e *software*, principalmente pelo desempenho obtido, sendo que do ponto de vista do usuário, é praticamente instantânea a ação exercida no *hardware* e a reação do *software* referente a esta ação, que para a finalidade do *hardware* desenvolvido é fundamental o tempo de resposta. Outro fator de grande relevância é o custo para a confecção do *hardware*, o qual é demonstrado na tabela 2.

Tabela 2 – Custo para o desenvolvimento do *hardware*

DESCRIÇÃO DO PRODUTO	***VALOR UNITÁRIO***	***QUANTIDADE***	***TOTAIS***
Soquete para circuito integrado - 8 pinos	U$ 0,04	1	U$ 0,04
Soquete para circuito integrado - 14 pinos	U$ 0,06	2	U$ 0,12
Soquete para circuito integrado - 16 pinos	U$ 0,08	5	U$ 0,40
Soquete para circuito integrado - 20 pinos	U$ 0,10	5	U$ 0,50
Soquete para circuito integrado - 28 pinos	U$ 0,13	4	U$ 0,52
Diodo de Silício - 1n4148	U$ 0,02	64	U$ 1,28
Circuito integrado ADC0808N	U$ 7,50	2	U$ 15,00
Circuito integrado LM555	U$ 0,32	1	U$ 0,32
Circuito integrado SN74LS04	U$ 0,22	2	U$ 0,44
Circuito integrado SN74LS138	U$ 0,29	1	U$ 0,29
Circuito integrado SN74LS192	U$ 2,30	2	U$ 4,60
Circuito integrado SN74LS373	U$ 0,74	3	U$ 2,22
Circuito integrado SN74LS541	U$ 1,09	1	U$ 1,09
Chaves	U$ 0,09	19	U$ 1,71
Potenciômetros deslizantes	U$ 0,92	3	U$ 2,76
Potenciômetros convencionais	U$ 0,44	9	U$ 3,96
Conectores de 10 vias	U$ 0,17	10	U$ 1,70
Placa Universal 10cm x 10cm	U$ 2,70	3	U$ 8,10
Placa de Fenolite 30cm x 20cm	U$ 8,70	1	U$ 8,70
Capacitores	U$ 0,10	2	U$ 0,20
Resistores	U$ 0,03	10	U$ 0,30
Multímetro Digital	U$ 19,00	1	U$ 19,00
Alicates de Corte e Bico Fino	U$ 9,35	1	U$ 9,35
Kit p/ confecção de placas (solda, ferro de soldar, percloreto de ferro, sugador)	U$ 16,00	1	U$ 16,00
Cabo Flat 10 vias (metro)	U$ 0,60	2	U$ 1,20
TOTAL GERAL			***U$ 99,80***

Nesta planilha não estão inclusos os custos relacionados à pesquisa na internet, impressão de documentos diversos e *softwares* utilizados, tanto que os *softwares* utilizados, com exceção do Delphi 5 são *shareware* o que possibilita uma grande economia referente à aquisição de *softwares*.

8.1 EXTENSÕES

Para trabalhos futuros, é sugerido:

a) elaboração de um componente para o Delphi 5 que permita a manipulação da porta paralela de forma intuitiva;

b) inclusão de novas funcionalidades ao protótipo, de forma que seja possível controlar todas as funções do Virtual Turntables a partir do protótipo, como por exemplo: o botão *Jog Whell*, o botão *Eject*, entre outros;

c) inclusão de *displays* de cristal líquido para cada um dos *players*, para que sejam exibidas no *hardware* as informações referentes ao arquivo em execução no *player*, e com isto tornar a comunicação entre o *hardware* e o *software* bidirecional;

d) implementação de obtenção automática dos *handles* dos objetos a serem manipulados;

e) criação de um frontal para o protótipo, tornando a manipulação do *hardware* mais amigável;

f) integração dos CI’s da família TTL utilizados neste protótipo utilizando a tecnologia FPGA.

REFERÊNCIAS BIBLIOGRÁFICAS

ALTIUM, **Altium: Making Electronics Design Easier**. Disponível em: <http://www.altium.com>. Acesso em: 03 dezembro 2003.

AXELSON, Jan. **Making Printed Circuit Boards**. New York: TAB Books, 1993.

AXELSON, Jan. **Parallel Port Complete:** Programming, Interfacing & Using the PC's Parallel Printer Port. Madison: Lakeview Research, 2000.

BERGSMAN, Paul. **Controlling the World with your PC**. San Diego: Hightext, 1994.

CADSOFT, **CadSoft Online: Home of the EAGLE Layout Editor**. Disponível em: <http://www.cadsoftusa.com>. Acesso em: 03 dezembro 2003.

CARROT INNOVATIONS. **Homepage official**. Disponível em: <http://www.carrotinnovations.com>. Acesso em: 28 novembro 2003.

DIAL ELECTRONICS DATASHEETS, **SN74LS192 Datasheet**, 2003. Disponível em: <http://www.dialelec.com/714.html>. Acesso em: 28 novembro 2003.

ENTECH TAIWAN. **TVicLPT – The toolkit for Directly Accessing LPT Ports From Win32 Applications**. Disponível em: <http://www.entechtaiwan.com/tviclpt.htm>. Acesso em: 03 dezembro 2003.

FAIRCHILD SEMICONDUCTOR, **DM74LS138 Datasheet**, 2003. Disponível em: <http://www.fairchildsemiconductor.com/pf/DM/DM74LS138.html>, Acesso em: 28 novembro 2003.

GREGO, Maurício. Diversão é aqui. **Info Exame**, São Paulo, v. 201, p. 48-66, dezembro 2002.

GUIA DO MP3. **Info Exame**, São Paulo, 2002. Edição 194-A

MÚSICA NO MICRO. **Info Exame**, São Paulo, 2003. Edição 206-A

NATIONAL SEMICONDUCTOR, **ADC0808 Datasheet**, 2003. Disponível em: <http://www.national.com/pf/AD/ADC0808.html>. Acesso em: 28 novembro 2003.

NATIONAL SEMICONDUCTOR, **LM555 Datasheet**, 2003. Disponível em: <http://www.national.com/pf/LM/LM555.html>. Acesso em: 28 novembro 2003.

PACHECO, Xavier; TEIXEIRA, Steve. **Delphi 5 developer's guide.** Indianapolis: Sams, 1998.

ROGERCOM, **Pesquisa e Desenvolvimento**, 2003. Disponível em: <http://www.rogercom.com>. Acesso em: 28 novembro 2003.

SALIBA, Walter Luiz Caram. **Técnicas de Programação,** uma abordagem estruturada. São Paulo: McGraw-Hill do Brasil, 1992.

SUCHEUSKI, Maurício.. **Desenvolvedor Profissional,** Delphi 3.0. Curitiba: Lísias Editora, 1998.

TAUB, Herbert. **Circuitos digitais e microprocessadores.** São Paulo: McGraw-Hill do Brasil, 1984.

TEXAS INSTRUMENTS, **SN74LS04N datasheet**, 2003. Disponível em: <http://focus.ti.com/docs/prod/folders/print/sn74ls04.html>. Acesso em: 28 novembro 2003.

TEXAS INSTRUMENTS, **SN74LS373N Datasheet**, 2003. Disponível em: <http://focus.ti.com/docs/prod/folders/print/sn74ls373.html>. Acesso em: 28 novembro 2003.

TOCCI, Ronald J.; WIDMER, Neal S. **Sistemas Digitais:** Princípios e Aplicações. São Paulo: Prentice Hall, 2003.

TTL DATABOOK MOTOROLA, **SN74LS540 Datasheet**, 1997. Disponível em: <http://noel.feld.cvut.cz/semi/motorola/html/dl121_index.htm>. Acesso em: 28 novembro 2003.

VISIOSONIC. PCDJ. Florida, 2003. Disponível em <http://www.visiosonic.com>. Acesso em: 28 novembro 2003.

WILKEN, Peter; HONEKAMP, Dirk. **Windows System Programming:** Program your Windows applications faster and easier. Duesseldorf: Abacus, 1991.