Sistema de Aquisição de Dados via USB usando Interface Genérica de Áudio

Jader Garcia da Silveira[1], Rômulo Silva de Oliveira[1]

[1]Dep. de Automação e Sistemas – Universidade Federal de Santa Catarina (UFSC)
Caixa Postal 476 – 88040-900 – Florianópolis – SC – Brasil

jadergs@gmail.com, romulo@das.ufsc.br

Abstract. *USB (Universal Serial Bus) is one of the most popular interface standards for external computer devices. Among USB possible uses we have the connection between real-time data acquisition boards and desktop computers. This paper shows how it is possible to achieve data transference between device and computer up to 890 kbytes/s by making careful use of the generic device-drivers originally developed for audio devices. At the same time, function HID was used to support the exchanging of control data between the computer and the data acquisition board.*

Resumo. *A USB (Universal Serial Bus) é um dos padrões de interface para periféricos externos ao computador mais populares já criados. Entre os usos possíveis da USB, está a conexão de placas de aquisição de dados em tempo real a computadores convencionais. Este artigo mostra como é possível obter uma transferência de dados entre dispositivo e computador de até 890 kbytes/s através do bom aproveitamento dos drivers genéricos para dispositivos de som. Em paralelo, a função HID foi usada para suportar que alguns dados de controle sejam trocados entre o computador e a placa de aquisição de dados.*

1. Introdução

A USB (Universal Serial Bus) é um dos padrões de interface para periféricos externos ao computador mais populares já criados. Ela interconecta um mestre chamado de host e escravos que são os dispositivos ou periféricos.

Entre os usos possíveis da USB, há a conexão de placas de aquisição de dados em tempo real a computadores convencionais. Por exemplo, no contexto dos processos de soldagem, no qual este trabalho originou-se, busca-se uma solução para aquisição de dados que consiga operar nas velocidades necessárias, disponibilizando os dados em um computador convencional, onde ferramentas gráficas e estatísticas podem ser usadas posteriormente na análise dos dados coletados. Uma solução típica inclui uma placa de aquisição de dados microprocessada, colocada junto ao equipamento de soldagem, a qual inclui transdutores e realiza condicionamento de sinal e linearizações, além de possuir a capacidade de comunicação via USB. Também inclui um computador convencional, executando o sistema operacional Windows.

O objetivo deste artigo é descrever o funcionamento da USB no contexto de promover um meio de comunicação direto entre computador convencional e placa de aquisição de dados. Essa comunicação, no caso de um fluxo contínuo, deve alcançar

taxas de transferência de, no mínimo, 100 kbytes/s. O foco principal deste trabalho foi a utilização de device drivers nativos do sistema operacional, especificamente do sistema de som, no desenvolvimento de uma solução capaz de realizar aquisição de dados em tempo real via interface USB. O foco principal deste trabalho foi a utilização das interfaces genéricas do USB para a conexão de periféricos de áudio no desenvolvimento de uma solução capaz de realizar aquisição de dados em tempo real via interface USB. O objetivo inicial do sistema era a aquisição de dados em processos de soldagem a arco. Entretanto, houve também a implementação de uma função bidirecional de troca de dados intermitentes entre computador e placa de aquisição de dados, através do USB.

2. Sistema de Aquisição de Dados para Processos de Soldagem

O conceito atual de soldagem é baseado apenas na união de peças, assegurando na junta, a continuidade das propriedades físicas e químicas. Para ocorrer a soldagem, é preciso transferir energia para a zona da junta do material de base. Com metais, isso pode ser feito com arco voltaico (ou arco elétrico) [CIMM 2007].

Arcos voltaicos são estabelecidos por meio de um circuito elétrico que é fechado no momento em que o eletrodo encosta no material de base. Estabelecendo-se um arco desse tipo, em sua extensão é possível medir tensões elétricas que variam entre 0 e 40 V, dependendo do processo. Tensão elétrica provavelmente é a principal variável de medida em um processo de soldagem a arco, pois muitas observações e conclusões podem ser feitas a partir dos gráficos gerados. Uma demonstração disso é o fato de a tensão aumentar quase em proporção direta com o comprimento de arco, praticamente sem se importar com a corrente, pois as maiores quedas de tensão residem nas interfaces entre o meio metálico e o ar, o que é tema para uma discussão mais profunda. Essa proporcionalidade permite utilizar o próprio arco como sensor de altura, por exemplo.

3. USB - Universal Serial Bus

A USB, sigla para Universal Serial Bus, é o padrão de interface para periféricos externos ao computador provavelmente mais popular dos já criados. Um sistema USB é composto por hardware mestre e escravo. O mestre é chamado de host e o escravo denomina-se dispositivo ou simplesmente periférico. Todas as transferências USB são administradas e iniciadas pelo host. Mesmo que um dispositivo queira enviar dados, é necessário que o host envie comandos específicos para recebê-los.

Há três classes de dispositivos USB: low-speed, full-speed e high-speed. As duas primeiras pertencem às especificações 1.x e a última à 2.0. Há uma pequena diferença entre os dispositivos high-speed e os anteriores que é a divisão de tempo. Os primeiros utilizam quadros divididos em 1 ms, chamados de frame enquanto o último usa uma base de 125 ms denominada de microframe. As velocidades máximas de transferência para cada tipo de dispositivo são: 1,5 Mbits/s (low-speed), 12 Mbits/s (full-speed) e 480 Mbits/s (high-speed). Low-speed (1,5 Mbits/s), Full-speed (12 Mbits/s) e High-speed (480 Mbits/s). [Axelson 2005]

3.1. Comunicações USB

As comunicações USB podem ser classificadas em dois tipos principais: as de aplicação e as de enumeração. As trocas de dados de aplicação são o verdadeiro papel de um meio de conexão de dispositivos. É a forma de comunicação que se espera ocorrer quando um

periférico é desenvolvido e é somente ela a utilizada depois que o periférico USB está devidamente preparado pelo sistema operacional.

A fase de preparação, conhecida como enumeração, a grosso modo, acontece logo depois de quando o dispositivo USB é fisicamente conectado ao computador. Nesse momento, o sistema operacional realiza vários pedidos ao dispositivo para que suas características de funcionamento sejam reconhecidas. O sistema operacional, com a obtida noção do periférico USB, atribui-lhe um endereço e seleciona a configuração mais apropriada de acordo com certos critérios. Com mensagens de confirmação do dispositivo indicando que essas duas últimas operações foram corretamente aceitas, a enumeração é finalizada e o sistema fica pronto para o uso.

A especificação USB define quatro tipos de transferências USB: de controle (control), interrupta (interrupt), isócrona (isochronous) e massiva (bulk). As transferências de controle servem normalmente para as fases de SETUP, STATUS e CONFIGURAÇÃO do dispositivo USB. Na prática, além das funções mais importantes passarem por esse canal, é por onde as configurações e interfaces são selecionadas.

Transferências interrupta são geradas pelo dispositivo. O dispositivo deve requerer atenção do host para poder começar a enviar os dados. O tempo entre essa requisição de atenção e a resposta do host, configurado no descritor USB, é limitado a 255 ms nos dispositivos low-speed e full-speed. A interface HID deste trabalho utiliza transferência interrupta para enviar dados ao host.

Isócrona: é o tipo de transferência utilizada para fluxos contínuos. Ela dispensa retransmissões (em caso de erro) e é executada periodicamente gerando velocidade praticamente constante. Essa taxa quase constante de transmissão é possível devido à garantia de largura de banda que existe para as isócronas, por responsabilidade do sistema operacional. Um fato importante para este trabalho é que, mesmo que não haja correções de erros de transmissão, eles são detectáveis através de CRC.

Massiva: esse é o tipo de transferência que pen-drives, impressoras, scanners, câmeras digitais, entre outros, utilizam. As conexões massivas, como diz o próprio nome, servem para os maiores volumes de dados da USB. Contudo, devido à ausência de garantia de largura de banda, a velocidade varia bastante.

A largura de banda é determinada pelo host. Na fase de enumeração, 90% da banda é reservada para as transferências periódicas (vide interrupta e isócrona) e 10% é disputada entre as transferências de controle e massiva. Isso em full-speed. Em high-speed, a divisão é de 80% para periódicas e 20% para o resto.

Por haver dispositivos bastante comuns, tal como o caso das impressoras, foram criadas as chamadas “Classes de Dispositivos USB” que, a princípio, serviriam para fornecer drivers genéricos para os fabricantes, em vez de todos precisarem escrever novos drivers para cada periférico. Há especificações para algumas classes, mas a disponibilidade dos drivers genéricos depende do sistema operacional. As classes de dispositivos USB previstas são para áudio, smart card, comunicações, segurança de conteúdo, atualização de firmware de dispositivo, dispositivo de interface humana (HID), ponte IrDA, armazenamento em massa, impressora, scanner e câmera, teste e medida e vídeo.

A sigla para Human Interface Device é a classe USB que foi criada para lidar com periféricos de entrada tais como mouses e teclados. A HID também pode enviar comandos aos periféricos, tal como num joystick que possua force-feedback. Dispositivos HID não necessariamente precisam ser interfaces humanas; apenas devem seguir o padrão USB para funcionar. Todos os dados compõem-se de estruturas denominadas reports. O host envia e recebe dados estruturados nessa forma através de transferências de controle e interruptas. A velocidade máxima de transferência é de 800 bytes/s para dispositivos low-speed, 64 kbytes/s para full e 24 Mbytes/s para high.

4. Considerações sobre Emprego de USB em Sistemas de Aquisição de Dados

Os drivers são responsáveis pela abstração de comandos e dados, normalmente de dispositivos físicos, para que aplicativos possam manipulá-los independentemente de conhecimento do hardware. De mesma forma, para acessar dispositivos USB, cada um deve ter um driver correspondente. Normalmente, para alguns dispositivos conhecidos, o sistema operacional provê drivers próprios dos fabricantes ou genéricos. Exemplos são os conhecidos pendrives que, do ponto de vista do sistema operacional são tratados como dispositivos de armazenamento em massa (mass-storage device) genéricos, dispensando a necessidade de instalações com intervenção do usuário.

Já aqueles dispositivos que não se encaixam em classes genéricas, ou que forneçam recursos não previstos pelo sistema operacional, devem ser acompanhados de drivers específicos. Neste caso, por exemplo, o descritor de dispositivo da USB tem o código 0xFF no campo bDeviceClass para indicar ao sistema que um driver específico precisa ser instalado. Assim, o descritor de configuração pode ser escrito mais facilmente e personalizado de acordo com o fabricante do dispositivo, mas ainda assim respeitando o padrão USB.

É necessário decidir qual tipo de driver utilizar para acessar a interface USB com o propósito de aquisição de dados em tempo real. Essa decisão considera vários fatores, como o tipo de transferência USB (isócrona, de controle, interrupta ou massiva), o firmware da placa de aquisição de dados e o aplicativo no computador. Utilizar um driver HID é bastante conveniente devido a sua disponibilidade nativa no sistema operacional, mas insuficiente em velocidade para a aquisição de dados em tempo real. Por outro lado, a conexão isócrona da USB possui características adequadas para o objetivo considerado, mas falta um driver para ela. [EMBEDDEDRELATED 2007]

Existem soluções de drivers genéricos para qualquer dispositivo USB. Uma é o USBIO [THESYCON 2007] da empresa Thesycon Software & Services GmbH e a outra é o Lib-USB [LIBUSB 2007]. Os drivers genéricos costumam ser compostos de um arquivo .SYS, que é o driver propriamente dito, e outro .DLL, que disponibiliza os recursos da camada de driver para os aplicativos. O Lib-USB, de distribuição livre, existe em versões para Linux e para Windows, porém não suporta conexões USB isócronas. O USBIO é pago e custa 2500 euros.

O uso de drivers genéricos apresenta vantagens em relação a drivers personalizados que bem se adequam ao trabalho desenvolvido:

• independência de plataforma pois existem drivers para classes genéricas;

• 100% Plug-and-Play pois não há intervenção do usuário;

• o tempo de desenvolvimento do dispositivo é reduzido por eliminar a necessidade de criar um novo driver;

• a cada nova versão do Windows, o driver já será digitalmente assinado;

• já existe documentação sobre a utilização dos drivers genéricos, principalmente para classes totalmente arraigadas no meio computacional, como áudio; e

• permite que os desenvolvedores focalizem-se no dispositivo e no aplicativo do computador em vez de se preocupar com o meio de comunicação (no caso, USB).

Para a parte de comando digital, já estava definido, desde o princípio, que uma interface HID seria adequada. O sistema operacional já oferece suporte a ela. Já para a aquisição, desejava-se transferência USB isócrona devido ao seu poder de garantia de velocidade. As classes de dispositivos que se encaixam nesse perfil isócrono são as de áudio e de vídeo. Preferiu-se a de áudio porque a de vídeo não seria suportada em algumas versões do Windows. Com isso em mente, bastaria escrever um descritor USB que se apresentasse como um microfone (Figura 1).

Isto feito, conseguiu-se um dispositivo USB de duas interfaces: uma HID e outra de ÁUDIO. Assim, o dispositivo é totalmente Plug-and-Play. A princípio, deve ser compatível com todas as versões do Windows a partir do 98. O que restou era realizar testes preliminares para se assegurar de que utilizar drivers genéricos de áudio do sistema operacional garantissem perfeição ao conduzir os dados da placa de aquisição de dados para o aplicativo no computador. A ponte entre a USB e a camada de aplicação é formada por uma estrutura de drivers, os quais são chamados de “Kernel-Mode WDM5 Audio Components” de acordo com a Microsoft, mas nem todos são sempre utilizados ([MSDN2007a], [MSDN 2007b], [MSDN 2007c]).

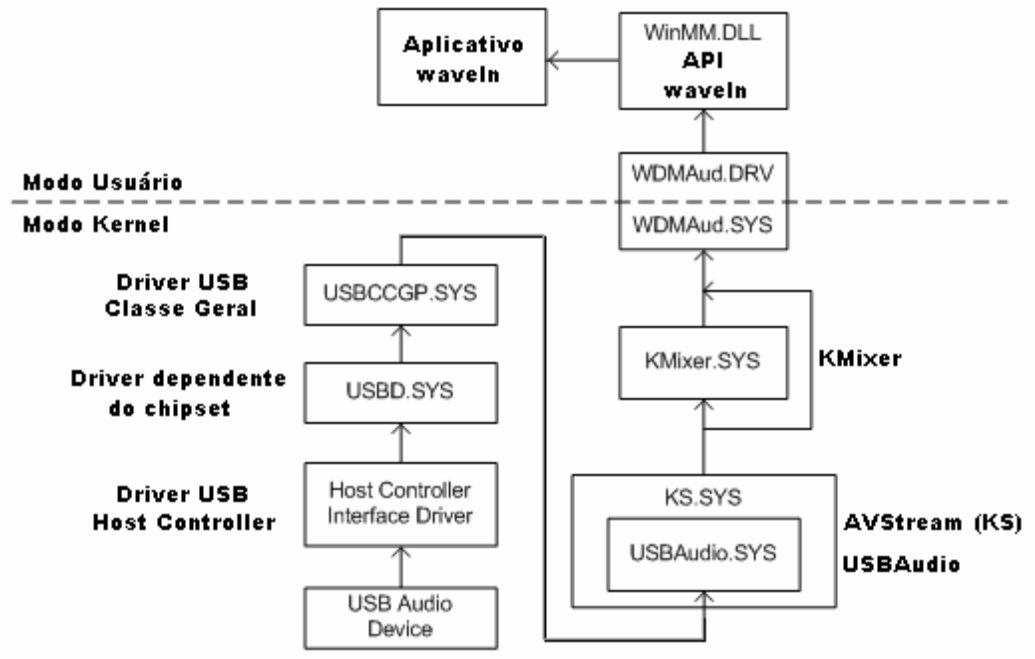

Figure 1. Estrutura dos drivers de áudio USB

O importante é que a captura de “som” via USB não passe pelo KMixer, o que já é feito por padrão. Esse driver faz a mistura de canais de fluxo de mídia do sistema operacional e poderia ocasionar distorção dos dados, o que é inadmissível tratando-se de um sistema de aquisição. Para reprodução de som, a passagem pelo KMixer é obrigatória. Com o DirectX, é possível pular o KMixer. Deve-se, então, haver atenção a esse detalhe caso se deseje implementar um canal endpoint de controle de sinais via sistema de som.

5. Configuração da USB para Aquisição de Dados

Com o objetivo de completar o sistema de aquisição, foi criado um software para explorar, via USB, os recursos disponibilizados pela placa de aquisição de dados. Como já foi mencionado, além da função de aquisição, uma outra, de trocas de dados intermitentes (HID), foi também desenvolvida.

A princípio, o software deveria utilizar somente a função de aquisição para gerar gráficos na tela. Entretanto, houve a possibilidade de implementar o código de acesso à interface HID, então ambas as funções da placa de aquisição de dados puderam ser exploradas no aplicativo. Com a função HID, o programa poderia prover maneiras de enviar e receber dados esporádicos, tais como botões liga-desliga e LEDs, se fosse o caso, para alterar e obter estados da placa de aquisição de dados.

5.1. Aquisição de Dados

Neste trabalho, quando se refere à parte do sistema de aquisição no computador, fala-se sobre sistema de som ou sistema de áudio. Ao mencionar essas expressões, deve-se diretamente pensar no acesso ao dispositivo USB. Como já discutido, a estrutura de multimídia do Windows foi utilizada como canal entre o hardware USB e a camada de aplicativo. Então, imaginando-se um mapa entre o sistema de aquisição e o sistema de som, as funções do programa resumem-se em: acessar dispositivos de som; obter os dados de som; desenhar gráficos na tela. Para implementar essas funções no computador nenhuma biblioteca nova precisa ser utilizada. Os sistemas de som são muito difundidos no meio computacional.

5.2. Operação do Sistema de Som

O funcionamento básico de gravação de som no Windows é de forma assíncrona, ou seja, não depende estritamente do tempo, mas sim da quantidade de dados que se obtém. Depois de acessar o dispositivo, o Windows aciona o sistema de som para iniciar a recepção dos dados representativos de amplitude de som e colocá-los em buffers.

Quando um buffer estiver carregado, o Windows chama o aplicativo e provavelmente haverá algum processamento com esses dados. Há algumas maneiras de chamar o aplicativo para avisá-lo de que um buffer está pronto. A opção mais simples é o programa ler um flag ligado pelo Windows até que um buffer fique carregado. Essa opção certamente provoca alta carga de CPU, o que é completamente indesejado já que se trata de um sistema de aquisição com restrições temporais. A segunda opção é registrar o aplicativo como interceptador de mensagens do Windows

A terceira opção, a mais comum a partir dos Windows de 32 bits, consiste em criar uma thread suspensa com a lógica de processamento de som. A thread é ativada

quando o Windows dispara um evento (CALLBACK) avisando o aplicativo de que o buffer está pronto. Ela realiza as tarefas com aquele buffer e entra em suspensão à espera de novos eventos. O aplicativo fica livre, sem processamento, enquanto o Windows não o chama novamente. Com disparada vantagem, esta última opção foi a utilizada. Uma explicação visual do que a terceira opção faz é sintetizada na Figura 2.

O aplicativo deve apenas iniciar a gravação, esperar pelas chamadas de sistema e desenhar os gráficos. A tarefa de obtenção dos dados da placa de aquisição de dados é totalmente do sistema operacional aliado aos drivers do sistema de áudio.

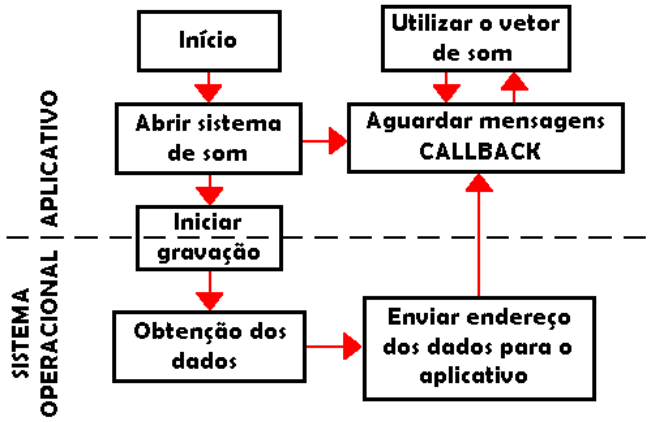

Figure 2. Resumo de funcionamento da captura de dados

O procedimento de captura exige que pelo menos um buffer seja entregue ao sistema operacional (enquanto outra fica livre) para que, depois de preenchido, os dados sejam transferidos para o aplicativo. A operação acontece de forma cíclica. Um dos buffers sempre deve estar livre para que a thread possa lê-lo e desenhar os gráficos. Depois disso e do sistema operacional entregar de volta um buffer cheio, aquele que estava em uso pela thread é preparado e adicionado no lugar do anterior. A solução adotada para dividir as tarefas de obtenção de dados e de desenho dos gráficos, através de thread, exige a utilização de dois deles, no mínimo. O sistema de som admite incontáveis buffers, mas há de se ponderar entre o consumo de memória RAM e o desempenho de captura. Realizou-se medições nesse aspecto e os procedimentos e resultados estão na próxima seção.

6. Medições no Protótipo

Tratando-se de um sistema de aquisição de dados em tempo real, com o qual pesquisas serão realizadas e inferências teóricas sobre fenômenos diversos serão feitas baseadas nos sinais registrados, ainda era necessário executar a etapa de testes e de validação dos resultados para corrigir erros de planejamento e de implementação e para levantar aspectos relativos às limitações de operação. Uma razão a mais para realizar testes sobre o sistema de aquisição é a sua forma de implementação, a qual foi feita utilizando os drivers de dispositivos de som nativos do sistema operacional, o que é incomum.

Os testes estiveram focalizados no percurso desde um sinal aplicado na placa de aquisição de dados até a exibição das amplitudes na tela do computador, visando observar três aspectos: tempo real de atualização de telas; erros grosseiros de captura; e continuidade de telas. A imagem obtida do aplicativo de aquisição desenvolvido neste

projeto mostra o resultado que se esperava e o que se obteve com um sinal de onda quadrada de 5 kHz injetado na entrada analógica da placa de aquisição de dados.

Caso o único fator limitante de velocidade de transferência de dados fosse a USB, poder-se-ia realizar aquisições com volume de dados no total de aproximadamente 800 kbytes/s. A velocidade máxima alcançada com a conexão isócrona USB chegou a 890 kbytes/s, mas essa é uma taxa sem garantia. Para 800 kbytes/s, então, uma aquisição de 4 canais a 16 bits poderia ser feita a 100 kHz.

A lógica de funcionamento do sistema de aquisição é baseada nas mensagens CALLBACK efetuadas pelo sistema operacional com destino para o aplicativo de visualização de dados. A USB recebe os dados da placa de aquisição de dados, entrega para os drivers de som que, por vez deles, despacham vetores para o aplicativo. Esses vetores têm tamanhos definidos pelo usuário do aplicativo, escolhendo-os de acordo com a largura da janela de tempo desejada para observar na tela.

Uma das dúvidas surgidas com a implementação da tela de visualização dos gráficos de aquisição era sobre a continuidade entre as telas. Ora, cada tela é um vetor. No fundo queria-se ter certeza de que todos os vetores fossem concatenáveis porque, apesar de uma imagem da tela representar corretamente um certo tempo de aquisição, é necessário, além disso, unir todos os vetores representativos das telas em um único para haver sua posterior gravação em arquivo.

Deve-se ressaltar que esses mencionados vetores são os buffers do sistema de som, e não da USB. A USB em conexão isócrona sempre transmite o mesmo número de bytes em um frame, apesar disso não ocorrer em todos os frames. Os buffers de recepção do sistema de som são independentes da USB e, inclusive, possuem tamanhos e tratamentos diferentes. Partiu-se do pressuposto, baseado em testes preliminares, de que a integridade de dados era garantida se dependesse apenas da USB. Caso houvesse descontinuidade de telas, provavelmente seria responsabilidade do sistema de som. O teste seria bem sucedido se a condição de continuidade de telas fosse satisfeita.

Visando-se testar o sistema, algumas modificações tiveram de ser implementadas. Na placa de aquisição de dados o programa envia um padrão fixo e praticamente contínuo de dados. No aplicativo a condição para o êxito no teste é verificada. Foi incluída uma opção para alterar a quantidade de buffers porque, apesar desse valor ser ajustável, assumira-se que 2 era suficiente devido ao costume nessa escolha na maioria dos softwares de acesso ao sistema de som. Ao realizar os primeiros testes, percebeu-se que a suposição de que 2 buffers era suficiente estava incorreta.

6.1. Testes

A configuração do computador onde os testes ocorreram é: Processador AMD Sempron 64 2800+, 1,6 GHz; Memória RAM: 512 Mbytes; Sistema operacional: Windows XP com Service Pack 2. Os primeiros testes mostraram que a condição de continuidade de telas já falhava. O primeiro exemplo foi uma aquisição com janela de tempo de 500 ms e com 2 buffers de som, no qual 400 aquisições foram perdidas.

Percebeu-se que quanto menor o tamanho do buffer, mantendo-se a quantidade deles, maior era a diferença. Esse fato conduziu à suposição de que a perda dos dados estaria associada a algum excesso de carga de processamento em algum trecho no caminho percorrido pelos dados nos drivers de som. Por conseqüência, de alguma

forma, o número de buffers do sistema de som também teria alguma relação. O número de buffers define quantos vetores comporão um buffer circular maior, que o sistema de som utiliza para compartilhar os dados com a thread de montagem do gráfico do aplicativo. Pelo mesmo motivo de compartilhamento, o número mínimo é 2. Com isso, entende-se a razão de haver aumento de perdas de dados com a diminuição da janela de período: o sistema de som fica incapaz de passar para o aplicativo todos os dados em tempo hábil antes de reescrever um dos buffers com novos dados da USB.

Desse modo, supõe-se que basta utilizar um número adequado de buffers para evitar qualquer perda de dados. Realizando-se testes com números maiores, chegou-se a conclusão de que isso é suficiente. Objetivando-se a análise da influência do número de buffers e do tamanho da janela de tempo de exibição de gráfico sobre a perda de dados, foi organizado um pequeno ambiente de testes com 24 cenários. Foram combinadas janelas de tempo com 10 ms, 50 ms, 100 ms, 500 ms, 1 s e 5 s com buffers na quantidade de 2, 4, 8 e 16. Em cada cenário, a aquisição foi executada durante mais de 1 minuto para, primeiramente, avaliar duas situações: Perdas de dados SEM intervenção de usuário (aplicativo fica em primeiro plano) e perdas de dados COM intervenção de usuário (aplicativos paralelos, troca de janelas e similares).

Os resultados obtidos para a quantidade de pontos perdidos para cada uma das situações estão resumidos nas tabelas 1 e 2. As perdas foram registradas com base na diferença máxima obtida durante 1 minuto de execução do aplicativo. Nota-se, como era esperado, que a utilização de outros aplicativos provoca maiores perdas de dados.

Tabela 1. Perdas de dados sem intervenção (em words)

	2 buffers	4 buffers	8 buffers	16 buffers
10 ms	1000	840	0	0
50 ms	520	0	0	0
100 ms	400	0	0	0
500 ms	400	0	0	0
1 s	480	0	0	0
5 s	480	0	0	0

Tabela 2. Perdas de dados com intervenção (em words)

	2 buffers	4 buffers	8 buffers	16 buffers
10 ms	2920	11960	4720	0
50 ms	1160	680	0	0
100 ms	1040	0	0	0
500 ms	400	0	0	0
1 s	3360	0	0	0
5 s	480	0	0	0

7. Conclusões

Este artigo descreveu o funcionamento da USB como meio de comunicação direto entre computador convencional e placa de aquisição de dados. A velocidade de transferência de dados via USB, que deveria garantir, no mínimo, 100 kbytes/s entre dispositivo e computador, foi de 890 kbytes/s através do bom aproveitamento dos drivers de som. Em paralelo, a função HID implementada já oferece o suporte para que alguns dados de controle sejam trocados entre o computador e a placa de aquisição de dados, além da função de aquisição. Idealmente, o sistema operacional já deveria fornecer device drivers adaptáveis genéricos para padrões como a USB, que é o que já acontece com o Windows Vista parcialmente, apenas indispondo de conexões isócronas.

De qualquer forma, os resultados deste trabalho mostram que, apesar de o padrão USB ser aparentemente menos flexível do que as antigas interfaces de conexão, tais como a RS-232, observa-se que a maior demanda de tempo de desenvolvimento de um sistema USB completo é para o device driver. No entanto, como foi aqui descrito, esse trabalho pode ser normalmente dispensado se os recursos próprios do sistema operacional forem bem utilizados.

References

Axelson, J. L. (2005) “USB Complete: Everything You Need to Develop USB Peripherals”. Lakeview Research, 3ª edition.

CIMM (2007) “Centro de Informação Metal Mecânica: Soldagem”, http://www.cimm.com.br/cimm/didacticMaterial/soldagem.html, Setembro.

EMBEDDEDRELATED (2007) “LPC2148: What endpoint(s) (bulk or isochronous) to choose for big amount of data ?”, http://www.embeddedrelated.com/groups/lpc2000/show/8818.php, Setembro.

LIBUSB (2007) “LisbUsb-Win32”, http://libusb-win32.sourceforge.net/#about, Setembro.

MSDN (2007) “Recording and Playing Waveform Audio”, http://msdn.microsoft.com/archive/default.asp?url=/archive/enus/dnarmulmed/html/sdn_spellit.asp, Setembro.

MSDN (2007) “Kernel-Mode WDM Audio Components”, http://msdn2.microsoft.com/en-us/library/ms789375.aspx, Setembro.

MSDN (2007) “Wave and DirectSound Components”, http://msdn2.microsoft.com/en-us/library/ms790062.aspx, Setembro.

THESYCON (2007) “USBIO - Reference Manual”, http://www.thesycon.com/usbio/usbioman.pdf, Setembro.