Arquitectura para um registador de dados

P.Fazenda(1)(2), M.Gomes(1)(2), A.Pinto(1)(2), V.Costa(1)(2)

(1) Dept. De Eng. De Electrónica e Telecomunicações e de Computadores
Instituto Superior de Engenharia de Lisboa
Rua Conselheiro Emídio Navarro, N. 1, 1950-062 Lisboa
pfazenda@cedet.isel.ipl.pt Tel. 218317281
http://www.isel.ipl.pt

(2) Centro de Estudos e Desenvolvimento de Electrónica e Telecomunicações
Instituto Superior de Engenharia de Lisboa
Rua Conselheiro Emídio Navarro, N. 1, 1950-062 Lisboa
cedet@cedet.isel.ipl.pt Tel. 218317284
http://www.cedet.isel.ipl.pt

A aquisição de dados é um procedimento utilizado em várias áreas. O presente trabalho tem como objectivo desenvolver um registador que armazena dados obtidos de uma ou mais fontes. Para esse efeito propõe-se uma arquitectura baseada num computador pessoal cujas dimensões e interface de expansão obedecem à norma industrial PC/104. Foi desenvolvida uma placa de aquisição para esta interface e foi utilizado o Linux como sistema operativo. Para cumprir os requisitos de tempo real do sistema utilizou-se o Real Time Linux, que corre o Linux como uma tarefa de baixa prioridade, permitindo usufruir de todos os seus serviços e funcionalidades.

A aplicação que gere o registador e a interface de configuração foi programada em linguagem Java. Esta aplicação comunica com um módulo desenvolvido no espaço do Real Time Linux para efectuar o ciclo de aquisição, em determinados intervalos temporais pré-programados. Do ponto de vista do utilizador, a arquitectura interna do sistema tem por orientação as normas industriais existentes para instrumentos programáveis.

A configuração do registador é efectuada remotamente via TCP/IP. Adicionalmente, e seguindo a tendência que existe para este tipo de aplicações, foram desenvolvidos gestores de dispositivo para a utilização com o programa LabVIEW.

Introdução

No mercado existem vários tipos de registadores de dados para as mais variadas aplicações. A escolha de um registador para uma determinada aplicação passa por considerar, entre outras funcionalidades, características como o número de canais, frequência de amostragem, consumo energético, capacidade de armazenamento e processamento. A frequência com que varia o fenómeno em observação e a resolução necessária na sua aquisição são factores determinantes nessa escolha. Outro factor de relevo é o consumo energético, importante para a autonomia do registador se este ficar isolado por um determinado período de tempo. Poderá também haver a necessidade de processamento local para que, após a recolha dos dados, se proceder a cálculos, onde a capacidade de processamento e a versatilidade com que se programa as aplicações são importantes. Todos estes parâmetros estão interligados e por vezes são mutuamente exclusivos. Há que encontrar um compromisso adequado à aplicação.

A proposta que apresentamos com este trabalho é a de um registador de dados que se compatibiliza com as várias soluções que existem para os computadores pessoais, aproveitando tecnologias que disponibilizam ferramentas para tratamento de dados, nomeadamente, ferramentas para análise, suporte ao armazenamento, apresentação e publicação. O objectivo passa também por encontrar uma estrutura modular que permita alterar e substituir módulos, conforme as necessidades de aplicações futuras. Pretende-se, assim, que a estrutura desenvolvida seja adaptável a um conjunto variado de problemas.

Para este efeito propõe-se uma arquitectura baseada num computador pessoal cujas dimensões e interface de expansão obedecem à norma industrial PC/104. Foi desenvolvida uma placa de aquisição para esta interface e foi utilizado o Linux como sistema operativo. Para cumprir os requisitos de tempo

real do sistema utilizou-se o Real Time Linux, que corre o Linux como uma tarefa de baixa prioridade, permitindo usufruir de todos os seus serviços e funcionalidades.
A aplicação que gere o registador e a interface de configuração foi programada em linguagem Java. Esta aplicação comunica com um módulo desenvolvido no espaço do Real Time Linux para efectuar o ciclo de aquisição, em determinados intervalos temporais pré-programados. Do ponto de vista do utilizador, a arquitectura interna do sistema tem por orientação as normas industriais existentes para instrumentos programáveis. Na instrumentação utiliza-se vulgarmente as interfaces *General Purpose Interface Bus* (GPIB ou 488.1) e RS232 para efectuar as comunicações [8][12]. Este trabalho propõe a utilização da Ethernet.
Este artigo começa por apresentar a arquitectura interna do registador de dados. Aborda de seguida a placa de aquisição desenvolvida, a descrição do programa principal do sistema, o aparato experimental montado para testar o registador e as conclusões finais, nas quais são realçadas as características conseguidas com vista ao cumprimento dos objectivos propostos.

Arquitectura do sistema

O sistema proposto para o registador de dados é composto por vários módulos. Foi utilizado como sistema operativo o Linux e uma máquina virtual Java para desenvolver a aplicação do registador. Como o Linux apresenta diversas limitações que o impedem de ser utilizado como um sistema de tempo real, o seu núcleo foi alterado para incluir o RTLinux [4][11][13][14]. Neste espaço do sistema desenvolveu-se todo o código que apresentava requisitos de tempo real, nomeadamente a tarefa que comunica com a placa de aquisição e que gere o processo de amostragem. A figura 1 mostra a estrutura interna do sistema. Um dos objectivos que se pretendeu cumprir com esta estrutura foi a minimização do acoplamento entre módulos. A aplicação DLOG_SYS_App (acrónimo derivado das palavras inglesas *Data Logging System Application*) é o programa central do registador que gere a interface de configuração e interrogação, os parâmetros do registador, o processo de armazenamento e configura o processo de aquisição. Este programa foi desenvolvido em linguagem Java e corre sobre uma máquina virtual, ficando assim independente do sistema subjacente.
O módulo do registador que reside no RTLinux foi programado em linguagem C. Contém uma tarefa servidora que comunica com a aplicação DLOG_SYS_App através de listas do tipo primeiro a entrar - primeiro a sair (First In First Out- FIFO). Com esta abordagem a aplicação DLOG_SYS_App mantém-se utilizável para outras configurações de sistemas subjacentes, se a interface dos FIFOs se mantiver. Outras abordagens podem ser utilizadas para substituir os FIFOs, caso sejam necessárias, sem grande prejuízo em termos de alteração dos códigos dos programas.

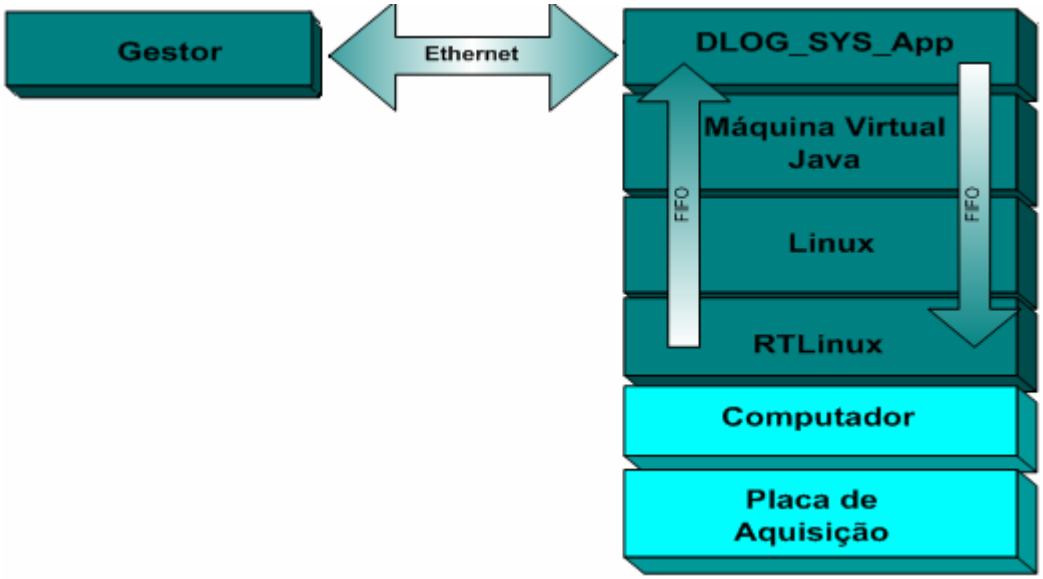

Figura 1 – Arquitectura do sistema.

A interface apresentada ao exterior pelo DLOG_SYS_App está orientada segundo a norma industrial *Standard Commands for Programmable Instruments* (SCPI) [2][7]. O registador realiza um

subconjunto de comandos desta norma, que vêm publicados nas suas especificações. A aplicação serve os gestores que enviam remotamente comandos via TCP/IP.
Para facilitar o desenvolvimento das aplicações que fazem a interface com o registador foi desenvolvido um conjunto de gestores de dispositivo para utilização em ambiente LABView, que organizam conjuntos de comandos SCPI em funcionalidades de nível superior [19]. A partir de uma aplicação remota um operador pode definir os canais de aquisição, a frequência de amostragem, o ficheiro de armazenamento, pode consultar os valores à entrada de cada canal, entre outras funcionalidades. A consulta dos dados pode também ser efectuada utilizando protocolos de transferência de ficheiros e outros programas de comunicação disponíveis no sistema operativo Linux.

A placa de aquisição

Foi desenvolvida uma placa de aquisição para ligar à placa PC através do barramento PC/104. Esta unidade de aquisição é, do ponto de vista do sistema, facilmente alterável e substituível, conforme as necessidades das aplicações. O barramento PC/104 permite também empilhar vários módulos de aquisição, como define a norma. O registador de dados recolhe dados através desta placa, que tem como elemento principal o conversor ADS1218 fabricado pela Texas Instruments. Os restantes circuitos servem a comunicação, descodificam endereços, filtram as entradas e regulam as tensões. A figura 2 mostra o diagrama de blocos da placa de aquisição.

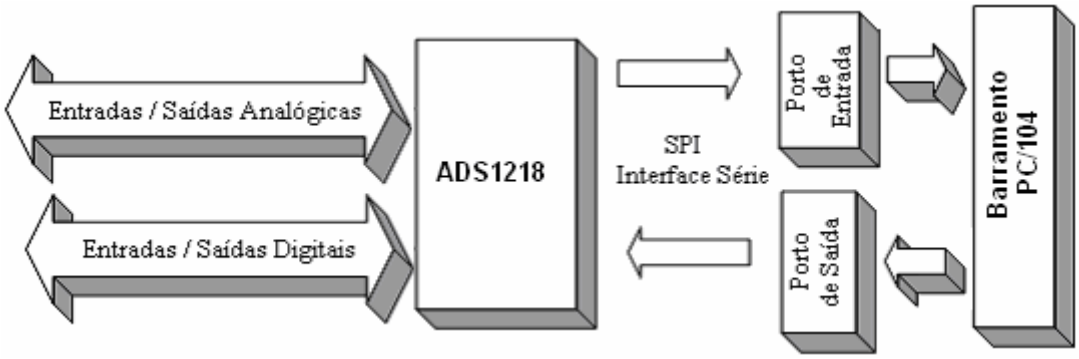

Figura 2 – Diagrama de blocos da placa de aquisição

A interface de aquisição está definida de acordo com as características do conversor ADS1218. Este circuito integrado apresenta funções de conversão analógica para digital com uma resolução de 24 bits, assim como 8 linhas digitais bidireccionais e dois canais analógicos de saída. O controlo deste circuito integrado é efectuado através de um banco de registos e de um conjunto de comandos. Os comandos permitem ler e escrever nos registos assim como efectuar operações de calibração, de sincronismo e de reinício de sistema (*Reset*).
A transmissão de dados e de comandos é efectuada através do protocolo série *Serial Periferical Interface* (SPI). Para realizar a comunicação SPI na placa de aquisição utilizam-se dois portos, cujos endereços dependem do espaço livre de endereços da placa PC [9].
Entre a placa de aquisição e a aplicação DLOG_SYS_App existe a tarefa em execução no espaço do RTLinux que comunica com ambos os módulos. A tarefa realiza o protocolo SPI para comunicação com o conversor ADS1218.
No registador de dados a direcção predominante na comunicação entre módulos é no sentido dos canais de entrada para os ficheiros de armazenamento ou para a interface de rede. Uma excepção é a actuação da aplicação nas saídas digitais do ADS1218 com vista a sinalizar, num conjunto de LEDs, alguns dos estados do sistema. Porém, a sua utilização poderá ter outras aplicações, nomeadamente, permitir que programas em execução paralela no espaço do Linux possam actuar, através da interface da aplicação DLOG_SYS_App, com o meio ambiente. Esta funcionalidade permite, por exemplo, ao utilizador programar aplicações que analisem os dados com condições de executar, em situações de alarme, procedimentos de emergência e actuar no meio ambiente onde o registador está inserido.

O programa DLOG_SYS_App

O modelo de programação do registador requer que todas as tarefas que apresentem requisitos de tempo real sejam executadas no espaço do RTLinux, deixando as restantes para execução em espaço Linux, por forma a tentar minimizar ao máximo a dimensão do código no espaço do RTLinux [3]. A figura 3 mostra a ligação entre ambos os programas. O módulo RT_DLOG_SYS_Software contém a programação da tarefa que corre no espaço do RTLinux. Esta tarefa serve a aplicação DLogSystem_App, executando comandos que lhe são enviados através do canal cmdFIFO. As respostas a estes comandos são enviadas de volta à aplicação através do canal readFIFO. Os sinais de controlo de fluxo do processo de amostragem e os dados provenientes da aquisição são enviados pelo dataFIFO, para aliviar ao máximo a necessidade da aplicação DLogSystem_App discriminar o tipo de informação recebida.

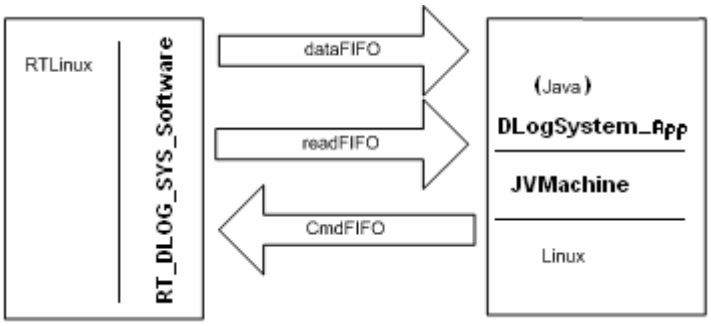

Figura 3 – Comunicação entre o programa DLogSystem_App e o módulo RT_DLOG_SYS_Software.

A aplicação DLOGSystem_App serve, através de uma interface Socket, os clientes que se ligam para efectuarem os serviços de configuração. Do ponto de vista dos utilizadores, a arquitectura interna caracteriza-se de acordo com os modelos para instrumentos programáveis apresentados na norma SCPI. A divisão dos vários módulos internos do registador acima apresentados é, para eles, transparente.
A norma SCPI utiliza modelos que reúnem funcionalidades que são compatíveis entre vários tipos de instrumentos diferentes como, por exemplo, osciloscópios, multímetros e registadores de dados. Nestes modelos existe uma hierarquia de blocos pela qual circulam fluxos de informação e controlo. A figura 4 mostra o modelo do bloco de medição de um *instrumento sensor*, do qual deriva o registador. Neste tipo de instrumento o sinal é recebido através do circuito de aquisição e convertido numa representação interna, após a qual pode ser: analisado ou manipulado de acordo com algumas regras de cálculo, armazenado, enviado para um mostrador, etc. Estas funcionalidades são realizadas por blocos. Cada um destes blocos desempenha uma ou mais funções no sistema no que diz respeito ao encaminhamento e tratamento do sinal, conversão, cálculos, armazenamento, etc. Cada bloco tem pontos por onde os fluxos de dados são absorvidos e emitidos, podendo receber dados de vários blocos mantendo porém, restrições na origem dos dados. São permitidas também ligações dinâmicas e programáveis entre alguns dos blocos.

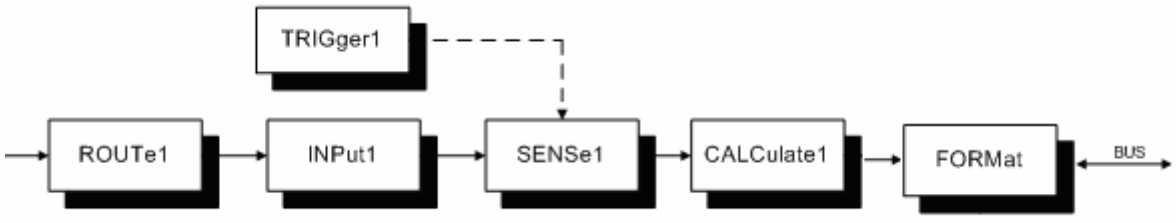

Figura 4 – Modelo do bloco de medição de um instrumento sensor [2].

A figura 5 mostra parte do modelo realizado pelo registador. O bloco SENSe contém várias saídas que fornecem as tensões de cada canal da placa de aquisição. A saída de um canal em leitura pode ser ligada dinamicamente a um bloco DLOG_SYS_FileManager, residente dentro da hierarquia MMEMory. Este bloco comprime os dados e canaliza-os para um ficheiro.
Os comandos SCPI que são enviados para configurar e interrogar o equipamento estão relacionados com os vários blocos e organizam-se hierarquicamente numa estrutura em árvore.
A arquitectura do programa DLOGSystem_App contém um conjunto de objectos que estão relacionados com os diversos comandos aceites pelo instrumento e com os vários blocos internos do modelo SCPI. O nome que estes objectos apresentam sugere automaticamente qual o bloco a que pertencem, assim como o caminho da hierarquia. Tal como a organização em árvore tomada pelos comandos da norma, também os objectos se organizam numa estrutura semelhante. A gestão de cada comando é efectuada por um objecto cujo nome se assemelha a este. Na análise sintáctica de um comando SCPI a árvore interna de objectos é percorrida e é invocado o método de execução ou interrogação do objecto que gere o respectivo comando. A pesquisa na árvore orienta-se dinamicamente por forma a optimizar a execução dos comandos executados mais frequentemente.

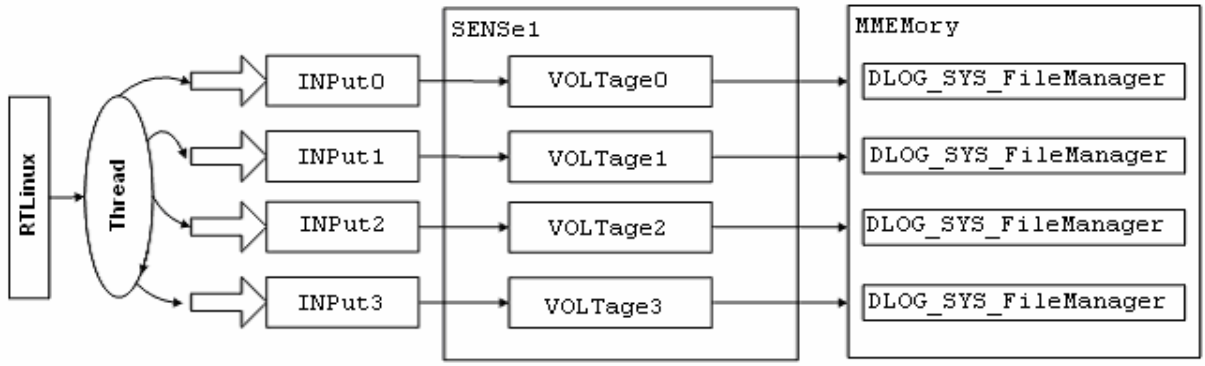

Figura 5 – Modelo interno do registador de dados segundo a norma SCPI.

Utilização do ambiente LabVIEW

Para a gestão de dispositivos de aquisição é frequente a utilização do ambiente de programação LabVIEW. Para utilizar o registador neste ambiente, os utilizadores podem utilizar os gestores de dispositivo. Estes gestores foram desenvolvidos segundo as recomendações apresentadas pela National Instruments [19]. Cada gestor gere uma funcionalidade, que é realizada com a execução de um ou mais comandos do registador. Ao utilizar os gestores, os clientes podem abstrair-se deste conjunto de comandos, que são executados automaticamente. Conforme se vão adicionando novas funcionalidades ao registador, há que criar o respectivo gestor. A figura 6 mostra a forma como cada gestor aparece no ambiente LabVIEW.

Figura 6 – Gestores de dispositivo para o ambiente LabVIEW.

Aparato Experimental

Para testar o registador foi montado um aparato experimental que consistia em ligar a três entradas do registador as saídas de um geofone, para detectar actividade sísmica segundo os eixos X,Y e Z. O registador foi configurado para efectuar aquisições de cada um dos canais, com vários intervalos de amostragem. No final do processo de análise os dados adquiridos constavam em ficheiros que foram recolhidos utilizando ferramentas de comunicação por *secure file transfer protocol* (Secure FTP).

A configuração do registador foi realizada através de uma aplicação de gestão, desenvolvida em ambiente LabVIEW, que tinha também por objectivo exemplificar a utilização dos gestores de dispositivos supracitados. A figura 7 mostra parte do código gráfico da aplicação.

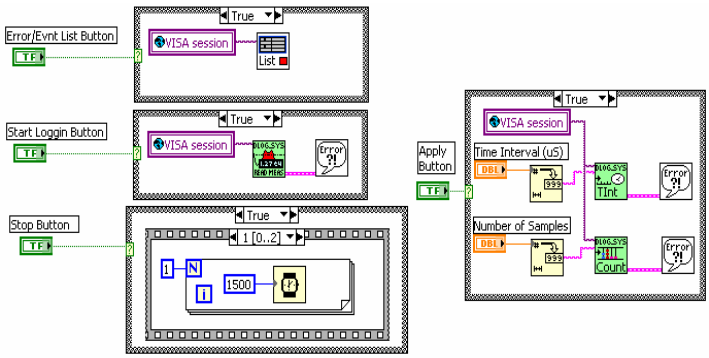

Figura 7 – Linguagem gráfica da programação da aplicação em ambiente LabVIEW, utilizando os gestores de dispositivo.

A aplicação é composta por vários painéis que permitem configurar variáveis do funcionamento do registador como, por exemplo, definir canais activos, nome dos ficheiros de armazenamento, frequências de amostragem, etc. A aplicação contém também um painel que permite monitorizar o comportamento dos canais em tempo real. A figura 8 mostra esse painel com o traçado do sinal recebido pelo canal 0, depois de se produzir propositadamente perturbações no suporte do geofone.

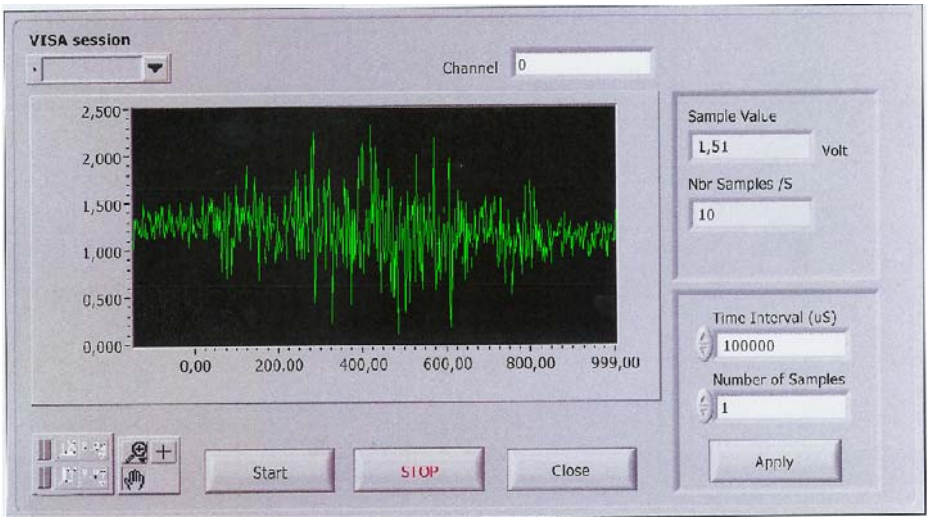

Figura 8 – Um dos paineis da aplicação para a gestão remota do registador.

Conclusão

Neste artigo foi proposta e testada uma arquitectura para um registador de dados que utiliza um computador com expansão PC/104. A abordagem utilizada abre caminho para um conjunto alargado de projectos de interesse académico e eventualmente, aplicações comerciais. Como referência para trabalhos futuros fica a hipótese de se expandir a arquitectura para utilização em projectos como robôs móveis, sistemas de aquisição e telemetria, controlo de sistemas, entre outros.

Durante o desenvolvimento deste projecto identificaram-se algumas limitações do sistema construído, sendo a mais relevante a máxima frequência de aquisição. Neste sistema, e utilizando apenas um canal, a frequência de amostragem pode ser, no máximo, de 100 amostras/segundo. Esta limitação tem várias origens. A velocidade de transferência de dados da placa de aquisição para a memória do computador é uma das causas. Sendo esta transferência gerida por uma tarefa de tempo real, o

aumento da frequência de aquisição reduz o tempo de processamento das tarefas que correm no espaço do Linux chegando, no limite, a parar todos os seus serviços e aplicações.
Verificou-se igualmente que a velocidade da aplicação programada em Java limitava a frequência de aquisição, devido ao tempo que esta levava a ler os dados do FIFO, transmiti-los através dos diversos blocos e gravá-los em ficheiros comprimidos. Em certas situações, devido à lentidão de resposta, verificou-se que o FIFO ultrapassava a sua dimensão preestabelecida. Verificou-se ainda que a sua utilização introduziu latências. Para futuras versões, considera-se a hipótese de utilizar um conversor para cada canal, permitido assim, efectuar aquisições simultâneas – o que não é permitido nesta versão devido à multiplexagem dos canais de aquisição – e aumentar a frequência de amostragem. Os dados podem ser agrupados num FIFO na placa de aquisição e transmitidos para a memória do computador através de DMA. Os FIFOs de comunicação entre o RTLinux e o Linux podem ser substituídos por memória partilhada, e a aplicação que corre no Linux sobre a máquina virtual Java pode ser compilada para código nativo. Com estas alterações julga-se que se poderá melhorar a velocidade de aquisição do sistema.

Referências

[1] Pedro Fazenda, Miguel Campilho Gomes, António C. Pinto, Vítor S. Costa, "Registador de Dados", Relatório do Projecto final de curso, 09-12-2003, DEETC, ISEL.
[2] SCPI Consortium, "Standard Commands for Programmable Instruments" (SCPI), http://www.scpiconsortium.org/scpistandard.htm.
[3] Matt Sherer, "Writing Applications with RTLinux", Finite State Machine Labs (FSM), http://www.fsmlabs.com.
[4] Cort Dougan, "Creating a WebPage Controlled 2-Axis Movable Camera with RTLinux/Pro, (FSM).
[5] A. Bruce Carlson, "Communication Systems", Third Edition, Mc Graw Hill.
[6] National Instruments, "Increasing Test System Compatibility and Productivity with IEEE 488.2", http://www.ni.com.
[7] John M. Pieper, "SCPI and VISA a valuable combination", Agilent, http://www.agilent.com
[8] Agilent, "A Tutorial Introduction to HPIB"
[9] Tom Shanley, Don Anderson, "ISA System Architecture", Third Edition, Addison-Wesley Publishing Company.
[10] Keithley, "Data Acquisition and Control Handbook", http://www.keithley.com
[11] Matt Sherer, "Writing Applications With RTLinuxPro", (FSM).
[12] VXI Consortium, "VMEbus Extensions for Instrumentation, TCP/IP-IEEE 488.2 Instrument Interface Specification" – http://www.vxibus.org.
[13] Michael Barabanov, "A Linux-based Real-Time Operating System", www.rtlinux.org
[14] FSM Labs, Inc. April 20, 2001 – "Getting Started with RTLinux"
[15] Michael Barabanov, Vitor Yodaiken, Edgar Hilton, "RTLinux FAQ", www.rtlinux.org
[16] Michael K.Johnson, "Writing Linux Device Drivers", www.redhat.com.
[17] Victor Yodaiken, "Cheap operating systems reserch and teaching with Linux", Department of Computer Science new Mexico Tech.
[18] National Instruments, "LabVIEW VISA Tutorial", www.ni.com
[19] National Instruments, "Developing a LabView Instrument Driver", Application Note 006, www.ni.com.
[20] Mathworks, "Controlling Instruments Using TCP/IP and UDP", Instrument Control Toolbox.
[21] Bruce Eckel, "Thinking in Java", http://www.eckelObjects.com.
[22] National Instruments, "LabVIEW", User Manual.
[23] National Instruments, "G Programming Reference Manual".
[24] National Instruments, "LabVIEW Basics".
[25] National Instruments, "Function and VI Reference Manual".
[26] National Instruments, "QuickStart Guide".
[27] Barry E. Paton,"Sensors, Transducers & LabVIEW", National Instruments.