Desenvolvimento de Laboratório Remoto Utilizando Módulo Didático para Ensino de Microcontroladores

Flávio Henrique Toribio Destro[1], Fábio Iaione[2]

[1,2]Faculdade de Computação – Universidade Federal de Mato Grosso do Sul (UFMS)
79070-900 – Campo Grande – MS – Brasil

flavio_toribio@hotmail.com, iaione@facom.ufms.br

Abstract. *Due to the growth of distance education in universities, laboratory activities are replaced by other more theoretical forms of teaching. However, in the area of microcontrollers, the laboratory in-person experience using hardware is much more interesting and useful for students. Given the above, this work proposes the development of a remote laboratory using a hardware module to teach microcontrollers. The student can interact through a Web browser with several functionalities of the hardware module, located remotely in a physical laboratory of the university. Thus, the student gains a real experience, as if he were in person in the laboratory.*

Resumo. *Devido ao crescimento do ensino a distância nas universidades, atividades de laboratório acabam sendo substituídas por outras formas mais teóricas de ensino. Porém, na área de microcontroladores, a experiência presencial no laboratório utilizando hardware é muito mais interessante e útil para os alunos. Dado o exposto, este trabalho propõe o desenvolvimento de um laboratório remoto usando um módulo de hardware para o ensino de microcontroladores. O aluno pode interagir por meio de um navegador Web com várias funcionalidades do módulo de hardware, localizado remotamente em um laboratório físico da universidade. Sendo assim, o aluno ganha uma experiência real, como se estivesse presencialmente no laboratório.*

1. Introdução

Com o recente crescimento da educação à distância nas universidades, é necessário desenvolver novas metodologias de ensino para atividades que geralmente são trabalhadas exclusivamente em laboratório. Tais atividades possuem componentes físicos que devem ser manipulados presencialmente, tornando-se um problema para um sistema de educação à distância. Por conta disso, na maioria das "universidades virtuais", os estudantes ainda precisam atender a eventos presenciais, tais como aulas de laboratório [Kaderali et al., 2001].

Uma das metodologias já apresentadas em diversos trabalhos [Djordjevic et al., 2005, Ko et al., 2001, Swamy et al., 2002, Tan et al., 2002] baseia-se inteiramente na simulação de placas de *hardware* via *software*. Isso soluciona o problema da presença física no laboratório, mas apresenta uma grande limitação, que é a incapacidade de um simulador reproduzir todas as limitações físicas e a complexidade de um sistema de *hardware*. Normalmente, a simulação não consegue considerar absolutamente todas as

DOI: 10.5753/cbie.sbie.2019.149

variáveis que influenciam o sistema de *hardware*. Alguns estudos concluíram que o desenvolvimento das habilidades "práticas" do aluno, por meio da execução de experimentos reais envolvendo sistemas embarcados, por exemplo, são essenciais para uma boa aprendizagem nessa área de estudo [Cooper, 2000]. Portanto, desenvolver metodologias de ensino a distância que apresentem uma experiência mais "*hands-on*" com laboratórios virtuais, continua sendo um desafio [Eppes and Schuyler, 2004]. Devido a fatores desfavoráveis como custo, espaço e tempo, referentes ao ensino em laboratório presencial [Luthon et al., 2009], soluções alternativas para a introdução de laboratórios remotos mais "reais" continuam sendo um assunto discutido na literatura.

Existem vários trabalhos já desenvolvidos com o objetivo de solucionar o problema dos experimentos práticos no ensino a distância, na área de hardware. Dentre eles, pode-se citar o DIESEL, o REDLART e o LaboRem.

O DIESEL (*Distant-Internet Embedded Systems Engineering Laboratory*) [Yue et al., 2009] foi um projeto desenvolvido na University of Ulster, que oferece uma solução modular para o problema. O sistema também é genérico no sentido de aceitar vários tipos de sistemas embarcados como microcontroladores, DSPs, FPGAs e SoCs em sua plataforma. Os autores utilizaram um *design* escalável, que aceita adições futuras de novos tipos de sistemas embarcados. A parte de experimentação do sistema possui a compilação remota de *firmware*, e a visualização remota dos instrumentos utilizados (osciloscópio, gerador de função, multímetro e analisador lógico) pela interface do cliente, que é providenciada pelo sistema LabVIEW, da National Instruments. Foram utilizados dois módulos de hardware para a comunicação e interação entre as placas de experimento e o computador, que age como servidor no laboratório, o EBIM (*Experiment Board Interface Module*) e o IIM (*Instrumentation Interface Module*). Esses módulos que possuem a função de "*middleware*" entre o servidor e os circuitos constituem um sistema extensivo e complexo, formado por vários componentes com funções diferentes.

O REDLART (*REconfigurable Digital Laboratory for Advanced Research and Teaching*) [Moreira, 2009] tem o objetivo de trazer a prototipação rápida de novos experimentos em sistemas digitais nas áreas de engenharia, telecomunicações, computação e afins, através de dispositivos semicondutores do tipo FPGA. O sistema desenvolvido também é modular, permitindo que o foco dos experimentos seja maior no problema que se deseja resolver, e não na complexidade do hardware utilizado, que é isolada pelo conjunto de serviços de *software* oferecido pelo sistema.

O LaboRem [Luthon et al., 2009] é uma arquitetura de instrumentação remota de um laboratório virtual para estudantes de graduação cursando matérias que abordam processamento de sinais, depuração de circuitos e testes de sistemas. Esse sistema tem uma abordagem mais voltada aos jogos educacionais, tornando o ensino mais didático. O sistema também é implementado com LabVIEW, mas possui o adicional do *streaming* de vídeo em tempo real do circuito.

Dado o exposto, este trabalho tem como objetivo desenvolver um laboratório remoto, baseado em um módulo didático de *hardware*, para o ensino de microcontroladores. O aluno poderá interagir via interface World Wide Web (WWW) com várias funcionalidades do módulo didático, localizado remotamente em um

laboratório físico da universidade, e disponível em qualquer dia e horário. Tais interações podem ser analisadas pelo *streaming* de vídeo que é transmitido em tempo real durante o experimento, mostrando os visores do módulo didático. Como a solução apresentada não é baseada em simulação, os alunos terão uma experiência real, como se estivessem presentes no laboratório, melhorando a aprendizagem.

2. Metodologia

Nessa seção são descritos os materiais e métodos utilizados no desenvolvimento do laboratório remoto, envolvendo *hardware*, *software* e *firmware*.

2.1. Módulo Didático para Ensino de Microcontroladores

O módulo principal utilizado no sistema corresponde a placa de hardware BIG8051 (Figura 1). O módulo didático BIG8051, fabricado pela MikroElektronika, é um sistema que fornece um ambiente de desenvolvimento para a programação e experimentação de microcontroladores com núcleo CIP-8051 (núcleo MCS-51 acrescido de *pipeline*), da Silicon Laboratories. A placa é fornecida com o microcontrolador C8051F040, que executa até 25 MIPS (@ 25 MHz), e possui internamente: 64 kB de memória *flash* (programa), 4352 bytes (256 bytes do núcleo MCS-51 acrescido de 4 kB) de memória RAM (dados), 64 pinos de I/O, portas seriais UART (duas), SPI, I2C, CAN, cinco temporizadores/contadores (16 bits) de uso geral, matriz programável de temporizadores/contadores (16 bits) com seis módulos de captura/comparação, conversor analógico-digital (ADC) e digital-analógico (DAC) de 12 bits, tensão de referência, três comparadores de tensão, entre outros. A placa pode ser programada e depurada utilizando-se um módulo USB, conectado via interface JTAG ao microcontrolador.

Figura 1. Módulo didático BIG8051 para ensino de microcontroladores (26,5 x 22 cm).

No módulo BIG8051, externamente ao microcontrolador, estão disponíveis os seguintes dispositivos: 64 teclas e 64 LEDs, para definição e visualização dos sinais nos 64 pinos de I/O; conectores para acesso direto aos 64 pinos de I/O do microcontrolador; *buzzer* piezoelétrico; *display* de cristal líquido alfanumérico (2 linhas x 16 caracteres); *display* de cristal líquido gráfico (128 x 64 *pixels*); memória EEPROM I2C de 128 bytes; memória SRAM SPI de 8 kB; memória *flash* SPI de 8 Mbits; conector para

cartões de memória MMC/SD; conectores ethernet RJ-45, EIA/TIA-232 DB-9 (2), USB (fonte de energia e porta serial virtual); conectores para 4 entradas e 2 saídas analógicas; conector para sensor de temperatura digital; e conector para módulo de RF ZigBee, entre outros. Como a quantidade de pinos de I/O do microcontrolador é insuficiente para a conexão simultânea de todos os dispositivos presentes na placa, existem chaves (*DIP switches*) que permitem a conexão de um ou outro dispositivo aos pinos de I/O [MikroElektronika, 2010].

A programação do microcontrolador pode ser feita em linguagem de montagem ou em linguagens de programação de alto nível, sendo a linguagem C a mais utilizada.

2.2. Arquitetura do Sistema

O sistema desenvolvido consiste de um lado servidor e um lado cliente. O lado cliente é a interface *web* utilizada pelo aluno, que acessa remotamente o laboratório de ensino utilizando um aplicativo "navegador". Já o lado servidor é responsável pela hospedagem do *website*, pela hospedagem da *webcam*, e pela interface entre o cliente do laboratório remoto e a placa BIG8051, utilizando um módulo programador, e um Arduino Mega 2560 para gerar e ler alguns sinais (Figura 2).

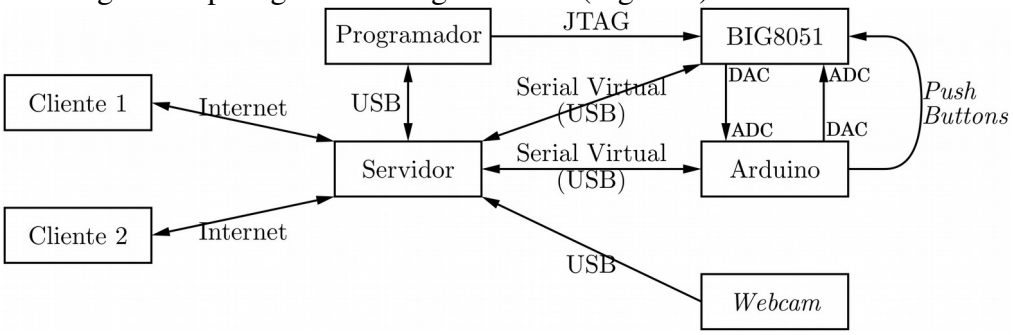

Figura 2. Diagrama de blocos do sistema.

2.3. Lado Cliente

O lado cliente é uma página web (Figura 3) constituída de seis componentes: *push buttons,* controle de tensão para enviar ao ADC, monitor de tensão lida na saída do DAC, webcam, console serial e editor de código (compilador/uploader).

Todas as requisições feitas pelo cliente (ex.: um clique em um *push button* ou no botão "Compilar") são enviadas para o servidor através do protocolo *WebSocket* [Lombardi, 2015], resultando em uma maior velocidade de comunicação entre os pares devido à sua conexão persistente. Isso é altamente desejável em uma aplicação de tempo-real como a apresentada aqui, sendo a única desvantagem o fato dessa tecnologia estar disponível somente em navegadores mais atuais que suportam HTML5.

A página envia uma requisição ao servidor tanto ao pressionar, como ao soltar os push buttons da Figura 3, simulando não apenas o clique como também o click and hold de uma aplicação real. O middleware — que simula o clique do botão — adiciona um bounce [Morton, 2001] de 500 microssegundos a cada mudança de estado do componente. Isto faz com que o sistema comporte-se como uma tecla real, forçando o aluno a utilizar um algoritmo de debouncing no firmware do microcontrolador.

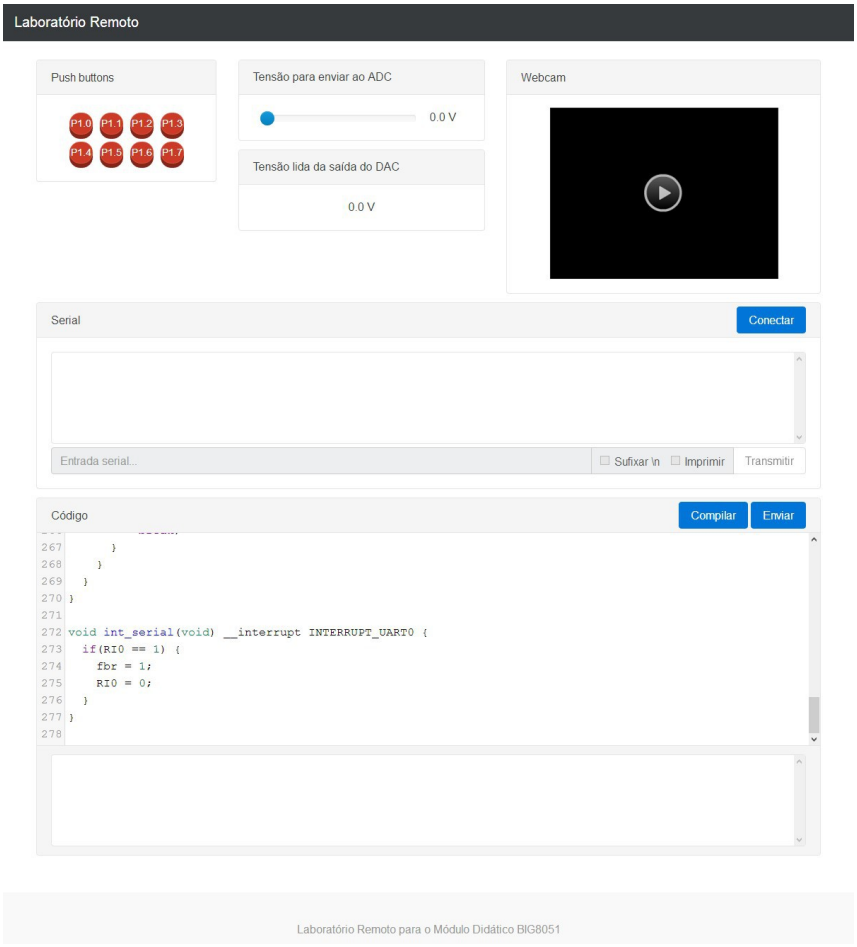

Figura 3. Visão geral do *front-end* do *website*.

Por meio de um componente *slider*, o usuário pode enviar uma tensão elétrica entre 0 e 3,3 Volts para o conversor analógico-digital do microcontrolador (canal 0).

A tensão elétrica gerada pelo conversor digital-analógico (DAC) do microcontrolador pode ser visualizada no componente da Figura 3, que é atualizado a cada 250 ms (apenas se houver mudança, para economizar banda de rede).

A webcam (Figura 3) pode ser utilizada para visualizar certos dispositivos da placa BIG8051, como por exemplo, os textos e imagens mostrados pelos dois displays de cristal líquido e pela matriz de LEDs. A resolução de vídeo e a taxa de atualização de quadros da webcam são dependentes do servidor específico hospedando a aplicação. Nos testes de atraso de quadros, foi configurada uma taxa de quadros de 20 FPS e resolução de 320x240 *pixels*.

Por meio do Console serial, o usuário pode interagir com a porta serial UART do microcontrolador da BIG8051. Todos os caracteres enviados pelo microcontrolador são mostrados nesse console, assim como, caracteres digitados no console podem ser enviados ao microcontrolador.

O usuário deve escrever o *firmware* do microcontrolador em linguagem C, no componente Código (Figura 3), podendo compilá-lo e depois enviá-lo à placa BIG8051.

2.4. Lado Servidor

O lado servidor é implementado em um computador localizado na instituição de ensino, possuindo os seguintes objetivos:

- Hospedar um servidor HTTP e WebSocket;
- Compilar e gravar os *firmwares* na placa didática BIG8051;
- Interagir com a placa BIG8051 por meio de um Arduino Mega 2560 e de uma porta serial UART virtual (USB);
- Hospedar um servidor RTMP para transmitir, em tempo real, imagens capturadas da placa didática pela *webcam*.

O servidor HTTP e WebSocket foi programado usando os frameworks Express e Socket.IO do Node.js. Esta solução acabou sendo elegante devido à natureza assíncrona do protocolo WebSocket. Anteriormente na etapa de levantamento de requisitos e projeto da aplicação, haviam sido pensadas outras alternativas que necessitariam de codificações mais complexas para implementar a comunicação entre o servidor HTTP e WebSocket. Por exemplo, PHP com Apache e Ratchet, Ruby on Rails com Faye WebSockets, entre outros. A codificação é quase que totalmente feita em JavaScript, exceto nos casos de arquivos de configurações, arquivos de estilo em cascata e/ou arquivos de *views*.

O compilador utilizado é o *Small Device C Compiler* (SDCC), sendo que a compilação é realizada através da criação de um processo do SDCC, diretamente pelo *back-end* do servidor. Quando o usuário envia a ação de compilar, a aplicação envia uma requisição WebSocket com o código do usuário, e faz o pré-processamento para checar se há alguma violação de segurança (como por exemplo a inclusão de arquivos de cabeçalho com caminho de diretórios do sistema). Após essa etapa, é iniciado o processo de compilação e acoplamento. Em qualquer uma dessas etapas, se houver um erro, o usuário recebe uma mensagem imediatamente. A gravação do *firmware* é realizada utilizando o módulo *USB Debug Adapter/Programmer,* da Silicon Laboratories, ligado diretamente no conector JTAG da placa didática, e a mesma rotina de criação do processo de compilação, porém, o processo criado é o da ferramenta FlashUtilCL.exe. Esta ferramenta está disponível apenas para Microsoft Windows, sendo esse o motivo do sistema desenvolvido funcionar apenas neste sistema operacional. No entanto, o sistema já está preparado para outras plataformas, como Linux e Mac OS, considerando a existência da ferramenta de gravação para esses sistemas operacionais.

Como mostrado na Figura 2, o Arduino funciona como um *middleware*, simulando o pressionamento de 8 teclas (*push buttons*) da placa BIG8051, colocando pinos de I/O do microcontrolador em nível lógico 0 exatamente da mesma forma que as teclas fazem ao serem pressionadas. Os pinos P2 a P9 do Arduino controlam os níveis lógicos aplicados aos pinos P1_0 a P1_7, includindo *bounce* de 500 microssegundos ao pressionar e ao soltar. Além disso, o Arduino lê tensões elétricas geradas pelo conversor digital-analógico (DAC), por meio da conexão entre a saída DAC0 da BIG8051 e a entrada A0 do Arduino. Ainda, o Arduino envia tensões elétricas para o conversor analógico-digital (ADC) do módulo BIG8051, por meio de seu pino P11 (configurado

como saída *PWM-Pulse Width Modulation*), que para fornecer um sinal realmente analógico passa por um filtro passa-baixas RC (Fc=10 Hz), antes de ser aplicado ao pino CH0 (canal 0 do ADC) da BIG8051. A comunicação entre o Servidor e o Arduino é realizada por meio de uma porta serial UART virtual (115200 bps), via USB, e o protocolo usado é extremamente simples, composto de um byte "ação a tomar" e um byte de parâmetro da ação.

A comunicação serial UART entre o servidor e a placa didática BIG8051 é realizada por meio de uma porta serial virtual, implementada via porta USB. Esta ligação é utilizada para estabelecer a comunicação serial UART entre o cliente e o microcontrolador (9600 bps, 8N1). A comunicação serial com a placa BIG8051, como também com o Arduino, é realizada através da biblioteca Node Serialport1, do Node Package Manager (NPM). O administrador do servidor pode configurar as portas COM por busca automática ou especificá-las manualmente no arquivo de configurações app/config/index.js.

O conjunto de servidor e cliente RTMP (*Real Time Messaging Protocol*) é responsável pelo *broadcast* da *webcam* para os usuários. Em etapas anteriores do desenvolvimento do trabalho, foi especificado o uso de um formato de *streaming* de vídeo capaz de ser visualizado diretamente em navegadores que suportam HTML5. Foram levantados os requisitos dos formatos HLS, Dash e MJPEG. A escolha final foi o formato RTMP, devido aos seguintes benefícios: *streaming* em tempo real, bom suporte e fácil implementação. O servidor RTMP utilizado foi o Nginx RTMP Module, que é fácil de configurar e mostra-se extremamente escalável. Para a configuração final do servidor, foram necessárias apenas 17 linhas.

No caso do cliente RTMP, pode-se utilizar qualquer software. Durante o desenvolvimento do trabalho, o software que se apresentou mais confiável foi o Adobe Flash Media Live Encoder. Outras alternativas são: Open Broadcaster Software (OBS), ffmpeg, e outros.

Cabe observar que o servidor atende um cliente por vez, pois existe apenas um módulo didático conectado ao mesmo. Quando um segundo cliente tenta utilizar o sistema simultaneamente, este recebe uma mensagem de ocupado no navegador. O sistema é liberado quando o cliente que está usando o sistema fecha a aba do seu navegador, ou fica inativo por três minutos.

3. Testes e Resultados

Nesse capítulo são apresentados os testes realizados e os resultados obtidos, após a implementação do sistema. Os testes contemplam a verificação do correto funcionamento do sistema, assim como os atrasos de tempo na interação entre cliente-servidor, considerando duas situações, cliente-servidor na mesma intranet e cliente-servidor na internet. Os atrasos na internet foram verificados entre dois pares localizados em diferentes estados do país, Mato Grosso do Sul e Paraná.

3.1. Atraso da Webcam

Na intranet, o método de medição do atraso utilizou uma câmera para capturar a tela do cliente em conjunto com a tela do próprio servidor, na qual era visualizado um

cronômetro com precisão de milissegundos (Figura 4). Ao fotografar 40 quadros arbitrários do cenário, pode-se verificar o atraso de tempo para uma imagem capturada pela *webcam* no servidor, aparecer na tela do cliente. A média de atraso da *webcam* na intranet foi de 676,87 ms, com desvio padrão de 110,94 ms.

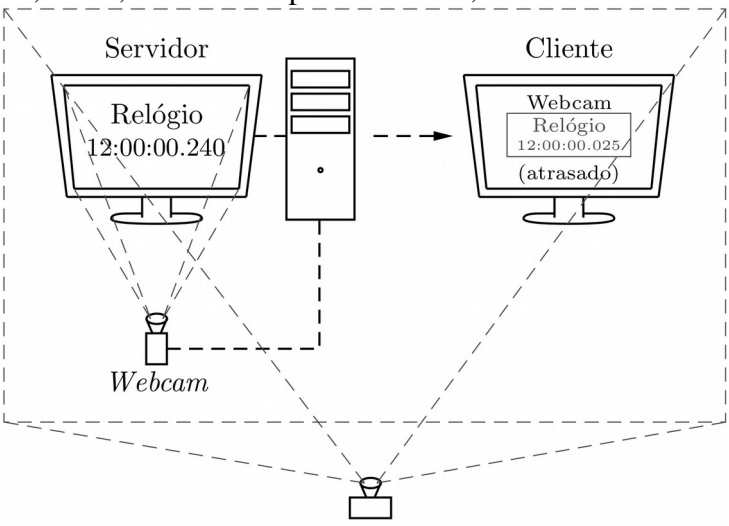

Figura 4. A *webcam* está filmando um cronômetro no próprio servidor, que então envia o vídeo para o cliente, que é visualizado no navegador. Percebe-se que nesse exemplo houve um atraso de 12:00:00.240–12:00:00.025 = 215 ms.

O atraso da webcam na internet foi verificado baseando-se na sincronização via NTP [Mills, 1991], entre o relógio do servidor e do cliente. A sincronização foi feita com precisão de segundos entre dois pares em diferentes estados do país. Novamente, para medir a diferença dos tempos, a webcam foi posicionada de forma a filmar o relógio do servidor, para então o cliente capturar a tela de visualização da webcam e seu próprio relógio, sincronizado anteriormente com o servidor. A média de atraso da *webcam* na internet foi de 1 s, com desvio padrão de 0,5 s.

3.2. Atraso da Comunicação Serial

O método utilizado no atraso da comunicação serial baseou-se na medição do tempo de ida e volta de um caractere transmitido do cliente para o servidor, executando no módulo BIG8051 um *firmware* que implementava um "ping-pong" na porta serial (transmitia para o cliente um caractere, imediatamente após receber um caractere).

Na intranet, a média do atraso de ida e volta de um byte foi de 21,82 ms, com desvio padrão de 7,68 ms. Na internet, a média de atraso de ida e volta foi de 132,97 ms, com desvio padrão de 22,23 ms.

3.3. Atraso dos Push Buttons

A verificação do atraso dos *push buttons* foi realizada pela medição do tempo entre o clique inicial do usuário e o retorno de um byte pela porta serial (console serial). Foram coletadas 40 amostras de tempo de atraso. A média de atraso na intranet foi de 15,32 ms, com desvio padrão de 4,26 ms, e na internet foi 29,05 ms, com desvio padrão de 9,44 ms.

3.4. Atraso do DAC e ADC

A verificação de atraso dos componentes DAC e ADC foi realizada simultaneamente. Um valor de tensão foi transmitido pelo *slider* de tensão do componente ADC (Figura 3), e um *firmware* de teste na BIG8051 mediu essa tensão e gerou o mesmo valor no seu DAC, que por sua vez, atualizou o componente DAC da página (Figura 3). Na intranet, a média do atraso foi de 73,70 ms, com desvio padrão de 6,96 ms, e na internet foi de 235,35 ms, com desvio padrão de 11,01 ms.

4. Conclusão

O laboratório remoto desenvolvido fornece ao aluno, participante de uma turma à distância, a possibilidade de utilizar um módulo didático real, para o ensino de microcontroladores. O sistema proporciona uma interação entre o aluno e a placa didática, à distância, que é como se o aluno estivesse presencialmente no laboratório. Com o uso de *streaming* de vídeo, o aluno pode visualizar os dois displays de cristal líquido e a matriz de LEDs da placa. Além disso, pode realizar o pressionamento de *push buttons* (com *bouncing)*, gerar tensões para serem aplicadas na entrada do ADC, ver a tensão gerada na saída do DAC, e interagir com a porta serial do microcontrolador por meio de um console serial. O sistema proporciona uma forma de escrever o *firmware* para o microcontrolador da placa didática em linguagem C, utilizando apenas o navegador, sem a instalação de IDEs, compiladores e outras ferramentas. Cabe observar que utilizando o sistema proposto é possível realizar uma grande variedade de experimentos, pois além dos recursos internos do microcontrolador, pode-se utilizar as memórias EEPROM (I2C) e *flash* (SPI) disponíveis na placa, assim como outros dispositivos.

Levando em consideração os resultados obtidos nos testes do sistema, é aceitável afirmar que a aplicação atende às propostas iniciais de forma satisfatória. Um dos desafios analisados no começo do projeto foi o atraso existente na interação do aluno com o laboratório, devido a todos os obstáculos que um pacote de dados percorre desde o momento que é enviado pelo servidor, até o momento que é recebido e visualizado na tela do cliente. Mesmo assim, o sistema apresentou um atraso perfeitamente aceitável para o vídeo (1±0,5 s). Os outros componentes também apresentaram atrasos relativamente pequenos: 132,97 ms para a comunicação serial, 129,05 ms para os *push buttons* e 235,35 ms para o DAC/ADC. Portanto, pode-se afirmar que o sistema alcançou seu objetivo inicial, superando um dos maiores desafios analisados no levantamento de requisitos.

Inicialmente, o sistema desenvolvido será utilizado na disciplina que aborda microcontroladores, no curso de Engenharia de Computação, permitindo a realização de experimentos extraclasse. Assim, a carga horária da disciplina que pode ser ministrada à distância (10 h) poderá incluir aulas de laboratório. Além disso, o sistema permitirá a reexecução de experimentos realizados presencialmente, melhorando o entendimento e a fixação dos conceitos. Em um segundo momento, pretende-se utilizar o *feedback* dos alunos para implementar melhorias no sistema.

Dado o exposto, acredita-se que esse trabalho contribuiu com a área de ensino a distância, especificamente na área de laboratórios remotos.

References

Cooper, M. (2000). The challenge of practical work in an euniversity — real, virtual and remote experiments. In Information Society Technologies Conference: The Information Society for All, Proceedings of the, páginas 34–40.

Djordjevic, J., Nikolic, B., e Milenkovic, A. (2005). Flexible webbased educational system for teaching computer architecture and organization. Education, IEEE Transactions on, 48(2):264–273.

Eppes, T. e Schuyler, P. (2004). Work in progress - a distance laboratory system using agilent test equipment. In Frontiers in Education, 2004. FIE 2004. 34th Annual, páginas T3C/20–T3C/21 Vol. 1.

Kaderali, F., Steinkamp, G., e Cubaleska, B. (2001). Studying electrical engineering in the virtual university. Internal Journal on Engineering Education, páginas 119–130.

Ko, C. C., Chen, B., Hu, S., Ramakrishnan, V., Cheng, C. D., Zhuang, Y., e Chen, J. (2001). A web-based virtual laboratory on a frequency modulation experiment. Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on, 31(3):295–303.

Lombardi, A. (2015). WebSocket: Lightweight Client-Server Communications, página 1. O'Reilly Media.

Luthon, F., Petre, A., Steriu, D., e Besleaga, A. (2009). Laborem: open lab for remote work. In Signals, Circuits and Systems (SCS), 2009 3rd International Conference on, páginas 1–6.

MikroElektronika (2010). BIG8051 Development System User Manual. MikroElektronika. Disponível em: http://www.mikroe.com/big8051/ [visitado em 02/03/2015].

Mills, D. L. (1991). Internet time synchronization: the network time protocol. IEEE Transactions on Communications, 39(10):1482–1493.

Moreira, V. R. (2009). Plataforma em hardware reconfigurável para o ensino e pesquisa em laboratório de sistemas digitais a distância. Universidade Estadual de Campinas (UNICAMP). Faculdade de Engenharia Elétrica e de Computação. Dissertação de mestrado. Orientador Arantes, D. S.

Morton, J. (2001). Exploring the pic5x series. In PIC: Your Personal Introductory Course, IDC Technology, páginas 59–60. Newnes, 2ª edição.

Swamy, N., Kuljaca, O., e Lewis, F. (2002). Internet-based educational control systems lab using netmeeting. Education, IEEE Transactions on, 45(2):145–151.

Tan, K. K., Lee, T. H., e Soh, C. Y. (2002). Internet-based monitoring of distributed control systems-an undergraduate experiment. Education, IEEE Transactions on, 45(2):128–134.

Yue, X., Drakakis, E. M., Harkin, J., Callaghan, M. J., McGinnity, T. M, Maguire, L. P. (2009). Modular hardware design for distant-internet embedded systems engineering laboratory. Computer Applications in Engineering Education, 17(4):389–397.