VII IBERCHIP WorkShop
Montevidéu – Uruguai
Março - 2001

SISTEMA INTEGRADO E MULTIPLATAFORMA PARA CONTROLE REMOTO DE RESIDÊNCIAS

Fernando Gehm Moraes[1]
Alexandre Morais Amory [2]
Juracy Petrini Júnior [2]

Pontifícia Universidade Católica do Rio Grande do Sul (PUCRS)
Faculdade de Informática
Av. Ipiranga, 6681 - Prédio 30 / BLOCO 4
Telefone: +55 51 320-3611 - Fax: +55 51 320-3621
CEP 90619-900 - Porto Alegre - RS - BRASIL

Abstract

This work presents the implementation of a control system for house applications, belonging to a domain called domotics or smart house systems [1-2]. The approach combines hardware and software technologies. The system is controlled through the Internet, the home devices being connected using the control protocol. Users can remotely control their houses using a standard Web navigator, characterizing a multiplatform system. The remote application is the C*lient* program. Locally, instructions sent or received from the Client are translated by the server, which distributes the commands to a master controller. This master controller is a hardware bridge to the domestic applications. The implemented system has the following characteristics, which distinguish it from existing approaches [3-8]: (*i*) the client interface is automatically updated; (*ii*) a standard communication protocol is used in the hardware implementation, providing reliability and error control; (*iii*) new applications are easily inserted in the system database; (*iv*) system security is provided by user authentication; (*v*) user rights can be set up by an administration interface.

Resumo

O presente trabalho apresenta a implementação de um sistema de controle para aplicações domésticas, visando a automação de lares (também chamada domótica [1-2]), integrando *hardware* e *software*, e prototipado via utilização de dispositivos programáveis do tipo FPGA. Os dispositivos da casa são interligados pelo protocolo de controle (CAN). O sistema é acessível via Internet, sendo que os usuários (moradores) podem controlar e administrar seu lar a distância usando um navegador Web comum (multiplataforma). A aplicação remota é o programa Cliente. Localmente, o Servidor traduz as instruções de e para o navegador Web e as distribui para as aplicações domésticas. O sistema proposto neste trabalho apresenta como diferencial dos sistemas existentes [3-8] as seguintes características: (*i*) atualização automática da interface do Cliente; (*ii*) protocolo padrão de comunicação utilizado no *hardware*, o qual provê robustez e controle de erros; (*iii*) fácil inserção de novas aplicações no banco de dados do sistema; (*iv*) mecanismos de autenticação de usuários garantindo a segurança do sistema; (*v*) interface de administração dos usuários do sistema.

Palavras Chave : Integração hardware/software/Internet, domótica, teleação, VHDL, FPGA, Java, automação.

---

1 Doutor em Informática, opção Microeletrônica (LIRMM, França, 1994), Engenheiro Eletrônico (UFRGS, 1987), Professor Adjunto da Faculdade de Informática/PUCRS.
E-mail: moraes@inf.pucrs.br
2 Bacharel em Ciência da Computação, PUCRS, 2001.

1 Introdução

Domótica é a integração de tecnologias e serviços aplicados a lares, escritórios e pequenos prédios, com o propósito de automatizá-los obtendo um aumento de segurança, conforto, comunicação e economia de energia [1] [2]. A domótica pode substituir o homem em diversas atividades rotineiras de forma a propiciar uma otimização nas condições de vida em uma casa. O próprio sistema zela pela satisfação dos moradores, sem a necessidade da contínua intervenção dos mesmos. Teleação é a capacidade de se controlar algum dispositivo remotamente [2].

Unindo os dois conceitos acima descritos (domótica e teleação) surgiu a idéia de interligar a rede interna de uma casa (domótica) com a rede externa à casa (Teleação/Internet) de forma que os moradores da casa possam controlar e administrar seu lar a distância. Unimos neste projeto uma realidade atual (Internet e micro computadores domésticos) com uma tendência para o futuro (domótica).

Os benefícios de domótica são classificados em quatro grupos: segurança, conforto, economia de energia e comunicação. Segurança trata de proteger moradores e seus pertences em casos de eventualidades como invasão, vazamento de água, vazamento de gás, incêndio, doenças, etc. Como conforto adicional que o sistema traria para seus moradores podemos citar controle automático de iluminação e temperatura, programação e escalonamento de eletrodomésticos. Um sistema com controle inteligente pode evitar desperdícios de energia, contemplando o quesito de economia de energia. Já a comunicação trata da facilidade adicional que os moradores terão a respeito de comunicação através do uso de videofone, chamada automática a bombeiros e polícia e envio automático de mensagens eletrônicas.

Como exigências técnicas de um sistema de domótica podemos citar baixo custo (sistema de fácil instalação e dispositivos baratos), *plug and play*, flexibilidade (sistema modular e extensível), confiabilidade e fácil utilização. O sistema apresentado neste artigo contempla estes requisitos (alguns parcialmente) por se tratar de um sistema estruturado de forma modular, utilizando recursos padrão, tanto para software quanto para hardware.

Uma série de outros sistemas com propostas semelhantes a esse projeto foram encontradas na Internet. Entre eles citamos: HAI [3], Home Seer [4], HAL 2000 [5], Mister House [6] e Omnipotence[7], sendo que a referência [8] é voltada para automação industrial.

Tem-se por motivação para o desenvolvimento deste trabalho o projeto integrado de *hardware* (o protocolo de controle) e *software* (a comunicação Cliente/Servidor e a interface RS-232C), e o desenvolvimento da interface da aplicação (*home page* da casa). Outra motivação ao desenvolvimento deste trabalho é a criação de uma nova aplicação para computadores e Internet. Essa aplicação poderá contribuir para que o uso de computadores faça cada vez mais parte do dia-a-dia do usuário comum.

Esse artigo é organizado da seguinte forma. A Seção 2 descreve a arquitetura do sistema. As Seções 3 e 4 descrevem as implementações do software e hardware, respectivamente. A Seção 5 apresenta algumas conclusões e trabalhos futuros.

2 Arquitetura do Sistema

O sistema desenvolvido é baseado na comunicação entre dois ou mais computadores através do protocolo de comunicação HTTP. Um destes computadores é denominado de Servidor e os restantes denominados Clientes. Internamente à casa existe uma outra rede utilizando o protocolo CAN, responsável por interligar os diversos periféricos da casa. O protocolo *Controller Area Network* (CAN [17]) é um protocolo de controle para diversos ambientes industriais. Através de um Controlador Mestre (placa de aquisição CAN), esta rede interna comunica-se com o computador Servidor através de uma porta serial RS-232C. A Figura 1 ilustra a estrutura geral do sistema.

O computador Cliente é responsável por permitir que o usuário possa interagir com sua residência. Para que isso seja possível, há no computador Cliente um software responsável pelo envio de sinais de controle denominados pacotes de controle. Estes pacotes serão recebidos pelo Servidor que se encarrega de tratá-los de maneira adequada. Devido ao crescimento e popularização da Internet, o software utilizado no lado Cliente é um navegador Web. Sendo assim, o navegador Web é responsável em enviar os pacotes de controle ao Servidor através do protocolo HTTP, e ao mesmo tempo é responsável por manter a interface com o usuário atualizada à medida que o Servidor envia atualizações.

O Servidor é responsável por receber pacotes de controle do Cliente (Figura 1-1) que serão enviados a partir do navegador Web que estiver rodando na máquina Cliente. O Servidor por sua vez interpreta estes pacotes de controle

recebidos do Cliente, atualiza o banco de dados (Figura 1-4) e repassa estes pacotes de controle através da porta serial RS-232C (Figura 1-5) para o Controlador Mestre da rede interna da casa (Figura 1-6).

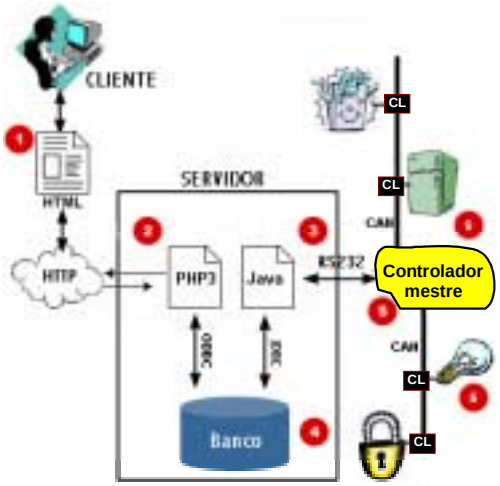

Figura 1 – Estrutura completa do sistema.

Um servidor Web Apache, instalado no Servidor, é responsável por disponibilizar a interface do(s) Cliente(s). Programas desenvolvidos em PHP (Figura 1-2) recebem os pacotes de controle enviados pelo usuário através do navegador Web, atualizando um banco de dados (Figura 1-4) contido no Servidor. Este banco de dados tem como principal função manter o estado atualizado sobre cada aplicação doméstica que está sendo controlada remotamente. Também é responsabilidade do Servidor a atualização, de maneira transparente ao usuário, da interface do navegador Web do Cliente. Este geração dinâmica de páginas é gerenciada pela aplicação escrita em PHP. Em outras palavras, isso possibilita que o usuário tenha sempre dados atualizados de sua residência sem a necessidade de ficar acionando um comando de atualização da interface (navegador).

Ainda no contexto do Servidor, existe uma aplicação escrita em Java (Figura 1-3), responsável pela comunicação com a porta serial RS-232C, possibilitando o envio e recebimento de pacotes de controle do Controlador Mestre. Outro objetivo da aplicação Java é garantir a consistência das aplicações domésticas da residência de acordo com o seu respectivo *status* no banco de dados, pois este último reflete diretamente os comandos de atualização enviados pelo usuário a sua casa.

A autenticação de usuários e encriptação dos dados são mecanismos utilizados para garantir a segurança do sistema, impedindo que pessoas desautorizadas tenham acesso ao mesmo.

O Controlador Mestre da rede residencial faz a codificação dos dados recebidos pelo Servidor para o protocolo CAN. Além disso, tarefas como endereçamento dos pacotes, gerenciamento do barramento e decodificação dos dados recebidos dos periféricos para o Servidor, também são tarefas do Controlador Mestre.

Uma rede serial com protocolo CAN é instalada na casa. Essa rede interliga todos os periféricos e nela trafegam os comandos e dados necessários para se atingir o objetivo de automatizar a casa. Cada periférico tem um Controlador Local (CL na Figura 1), que recebe (*ou* envia) comandos do (*ou* para o) barramento serial, controlado pelo Controlador Mestre. O protocolo CAN é implementado utilizando um IP C*ore* escrito em VHDL.

3 Implementação do Software

A implementação do software é dividida em dois componentes, conforme explicado na Seção anterior. De uma lado tem-se o Cliente, que envia/recebe pacotes de controle através da internet, utilizando o protocolo de comunicação HTTP (*HyperText Transfer Protocol*), e de outro lado o Servidor, que atua como uma interface entre os periféricos da casa e o Cliente.

3.1 Implementação do Cliente

A interface do Cliente é um navegador Web. Sua principal função é exibir ao usuário o *status* de sua residência, além de possibilitar a interação com este, permitindo que o mesmo possa executar comandos remotos na sua

residência, como por exemplo desligar a lâmpada da sala, capturar a imagem de vídeo de uma câmera na entrada da casa, ou receber mensagens de alarme.

Para o desenvolvimento da interface foram utilizadas três linguagens de programação: (*i*) HTML - usada para montar estaticamente a interface da aplicação no Cliente; (*ii*) Javascript - responsável por tornar a interface dinâmica a medida que o usuário interage com a mesma e para gerar atualizações na interface quando ocorrem mudanças na aplicação final (residência); (*iii*) PHP [9]- geração de páginas dinâmicas para atender a interação entre cliente-servidor.

A interface Web possui os seguintes frames: *Frame* 1 – Biblioteca de funções Javascript; *Frame* 2 – Programa PHP de leitura do banco de dados; *Frame* 3 – Programa PHP para atualização do banco de dados; *Frame* 4 – Menu de opções; *Frame* 5 – Aplicações encontradas na residência.

Os *frames* 1, 2 e 3 são invisíveis para o usuário, pois são responsáveis unicamente pelo processamento de informações que ocorrem em segundo plano na interface Cliente.

No *frame* 2 existe um programa desenvolvido em PHP cuja principal função é ler o banco de dados contido no Servidor, verificando quais dos registros estão desatualizados em relação ao computador Cliente. Com base nestes registros, o programa PHP gera comandos Javascript que atualizam a interface Web. Como se fosse uma *thread*, este programa é invocado de tempos em tempos garantindo a atualização da interface Web. Com a utilização de comandos Javascript para atualização da interface, esta ocorre de maneira automática sem que o usuário precise apertar em um botão "Atualizar" a todo instante, desde que esteja associada à constante monitoração automática do banco de dados. Esta é uma importante vantagem do sistema, uma vez que o usuário é automaticamente informado de tudo que é monitorado na residência, possibilitando, inclusive, o recebimento de mensagens de alarme na interface.

Devido à dinâmica utilizada para a atualização da interface Cliente, criou-se uma biblioteca de funções Javascript. Esta biblioteca está localizada no *frame* 1 e atualmente agrupa um número determinado de funções para as aplicações já desenvolvidas. À medida que novos controles da residência são implementados na interface, esta biblioteca Javascript deve agregar as novas funcionalidades necessárias.

A Figura 2 ilustra um exemplo de interface do Cliente, tendo como periféricos lâmpadas, *dimmers* e uma câmera de vídeo.

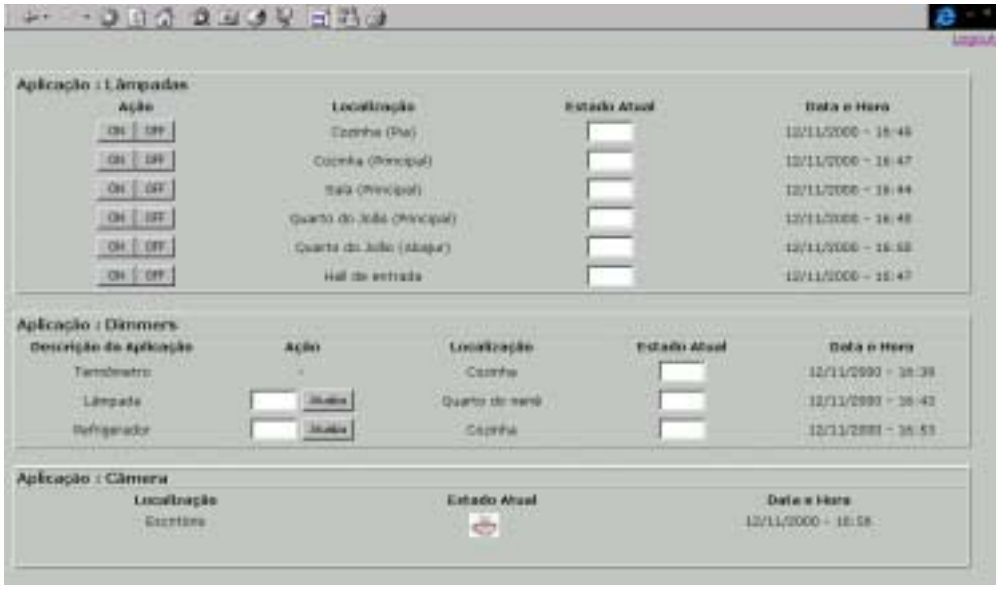

Figura 2 – Exemplo de interface do Cliente.

3.2 Servidor

O Servidor é o computador da residência e faz a interface entre o Cliente e os periféricos da casa. Dentre as funcionalidades realizadas pelo Servidor, citamos: (*i*) envio, recepção e interpretação de pacotes de controle através da porta RS-232 para o Controlador Mestre; (*ii*) monitoração do banco de dados em busca de alterações no *status* da aplicação; (*iii*) atualização do banco de dados; (*iv*) atualização dinâmica da interface no(s) Cliente(s).

As funcionalidades do Servidor são divididas em três conjuntos: comunicação com o Cliente, banco de dados e comunicação com o Controlador Mestre.

A comunicação com o Cliente ocorre através do protocolo HTTP. A distribuição das informações através deste protocolo é feita através de um servidor Web Apache [10] rodando no Servidor. Todas as requisições que partirem do Cliente são recebidas pelo servidor Web. O Servidor é responsável por gerenciar estas requisições e respondê-las ao longo do tempo.

A linguagem PHP é utilizada para gerar saídas de texto no formato HTML/Javascript que, quando enviadas através do protocolo HTTP ao Cliente (navegador Web), são interpretadas, montando a interface visual e um conjunto de funcionalidades oferecidas ao usuário.

A Figura 3 apresenta a estrutura de banco de dados utilizado no sistema, assim como seus respectivos relacionamentos.

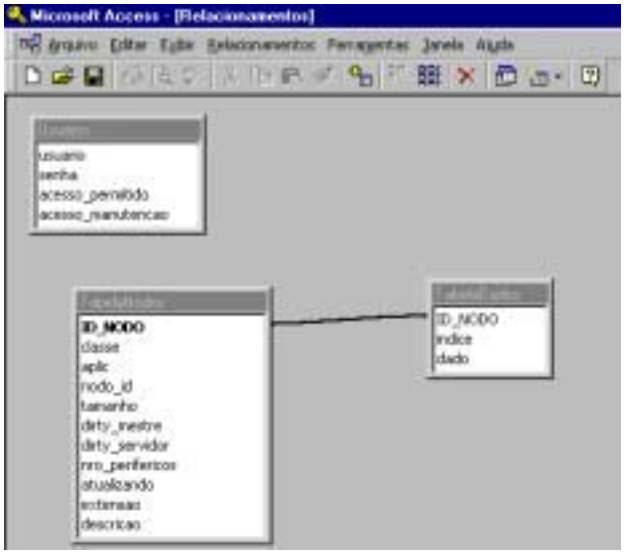

Figura 3 – Tabelas do Sistema.

A Figura 3 ilustra no canto superior esquerdo a tabela **Usuarios.** Sua principal função é armazenar os usuários que possuem permissão de acesso à *homepage* da residência e, consequentemente, o acesso às aplicações desta. Também indica quais usuários possuem acesso à interface de manutenção de usuários. Visando aumentar a segurança e a confiabilidade do sistema, a senha do usuário é criptografada e armazenada no banco de dados. O algoritmo de criptografia utilizado pelo sistema é o *md5*, com *fingerprint* de 128 bits [11].

A **TabelaDados** é responsável por armazenar os dados referentes às aplicações existentes na residência. Outra tabela existente no banco de dados é a **TabelaNodos**. Esta tabela é responsável por armazenar as informações referentes aos periféricos existentes na residência. O relacionamento criado entre as tabelas TabelaNodos e TabelaDados existe para permitir identificar quais são os dados pertencentes a cada aplicação (nodo).

Esta organização do banco de dados garante independência das aplicações na atualização dos dados. Em outras palavras, pacotes de controle que chegam ao Servidor são gravados diretamente no banco de dados, não importando que tipo de aplicação está enviando estes pacotes.

Para a comunicação com o Controlador Mestre desenvolveu-se um software em Java [12] [13] responsável pelo controle e atualização do banco de dados, juntamente com o controle da porta de comunicação RS-232. O acesso ao banco de dados acontece através dos métodos da biblioteca JDBC, utilizando o driver JDBC-ODBC, que é responsável por atualizar o banco de dados em relação aos periféricos da casa e vice-versa. Para a implementação da comunicação serial utilizando RS-232, utilizou-se o pacote desenvolvido pela *Sun Microsystems* denominado de *CommAPI* [14].

Todas as operações realizadas pelo software são executadas através da chamada de métodos de objeto em Java. A utilização da linguagem Java também garante a portabilidade do sistema entre diferentes plataformas. A Figura 4 ilustra o relacionamento dos métodos utilizados pela aplicação para a comunicação com o banco de dados e porta de comunicação RS-232.

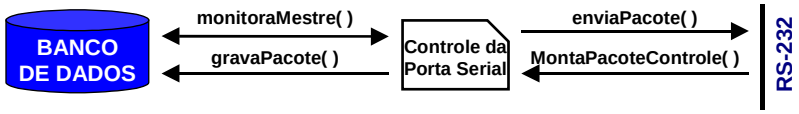

Figura 4 – Métodos de acesso ao banco de dados e à porta serial de comunicação RS-232.

O módulo de comunicação serial gerencia o envio e a recepção de pacotes de controle em paralelo. A qualquer momento o controlador mestre pode enviar pacotes. Utilizou-se *threads* para possibilitar a monitoração da porta serial, pois ao mesmo tempo em que a porta serial é monitorada, outras operações devem ser realizadas pelo *software*.

Para conseguir classificar os bytes que chegam na porta serial e montar um pacote de controle, seja ele de que tipo for, criou-se um método chamado de *montaPacoteControle*(), o qual sempre recebe o byte proveniente da porta serial. Internamente a este método, mecanismos de controle e sequenciamento foram implementados para que o pacote de controle seja identificado dentro da seqüência de bytes que chegam. Atualmente classificamos os pacotes como sendo de dois tipos: (*i*) pacotes pequenos – pacotes com até 8 bytes de tamanho; (*ii*) pacotes grandes – pacotes maiores que 8 bytes.

No módulo de acesso ao banco de dados existem dois eventos que necessitam que ocorra o acesso automático aos dados (registros): (*i*) chegada de pacotes através da serial; (*ii*) envio de pacote através da porta serial. Como conseqüência da chegada de pacotes de controles da casa, necessita-se atualizar o banco de dados com as informações provenientes deste pacote. Visando alcançar este objetivo, criou-se um método denominado *gravaPacote*(). Internamente a este método é realizada uma atualização do registro, localizado na estrutura TabelaNodos. Criou-se o método *monitoraMestre*() cuja função principal é procurar no banco de dados aplicações que estejam com o status desatualizado em relação ao controlador mestre e enviar estas informações através da porta serial, utilizando o método chamado *enviaPacote*().

4 Implementação de Hardware

A crescente demanda de comunicação, conectividade e fluxo de informações impulsionou a criação de vários protocolos de comunicação. O estudo das características de protocolos depende muito da aplicação destes. Por exemplo, se procurarmos um protocolo para ser usado em uma grande indústria metalúrgica esse protocolo deve ter longo alcance para poder se estender por toda empresa, poder ligar vários nodos, ter grande imunidade a ruídos usando técnicas de detecção e correção de erros, ter alta disponibilidade e tolerância a falhas, uma vez que grandes valores e vidas humanas podem estar em jogo. Já um protocolo para aplicação de entretenimento, basicamente usado somente dentro do ambiente doméstico, não necessita ter alta disponibilidade, precisa ter grande taxa de transferência para transportar vídeo e som, um alcance curto é o suficiente, compatibilidade entre vários fabricantes é necessária, etc.

Os dois exemplos citados acima, de uma certa forma ortogonais, nos mostram que as características e requisitos de um protocolo podem variar muito, dependendo exclusivamente de sua aplicação. Porém, conhecendo-se apenas a aplicação não se faz uma escolha adequada de protocolo. A seguir descrevemos mais algumas características técnicas que devem ser avaliadas: (*i*) a relação custo/benefício da técnica de cabeamento. A técnica de cabeamento e conexão é um dos itens que mais influenciam o custo total de instalação de uma rede; (*ii*) *plug and play*, confiabilidade, disponibilidade, flexibilidade, compatibilidade, sincronização e padronização; (*iii*) variabilidade de aplicações – o mesmo protocolo pode ser empregado em aplicações diferenciadas?; (*iv*) existência de protocolos de nível de aplicação; (v) ferramentas para teste e diagnóstico, interface com PC e *drivers* de *hardware* e *software*; (*vi*) taxa de comunicação, resposta em tempo real, técnica de acesso ao meio, topologia, comprimento físico máximo da rede e número de bytes de dados; (*vii*) a rede deve ser multi-mestre ou com um único mestre? suporta *broadcasting* e *multicasting*? suporta requisição remota de dados e reconhecimento ? (*viii*) maior aceitação no mercado e potencial de crescimento.

Com o objetivo de encontrar um protocolo de controle que atendesse os requisitos citados acima, analisou-se uma série de protocolos de controle dentre eles EHS, BDLC, LON, EIB, X-10, CeBus e CAN. Detalhes sobre a escolha do protocolo e uma tabela comparativa entre estes protocolos pode ser encontrada em [1].

4.1 Controller Area Network (CAN)

Dentre os protocolos estudados, o *Controller Area Network* [15][16][17] foi o protocolo que atendeu o maior número de itens e o que se adaptou melhor ao desenvolvimento deste trabalho.

CAN é um protocolo de comunicação serial que suporta controle em tempo real distribuído, com um grande nível de segurança. Foi originalmente desenvolvido para aplicações automotivas, mas hoje em dia já existem outras aplicações variando de sistemas de controle industrial a sistemas de aplicação específicas como satélites artificiais e sistemas médicos. Neste trabalho propomos o emprego de CAN a uma nova aplicação. Essa aplicação é um sistema de supervisão de domótica via *web*, servindo o protocolo como rede de controle da casa.

Dentre as principais características do protocolo CAN citamos: (*i*) método de acesso ao barramento com priorização de mensagens; (*ii*) taxa de transferência de até 1Mbps; (*iii*) garantia do tempo de latência (tempo real); (*iv*) flexibilidade de configuração; (*v*) recepção *multicast* com sincronização; (*vi*) várias técnicas para manter consistência de dados e sinalização de erros; (*vii*) multi-mestre; (*viii*) técnicas de detecção e sinalização de erros; (*ix*) desligamento automático de nodos com defeitos; (*x*) possui técnica de requisição remota de dados; (*xi*) possui até 8 bytes no campo de dados.

A Figura 5 ilustra o formato do pacote de dados CAN.

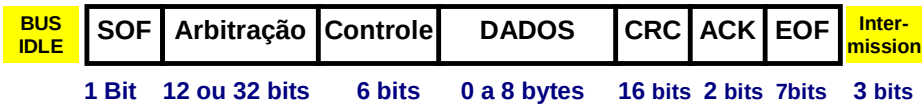

Figura 5 – Formato de um pacote de dados CAN.

Para a implementação deste trabalho foi adaptado um IP Core chamado HurriCANe que implementa o protocolo CAN. Esse IP Core foi descrito em VHDL e desenvolvido na ESA (European Space Agency) [18].

A Figura 6 ilustra a estrutura hierárquica do HurriCANe com a ocupação de *slices* (bloco lógico básico do FPGA) para um FPGA xcv300-5-bg352 da família Virtex 300 da Xilinx. Detalhes de como esse IP Core foi utilizado neste trabalho são encontrados na referência [1].

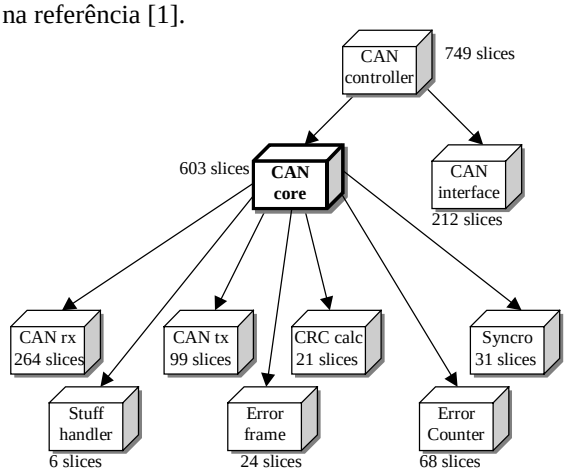

Figura 6 – Estrutura do HurriCANe.

O módulo **Interface** provê ao usuário do *core* uma interface padrão entre dispositivos, baseando-se em comandos de escrita (transmissão) e leitura (recepção). Esse módulo é especialmente importante caso use-se o HurriCANe associado a um processador comercial, pois esta interface facilitará o controle e o tráfego de dados entre a aplicação do usuário e o *core* CAN. O módulo **CAN Core** é o principal módulo do protocolo. Ele interliga todos os módulos que estão abaixo na hierarquia. Pelo fato de não ter sido proposto neste projeto o desenvolvimento da interface entre CAN e um processador, *usou-se somente este módulo*, CAN core.

4.2 Nodos Escravos

Os nodos Escravos, também chamados de nodos de aplicação, são os nodos que estão distribuídos pela casa responsáveis por controlar periféricos variados. Os nodos Escravos variam entre si de acordo com sua aplicação. Podemos ter, por exemplo, um nodo que controla lâmpadas, sensores de temperatura, entre outras aplicações possíveis.

Cada aplicação controlada existente na casa deve possuir um suporte tanto em hardware quanto em software. Por exemplo, para se fazer o controle de lâmpadas é necessário existir no sistema módulos (hardware) que fazem a

interface das lâmpadas com o barramento CAN, e métodos (software) no Servidor para gerenciar lâmpadas. Foi implementado para este projeto uma aplicação de lâmpadas, que demonstra o funcionamento do sistema.

A aplicação de lâmpadas foi implementada em VHDL [19] em uma plataforma de prototipação XS40 [20] com FPGA xc4010-3-EPC84 com capacidade de cerca de 10.000 portas lógicas. Esta aplicação suporta até 64 lâmpadas, uma vez que o protocolo CAN possui 8 *bytes* de dados (8*bytes* x 8*bits*). Cada bit do campo de dados representa o estado de uma lâmpada (acesa ou apagada). Mais detalhes da arquitetura dos nodos de lâmpadas são encontrados na referência [1].

Dentre as informações geradas no relatório de síntese da ferramenta Xilinx Foundation em relação a um FPGA xc4010e-3-pc84, destacamos a área em número de CLBs (294 CLBs com 73% de utilização total) e o relatório de temporização com uma freqüência máxima (9.018MHz) que essa descrição suporta.

4.3 Nodo Mestre

O Nodo Mestre é responsável por fazer a interface entre a rede CAN e o Servidor. Ele foi implementado na plataforma de prototipação VW300 [21] com um FPGA xcv300-5-bg352 [22], possuindo 3072 slices (cerca de 300.000 portas lógicas equivalentes).

A arquitetura do nodo Mestre pode ser visualizada na Figura 7. Ela é baseada em um sistema de filas pelo fato das duas interfaces (serial e CAN) terem velocidades de transmissão diferentes. Quando a interface CAN recebe um pacote, o mesmo é dividido em até 13 bytes e gravado byte a byte na *fila CAN*. Enquanto a fila CAN não está vazia a interface serial transmite dados desta fila pela serial. De forma análoga, quando a interface serial recebe algum dado, a mesma o grava na *fila Serial*. Enquanto essa fila não está vazia, a interface CAN transmite dados pelo barramento CAN para que o nodo destino possa recebê-lo.

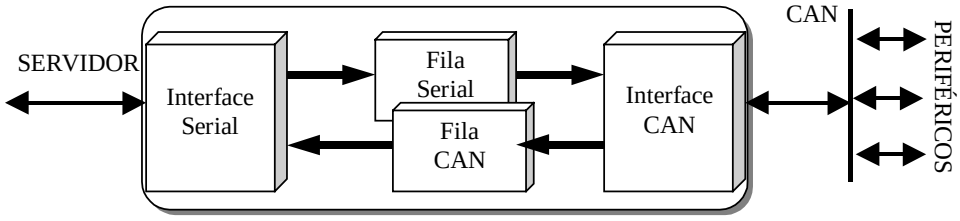

Figura 7 – Arquitetura do nodo Mestre.

As filas da interface CAN e serial são implementadas utilizado memórias RAM internas ao FPGA chamadas *Block Select Ram* [23]. Nessa implementação o *Block Select Ram* está configurado para ser uma memória *dual port* (sendo uma porta somente de escrita e outra somente de leitura) de 8 bits de tamanho de palavra, com 512 palavras

A interface serial é responsável por enviar/receber dados ao/do Servidor. Esse módulo funciona sem paridade, com 1 *stop bit* e com taxa de transmissão de 19200 bps expansível até 115200 bps.

A interface CAN é composta pelo IP Core HurriCANe com alguma lógica adicional específica para essa implementação.

A Tabela 1 apresenta um resumo com as principais informações geradas durante o processo de síntese do nodo Mestre, utilizando o sistema de CAD Foundation, fornecido pela XILINX. Esses dados foram gerados levando-se em consideração a utilização do FPGA xcv300-5-bg352, sem uso de esforço máximo de síntese.

Módulos	Ocupação (Slices)	Ocupação (%)	Freqüência Máxima (MHz)
Mestre	731	23	26.134
Interface CAN	512	16	25.028
Interface Serial	65	2	85.434
Fifos	116	3	37.448

Tabela 1 – Resumo de relatórios de síntese do Mestre (freqüências obtidas por sínteses independentes).

O pacote CAN recebido ou enviado pelo Controlador Mestre é dividido em até 13 pacotes de um byte no protocolo serial. Os byte 1 a 4 formam o identificador da mensagem, o byte 5 é de controle e os 8 bytes restantes contém os 8 bytes de dados do protocolo CAN.

4.4 Protocolo de Aplicação

O protocolo CAN possui um limitação para ser aplicado a domótica. Ele transmite somente até 8 bytes de dados. Para que aplicações que contém pacotes de dados com tamanho superior a 8 bytes é necessário acrescentar um nível de abstração superior ao protocolo CAN, denominado de protocolo de aplicação. Os autores propõem neste trabalho uma forma de transmitir informações complexas, tais como áudio, imagens e vídeo utilizando CAN. A idéia é dividir os pacotes CAN em dois grupos: pacotes de dados e de controle. Diferenciam-se pacotes de dados e pacotes de controle pelo número de identificação do nodo ou serviço (Id, inserido no campo Arbitração da Figura 5). Ids ímpares identificam mensagens de controle, pares identificam mensagens de dados.

Os pacotes de dados são os pacotes normalmente transmitidos pela rede. Esse pacote possui o limite de 8 bytes de dados. Os pacotes de controle são serviços especiais do sistema que solicitam ou informam alguma característica específica para determinada aplicação. Por exemplo, uma câmera de vídeo pode enviar um pacote de controle para o Servidor informando que começará a transmitir uma imagem de 100 Kbytes. Outro possível exemplo seria a transmissão de uma mensagem de controle do Servidor para um nodo de lâmpadas solicitando um auto teste do nodo.

Vale ressaltar que com essa técnica a compatibilidade com o protocolo CAN foi preservada. As alterações refletem-se em um nível de abstração acima do CAN Core, nível este onde são implementadas as aplicações que controlam os periféricos em questão.

Essa abordagem possibilitou uma *grande flexibilidade* ao sistema. Quando for um pacote de controle, dos 8 bytes de dados, 1 é utilizado para identificar o serviço e 7 para identificar o(s) parâmetro(s) de execução. Assim, há suporte para a criação de até 256 serviços para cada aplicação existente no sistema, sendo que cada serviço pode ter até 7 bytes de parâmetros. Por exemplo, se o serviço requer mais de 8 bytes de informação, e utiliza os 7 bytes restantes par especificar o tamanho (em bytes) da informação a ser transmitida, o *hardware* suporta uma transmissão teórica de até 2^56 (8bits*7bytes) bytes de dados.

4.5 Validação

Para depuração do sistema de hardware foi utilizado o Simulador VHDL da Mentor Graphics chamado QHSIM, osciloscópio Tektronix TDS 220 e analisador lógico HP 16663E. A plataforma de teste de síntese usada para validar sistema de *hardware* é apresentada na Figura 8. Os seguintes recursos foram usados: (*i*) um micro-computador que funciona como o Servidor do sistema, comunicando-se com o Controlador Mestre pela porta serial; (*ii*) osciloscópio que monitora o sinal no barramento CAN; (*iii*) analisador Lógico que monitora sinais do Mestre usados para depuração; (*iv*) placa de prototipação VW300 [21] onde a descrição VHDL do Controlador Mestre é executada; (*v*) Placa de prototipação XS40 [20] onde a descrição VHDL da aplicação de lâmpadas é executada.

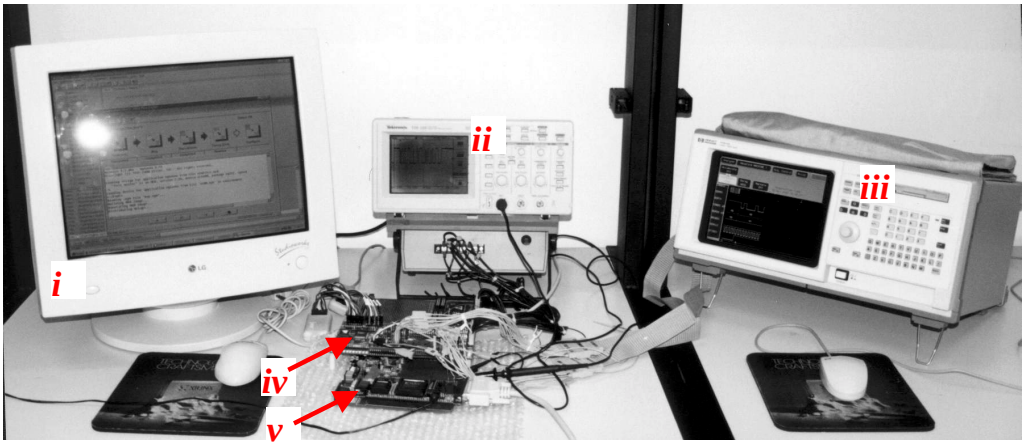

Figura 8 – Ambiente de teste do Controlador Mestre

5 Conclusões e Trabalhos Futuros

Este trabalho apresentou a implementação de um sistema de domótica que possibilita o controle remoto de residências com o uso de um navegador web. Este trabalho integrou diversas tecnologias em níveis de abstração

variados (hardware/software/Internet). O hardware foi totalmente implementado em VHDL, adaptando um IP Core que implementa o protocolo CAN. O software foi desenvolvido em Java com métodos de acesso a interface serial para comunicação com hardware e métodos para acesso ao banco de dados. A Internet foi empregada a partir de aplicações desenvolvidas nas linguagens HTML e PHP, responsáveis pela implementação da interface do sistema que roda em um navegador web.

Além de integrar hardware e software, o sistema é multiplataforma, pois o Cliente necessita apenas um navegador *web* para controlar sua residência. O servidor também é multiplataforma, pois o software Apache é disponível para diferentes arquiteturas, além de utilizarmos linguagens padrão, como Java e PHP.

Como proposta para trabalhos futuros pode-se citar: (*i*) a mudança da interface de comunicação entre hardware e software de serial para PCI; (*ii*) o serviço que implementa um protocolo *Plug and Play;* (*iii*) para verificar o correto funcionamento dos nodos de aplicação distribuídos pela casa sugerimos técnicas de *keep alive*, testabilidate remota usando auto teste (BIST); (*iv*) serviço de reconfiguração remota usado para atualização de versão e manutenção dos nodos de aplicação; (*v*) mecanismo de *time to live* para pacote segmentados (dados com mais de 8 bytes); (*vi*) controle de acesso Web com múltiplas permissões possibilitando desabilitar a interação de, por exemplo, crianças com o sistema de segurança.

Agradecimentos

Agradecemos as contribuições dadas por Eduardo Bezerra (E.A.Bezerra@sussex.ac.uk) a este trabalho e a Luca Stagnaro (lstagnaro@estec.esa.nl), da Agência Espacial Européia (ESA), por ter cedido o IP Core HurriCANe. O autor Fernando Gehm Moraes agradece o suporte do CNPq (projeto integrado número 522939/96-1) e da FAPERGS (projeto número 96/50369-5).

Bibliografia

[1] Moraes, Fernando; Amory, Alexandre M.; Petrini, Juracy Jr. Sistema Cliente-Servidor para Supervisão de Processos através da Web. trabalho de conclusão do curso de informática da PUCRS. dezembro de 2000. 167p. [http://www.ee.pucrs.br/~amory/Trabalho_de_Conclusao/trabalho_de_conclusao.html]

[2] F. Baumann; B. Jean-Bart; A. Kung, P. Robin. Eletronic Commerce Services for Home Automation. Trialog. [http://www.trialog.com/emmsec9-97.pdf].

[3] Home Automation, Inc. HAI Web-Link Software Homepage. [http://homeauto.com/web-link/index.htm].

[4] Keware Tecnologies. HomeSeer Software. [http://www.homeseer.com].

[5] Home Automated Living. HAL 2000 Software. [http://www.automatedliving.com/].

[6] MisterHouse. Home Automation Software. [http://www.misterhouse.net/].

[7] Omnipotence Software. Event Control System - ECS. [http://www.usit.com/omnip/home-automation-1.htm].

[8] Brandão, Eduardo Scharnberg. Sistema Cliente-Servidor para Supervisão de Processos através da Web. trabalho de conclusão do curso de informática da UFRGS. fevereiro de 1999. 57p.

[9] PHP Hypertext Preprocessor. [http://www.php.net].

[10] The Apache Software Foundation. [http://www.apache.org]

[11] The MD5 Message - Digest Algorithm. [http://www.faqs.org/rfcs/rfc1321.html]

[12] Cornell, Gary; HorstMann, Cay S. Core Java Second Edition. Califórnia, SunSoft Press, 1997.

[13] Wigglesworth, Joe; Lumby, Paula – Java Programming Advanced Topics. Thomson Learning, 2000.

[14] Java(TM) Communications API. [http://www.java.sun.com/products/javacomm/index.html]

[15] Bosch CAN Homepage. [http://www.bosch.de/k8/can/].

[16] ROBERT BOSCH GmbH. CAN Especification Version 2.0. Stuttgart. 1991. [http://www.bosch.de/k8/can/docu/can2spec.pdf].[http://www.can-cia.de/CAN20B.pdf].

[17] Lawrenz, Wolfhard. CAN System Engineering From Theory to Practical Applications. 1997. 468 pp.

[18] HurriCANe Home Page. [ftp://ftp.estec.esa.nl/pub/ws/wsd/CAN/can.htm].

[19] Mazor, Stanley; Langstraat, Patricia. “A guide to VHDL”. Boston : Kluwer Academic, 1996. 250p.

[20] XESS Inc. XS40 Board V1.4 User Manual . XESS Corporation. [http://www.xess.com/manuals/xs40-manual-v1_4.pdf].

[21] VCC Inc. The Virtual WorkBench Guide. Virtual Computer Corporation. [http://www.vcc.com].

[22] Xilinx Inc. Virtex 2.5V Field Programmable Gate Arrays (DS003). Virtex Series.

[23] Xilinx Inc. Using the Virtex Block SelectRAM+ Features (Application Note XAPP130). Virtex Series.