CENÁRIO DE CONVERGÊNCIA ENTRE A TV DIGITAL E DISPOSITIVOS ELETRÔNICOS EM SISTEMAS DE AUTOMAÇÃO RESIDENCIAL

Vandermi S. SILVA (1); Nairon S. VIANA (2); Orlewilson B. MAIA (3); Alexandre Lopes MARTINIANO (4); Vicente F. de LUCENA Jr. (5)

(1) UFAM - PPGI – Universidade Federal do Amazonas; Departamento de Ciência da Computação (DCC); Programa de Pós-Graduação em Informática, Av. General Rodrigo Otávio Jordão Ramos, 3000; CEP 69077-000; Manaus – AM; Tel.: (92)3647-4019 / (92)3647-4022 e-mail: vandermi@ufam.edu.br
(2) UFAM - PPGEE – Faculdade de Tecnologia (FT); Programa de Pós-Graduação em Engenharia Elétrica, Tel.: (92)3647 4495, Fax.: (92)3647 4494; e-mail: nairon_viana@ufam.edu.br
(3) UFAM - PPGEE – e-mail: orlewilson_maia@ufam.edu.br
(4) UFAM - PPGEE – e-mail: alexandrelopes@ufam.edu.br
(5) UFAM, CETELI e CEFET-AM – Centro de Pesquisa e Desenvolvimento em Tecnologia Eletrônica e da Informação; Centro Federal de Educação Tecnológica do Amazonas, e-mail: vicente@ufam.edu.br ou lucena@cefetam.edu.br

RESUMO

Este trabalho apresenta um cenário de convergência entre a TV Digital (TVD) e dispositivos eletrônicos em Sistemas de Automação Residencial, envolvendo o monitoramento de temperatura em ambiente controlado, e propõe uma arquitetura em que a TVD atua centralizando o processamento de uma rede de sensores. É descrito o processo de integração entre as tecnologias envolvidas, através do estudo de todos os níveis do sistema proposto. Em um mesmo ambiente, são contempladas tecnologias de *software,* especificações de *middlewares* para TVD (MHP) e *frameworks* para gerência de aplicativos em Redes Residenciais (OSGI), tecnologias de comunicação e formatação de dados usando a *Java Communication* API (JCA) e XML, até o nível de dispositivos, com o estudo e configuração de módulos embarcados para sistemas de monitoramento de temperatura, utilizando as plataformas *Texas Instruments* MSP-430 e *MCs-51*. O objetivo final do trabalho é apresentar o cenário desenvolvido e o processo de análise e configuração dos componentes de *hardware* e *software*, conversão e envio de dados e construção de componentes de *software* em acordo com as especificações MHP e OSGI, destacando principalmente a estratégia adotada para a utilização dessas tecnologias num modelo que dá suporte à operação da TVD em um ambiente típico de Automação Residencial.

Palavras-chave: Automação Residencial, TV Digital, Dispositivos Eletrônicos, Convergência.

1. INTRODUÇÃO

Em um sistema de Automação Residencial, vários dispositivos, como sensores de portas, temperatura e controle de acesso, são integrados em uma interface única que permite ao usuário o controle das operações que podem ser realizadas por cada dispositivo. O que se observa atualmente é a disponibilidade de uma variedade de tecnologias para esses dispositivos, sem, no entanto, haver um padrão comum que facilite o desenvolvimento de aplicações que os integre em um mesmo ambiente residencial. Mais ainda, não há um padrão aberto que seja aceito por todos os fabricantes dessas tecnologias. Algumas entidades da indústria definem especificações, no sentido de promover compatibilidade entre tecnologias heterogêneas. Como exemplos, pode-se citar UPnP (*Universal Plug and Play*) e OSGI (*Open Service Gateway Initiative*), que especificam modelos para a gerência e configuração de dispositivos de uma maneira comum e ambientes de redes automação e redes residenciais.

Um dos dispositivos eletrônicos mais comumente presente no cotidiano do usuário, principalmente nas residências brasileiras, é a televisão. O uso da TV atinge uma vasta gama de telespectadores que, com o advento da TV Digital Interativa (TVDI), agora são chamados usuários-telespectadores, pois podem utilizar a TV também como plataforma computacional, um ambiente para a execução de aplicativos, além da exibição de áudio e vídeo com qualidade digital. Para a TV Digital (TVD), algumas entidades também definem padrões ou modelos de referência no intuito de promover compatibilidade entre tecnologias, aplicações e as plataformas de execução (STB, *Set-Top Box*). Entre os principais modelos de referência podem ser citados: o europeu DVB (*Digital Video Broadcasting*); o americano ATSC (*Advanced Television Systems Comittee*); o japonês ISDB (*Integrated Services Digtal Broadcasting*) e, mais recentemente, o brasileiro ISDTV (*Internartional Standards for Digital Television*).

O suporte para a execução de aplicações e interatividade da TV Digital proporciona uma variedade de serviços ao usuário-telespectador. Dentro de um ambiente de automação residencial, entretanto, são relevantes apenas as aplicações que possam integrar tecnologias de TVD com outros dispositivos. A princípio isso não é uma tarefa fácil, exatamente pela grande quantidade de tecnologias disponíveis e a falta de um padrão que dê suporte a interoperabilidade das aplicações entre plataformas.

Alguns trabalhos exploram a idéia de integração (ou convergência) entre a TVD e dispositivos eletrônicos. Para isso, investigam modelos novos a nível de *software* que possam, dentro de um padrão comum, permitir que aplicações de TVD acessem e modifiquem dispositivos numa rede residencial. No trabalho apresentado em (MATSUBARA, 2005) é proposta uma arquitetura que centraliza na TVD funções de controle dos dispositivos da rede residencial e gerenciamento do conteúdo em formato heterogêneo proveniente de cada dispositivo, baseada nas tecnologias UPnP e DLNA (*Digital Living Networking Alliance*) e nos protocolos IP e IEEE 1394.

Nesse sentido, o presente trabalho traz a idéia do uso da TVD como centro de processamento de informações em uma rede residencial. O trabalho propõe um esquema simplificado para o acesso às informações de dispositivos (sensores de temperatura) por meio de aplicativos de TVD, baseando-se em esquemas XML. O trabalho é organizado como segue: na seção 2 são apresentadas separadamente cada tecnologia utilizada; a seção 3 descreve a arquitetura de um cenário de uso dessas tecnologias; na seção 4 são enumerados os módulos da implementação do trabalho e as ferramentas utilizadas; na seção 5 é apresentado o resultado do experimento; e finalmente na seção 6, as conclusões do trabalho.

2. TECNOLOGIAS PARA A CRIAÇÃO DO CENÁRIO

Para o desenvolvimento de uma arquitetura de convergência entre a TVD e Sistemas de Automação Residencial, utilizou-se tecnologias de *software* (MHP e OSGI), comunicação (JCA e XML) e *hardware* (plataformas MSP-430 e MCs-51). Nas subseções a seguir serão descritos com mais detalhes essas tecnologias.

2.1. MHP

Em uma arquitetura do sistema de TV Digital, o *middleware* consiste de um conjunto de padrões que oferecem uma interface que auxilia a comunicação entre duas camadas: camada de aplicações e camada de *hardware*, conforme mostrado na Figura 1. Através desses padrões, facilita-se o desenvolvimento de aplicações para a TV Digital, pois estes permitem o acesso a recursos disponíveis pelo *hardware* do dispositivo, como *drivers*, sistema operacional e aplicações nativas.

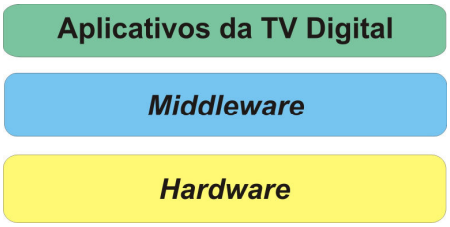


Figura 1 – Arquitetura simplificada de um sistema de TV Digital

Dentre os principais *middlewares* existentes no mundo, destacam-se: o europeu MHP (*Multimedia Home Platform*), o americano DASE (*Digital TV Application Software Environment*), o japonês ARIB (*Association of Radio Industries and Business*) e o brasileiro GINGA. Para representar o lado da TV Digital Como middleware da implementação deste trabalho foi adotado o MHP pelo fato de utilizar a linguagem de programação Java para desenvolver os aplicativos e por existirem emuladores *open source* para realizar testes dos aplicativos.

Os componentes básicos da especificação MHP estão bem definidos: um miniaplicativo (*Xlet*) que atua sobre o *middleware* no STB é responsável por realizar as ações básicas no ambiente de execução, em interação com o Gerenciador de Aplicações (*Application Manager*) por meio de uma entidade *Xlet- Context*, tendo acesso às funcionalidades de um conjunto de APIs padronizadas [MORRIS E SMITH-CHAIGNEAU, 2004]. Definido sob um modelo de uma máquina de estados, os *Xlets* (interfaces Java) têm seu comportamento controlado por um Gerenciador de Aplicações e possuem quatro estados: carregado, pausado, ativo e destruído (Figura 2). Em cada estado, o *Xlet* realiza operações em um contexto específico, como ativar uma conexão com a Internet, enviar uma mensagem de texto, apresentar opções de menu ao usuário ou liberar recursos para o ambiente. *Xlets* também podem interagir com outros *Xlets*, através da entidade *IXC Registry*.

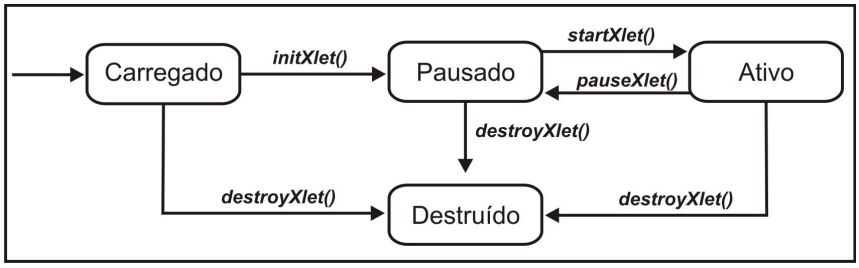


Figura 2 – Ciclo de vida de um *Xlet* (modificado de (CALDER *et al*, 2008))

2.2. OSGI

No campo de Sistemas de Automação Residencial, podem-se encontrar várias abordagens de comunicação entre dispositivos, como o acesso a informações de uma geladeira e envio para o celular através do *Bluetooth* (KANMA *et al.*, 2003) ou o controle e monitoramento de algum dispositivo doméstico (lâmpada, ventilador) utilizando a interface de uma página da Web (KIM *et al.*, 2006; KIM *et al.*, 2007). Porém para cada uma dessas abordagens é necessário um esforço dispendioso para desenvolver a comunicação entre esses dispositivos, pois normalmente são de diferentes fabricantes além da variedade de tecnologias de comunicação, como *Bluetooth*, Zigbee, Wi-Fi, UPnP, etc.

Diante dessa situação, um grupo de entidades entre empresas, instituições de ensino e pesquisa, criaram o OSGI (*Open Services Gateway Initiative*) com o objetivo de permitir uma comunicação de forma transparente entre vários dispositivos mesmo que sejam de diferentes fabricantes (OSGI, 2008). A especificação OSGI inclui um *framework* para o carregamento e gerenciamento de aplicativos encapsulados, denominados *bundles*. Um *bundle* é um conjunto de classes Java e outros recursos, denominados serviços, organizados em um arquivo *Java ARchive* (JAR). Cada serviço corresponde ao controle de um dispositivo específico da rede residencial, ou de alguma parte dele. Um *bundle* apresenta um ciclo de vida, definido como máquina de estados (instalado, desinstalado, pronto, iniciando, parando e ativo) (ver Figura 3) em um contexto representado pelo componente *BundleContext*, interagindo com uma entidade denominada *Service*

Registry. É através do *Service Registry* que os *bundles* notificam o ambiente da disponibilidade de seu serviço, bem como verificam as referências para os serviços providos por outros *bundles* (MARPLES E KRIENS, 2001).

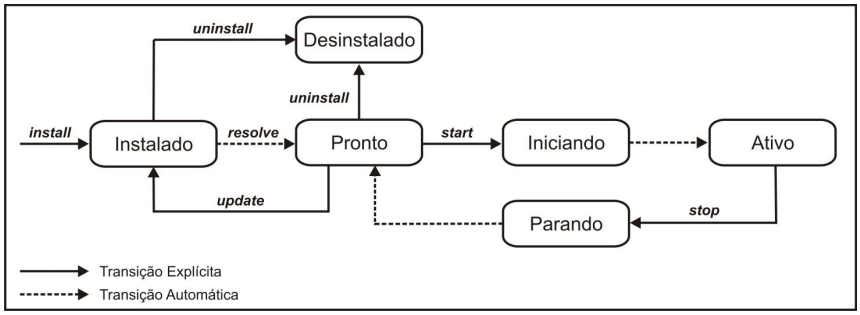


Figura 3 – Ciclo de vida de um *Bundle* (modificado de (MARPLES E KRIENS, 2001))

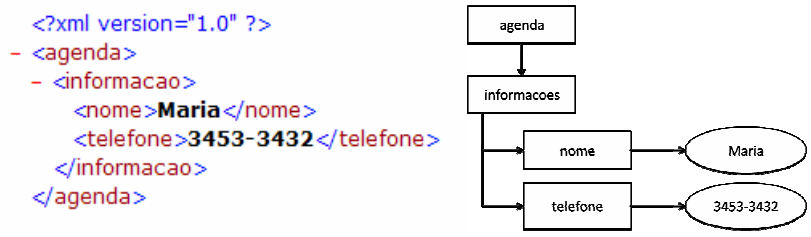
Há vários benefícios de se utilizar o OSGI como: facilidade para o desenvolvimento de soluções para Sistemas de Automação Residencial, suporte a vários tipos de comunicação, suporte a integração com outros componentes, além das vantagens decorrentes de o esquema OSGI ser baseado na tecnologia Java e no modelo orientado a serviços.

2.3. JCA

A JCA é uma extensão do Java que facilita o desenvolvimento de aplicações de plataformas de comunicação independente como *Smart Cards*, sistemas embarcados, fax, modems, terminais de acesso e robótica (SUN, 2008). Através dessa API, podem-se criar aplicativos para acessar informações enviadas através de portas seriais (RS-232) e paralelas (IEE-1284). Podem ser destacadas algumas características da JCA, como*:* Enumeração das portas; Configuração da porta (como velocidade, paridade, taxa de *baud*, *bit* de parada); e Opção de evento assíncrono para notificação de dados disponíveis.

2.4. XML

XML (*Extensible Markup Language*) é uma linguagem semi-estruturada que permite a padronização da descrição de dados de uma forma que eles se auto-descrevem, ou seja, um documento XML pode ser lido e seu conteúdo entendido de forma natural. A estrutura hierárquica de um documento XML é representada através de uma árvore. Na Figura 4 é mostrado um exemplo de documento XML e a sua hierarquia em forma de árvore.



(A) **(B)**

Figura 4 – Exemplo de documento XML (A) e sua hierarquia (B)

Os documentos XML devem ser bem-formatados, ou seja, o documento deve obedecer às especificações XML. Além de bem-formatado, um documento XML pode ser válido, ou seja, referenciar um documento externo que determina algumas regras que devem seguidas, como qual o tipo de conteúdo permitido de um elemento, a seqüência de elementos que deve ser seguida e assim por diante. Para criar essas regras, pode-se utilizar o DTD (*Document Type Definition*) ou o XML *Schema*.

Para manipular documentos XML, utiliza-se o DOM (*Document Object Model*) ou o SAX (*Simple API for XML*). Através do DOM, é possível criar, excluir, pesquisar e escrever dados em um documento XML. Uma desvantagem do DOM é que ele pode utilizar muito recurso de memória, pois através dele é criada uma árvore para representar o documento XML. Já o SAX é uma interface que permite somente a leitura dos dados contidos em um documento XML. Apesar disso, o SAX possui algumas vantagens: simples de usar; pode ler documentos XML de qualquer tamanho; e em relação ao acesso de informações de um documento XML, o SAX é mais rápido em relação ao DOM (W3SCHOOLS, 2008).

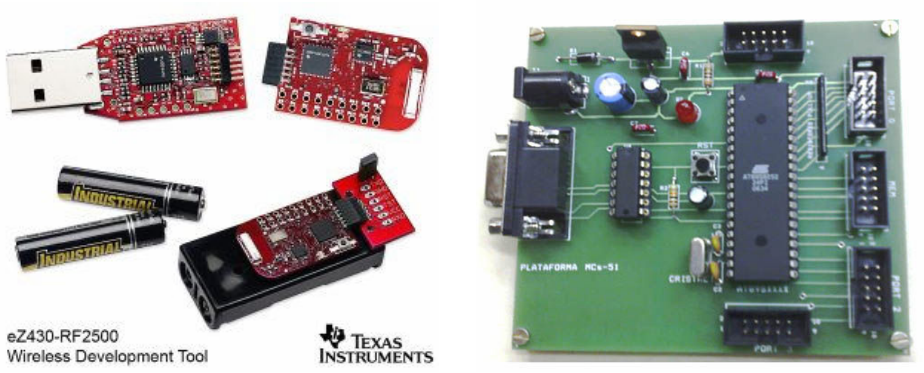
Para efeito de implementação, utilizam-se *parsers*, estruturas que oferecem funções para manipulação de documentos XML em alguma linguagem de programação. Como exemplo dessas tecnologias, pode ser citado nanoXML (NANOXML, 2008), utilizado neste trabalho, por oferecer um conjunto reduzido de componentes para formatação de dados em XML, além de ser definido pelo MHP como padrão para tratamento de conteúdo XML em TV Digital.

2.5. Plataformas MSP-430 e MCs-51

Um microcontrolador é um computador que possui componentes similares a um desktop, como processador, memória RAM, temporizadores, contadores e assim por diante. Uma característica importante de um microcontrolador é a quantidade de recurso disponível, ou seja, eles não possuem muita memória RAM, pouca memória para armazenamento de informações etc. Existem diversos microcontroladores no mercado, como MSP-430, Intel 8051, ARM 7, PIC etc. Dentre esses microcontroladores, foram utilizados no projeto o MSP-430 e o Intel 8051 para representar dispositivos de um Sistema de Automação Residencial.

O MSP-430 é um microcontrolador RISC de 16 *bits* que possui baixo consumo de energia (TEXAS, 2008). A sua CPU possui um conjunto de 51 instruções e um total de 16 registradores de 16 *bits*. Um dos *kits* de desenvolvimento disponível é o eZ430-RF2500 (ver Figura 5-A). Esse *kit* inclui um emulador USB da interface RS-232 serial, para testar e carregar as aplicações e duas placas que Wireless de 2.4-GHz.

A plataforma MCs-51 (ver Figura 5-B) utiliza o microcontrolador AT89S8252 CMOS de 8 *bits* que possui um baixo consumo de energia e alto desempenho. O microcontrolador possui 8Kbytes de memória Flash (programável e apagável), 2Kbytes de EEPROM (memória só de leitura), 256 bytes de RAM, 32 I/O *lines*, *watchdog timer* programável, três contadores/temporizadores de 16-*bits*, uma porta serial *full duplex*. O dispositivo é fabricado utilizando a tecnologia de memória não volátil de alta densidade da Atmel e é compatível com o padrão de instrução do núcleo 80C51. A plataforma foi desenvolvida na Universidade Federal do Amazonas (UFAM) com a concepção de modularidade, ou seja, para cada aplicação específica foi desenvolvido um módulo para conexão na plataforma MCs-51.



(A) **(B)**

Figura 5 – Kit de desenvolvimento: (A) eZ430-RF2500 (TEXAS, 2008) e (B) MCs-51

3. ARQUITETURA DO CENÁRIO

As tecnologias apresentadas na seção 2 são usadas nesta seção através da demonstração de um cenário integrado em um sistema de monitoramento de temperatura residencial, descrito na seção 3.1, onde se

visualiza a arquitetura, o processo e a integração das tecnologias envolvidas (Sistemas Embarcados, OSGI, MHP, XML , PHP e Java). A arquitetura é apresentada e seus componentes são explicados separadamente.

3.1. Descrição do Cenário

O cenário consiste em uma aplicação de monitoramento de temperatura, integrando as tecnologias de TVD, Sistemas Embarcados e Redes de Sensores, em que dado um conjunto de sensores ligados nas plataformas MCs-51 e MSP-430, valores de temperaturas são coletados de um ambiente residencial e enviados a um servidor de coleta e tratamento. O servidor de coleta armazena os dados em formato XML que é disponibilizado a seguir em um servidor Web implementado em PHP como se vê na Figura 6-A, 6-B, 6-C e Figura 6-D.

Para um melhor entendimento do problema, apresenta-se na Figura 6 o processo completo em forma de diagrama de estados, no qual são representadas as atividades de coleta, processamento e saída, de forma a modularizar a arquitetura para melhor integração com componentes externos. Nesse cenário, adota-se a TVD e a Web, como saídas padrão, no entanto nada impede que um dispositivo móvel seja integrado, por exemplo, um PDA ou um celular.

No módulo de sistemas embarcados usaram-se dois microcontroladores com sensores de temperatura, (Figura 6-A), um servidor com a aplicação de coleta e transformação dos dados implementado em Java com OSGI (Figura 6-B), um servidor Web Apache e PHP (Figura 6-C) e um STB com MHP configurado em uma rede IP para acessar os dados via XML (Figura 6-D). O fluxograma do processo de coleta e apresentação é mostrado no diagrama da Figura 6 na qual podem ser vistas partes isoladas do processo e a integração entre as diversas tecnologias utilizadas.

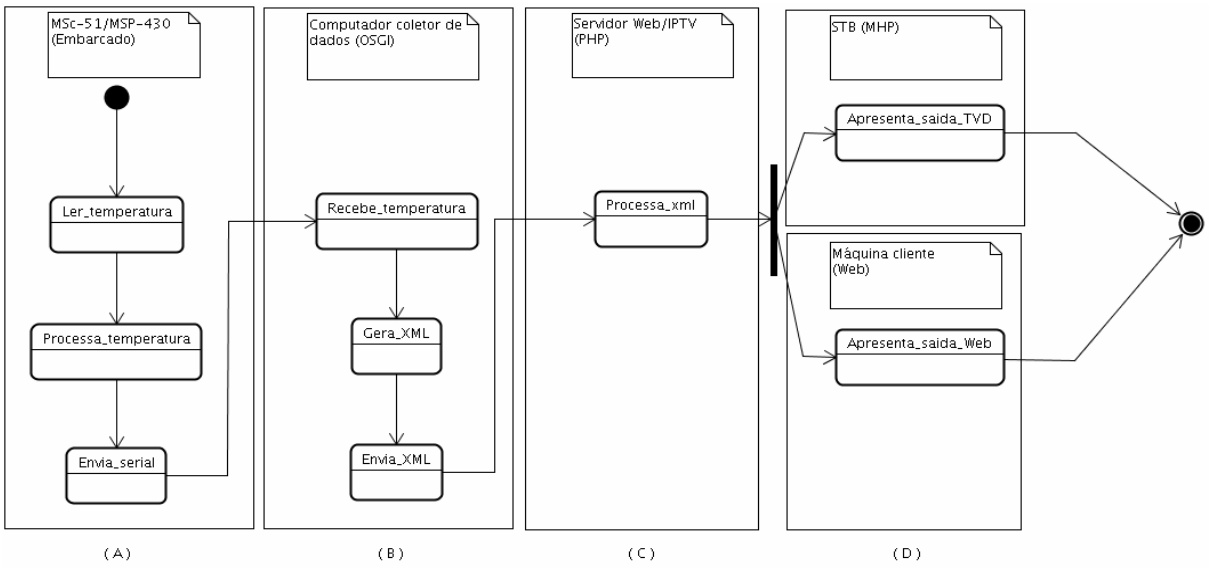


Figura 6 – Fluxograma do Processo de Coleta e Apresentação dos Dados:(A) Embarcado (B) OSGI (C) PHP/IPTV (D) STB/Cliente Web

Como visto na Figura 6, o fluxo do processo inicia com a coleta de dados através do processo *Ler_temperatura,* que coleta os dados dos sensores de temperatura acoplados ao MCs-51 e MSP-430, processando os dados em *Processa_temperatura* e os envia pela porta serial para o computador responsável pela coleta (ver Figura 6-A). O coletor de dados lê os dados em *Recebe_temperatura* usando uma aplicação Java SE, gera o arquivo XML em *Gera_XML* e os envia através de *Envia_XML* para o servidor Web (ver Figura 6-B). Por outro lado, o servidor Web processa o XML através de *Processa_xml*, uma aplicação PHP que executa um *parser* XML no servidor Web, para formatar os dados de saída, e disponibilizar para a aplicação cliente Web, usando *Apresenta_saida_Web*. Ainda nessa etapa, um aplicativo OSGI (*bundle*) executando no servidor atua sobre o arquivo XML e o disponibiliza para as APIs MHP da TV, através de *Apresenta_saida_TVD* (ver Figura 6-C e Figura 6-D), finalizando o fluxo do processo.

3.2. Arquitetura

Baseado no cenário descrito na seção 3.1, a arquitetura foi desenvolvida baseada em clientes e servidores, que representam respectivamente os sistemas embarcados, TVD, e servidores subdivididos em Web e coletor de dados. Na Figura 7 é apresentada em mais detalhes a arquitetura.

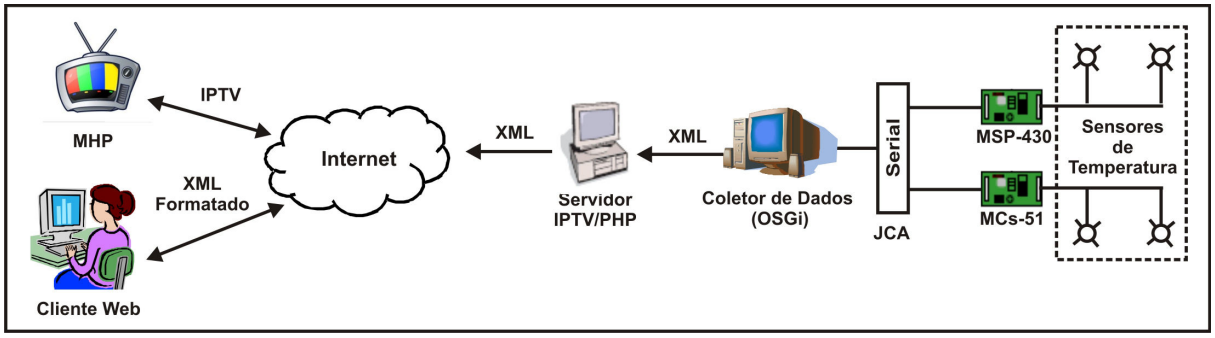


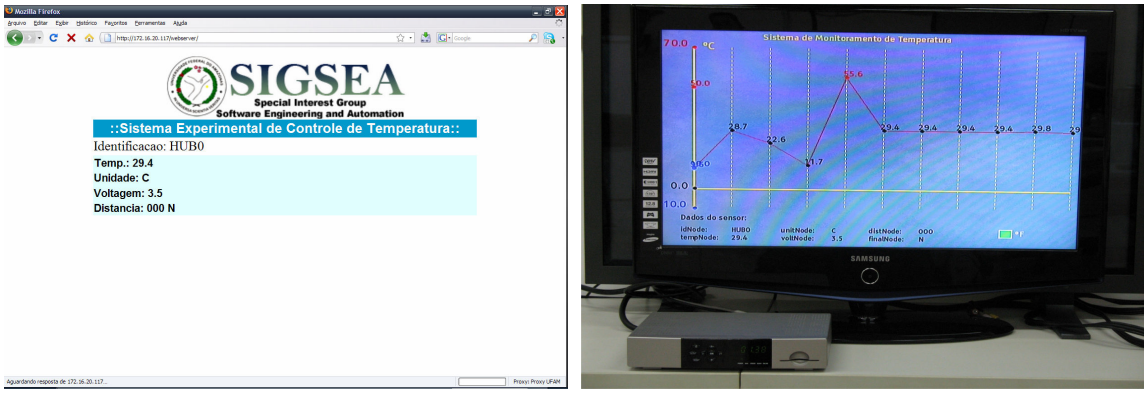
Figura 7 – Esquema geral de integração entre as tecnologias utilizadas no cenário.

Os sistemas embarcados nas plataformas MSP-430 e MCs-51 fazem a leitura dos sensores de temperatura, convertem o sinal analógico para digital através de conversores analógico digital (AD), tratam o sinal e o enviam para a porta serial do computador coletor. Esse processo é feito por uma aplicação embarcada no dispositivo, desenvolvida em linguagem de programação C.

O *software* coletor foi desenvolvido em Java usando a JCA, que permite tratar a comunicação via porta serial, utilizada para coletar as informações transmitidas pelos microcontroladores. Além da leitura dos dados, a aplicação (executando sobre o ambiente OSGI emulado no computador coletor) também constrói o arquivo XML, através do uso da API nanoXML, disponibilizando-o em seguida no servidor Web, através do uso das classes de IO existentes no Java. A leitura é feita com a taxa de transmissão de 9600 *baud*, para evitar perda do sinal e assim garantir a integridade dos dados.

O servidor Web foi desenvolvido em PHP e usa uma função de *parser*, para processar o XML e gerar a saída no navegador. A principal função do servidor é fazer a formatação do XML e atualizar a página Web. Essa página é construída dinamicamente a partir do arquivo gerado no coletor pelo *parser* da aplicação PHP, que por sua vez gera o HTML, como se vê na Figura 8-A.

A TVD solicita os serviços armazenados em um sistema de IPTV, através do *Set-Top Box*, e apresenta a informação na tela da TV, através de um gráfico de temperatura, que se comporta de acordo com as variações ocorridas no sensor (ver Figura 8-B). Isso foi possível porque o servidor cria um serviço que pode ser acessado via rede, usando um *Xlet* integrado aos dados do XML gerado pelo coletor.



(A) **(B)**

Figura 8 – Telas de Apresentação: Navegador Web do Cliente (A) e Interface da TVDI (B)

3.3. Integração dos componentes

O sistema foi desenvolvido separadamente pela equipe, baseado fortemente na arquitetura proposta, tendo como base comum o mesmo arquivo XML. Os trabalhos foram iniciados em paralelo, com equipes divididas nas diversas camadas da arquitetura, começando pela implementação da coleta e processamento dos dados dos sensores, a fim de decidir quais as informações seriam disponibilizadas. Em um segundo trabalho, mapeou-se os elementos de dados para o arquivo XML, padronizando o mesmo arquivo para toda a equipe.

Após o trabalho de mapeamento, cada membro da equipe desenvolveu parte da arquitetura, abstraindo as outras camadas, contudo preocupando-se em manter a *interface* para conexão com os outros componentes. Desta forma, o aplicativo Web/PHP, pôde ser testado separadamente, usando o XML construído seguindo o padrão pré-estabelecido. O mesmo ocorreu com os componentes de TVD e coletor de dados que foram desenvolvidos usando o mesmo padrão XML.

O projeto foi concebido, centrado na arquitetura pré-definida, o que possibilitou o desenvolvimento de forma ágil e independente de seus componentes, culminando com a integração de todo o código fonte no sistema de controle de versão.

4. IMPLEMENTAÇÃO

Nessa etapa são descritas as estratégias e modelos adotados para a criação dos componentes de *software* responsáveis pela arquitetura mostrada nas seções anteriores. Conforme descrito na seção 3.3, o sistema foi dividido em módulos integráveis. Em cada módulo, um conjunto de tecnologias foi utilizado. No centro do sistema está o arquivo XML, um formato conveniente para promover interoperabilidade de informações entre ambientes heterogêneos. O acesso à informação formatada em XML não é padronizado: cada plataforma seu conjunto de API's próprias que dão suporte ao tratamento dos dados XML. Por exemplo, utilizaram-se mecanismos diferentes para a interpretação do XML pelo *browser* do cliente *web* e pelo STB.

A implementação baseou-se na divisão do sistema em 4 módulos/subsistemas: (1) *Subsistema Sensor de Temperatura e Módulo Embarcado MCs-51/MSP-430* (Figura 9-A); (2) *Subsistema de Software APIs do OSGI, JCA e NanoXML* (Figura 9-B)*;* (3) *Subsistema Comunicação TVDi/Servidor módulo aplicativo STB* (Figura 9-C); e (4) *Subsistema cliente web* PHP (Figura 11-C); No primeiro é construída uma aplicação nas plataformas MCs-51 e MSP-430 para permitir o envio dos dados captados do sensor de temperatura para o servidor de aplicações; no segundo, um aplicativo trata os dados e os persiste em XML no servidor IPTV; nos dois últimos módulos, a aplicação de TVDI (MHP *Xlet*) e o cliente *web* PHP acessam em tempo real as informações do servidor e as apresentam na tela da TV e no *browser*, respectivamente.

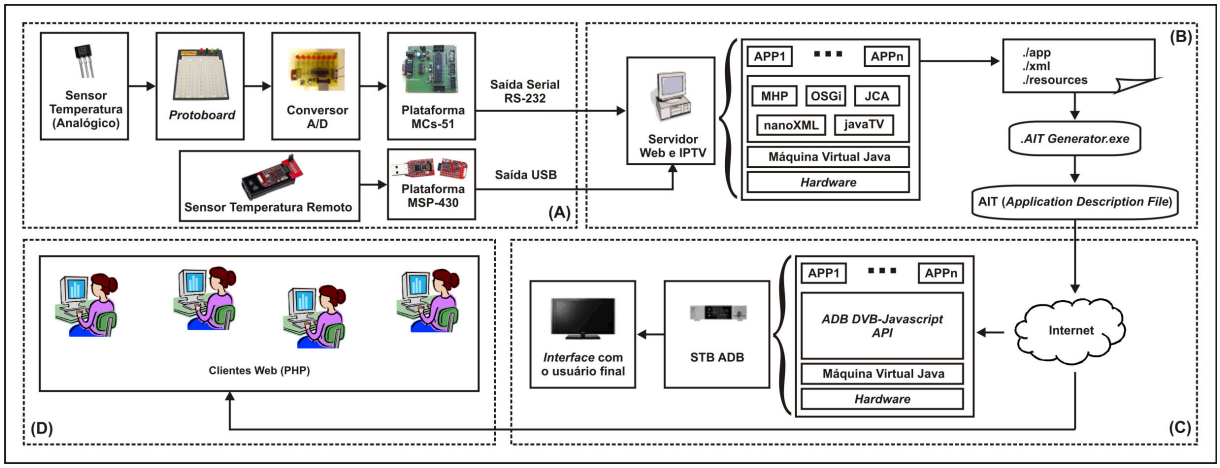


Figura 9 – Módulos da Implementação do projeto.

O módulo (1) define os procedimentos para configuração dos módulos de sistemas embarcados para formatação das informações coletadas no sensor de temperatura. Nessa etapa a programação é feita em *hardware* com a utilização de IDEs próprias para a tecnologia das plataformas MSP-430 e MCs-51.

O ambiente utilizado constitui-se de programas para a edição dos códigos em C a serem carregados nas placas. Para a plataforma MCS-51 utilizaram-se as IDEs *Keil* µVision, MIDE-51 e Eclipse SDK (com extensões de plugins para geração de código hexadecimal), a Máquina Virtual Java e o compilador SDCC (*Small Device C Compiler*), além do *software* para a gravação do programa no microcontrolador, o *ISP Flash Programmer*. Para a plataforma MSP-430 utilizaram-se ferramentas fornecidas pelo kit de programação *Texas Instruments MSP-430*: as IDEs *IAR Embedded Workbench* e *Code Composer Essentials* **MSP-CCE430** (*software* para geração de código compatível com MSP-430, baseado no Eclipse SDK).

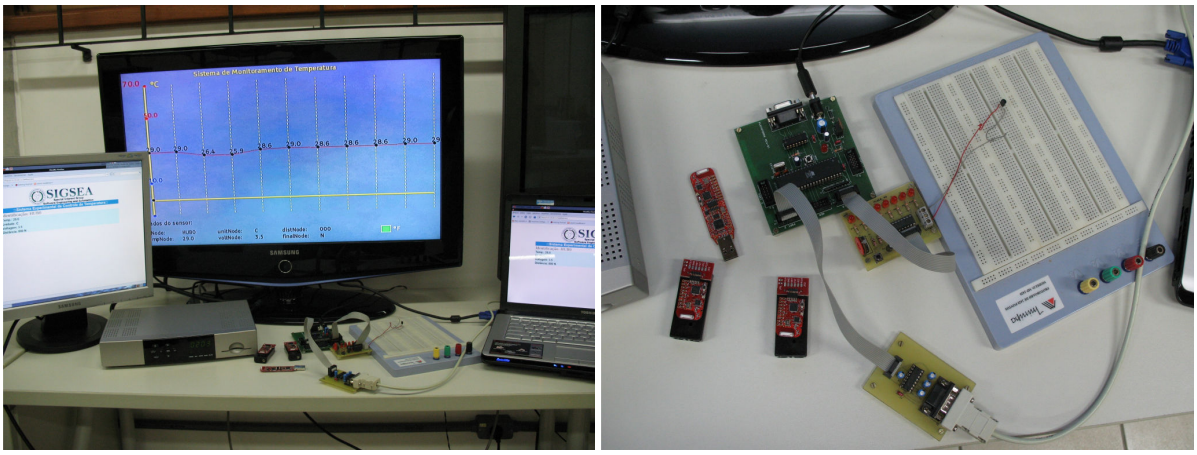
O módulo (2) define procedimentos para o tratamento do fluxo de informações provenientes do canal serial (*usando funções da API JCA*), tradução para o formato XML convencional (parser nanoXML) e comunicação com componentes de *software* da especificação de TVD (no caso, APIs do *middleware* MHP) e OSGI. Nesse módulo são criados componentes JCA que formatam os dados provenientes das plataformas, disponibilizando-os para os aplicativos OSGI (*bundles*). Nesse nível, essas informações são abstraídas no formato de *serviços OSGI*: um *bundle*, executando num ambiente emulado (nesse caso, o meio de execução é proporcionado pelo uso do emulador *Knopflerfish*, baseado em Java) provê para outros *bundles* do meio a informação do sensor de temperatura. Outro *bundle* que importa o serviço oferecido pelo primeiro *bundle* organiza essas informações e, utilizando recursos da API nanoXML, as grava no servidor IPTV/PHP (para acesso por meio do STB e por um cliente *web* comum, respectivamente) em formato XML.

No módulo (3) as aplicações MHP (*xlets*) são construídas e embarcadas no STB, acessando o XML e atualizando-o na tela do monitor de TV. Aqui foram usados componentes da API javaTV, além de componentes gráficos HAVI (bibliotecas adicionais para a criação de componentes gráficos em TVDi) para a interface do sistema. Como plataforma de STB, utilizou-se o ADB3000W-IPTV, da *Advanced Digital Broadcast* (ADB, 2008), sob o *middleware DVB-JavaScriptAPI* –IPTV.

No módulo (4) foi construída a aplicação do servidor PHP: uma interface que permite o acesso ao arquivo XML atualizado constantemente pelos componentes do módulo (3). Assim, é possível também o monitoramento da rede de sensores via Web, conforme visto a Figura 8-A.

5. RESULTADOS

A Figura 10 ilustra a interface final da aplicação no STB e no *browser* Web, bem como os dispositivos eletrônicos utilizados. Com 4 sensores do MSP-430 e 2 sensores do MCs-51, os resultados foram satisfatórios, pois foi possível verificar a atualização da temperatura dos sensores na tela da TV e no cliente Web em tempo real. O desempenho das plataformas foi semelhante, com uma leve vantagem observada para os sensores do MSP-430, no que diz respeito ao tempo de atualização das informações dos sensores. Não se utilizou algum mecanismo para a medição de outros parâmetros do sistema, como tamanho de código-fonte, consumo de memória, de energia, etc., pois uma análise desses parâmetros não fazia parte do escopo do trabalho.



(A) **(B)**

Figura 10 – Interfaces finais da aplicação (A) e módulos de dispositivos eletrônicos do projeto (B)

6. CONCLUSÕES

A TVD, como uma plataforma computacional, oferece recursos que possibilitam a construção de uma variedade de aplicações. Entre os usos mais recentes desta nova tecnologia destacam-se os que a integram com outros dispositivos eletrônicos como computadores, celulares, PDAs, etc. Dentro dessa linha de pesquisa, o trabalho apresentou uma estratégia para a construção de um cenário de integração entre a TVD e dispositivos de microcontroladores, sensores de temperatura, baseados nas plataformas MSP-430 e MCs-51. A estratégia adotada no trabalho centraliza o acesso de todos os dispositivos numa estrutura em formato XML. Assim, é possível verificar um cenário de monitoramento de temperatura numa rede de sensores de um ambiente residencial, utilizando-se como interface um STB e um cliente Web. Com este trabalho pretende-se contribuir ao relatar experiências de uso e integração entre tecnologias como microcontroladores a TV Digital. É intenção do trabalho contribuir também com a literatura relacionada às aplicações da TVD brasileira, ainda em fase de desenvolvimento.

REFERÊNCIAS

ADB. ***Advanced Digital Broadcast ltd***. Disponível em: <http://www.adbglobal.com> Acesso em: 04 ago 2008.

CALDER, B.; *et al*. **Java TV API Technical Overview: The Java TV API Whitepaper, version 1.0**. Disponível em: <http://java.sun.com/products/javatv/jtv-1_0-spec_overview.pdf> Acesso em: 04 ago 2008.

KANMA, H.; WAKABAYASHI, N.; KANAZAWA, R.; *et al*. ***Home appliance control system over Bluetooth with a cellular phone***. *In*: IEEE Transactions on Consumer Electronics, v. 49, n. 4, 2003.

KIM, K.-S.; PARK, C.; LEE, J. ***Internet Home Network Electrical Appliance Control on the Internet with the UPnP Expansion***. *In*: International Conference on Hybrid Information Technology, v. 2, 2006.

KIM, K.-S.; PARK, C.; SEO, K.-S.; *et al*. ***ZigBee and The UPnP Expansion for Home Network Electrical Appliance Control on the Internet***. *In*: The 9th International Conference on Advanced Communication Technology, v. 3, 2007.

NANOXML. ***A small XML parser for Java***. Disponível em: < http://nanoxml.sourceforge.net> Acesso em: 04 ago 2008.

MARPLES, D.; KRIENS, P. ***The Open Services Gateway Initiative: an introductory Overview***. *In*: Communications Magazine, v. 39, n. 12, 2001.

MATSUBARA, F., MIURA, S., IMAI, S., et al., "**DTV architecture design for multimedia network environments**", Consumer Electronics, 2005. ICCE. 2005 Digest of Technical Papers. International Conference on, pp. 147–148, 8-12 Jan. 2005.

OSGI. ***OSGi Service Platform, Release 3***.Disponível em: <http://www.osgi.org> Acesso em: 04 ago 2008.

SUN. ***Java Communications API Users Guide***. Disponível em: <http://java.sun.com/products/javacomm> Acesso em: 04 ago 2008.

TEXAS.***MSP430 Ultra-Low-Power Microcontroller Platform***. Disponível em: <http://www.ti.com> Acesso em: 04 ago 2008.

W3SCHOOLS. ***XML Tutorial***. Disponível em: <http://www.w3schools.com/xml> Acesso em: 07 ago 2008.