COBENGE 2005
XXXIII - Congresso Brasileiro de Ensino de Engenharia
"Promovendo e valorizando a engenharia em um cenário de constantes mudanças"
12 a 15 de setembro - Campina Grande Pb
Promoção/Organização: ABENGE/UFCG-UFPE

TUTORIAL: ETAPAS DE DESENVOLVIMENTO DE TRANSDUTORES WEB UTILIZANDO DISPOSITIVOS EMBARCADOS

Alfranque Amaral da Silva – alfranque@dee.ufcg.edu.br
Antonio Marcus Nogueira Lima – amnlima@dee.ufcg.edu.br
Péricles Rezende Barros – prbarros@dee.ufcg.edu.br
Universidade Federal de Campina Grande, Departamento de Engenharia Elétrica. Campus de Campina Grande 58000-100 – Campina Grande – Pb

Resumo: *Este artigo é um tutorial que descreve as etapas de desenvolvimento de transdutores web utilizando dispositivos embarcados. São apresentadas as etapas de desenvolvimento, compreendendo desde a medição e/ou atuação, até a construção da interface web, onde serão exibidos os dados medidos pelos sensores. Como exemplo da utilização de transdutores web foi desenvolvido um sistema de controle de temperatura que consiste em um sensor de temperatura LM35 e um ventilador. A leitura do sensor de temperatura e o controle para acionamento do ventilador foram implementados com o microcontrolador ADuC832. A interface web foi codificada em Java e executada na plataforma de desenvolvimento TBM390 (TINI - Tiny InterNet Interface - Board Model 390).*

Palavras-chaves: Transdutores web, Rede de transdutores, Sistemas embarcados, Microcontrolador.

1. INTRODUÇÃO

Um transdutor *web* pode ser definido como um sistema que é constituído de um transdutor analógico ou digital, um microprocessador e uma interface de comunicação. A evolução da microeletrônica e das redes de comunicação de dados contribuíram decisivamente para o desenvolvimento de aplicações baseadas em transdutores *web* conectados em rede. O uso de redes é eminentemente motivado pela redução do custo de cabeamento, modularização e flexibilidade na configuração do sistema (JOHNSON, 1997). Além disso, a utilização de transdutores em redes, com inteligência embarcada nos processadores dos nós da rede, constitui uma alternativa promissora para tratar adequadamente a complexidade inerente de sistemas distribuídos.

Um digrama ilustrativo do tipo de ambiente onde serão utilizados os transdutores *web* é apresentado na Figura 1.1. Nesse contexto, os transdutores *web* são devidamente inseridos em processos de medição e/ou controle industrial, residencial, predial, como ilustrado pela região delimitada pela linha tracejada da Figura 1.1. Nesta Figura os dados oriundos ou destinados ao transdutor podem ser visualizados ou definidos por meio de uma interface *web* escrita em Java e executa de dentro de uma página *web* escrita na linguagem html, através de um *applet*. A interface de comunicação com o transdutor é feita através da interface SPI (*Serial Peripheral Interface*), implementada para o TINI em uma biblioteca escrita em assembler e executada através de um método nativo (MN) (SUN, 2004) escrito em Java.

Este artigo é um tutorial detalhado das etapas de construção de um transdutor *web*. Nele é abordada a seqüência de passos que devem ser seguidos, desde a medição e/ou atuação, até a construção da interface *web*, onde serão apresentados os dados medidos, estados e limites de referências dos atuadores. Os dados resultantes da medição de grandezas físicas por sensores são mostrados em uma página *web* disponibilizada na *intranet/internet*. Já os comandos para acionamento e/ou atuação de atuadores devem ser ou enviados a partir de um *click* em botões presentes na interface da página *web* ou ativado por um limite de referência previamente definido.

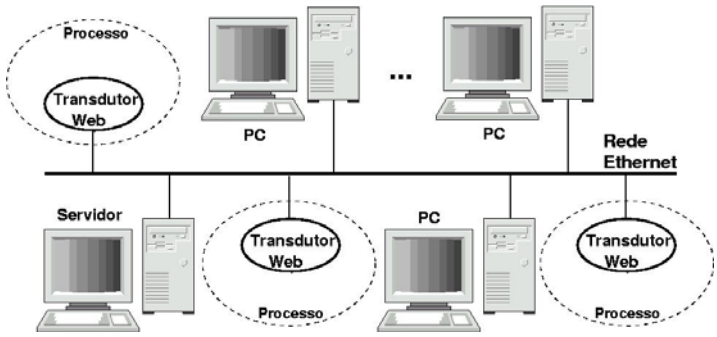

Figura 1.1: Rede Ethernet, com transdutores *web*, inseridos em processos de medição e/ou controle.

A descrição dos dispositivos utilizados para implementação de transdutores *web* e os passos para construção dos mesmos serão descritos nas seções seguintes. Na seção 2 é apresentado o módulo transdutor *web*. Na seção 3 é apresentada a interface SPI. Na seção 4 são introduzidos os conceitos de JNI (Java Native Interface) e definidos os MNs. Na seção 5 é apresentado o *hardware* usado na implementação. Na seção 6 são apresentados os passos para desenvolvimento de aplicativos para o ADuC832 e TINI. Na seção 7 é ilustrado o ambiente de desenvolvimento. Na seção 8 é apresentada a interface gráfica. Na seção 9 é apresentado um aplicativo *web* básico para a monitoração de temperatura. Finalmente, na seção 10, é apresentada as conclusões e as perspectivas de trabalhos futuros.

2. MÓDULO TRANSDUTOR WEB

O módulo transdutor *web* ilustrado na Figura 1.2 é constituído de transdutores; portas de I/O (presentes no microcontrolador ADuC832) e por uma placa do TINI (TBM390), que disponibiliza em uma página *web* os dados resultantes de medições e/ou atuações realizadas pelos transdutores conectados nas portas de I/O do ADuC832. Dados e comandos são transferidos do TINI para o ADuC832 e vice-versa, via interface SPI, onde o TINI é o mestre, e o ADuC832 é o escravo.

Transdutores *web* podem ser desenvolvidos utilizando-se o TINI ou outro *hardware* que disponibilize interface de rede. Dois exemplos de aplicações de transdutores *web*, desse tipo são mostrados em (SCHNEEMAN, 1999) e (MAXIM/DALLAS SEMICONDUCTOR, 2002). Em ambos os casos, os aplicativos são *applet*, isto é, aplicativos escritos em Java e executados de dentro de uma página *web* escrita em html. No primeiro exemplo o aplicativo é executado em um dispositivo da HP (*Hewlett-Packard*) e constitui a implementação do padrão IEEE 1451.1 (IEEE, 1999). O segundo é executado no TINI e é uma implementação para supervisão de temperatura via protocolo 1-wire, usando um sensor de temperatura DS1920 ou DS1820. A escolha do TINI juntamente com o ADuC832 e a interface SPI é

justificada pelo fato desse trabalho ser parte da implementação dos padrões para transdutores inteligentes IEEE 1451.1 (IEEE, 1999) e IEEE 1451.2 (IEEE, 1997). Estes padrões foram implementados respectivamente com TINI e o ADuC832.

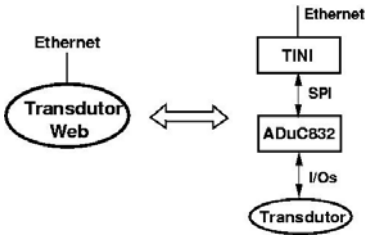

Figura 1.2: Módulo transdutor *web*.

O detalhamento dos procedimentos usados para construir transdutores *web* inicia com o processo de aquisição de dados e/ou acionamento realizado pelo ADuC832, via comunicação entre o ADuC832 (escravo) e o TINI (mestre), através da SPI e da conexão do TINI com o *hardware* de medição e/ou atuação à rede *Ethernet*.

3. A INTERFACE SPI

A interface SPI é uma porta de entrada e saída serial síncrona com velocidade da ordem de alguns *megabits* por segundo e permite que dados sejam sincronamente transmitidos e recebidos simultaneamente, isto é, *full duplex*. Na configuração padrão para um dispositivo, a SPI tem duas linhas de controle $\overline{SS}$ (*slave select*) e SCKL (*clock*) e duas linhas de dados MOSI (*Master Out Slave In*) e MISO (*Master In Slave Out*).

A comunicação SPI é do tipo mestre escravo (UBICOM, 2000). O mestre fornece o sinal de *clock* (SCKL) e determina o estado da linha $\overline{SS}$. Isto é, o mestre ativa o escravo e a comunicação com o mesmo. $\overline{SS}$ e SCKL são saídas do mestre e entradas para o escravo. A SPI é baseada em um simples registrador de deslocamento (NöLKER;KLEMENZ, 1999) a até um subsistema independente. Normalmente o registrador de deslocamento é de oito bits ou múltiplo de oito (MOTOROLA, 2002). O princípio básico do registrador de deslocamento está sempre presente. Código de comandos e dados são serialmente transferidos. Inseridos em um registrador de deslocamento e disponibilizado internamente para processamento paralelo (ESTL, 2002). Na Figura 1.3 é ilustrado o princípio de funcionamento da comunicação SPI de um mestre e um escravo (UBICOM, 2000), idêntico ao descrito neste artigo.

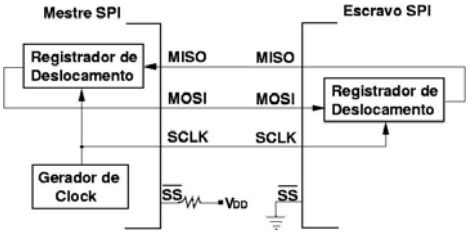

Figura 1.3: Diagrama de blocos da transferência mestre/escravo.

A interface SPI possui um par de parâmetros para configuração da polaridade e fase do sinal de *clock*. A parâmetro CPOL (*clock polarity*), determina a polaridade do sinal de *clock*. CPHA (*clock phase*) determina as bordas do sinal de *clock*, sobre a qual dados são amostrados e conduzidos (transmitidos/recebidos) (MOTOROLA, 2002). Cada um desses parâmetros

possui dois estados, possibilitando quatro combinações distintas. Estes parâmetros devem possuir a mesma configuração tanto no dispositivo mestre como no escravo. A transferência dos bits ocorre na transição central de cada pulso de *clock*. Na implementação descrita, foram testados os quatro possíveis modos de configuração de CPOL e CPHA.

4. CONCEITOS JNI

A JNI (SUN, 2004) é uma interface de programação nativa para Java que é parte do JDK (*Java Development Kit*). A JNI foi explorada durante o uso, no TINI, da biblioteca spi.tlib, a qual implementa por *software*, a interface de comunicação SPI (MAXIM/DALLAS SEMICONDUCTOR, 2001), usada para realizar a comunicação entre o TINI e o ADuC832.

Programas escritos usando a JNI têm a garantia de seu código ser completamente portátil sobre todas as plataformas. Além disso, a JNI possibilita que códigos escritos em Java, operem com aplicativos e bibliotecas escritas em outras linguagens, tais como C, C++ e assembler (SUN, 2004). Programadores usam a JNI para escrever MN para tratar situações em que um aplicativo não possa ser escrito inteiramente em Java. MN e a JNI são usados nas seguintes situações: a biblioteca de classes padrão Java não suporta as características da plataforma dependente, necessária para a aplicação; o usuário tem uma biblioteca ou aplicativo escrito em outra linguagem de programação e deseja torná-la acessível por aplicativos Java; ao desejar implementar um trecho de código com tempo crítico, em uma linguagem de programação de baixo nível, como assembler, tendo seu aplicativo escrito em Java, para chamar estas funções.

A JNI habilita o usuário a usar as vantagens da linguagem de programação Java do seu MN. Em particular, o usuário pode lançar exceções do MN e ter estas exceções tratadas no aplicativo Java. MNs também podem pegar informações sobre classes Java. Chamando funções JNI especiais, MNs podem carregar classes Java e obter informações de classes. MNs podem usar a JNI para desempenhar tipos de checagem de tempo de execução. Assim a JNI serve como uma "junção" entre aplicativos escritos em Java e aplicações nativas. Na Figura 1.4 é mostrado como a JNI interliga o lado assembler de um aplicativo ao lado Java.

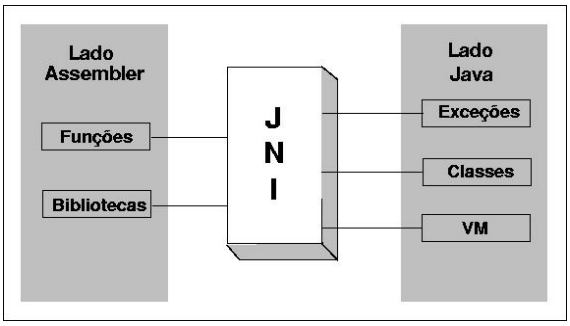

Figura 1.4: Conexão JNI do lado assembler de um aplicativo ao lado Java.

Nesta Figura é ilustrado como funções e bibliotecas escritas em assembler, são ligadas pela camada de interface nativa do *hardware* ao lado Java da aplicação. Assim parâmetros são passados do lado Java para o lado assembler através da chamada de um MN definido em uma classe Java e executado pela JVM (Java Virtual Machine). O lado assembler recebe esses parâmetros, atualiza o estado dos registradores, processa as operações implementadas nas funções da biblioteca e retorna as informações resultantes da execução para o lado Java. Seu conteúdo contém dados os quais são usados, entre outros fins, para o lançamento de exceções.

Na biblioteca SPI do TINI, implementada em assembler, a passagem de parâmetros, a chamada do MN que implementa a biblioteca (MAXIM/DALLAS SEMICONDUCTOR, 2001) e o tratamento de exceções são feitos em alto nível por um aplicativo Java de acordo com a documentação para construção de MN do TINI (SUN, 2004; DALLAS SEMICONDUCTOR, 2002b, 2002a).

5. O HARDWARE USADO

Na implementação do sistema foi utilizada uma placa de desenvolvimento SAR *Eval Board* Rev A3 (ANALOG DEVICES, 2002b) e a plataforma de desenvolvimento TINI modelo 390. Os detalhes sobre eles são mostrados nas subseções seguintes.

5.1 O ADuC832

O ADuC832 (ANALOG DEVICES, 2002a) é um microcontrolador com interface de usuário integrado em um único *chip*. Baseado no 80C52 da Intel, o ADuC832 possui os seguintes recursos: 1 MCU (*Microcontroller Unit*) de 8 *bits*; 2 conversores DA de 12 *bits*; 1 conversor AD de 12 *bits* do tipo aproximação sucessiva com 8 canais multiplexados com autocalibração; 4 portas de I/O serial (SPI, I2C e UART); 4 *kBytes* de memória *Flash/EE* não-volátil para dados; 256 *bytes* de memória RAM; 2 *kBytes* de memória RAM estendida; 62 *kBytes* de memória *Flash/EE* para memória de programa. Os aplicativos desenvolvidos para o ADuC832 são carregados e executados no *hardware* via porta serial com o WSD (*Window Serial Donwloader*), sem o auxílio de circuitos externos, caracterizando assim a programação ISP (*In-System Programming*). O ADuC832 é programado em linguagem C ou assembler, utilizando o compilador *Keil uVision* (KEIL SOFTWARE, 2001). O ADuC832 suporta os sistemas de desenvolvimento *QuickStart* e *QuickStart Plus*, os quais são ferramentas de desenvolvimento de *hardware* e *software* de baixo custo.

5.2 O TINI modelo 390

O TINI modelo 390 (DALLAS SEMICONDUCTOR CORPORATION, 2001) é usado em tarefas embarcadas tradicionais, tais como, monitoração e controle de dispositivos ou sistemas locais. É ferramenta de desenvolvimento para escrita de servidores *web* embarcados. Entretanto, a maioria de suas aplicações, inclusive a descrita neste artigo, utilizam sua capacidade de rede. O TINI tem um microcontrolador DS80C390 (MAXIM/DALLAS SEMICONDUCTOR, 2003) com a JVM embarcada. Integrado ao microcontrolador existe os seguintes protocolos de comunicação serial de baixo nível: *RS*232/*C*, UART, CAN, *1-Wire*. A interface SPI, existe implementada em *software* (MAXIM/DALLAS SEMICONDUCTOR, 2001). O TINI tem controlador *Ethernet*; 512 *kilobytes* de memória Flash ROM; 512 *kilobytes* memória RAM (expansível a até 2 *megabytes*); 2 portas serial integradas e suporte para 2 portas serial externa; 2 controladores CAN integrados; barramentos *1-wire* externo e interno; relógio em tempo real. Com API JDK 1.1 e um SO (Sistema Operacional) com multi-tarefas, multi-*threaded*, pilha TCP/IP completa, gc (*garbage collection*), porta serial e *drivers 1-Wire*, suporte PPP, um sistema *shell* Slush (DALLAS SEMICONDUCTOR, 2002d) baseado no Unix e com interface de acesso TTY, TELNET e FTP (DALLAS SEMICONDUCTOR, 2002c). Ele também possui I/O TTL de propósito geral com pinos bidirecionais. Seus aplicativos são desenvolvidos em Java, contudo, também é possível desenvolver aplicativos com MN escritos em assembler (SUN, 2004; DALLAS SEMICONDUCTOR, 2002b, 2002a).

6. DESENVOLVIMENTO DE APLICATIVOS

Nessa seção serão abordadas, passo a passo, as etapas para desenvolvimento e execução de aplicativos para o ADuC832 e para o TINI.

6.1 Escrevendo um aplicativo para o ADuC832

A construção e execução de um aplicativo para o ADuC832 é uma rotina que segue os seguintes passos (ANALOG DEVICES, 2001): (1) Cria-se um novo projeto usando o ambiente de desenvolvimento *Keil uVision* (KEIL SOFTWARE, 2001). Neste passo são gerados os arquivos: arquivo.uv2 e arquivo.plg; (2) Escreve-se o código C do aplicativo. Isso pode ser feito no próprio *Keil uVision;* (3) Utiliza-se o *Keil uVision* para compilar e depurar os arquivos.c do aplicativo escrito. Se o aplicativo for compilado sem erro, os seguintes arquivos são gerados: arquivo.hex, arquivo.m51, arquivo.lst, arquivo.obj, arquivo.Opt, arquivo.Inp; (4) Utiliza-se o WSD para carregar e executar o arquivo.hex no ADuC832. Tais passos são ilustrados no fluxograma da Figura 1.5a.

6.2 Escrevendo um aplicativo com MNs

O processo de escrita de MNs para um aplicativo Java segue os seguintes passos (SUN, 2004; DALLAS SEMICONDUCTOR, 2002b, 2002a): (1) Escreve-se o programa em Java. Cria-se a classe Java que declara o MN. Esta classe contém a declaração ou assinatura para o MN. Ela também inclui o método *main* que chama o MN; (2) Compila-se a classe Java que declara o MN e o método *main*; (3) Escreve-se a implementação do MN na linguagem de programação nativa. No caso do TINI, esta linguagem é *assembler*; (4) Compila-se os arquivos cabeçalho e a implementação em um arquivo biblioteca compartilhado. Para o TINI, corresponde a assemblar o código do MN segundo a API do TINI; (5) Carrega-se a biblioteca nativa no aplicativo desenvolvido. Para o TINI, isto é feito usando o TINIConvertor com o comando -n <NomeDaBiblioteca>" para especificar a localização da biblioteca nativa; (6) Usa-se o loadLibrary do aplicativo Java para carregar a biblioteca nativa; (7) Utiliza-se FTP para carregar e TELNET para executar no *hardware* o arquivo.TINI do aplicativo gerado nesse processo. A representação seqüencial dos passos usados para escrever aplicativos com bibliotecas nativas anexadas é mostrado na Figura 1.5b.

6. AMBIENTE DE DESENVOLVIMENTO

O ambiente de desenvolvimento é constituído por: uma placa de desenvolvimento do ADuC832 (SAR *Eval Board* Rev A3) (ANALOG DEVICES, 2002b), constituída pelo microcontrolador ADuC832 (ANALOG DEVICES, 2002a), circuitos de proteção, pinos de interface placa/usuário, que possibilitam acesso às portas de I/O do microcontrolador. Esta placa é usada para aquisição de dados e acionamento de atuadores; uma placa TINI modelo TBM390 (DALLAS SEMICONDUCTOR CORPORATION, 2001), que disponibiliza em uma página *web*, os dados adquiridos resultantes de medições realizadas por sensores conectados aos ADs do ADuC832; um *proto-board* para montagem de circuitos de acionamento e condicionamento de sinal; fonte de tensão DC de 5*V* e um PC com sistema operacional *windows*, usado no desenvolvimento dos aplicativos para o TINI e ADuC832.

Os detalhes de montagem do ambiente de desenvolvimento são mostrados na Figura 1.6. A aquisição dos dados é feita pelo conversor AD de 12 *bits* do ADuC832. O sinal é amostrado na saída do circuito de condicionamento de sinal, mostrado na parte inferior da Figura 1.6. Este circuito é constituído por um estágio de amplificação não inversor de ganho 5, realizado

pelo amplificador operacional não inversor LM324, cujo sinal de entrada é fornecido pelo sensor. Nesse experimento foi usado o sensor de temperatura LM35, entretanto nada impede o uso de qualquer sensor, desde que o condicionamento de sinal, seja feito adequadamente.

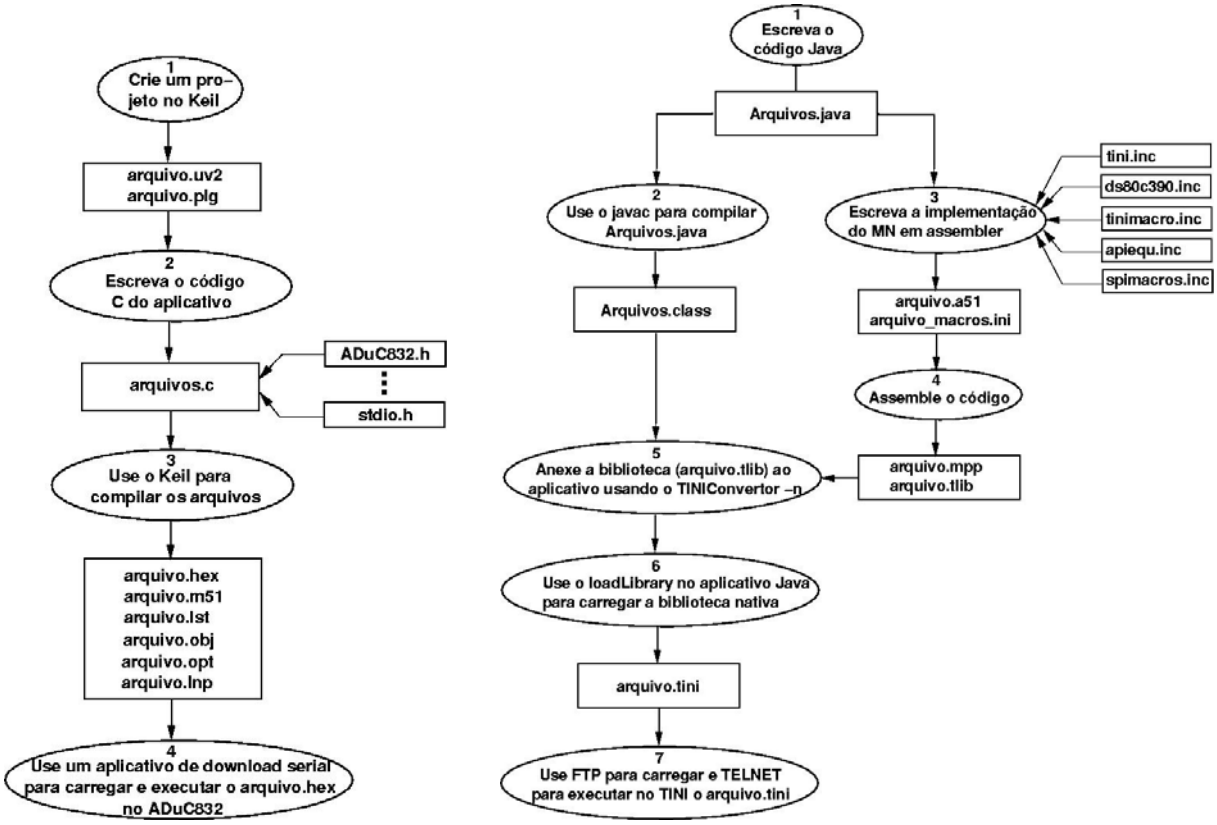

Figura 1.5: (a) Seqüência de passos para construção de um aplicativo para o ADuC832. (b) Diagrama seqüencial para construção de aplicativo com biblioteca nativa.

Na parte central da Figura 1.6 é mostrado o circuito de acionamento de um ventilador de um *cooler* de PC, acionado por uma porta de saída do ADuC832. Na parte superior da Figura 1.6 são ilustradas as conexões físicas da interface de comunicação SPI, entre o TINI (mestre) e o ADuC832 (escravo).

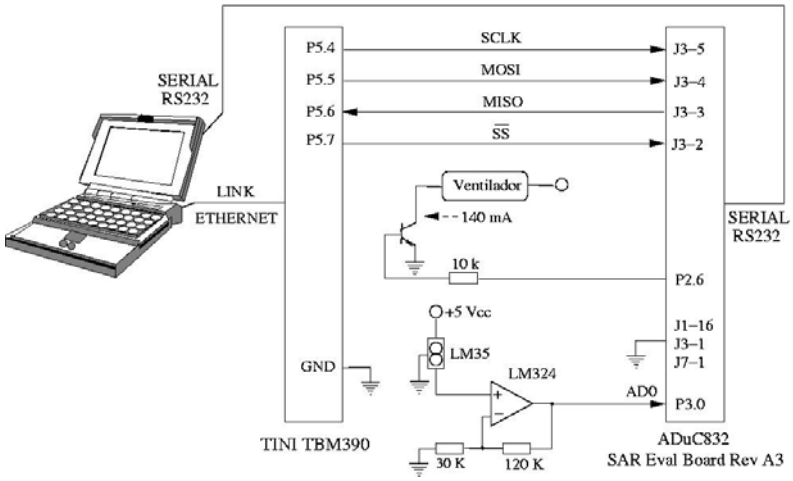

Figura 1.6: Ambiente de desenvolvimento.

Os dados são transferidos do ADuC832 para o TINI via SPI, após a cópia do conteúdo dos registradores ADCDATAH e ADCDATAL do conversor AD para o registrador de deslocamento SPIDAT da SPI, como mostrado no seguinte trecho de código.

```
//Carrega o conteú-do do registrador
//ADCDATAH para o registrador SPIDAT
SPIDAT=ADCDATAH;
while(!ISPI){} ISPI=0;
SPIDAT = ADCDATAL;
while(!ISPI){}  ISPI=0;
//O valor da temperatura em graus Celsius é:
Temperatura = (((((ADCDATAH &
0x0F)<<8) |  ADCDATAL)*0.00061)*20);
for(j=0;j<3000;j++) {}  //Imprime valor da
//temperatura na tela
TI = 0; // Assegura um tempo para rece-
//bimento dos bytes pela porta serial
SBUF = '\n';  while(TI!=1) { }
printf("O valor da temperatura em ºC:
%2.2f\n",Temperatura);
```

No TINI esses dados são novamente transformados em temperatura e exibidos em uma interface gráfica e em um *display*. Já no ADuC832 os dados são visualizados via *Hyper Terminal*, como mostrado na Figura 1.7.

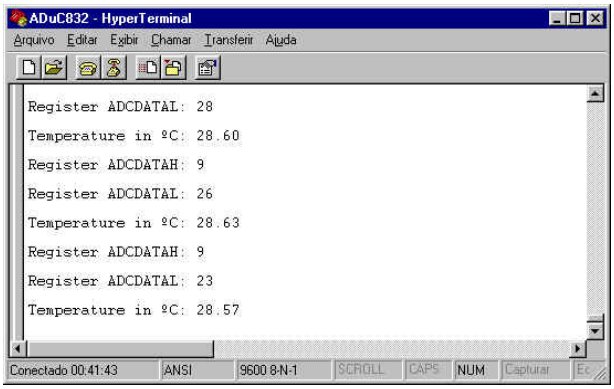

Figura 1.7: Janela do Hyper Terminal mostrando o conteúdo dos registradores ADCDATAH e ADCDATAL e o valor da temperatura em graus *Celsius*.

8. INTERFACE GRÁFICA

Nesta seção será descrito o *software* da interface gráfica do ponto de vista da comunicação entre servidores *web* e clientes *web*. Além disso, são mostradas as relações entre os blocos de *software* desenvolvidos e a interface gráfica onde os dados dos transdutores *web* são mostrados.

8.1 Relação entre *applets* e *servlets*

A estação de monitoração ligada à rede *Ethernet* com *browser* dotado de uma JVM que suporte *applets* Java é parte indispensável do sistema de monitoração dos transdutores *web*. *Applets* são aplicativos embarcados em páginas *web*, os quais podem ser usados para construir elementos GUI (*Graphical User Interfaces*) complexos sobre uma página *web* com transferência síncrona de informação de servidores *web* para clientes *web*. Tal mecanismo permite que dados de transdutores sobre a página *web* sejam atualizados automaticamente. *Applets* também podem ser usados em conjunto com aplicativos *servlets* para transmitir informações de clientes *web* para um servidor *web*. *Servlets* são executados sobre um servidor *web* e são criados para listar *strings* de comandos enviados pelos *applets*. O processo de comunicação entre *applets* e *servlets* é ilustrado na Figura 1.8 (COSTA, 2002).

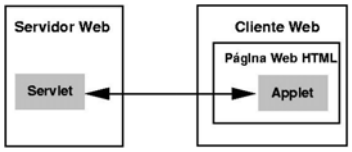

Figura 1.8: Processo de comunicação entre *applets* e *servlets*.

A relação complementar entre *applets* e *servlets* forma a base da rede de comunicação entre a estação de monitoração e o TINI. Inicialmente o TINI é ativado como um servidor *web* HTTP (*HiperText Transfer Protocol*) servindo o *applet* embarcado na página *web* da estação de monitoração. Depois um aplicativo *servlet* é executado no TINI fornecendo um canal de comunicação bi-direcional entre a estação de monitoração (cliente *web*) e o TINI (servidor *web*) (COSTA, 2002). Dois exemplos de aplicações de transdutores *web* que utilizam *applets* são mostradas em (SCHNEEMAN, 1999) e (MAXIM/DALLAS SEMICONDUCTOR, 2002).

8.2 Estrutura do Software

A estrutura dos módulos de *software* desenvolvidos para implementar a modularidade e interoperabilidade da interface gráfica pode ser vista em blocos separados pelas linhas tracejadas na Figura 1.9. De um lado está a estação de monitoração executando a GUI Java que atualiza os dados recebidos da comunicação entre o *applet* executado na estação de monitoração e o *servlet* executado no TINI. Do outro lado, entre o TINI e o ADuC832, ocorre a execução da comunicação via interface SPI, responsável pela conexão/comunicação entre o TINI e o ADuC832.

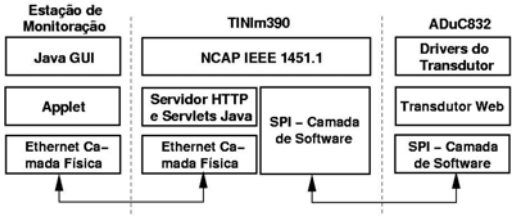

Figura 1.9: Estrutura dos módulos de *software* executados no TINI.

9. APLICATIVO BÁSICO

Neste trabalho é apresentado um aplicativo básico para a monitoração de temperatura. Inicialmente a tensão amostrada pelo conversor AD é tratada internamente no ADuC832, convertida em temperatura e disponibilizada, via porta serial, pelo Hiper Terminal do *Windows*. Após realizada a etapa de aquisição de dados o código em C deve ser modificado, acrescentando-se a parte de configuração do registrador SPICON da interface SPI e do carregamento dos dados dos registradores ADCDATAH e ADCDATAL do AD para o registrador SPIDAT da SPI do AduC832. O conteúdo dos registradores do AD do ADuC832 são transmitidos para o TINI via *interface* SPI, implementada através de um MN escrito em assembler. A classe responsável pela aquisição dos dados é mostrada a seguir.

```
public class SPIaquis{
   private SPI spi; //Cria um objeto spi da
   //classe SPI
   public SPIaquis(){
      spi = new SPI(0, true, true, true, true,
         false);
   } //Fim do método construtor
   public double convertAtoD(){
      int samp;
      double sample;
```

```
byte[] spidata = new byte[1];
int[] dado = new int[2];
for(int count=0; count<2; count++) {
   spi.xmit(spidata, 0, spidata.length);
   dado[count] = (int)spidata[0];
    // Imprime dado no terminal do TINI
   System.out.println("dado["+count+"]:
   "+ dado[1]);
}
 spidata[0]=(byte)0x00;
   if(dado[1] < 0x00){
      dado[1] = 256 + dado[1];
   }
   dado[0] = dado[0] & 0x0F;
   samp = dado[0]*256 + dado[1];
   System.out.println("samp: " + samp);
   sample = samp*0.00061*200.0;
   return sample;
  }
}
```

No TINI o conteúdo dos registradores do AD são novamente transformados em temperatura e exibidos em uma interface gráfica e em um *display*. Como atuador foi utilizado um ventilador de um *cooler* de PC o qual será ligado/desligado todas as vezes que a temperatura ultrapassar os limites de referência pré-estabelecidos. Na Figura 1.10, é mostrada uma janela com o navegador executando o *applet* o qual exibe, na forma gráfica e em um *display*, o valor da temperatura em graus *Celsius* ou *Fahrenheit*.

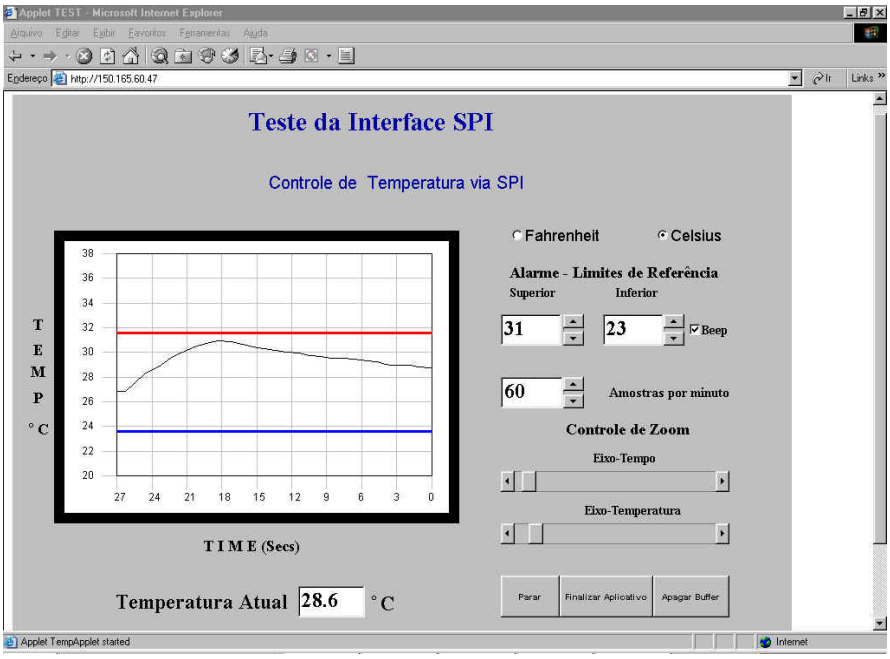

Figura 1.10: Página *web* mostrando temperatura.

Esta interface piloto é uma adaptação de uma *interface* desenvolvida pelo fabricante do TINI e construída para uma aplicação de monitoração de temperatura via *interface 1-wire* (MAXIM/DALLAS SEMICONDUCTOR, 2002) e está sendo reutilizada com as devidas adaptações para a aplicação aqui descrita. O gráfico da temperatura entre as linhas azul e vermelha é praticamente uma reta. Entretanto na Figura 1.10 este gráfico não é uma reta, devido ao fato de se ter provocado uma variação de temperatura para demonstrar a eficácia do sistema.

9. CONCLUSÕES

Neste trabalho foram apresentadas as etapas de construção de um ambiente de desenvolvimento de transdutores *web*. Foram utilizadas duas placas: a SAR *Eval Board* Rev

A3 para aquisição de dados e o acionamento de atuador e a do TINI para proporcionar interface com a rede *Ethernet*. A principal vantagem desse tipo de sistema está na possibilidade de construção de sistemas distribuídos para supervisão e controle (SCHNEEMAN, 1999) a um baixo custo.

Dentro do escopo proposto nesse trabalho, os resultados obtidos foram satisfatórios. Não apenas pelo fato da implementação do ambiente de desenvolvimento de transdutores *web*, mas por possibilitar avanços significativos no trabalho com sistemas embarcados voltados para instrumentação eletrônica e controle. Bem como, possibilitar a implementação da SPI que é o núcleo da comunicação entre o STIM (*Smart Transducer Interface Module*) padrão IEEE1451.2 (IEEE, 1997) e do NCAP (*Network Capable Application Processor*) padrão IEEE1451.1 (IEEE, 1999). O NCAP foi implementado para o TINI e o STIM foi implementado para o microcontrolador ADuC832.

9. REFERÊNCIAS BIBLIOGRÁFICAS

ANALOG DEVICES. MicroConverter Technical Note - uC002 Developing in C with the Keil uVision2 IDE. Ver 2.0. [S.l.], 2001.

ANALOG DEVICES. ADuC832, MicroConverter, 12-Bit ADCs and DACs with Embedded 62 kBytes Flash MCU. Inc., 2002. [S.l.], 2002.

ANALOG DEVICES. ADuC8xx SAR Evaluation Board Reference Guide, MicroConverter QuickStartTM Development System. Version a.3. [S.l.], 2002.

COSTA, B. Wireless Distributed Sensing and Actuation. Dissertação — University of Wollongong, October 2002.

DALLAS SEMICONDUCTOR. Documentation TINI Firmware 1.02f, Native_API.txt (DC, 22.03.02). [S.l.], 2002.

DALLAS SEMICONDUCTOR. Documentation TINI Firmware 1.02f, Native_Methods.txt (DC, 22.03.02). [S.l.], 2002.

DALLAS SEMICONDUCTOR. Documentation TINI Firmware 1.02f, README.txt (DC, 22.03.02). [S.l.], 2002.

DALLAS SEMICONDUCTOR. Documentation TINI Firmware 1.02f, slush.txt (DC, 22.03.02). [S.l.], 2002.

DALLAS SEMICONDUCTOR CORPORATION. O TINI Specification and Developer's Guide. First. [S.l.], 2001.

ESTL, H. Application Note, SPI interface and use in a daisy-chain bus configuration. V 1.2. [S.l.], 2002.

IEEE. Standard for a smart transducer interface for sensors and actuators – transducer to micro-processor communication protocols and transducer eletronic data sheet (teds) formats. IEEE, IEEE Std 1451.2-1997, p. 1–120, 1997.

IEEE. Standard for a smart transducer interface for sensors and actuators – network capable application processor (ncap) information model. *IEEE*, IEEE Std 1451.1-1999, 1999.

JOHNSON, R. N. Building plug-and-play networked smart transducers. Eletronics Development Corporation, October, p. 1–18, 1997.

KEIL SOFTWARE. Cx51 Compiler - Optimizing C Compiler and Library Reference for Classic and Extended 8051 Microcontrollers. User's guide 09.2001. [S.l.], 2001.

MAXIM/DALLAS SEMICONDUCTOR. Application Note, Designing IP sensors using TINI. 12/31/2001. ed. [S.l.], 2001.

MAXIM/DALLAS SEMICONDUCTOR. Application Note 198, Networked Temperature Monitoring. 052902. ed. [S.l.], 2002.

MAXIM/DALLAS SEMICONDUCTOR. High-Speed Microcontroller User's Guide*:* DS80C390 Supplement. Rev: 111103. [S.l.], 2003.

MOTOROLA. SPI Block User Guide, Original Release Date: 21 JAN 2000, Revised: 06 Mar 2002. V02.06. [S.l.], 2002.

NöLKER, N.; KLEMENZ, A. Interfacing Slow Peripherals to D.Module via SPI, Application Note. D.signt 1999. [S.l.], 1999.

SCHNEEMAN, R. D. Implementing a standards-based distributed measurement and control application on the internet). United Stades Department of Commerce, NIST, Manufacturing Engineering Laboratory, Automated Production Technology Division, June 1999, p. 1–34, 1999.

SUN. Java Native Interface, http://java.sun.com/docs/books/tutorial/native1.1/concepts/index.html. June, 2004. [S.l.].

UBICOM. Application Note, Serial Peripheral Interface (SPI) and Microwire/Plus implementation Using the SX Communications Controller. November 2000. [S.l.], 2000.

TUTORIAL: STEPS FOR THE WEB TRANSDUCERS DEVELOPMENT USING EMBEDDED DEVICES

Abstract: *This paper is a tutorial that describes the steps for the web transducers development using embedded devices. The steps of development are presented, covering since the measure and/or actuation until the construction of the web interface, where de measured data by the sensor are showed. A temperature control system was developed to exemplify the use of a web transducers, that system consists in a temperature sensor with LM35 and a fan. The reading of the temperature and the control to actuate the fan was implemented whit a ADuC832 Microcontroller. The web interface were codified in Java and executed in the development platform TBM390 (TINI Tiny InterNet Interface) Board Model 390.*

Keywords: Web transducers, Network of transducers, Embebed systems, Microcontroller.