Centro Universitário de Brasília – UniCEUB
Faculdade de Ciências Exatas e Tecnologia – FAET
Engenharia de Computação

Aquisição e controle remoto de temperatura de um ar-condicionado, através do aparelho celular e Internet

por

Clesio Pinto Rabelo Júnior

20177486 – FAET – UniCEUB

Trabalho Final de Graduação

Professora Maria Marony

Orientadora

Monografia apresentada ao Centro Universitário de Brasília, para obtenção do título de Bacharel em Engenharia da Computação.

Brasília - DF – Junho de 2006

AGRADECIMENTOS

Agradeço principalmente aos meus pais: Clesio e Fatima, irmãos e sobrinhos pela paciência e todo tipo de apoio. E à Nádia.

As teses de matemática não são certas quando relacionadas com a realidade, e, enquanto certas, não se relacionam com a realidade.

Albert Einstein

RESUMO

Este projeto consiste na implementação de um sistema de automação Liga-Desliga de ar-condicionado, que utiliza tanto uma aplicação web, como uma aplicação J2ME como interface para o usuário. Também, pode ser utilizado para levantamento e análise gráfica da temperatura em função do tempo de um certo ambiente que se deseje monitorar.

O projeto integra software para celular (J2ME), Web (JSP e Servlet), microcontrolador e interfaces (circuitos elétricos e eletrônicos). Estes circuitos serão responsáveis por controlar a temperatura de um ambiente utilizando uma máquina térmica (aparelho de ar-condicionado, por exemplo). Os valores de temperatura são armazenados durante um período de tempo em um banco de dados (MySQL), automaticamente. Com o auxílio do programa Excel, é traçado o gráfico de temperatura versus tempo, do local monitorado. Colaborando assim, com a análise da energia térmica.

Palavras-Chave: J2ME, Servlet, Java Communications API, Microcontrolador.

ABSTRACT

This project consists of the implementation of a system of Bind-Disconnect automation of air-conditional, that it uses an application in such a way web, as an application J2ME as interface for the user. Also, it can be used for survey and graphical analysis of the temperature in function of the time of a certain environment that if it desires to monitor.

The project integrates software for cellular (J2ME), Web (JSP and Servlet), microcontroller and interfaces (electric and electronic circuits). These circuits will be responsible for controlling the temperature of an environment using a thermal machine (device of air-conditional, for example). The values of temperature are stored during a period of time in a data base (MySQL), automatically. With the aid of the Excel program, the graph of temperature versus time is traced, of the monitored place. Thus collaborating, with the analysis of the thermal energy.

Word-Key: J2ME, Servlet, Java Communications API, Microcontroller.

SUMÁRIO

LISTA DE SÍMBOLOS

A/D	Conversor Analógico Digital
API	Interface de Programação de Aplicativos
BTU/H	Brithish Thermal Unit for Hour
CDC	Connected Device Configuration
CLDC	Connected Limited Device Configuration
HTML	Hiper Text Markup Language
HTTP	Hyper Markup Language
IN	Entrada
INTR	Interrupção
IP	Internet Protocol – Protocolo de Internet
J2EE	Java 2 Enterprise Edition
J2ME	Java 2 Micro Edition
J2SE	Java 2 Standard Edition
JDBC	Java Database Connectivity
JSP	Java Server Pages
KVM	K Virtual Machine
MHZ	Mega Hertz
MIDP	Móbile Information Device Profile
MSB	Most Significant Bit – Bit mais Significativo
SQL	Structured Query Language – Linguagem de Consulta Estruturada
ºC	Graus Celcius
ODBC	Open Data Base Connectivity
ON-OFF	Liga-Desliga
OUT	Saída
PC	Personal Computer - Computador Pessoal
PID	Proporcional - Integral - Derivativo
RD	Reading
RX	Receptor
SGDB	Sistema Gerenciador de Banco de Dados
TCP	Transmission Control Protocol
TX	Transmissor
URL	Universal Resource Locator
USB	Universal Serial Bus
V	Volts
WR	Writing

ÍNDICE DE FIGURAS

ÍNDICE DE TABELAS

ÍNDICE DE EQUAÇÕES

LISTA DE TRECHOS DE CÓDIGO

CAPÍTULO 1 – INTRODUÇÃO

A automação de sistemas é uma área da tecnologia de crescente importância e aplicação em diversos setores da indústria e da sociedade moderna. Diversos são os tipos de sistemas existentes hoje em dia que são automatizados. Como, por exemplo, o controle de ponto eletrônico de funcionários de uma empresa, um foguete espacial ou um sistema de ar-condicionado.

Na automação de ar-condicionado, existem vários tipos de interfaces para o usuário. Tais como, um controle remoto convencional que é fornecido pelo fabricante do aparelho e aplicação do tipo desktop, criada por exemplo, pela empresa de automação Johnson Control, utilizada em sistemas de refrigeração de grande porte. Uma alternativa de interface para este tipo de sistema, seria uma que pudesse receber ordens vindas de um aparelho celular ou aplicação web, fornecendo mobilidade para o usuário.

O desenvolvimento de um sistema de automação térmico integrado com software Java para a web, para celular e microcontrolador da família 8051, seria uma solução de sistema que pudesse fornecer mobilidade para o usuário, e que pudesse ser utilizado como exemplo para outros tipos de sistemas automáticos.

1.1 – Objetivos do Projeto

- Criar um sistema de controle automático do tipo liga-desliga de temperatura de um ambiente previamente configurado, utilizando um microcontrolador.
- Demonstrar a viabilidade da utilização de um aparelho de celular e/ou sistema web, como interface para o usuário de um sistema de automação de ar-condicionado e obter informações sobre um sistema de controle automático de ar-condicionado.

- Armazenar os resultados da entrada e saída (temperatura) do sistema, durante um intervalo de tempo, em banco de dados MySQL. Dessa forma, é possível, com o uso de programa como o Excel, analisar graficamente os resultados de entrada e de saída.
- Fazer com que o sistema de controle receba comandos de entrada através de um programa J2ME que possa ser instalado em um celular ou de uma aplicação JSP (Web) que possa se acessada de um browser que esteja conectado à Internet, ambos utilizando conexão HTTP.

1.2 – Motivação

A primeira motivação é a possibilidade de poder contribuir para a área de automação de sistemas termomecânicos com a adição de novas entidades como: microcontrolador e software de alto nível orientado a objetos.

Também, por interesse pessoal e por possuir experiência prévia em desenvolvimento de sistemas J2EE e automação de sistemas termomecânicos, adquirida durante o decorrer do curso, agregando conhecimento teórico à prática.

Outra motivação é a aplicação de conhecimentos adquiridos durante o curso de Engenharia de Computação. Visto que é um dos poucos que capacitam para elaboração de projetos que envolvem programação tanto de baixo nível como também de alto nível junto com conceitos de hardware, controle e redes de computadores.

1.3 – Visão Geral do Projeto

O projeto pode ser representado de forma geral por dois esquemas que são apresentados nas **Figuras 1.1 e 1.2**. Sendo que a Figura 1.1 representa o software e a Figura 1.2 representa o hardware.

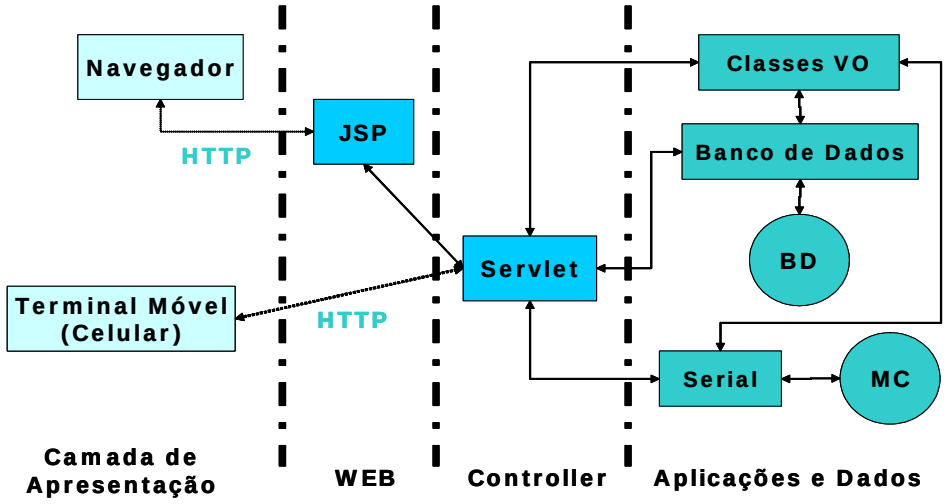


Figura 1. 1 - O software dividido em camadas

Onde:

BD – Banco de Dados

MC – MicroControlador

VO – Objeto de Visão

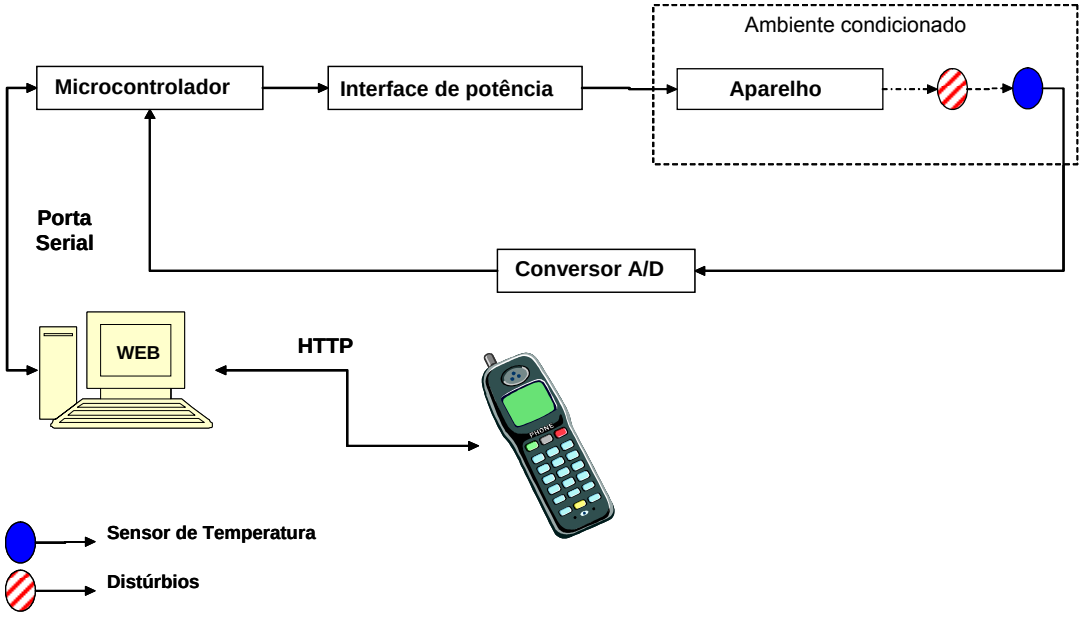


Figura 1. 2 - Diagrama em blocos simplificado do sistema

Pela Figura 1.1, observa-se que o software pode ser dividido em diferentes camadas. A camada de apresentação é a camada onde o usuário poderá interagir com o sistema. A camada de controle (controller) será formada por uma única aplicação chamada de Servlet, sendo a responsável por receber requisições HTTP. A camada de aplicações será a responsável por diversas funções no sistema, como interação com o banco de dados e comunicação através do da porta serial com o microcontrolador.

Na Figura 1.2 é mostrado como ocorre a interação entre os hardwares e interfaces que são usadas no sistema de forma resumida.

No celular, há um software do tipo cliente que pode se comunicar com um software do tipo servidor instalado em um servidor Web (PC – Personal Computer , Computador Pessoal). O PC, utilizando a sua interface serial irá se comunicar com o controlador digital (microcontrolador), que através de uma interface de potência, irá ligar ou desligar um aparelho de ar-condicionado. Para receber e processar a temperatura do ambiente, um sensor de temperatura e um conversor A/D são utilizados entre o aparelho de ar condicionado e o controlador.

Um software, em linguagem C, instalado no controlador digital é o responsável por realizar o controle em malha fechada da temperatura, em um ambiente previamente configurado. Para isso, deve ligar ou desligar o aparelho resfriador ou aquecedor, dependendo do sinal de entrada (set-point) e do sinal de saída do sistema (temperatura medida pelo sensor de temperatura). É utilizada a estratégia de controle do tipo ON-OFF. Assim, quando a temperatura medida pelo sensor estiver acima ou for igual temperatura máxima (set-point + atraso), então o controle irá ativar o aparelho refrigerador (caso aparelho aquecedor terá que desativar), caso a temperatura esteja abaixo ou igual a temperatura mínima (set-point – atraso) o controle terá que desativar o aparelho refrigerador (caso aparelho aquecedor terá que ativar).

1.4 – Organização da Monografia

- Na Introdução é demonstrado, de forma geral, o projeto.
- No Capítulo 2, “Hardwares e Interfaces do Sistema”, são abordados todos os circuitos elétricos e eletrônicos utilizados pelo sistema e os seus conceitos.
- No Capítulo 3, “Software Para Celular e Aplicação Web”, é descrito a aplicação cliente, que é formada por aplicação para celular (J2ME) e página para Web (JSP), e os conceitos envolvidos.
- No Capítulo 4, “Software do Servidor”, são mostrada as classes responsáveis pela comunicação com os clientes e microcontrolador. Como classe Servlet e classes que utilizam a API Java Comunications para estabelecer comunicação serial com o microcontrolador.
- No Capítulo 5, “Banco de Dados e Seu Software”, são apresentados o banco de dados utilizado no projeto e classes relacionadas com o mesmo.
- No Capítulo 6, “Software para Microcontrolador”, pode-se ver todo o software do microcontrolador, suas principais funcionalidades e alguns dos conceitos envolvidos sobre o Microcontrolador utilizado no projeto.
- No Capítulo 7, “Considerações Finais”, traz as dificuldades encontradas, os resultados obtidos, simulação, as conclusões e as sugestões para trabalho futuro.

CAPÍTULO 2 – HARDWARES E INTERFACES DO SISTEMA

2.1 – Introdução

Para a comunicação de um sistema com o outro é necessário à utilização de interfaces, sendo utilizadas para receber ou transmitir informações de e para o mundo externo. Para o sistema de controle de temperatura foi necessário o uso de quatro tipos de interfaces, interface de entrada, amplificadora e condicionadora de sinal, interface de potência e interface serial.

2.2 – Interface de Entrada

A interface de entrada é responsável por receber um sinal do tipo analógico e converter em digital para assim introduzir dados digitais dentro do controlador digital (microcontrolador) de forma que o mesmo possa entendê-lo e processá-lo. O tipo de sinal que será introduzido no microcontrolador para que ele possa processar e controlar será a temperatura ambiente, que é um tipo de grandeza física analógica que se comporta no tempo como uma função contínua, dependente do tempo e do espaço tridimensional. Será utilizado um sensor analógico de temperatura, que converte a temperatura em um sinal de tensão na razão (10mV/1ºC). Assim, para que seja possível que o controlador digital (microcontrolador) possa entender este tipo de sinal é necessária à conversão do sinal analógico em digital, onde um sinal digital será formado por uma cadeia de bits, representada por níveis lógicos 0 ou 1.

O conversor analógico digital (A/D) é o componente responsável pela transformação do sinal analógico em digital e este já tem internamente a sua própria interface de entrada.

2.2.1 – O Conversor A/D

No mundo real, os sinais são representados por funções contínuas no tempo, para que se possa trabalhar com o sinal digital deve-se converter o sinal do tipo analógico em digital. Este processo de conversão analógico para o digital consiste em:

a) Amostragem: converte o sinal de tempo contínuo em tempo discreto.
b) Quantização: é a representação dos sinais amostrados em um valor de amplitude definido analogicamente.
c) Codificação: é a geração de uma seqüência binária que represente o valor quantizado.

Sabe-se que o *teorema de Nyquist* estabelece que, um sinal analógico, o qual apresenta uma freqüência máxima (FMax), para que possa ser amostrado, processado e posteriormente recuperado e convertido novamente em analógico, a freqüência de amostragem mínima (inversa do período de amostragem) deve ser o dobro de FMax (freqüência máxima). O princípio citado, não precisa ser levado em consideração quando se trata da medição de sinais do tipo temperatura, pois este varia de forma muito lenta, sendo assim é comum amostrar esse sinal analógico a freqüências muito superiores ao dobro da máxima.

Trabalhar com o sinal digital possui várias facilidades, pois sinais em sistemas digitais podem ser armazenados e transmitidos com menor interferência a ruídos ou distorções, abre a possibilidade de inserirmos sistemas controlados

por software. Assim, o software será o responsável pelo algoritimo de controle e ou decisão, que muitas vezes não pode ser executado em sistemas analógicos.

O conversor utilizado no projeto é o "*ADC0804*" de 8 bits e que executa aproximações sucessivas para se chegar ao valor digital. Este tipo de conversor é muito utilizado pelo fato do tempo de conversão ser pequeno e não depender da entrada analógica. Sua resolução pode ser encontrada na **Equação 2.1**:

$$\text{Re}\,solução = \frac{5}{2^8} = \frac{5}{256} = 0{,}01953125 \qquad \textbf{Equação 2. 1}$$

A **Equação 2.1** é devido o conversor possuir uma tensão máxima na entrada analógica de 5 V e uma saída digital de 1 Byte. Dessa forma, a cada 0,01953125 V a saída digital irá mudar em um bit.

Uma lógica bastante conhecida dos profissionais de ilusionismo e de matemática, é a de usar algum tipo de algoritimo que os ajudem a adivinhar com o menor número de perguntas possível o número que uma pessoa pensou. A técnica de aproximações sucessivas utiliza em sua lógica de controle algo muito parecido.

Por exemplo, seja um conversor de 4 bits com resolução de 1 V e tensão de entrada analógica Va = 10,4 V.

Primeiro a tensão de referência (Tref) vai para 8V, pois o bit mais significativo é colocado em 1 ($1000_b=8_d$). Portanto, Tref < Va, e o (MSB) é mantido em 1. Depois o próximo bit é colocado em 1 ($1100_b=12_d$), assim Tref(12) > Va(10,4) e o bit volta a zero. Agora temos Tref=($1000_b=8_d$). Em seguida o terceiro bit é colocado em 1 ($1010_b=10_d$), então a lógica pergunta Tref(10) é menor ou maior que Va (10,4), recebe como resposta: "é menor", então mantém o terceiro bit em 1. Por fim, o ultimo bit é posto em 1 ($1011_b=11_d$) e Tref = 11 V, com Tref > Va o ultimo bit é zerado. No momento todos os 4 bits foram testados e o resultado

está guardado no registrador. Dessa forma o equivalente digital de Va (10,4 V) é de 10 V.

O Conversor A/D ADC0804 possui as seguintes características básicas:

Permite entrada Vin(+) e Vin(-) diferencial. Sendo que a entrada analógica real é a diferença entre as duas entradas (Vin(+) – Vin(-)). Normalmente usa-se conectar Vin(-) no terra ficando com a entrada analógica real igual a Vin(+).Utiliza entrada analógica de 0 a 5 V e é alimentado com Vcc = 5 V.

A saída digital desse conversor tem um *buffer tristate*, isto é, desconectado caso bit de controle (CS - Chip Select – seleção do chip) esteja em nível lógico zero.

Possui gerador de clock interno onde pela **Equação 2.2**, chega-se à freqüência de amostragem (fa) com R (resistência) e C (capacitor) externos.

$$fa = \frac{1}{1{,}1 * R * C}$$ **Equação 2. 2**

Possui um tempo de conversão de aproximadamente 100 μs, com uma freqüência de aproximadamente 606 kHz.

Possui 4 bits de controle CS, RD, WR e INTR, que são explicados em seguida.

O bit CS (Chip Select – seleção do chip) deve estar em nível lógico 0 para que as entradas RD e WR tenham efeito. Com CS em nível lógico 1 as saídas digitais ficarão com alta impedância.

O bit RD (Read – lido) deve estar em nível lógico 0 para que na saídas digitais constem o resultado da ultima conversão.

O bit WR (Write – escrever) deve receber um pulso para baixo para se iniciar uma nova conversão. Este pulso pode ser um tempo maior que 100 μs.

O bit INTR (Interrupt – interrupção) vai para nível lógico 1 no inicio da conversão e para 0 no fim da conversão.

Possui o pino para tensão de referência Vcc/2 que pode assumir vários valores mudando assim a resolução do conversor. Normalmente, usa-se este pino em 2,25 V, isto é, basta não conectar nada em seu terminal.

Possui também os pino Clock Out e Clock In, sendo que para habilitar a utilização do clock interno basta conectar um resistor ao pino Clock Out e um capacitor ao pino Clock In. Caso contrario pode-se utilizar clock externo conectado ao pino Clock In.

2.2.2 – O circuito utilizado como interface de entrada

Na **Figura 2.1** é mostrado o circuito utilizado como interface de entrada. O qual é constituído por um conversor A/D (ADC0804), de 8 bits de resolução, que se comunica de forma paralela com o microcontrolador e um circuito RC que terá uma constante de tempo "**T"** com valor igual à R*C (resistência vezes capacitância). Este fornece níveis de tensão de 0V ou 5V ao microcontrolador através dos pinos 11 ao 18 ou D0 a D7 (D0=P2_0), representando níveis lógicos 0 e 1 respectivamente. Os bits 1 ao 3 e 5 são bits de controle.

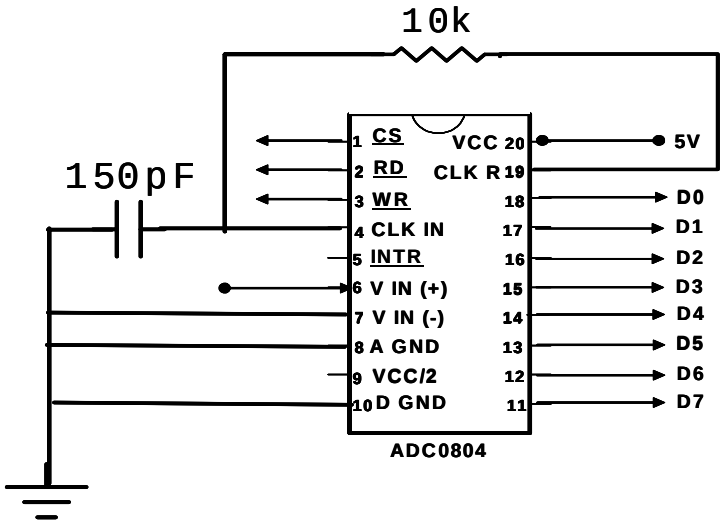


Figura 2. 1 - Conversor A/D utilizado como interface de entrada

2.3 – Interface Amplificadora e Condicionadora de Sinal

O sensor "*LM35DZ*" representado pela **Figura 2.2**, possui uma variação linear de 10mV para cada ºC. Sendo assim, em uma temperatura de 100ºC o este sensor apresentará em sua saída uma tensão de 1 V. Como já foi mencionado, o sensor é ligado ao conversor A/D para que o microcontrolador possa interpretar o sinal de temperatura. O conversor trabalha em uma faixa de tensão de 0 a 5V, para sinais de entrada. Neste caso, para elevar e condicionar a tensão na faixa especificada, antes de ligar o sensor ao conversor, este passará pela interface condicionadora e amplificadora de sinal, **Figura 2.3.** Fornecendo, desta forma, uma tensão máxima de 5 volts para 50ºC e no mínimo 0 volts para a temperatura de 0ºC, pois o sensor não trabalha com temperatura negativa, conforme diagrama em blocos da **Figura 2.4**.

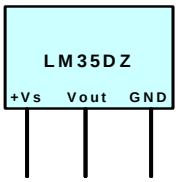


Figura 2. 2 - Sensor de temperatura LM35DZ

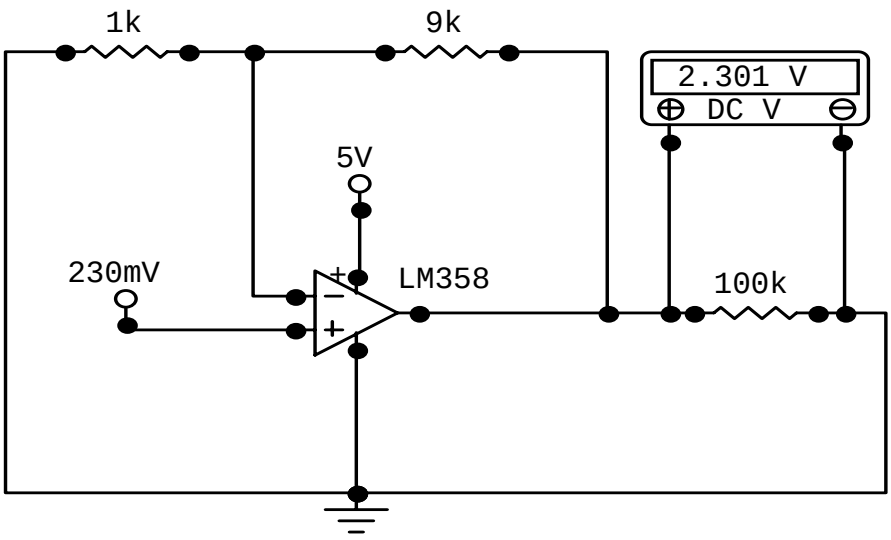


Figura 2. 3 - Interface condicionadora e amplificadora de sinal

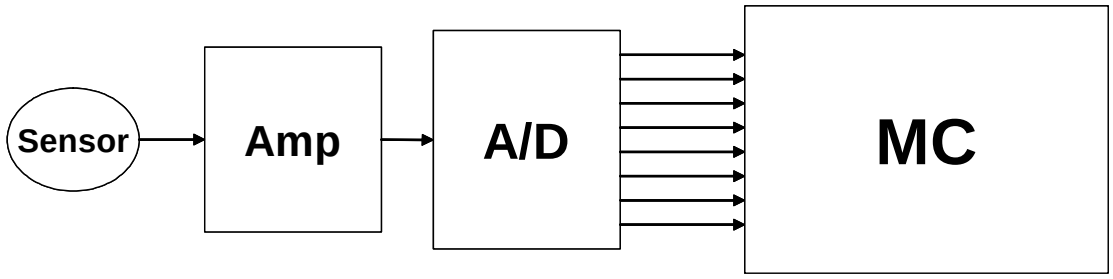


Figura 2. 4 - Diagrama em blocos representado a ligação entre o sensor com as interfaces e microcontrolador

2.3.1 – O Amplificador Operacional

2.3.1.1 – Introdução

O Amplificador Operacional é um circuito integrado muito utilizado na eletrônica e que possui o termo operacional porque é usado para implementar as operações matemáticas de integração, diferenciação, adição, mudança de sinal e multiplicação por um fator constante. Além de ser usado em muitas outras aplicações, como por exemplo, em filtros. No projeto, a operação de multiplicação por um fator constante, é a aplicação utilizada.

2.3.1.2 – Terminais do Amplificador Operacional

O amplificador operacional utilizado no projeto é o "*LM358*" da *National Semiconductor* que vem em um circuito integrado com encapsulamento chamado de DIP 8 pinos (Dual in-line package – encapsulamento duplo em linha), isto é, os terminais de cada lado do encapsulamento estão alinhados e os terminais em lados opostos do encapsulamento também estão alinhados. Como mostrado na **Figura 2.5**.

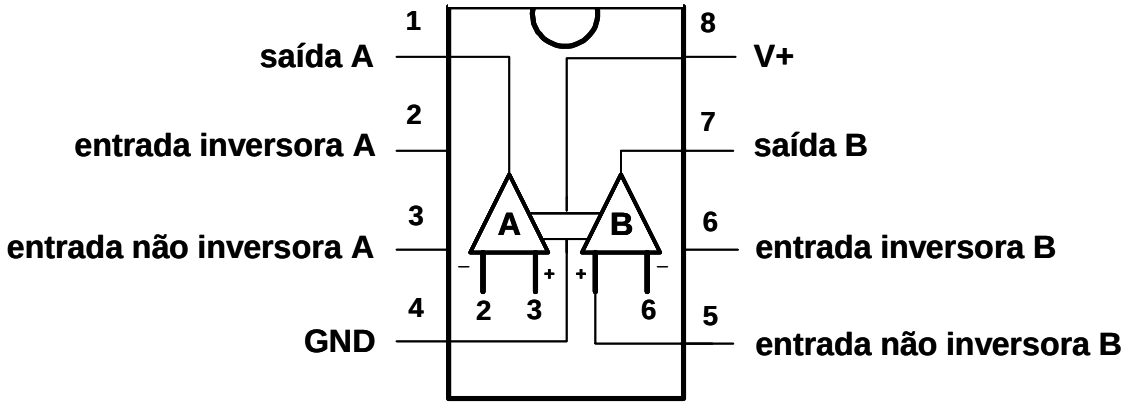


Figura 2. 5 - Terminais do Amplificador Operacional LM358

2.3.1.3 – Análise matemática do amplificado operacional

Na **Figura 2.6** observa-se o chamado ***curto-circuito virtual*** nas entradas mais (+) (não inversora) e menos (-) (inversora) do amplificador LM358. Isto se deve ao fato de que a entrada inversora está ligada ao terminal de saída. Esta ligação é chamada de realimentação negativa pois o sinal realimentado da saída para a entrada é subtraído do sinal de entrada. Pode ser explicado também pelo fato de a entrada possuir uma alta impedância, da ordem de 1 MΩ. Dessa forma, a corrente em R1 (resistor número 1) será igual a corrente em R2 (resistor número 2).

Assim, podemos identificar uma expressão capaz de relacionar R1, R2, Vs (tensão de saída) e Ve (tensão de entrada). Podem-se ver as deduções nas Equações 2.1 a 2.8. No entanto, uma forma rápida e prática para o levantamento correto dos resistores é através do software “Circuit Maker” que oferece o componente LM358 como amplificador operacional, assim, utilizando-se de valores comerciais de resistores pode ser feito a análise.

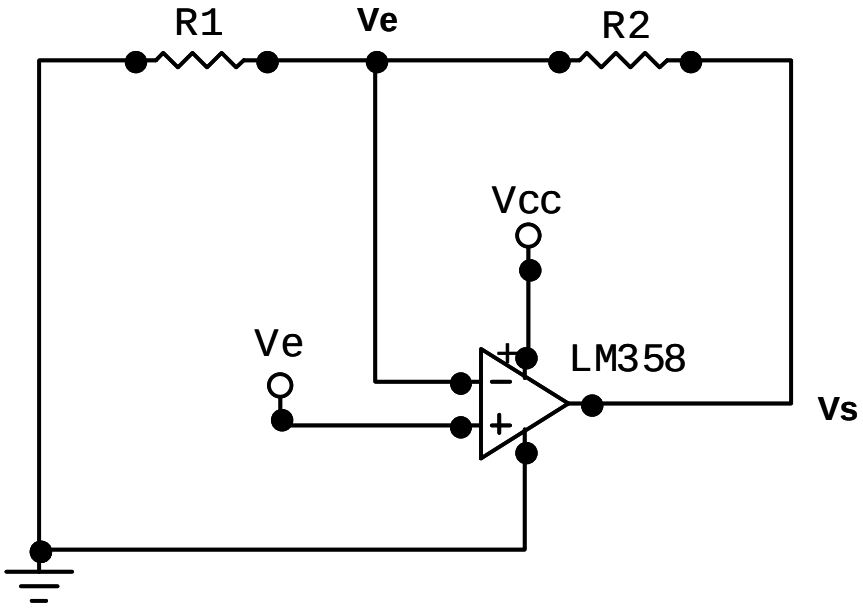


Figura 2. 6 - Circuito com o Amplificador Operacional

$$\frac{Ve-0}{R1}=I$$ **Equação 2. 3**

$$\frac{Vs-Ve}{R2}=I$$ **Equação 2. 4**

$$\frac{Ve-0}{R1}=\frac{Vs-Ve}{R2}$$ **Equação 2. 5**

$$Vs=\frac{Ve*R2}{R1}+Ve$$ **Equação 2. 6**

$$\frac{R2}{R1}=Kr$$ **Equação 2. 7**

$$Vs=Ve*Kr+Ve$$ **Equação 2. 8**

Caso **Ve** seja igual a 240 mV e **R1** = 1 KΩ e **R2** = 9 KΩ, teremos pela **Equação 2.9** o valor de Kr = 9. Assim, Vs pela **Equação 2.10** será de 2.4 V.

$$Kr=\frac{9K\Omega}{1K\Omega}=9$$ **Equação 2. 9**

$$Vs=Ve*9+Ve=10Ve=2.4volts$$ **Equação 2. 10**

2.4 – Interface de Potência

Para o controle da temperatura do ambiente utilizando um aparelho de ar-condicionado (tipo janela) por exemplo, será necessário que o aparelho forneça mais ou menos energia térmica ao ambiente. Para resfriar o ambiente e necessário ativar um componente do aparelho de ar-condicionado chamado de compressor, estes fornecem de 6.000 a 30.000 BTU/h de energia (sendo que 1 BTU/h é equivalente a 0,2931 W de potência média) a 220 V, dependendo do modelo e fabricante do aparelho de ar-condicionado.

Para ligar ou desligar o compressor, é necessário que se amplifique o sinal de controle vindo do microcontrolador, aumentando sua potência, para que este atue em dispositivos de manobra elétricos, relé e contatora. Portanto, utiliza-se a interface de potência composta pelo circuito apresentado na **Figura 2.7**. Sua função será de amplificar a potência dos sinais provenientes da saída do microcontrolador para aplicá-los sobre o elemento de manobra, o relé. Neste caso, o dispositivo de manobra responsável pelo acionamento do aparelho ou do compressor é composto de um relé de 12 V e contatora, conforme esquema proposto na **Figura 2.8**. A contatora 3TB44 da Siemens que foi utilizada no projeto, suporta cargas trifásicas com elevada corrente de pico, (7,5 a 12 Amperès, Bobina em 220 volts).

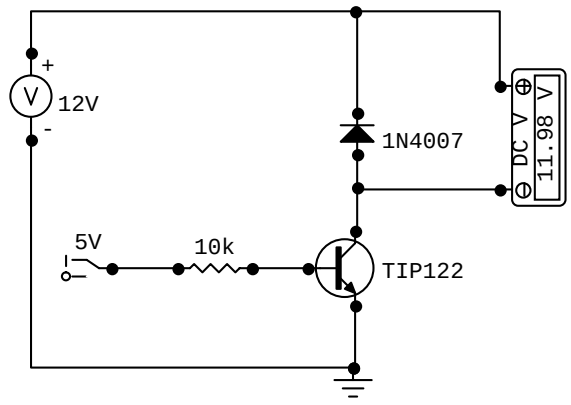


Figura 2. 7 – Circuito da interface de potência

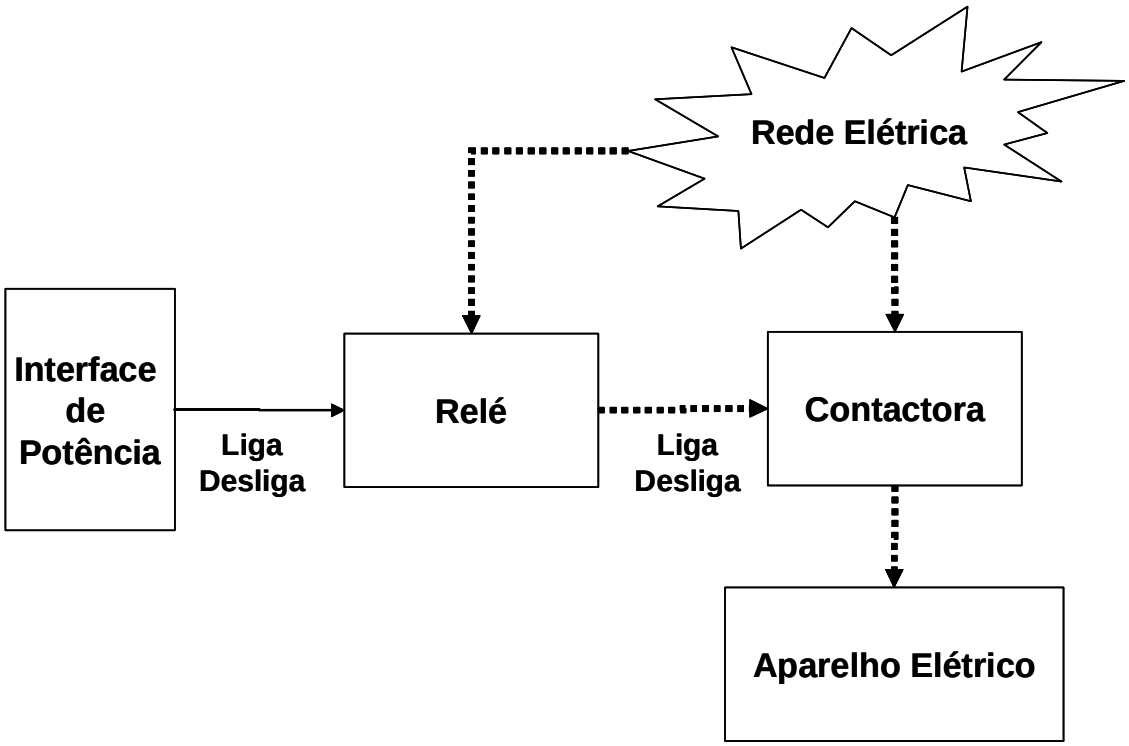


Figura 2. 8 - Interligação dos dispositivos de manobra, relé e contatora

O circuito apresentado na **Figura 2.7** é chamado de “Drive para Relay”, o qual recebe como entrada um bit com níveis de tensão de 0 ou 5V (**TTL**), e fornece a tensão e corrente necessária para o funcionamento do elemento atuador. Funciona da seguinte maneira: - quando a entrada for um nível lógico “1”, ou 5V, ativa o atuador, e quando na entrada da interface tiver um nível lógico “0”, ou 0V, o atuador será desligado.

O transistor funcionará em corte e saturação, ficando em corte quando em sua entrada tiver 0V, em saturação quando em sua entrada tiver 5V. Dessa forma, para o levantamento do circuito pode-se utilizar o software “Circuit Maker” de forma rápida e prática.

Para a proteção do transistor é utilizado um diodo inversamente polarizado em paralelo com a carga indutiva (relé), chamado de “diodo de Clamp” deve-se

utilizá-lo sempre que for colocada uma carga indutiva no coletor do transistor, assim quando é desligada a carga indutiva, a energia armazenada nesta é descarregada pelo diodo.

2.5 – Interface serial

2.5.1 – Introdução

Ainda hoje com o surgimento da tecnologia de comunicação USB, os computadores padrões vem equipados com porta serial e paralela. Sendo a porta paralela usada para interligação de impressoras, projetos específicos, etc. A porta serial para interligação de mouse, projetos específicos, comunicações como robôs, modens, etc. Esta interface será utilizada para a comunicação entre o microcontrolador e o Servidor. Conforme esquema mostrado na **Figura 2.9**.

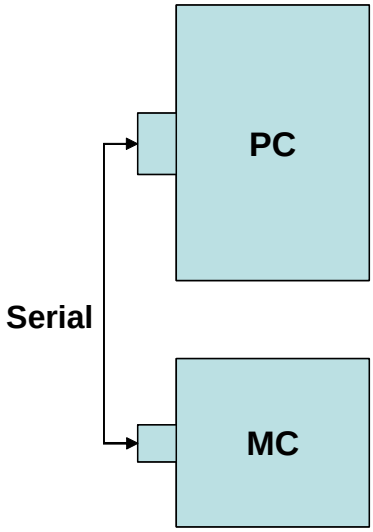


Figura 2. 9 - Ligação do Pc Servidor ao Microcontrolador

2.5.2 – O padrão RS-232C

O padrão chamado de “RS232” é uma norma da EIA (Eletronics Industries Associates – Associação das Industrias de Eletrônica) do tipo RS (Recommended Standard – Padrão Recomendado) que possui o nome de EIA – 232-C. Especifica

a transmissão entre os equipamentos TX e RX, padrão de conector e níveis de tensão.

Os conectores podem ser do tipo DB25, com 25 pinos ou DB9, com 9 pinos. No projeto será utilizado o conector DB25, conector macho (pinos para fora) para o Servidor e o DB9, conector fêmea (pinos para dentro, receptor) para o Microcontrolador. Computador pessoal (Servidor) do tipo padrão IBM com duas portas seriais e uma paralela.

Os níveis de tensão adotados pelo padrão RS-232 são diferentes dos níveis de 0 ou 5V. Pode ser verificado a diferença dos níveis na **Tabela 2.1**.

Tabela 2. 1 - Comparação entre tensões TTL e RS-232

Nível Lógico	TTL (V)	RS-232 (V)
0	0	+12
1	5	-12

Portanto, para se realizar a comunicação com o microcontrolador que enxerga níveis de 0 ou 5V, usa-se um circuito conversor TTL/RS232 e RS232/TTL. O chip utilizado para isso é o MAX232 da empresa MAXIM, maiores detalhes sobre este pode ser adquirido na folha de dados do fabricante fornecida gratuitamente no site “http://www.ortodoxism.ro/datasheets/maxim/MAX220-MAX249.pdf”. A maioria das placas para gravação no microcontrolador da família 8051 já vem com o chip MAX232.

A comprimento máximo do cabo para comunicação entre o ponto de transmissão e o ponto de recepção deve ser de 15 metros. Para uma comunicação full-duplex (o receptor e transmissor enviam e recebem ao mesmo tempo) assíncrona (sem bit de controle necessário para sincronismo entre transmissor e receptor). A comunicação utilizado no projeto será do tipo

assíncrona pois não há necessidade de sincronismo entre o PC e o microcontrolador.

2.5.3 – A comunicação serial

A comunicação serial é feita por apenas um canal de informação, onde os bits são enviados em série, um atrás do outro. Na comunicação assíncrona normalmente utiliza-se 8 bits para os dados e dois bits para controle (palavra), sendo um bit de start e o outro de stop, conforme **Tabela 2.2** . Pela tabela pode-se ver o envio da letra “a” em código ASCII.

Tabela 2. 2 - Comunicação Assíncrona de 8 bits de dados e dois de controle

start	**dados**								**stop**
Transição 1 para 0	**0**	**0**	**1**	**1**	**0**	**0**	**0**	**1**	Transição 0 para 1

Ao iniciar a transmissão será enviado um bit de start com transição de 1 para 0 indicando que os bits seguintes serão os de dados, ao término será enviado um bit de stop com transição de 0 para 1 indicando que chegou ao fim a transmissão. Pode ser enviado também um outro bit, chamado de bit de paridade (par ou ímpar), para facilitar a detecção de erros na transmissão e, caso eles ocorram, fazer com que a palavra seja retransmitida. Caso adotado a paridade do tipo par, isto é, o número de bits com níveis lógicos 1 de ser contado e se o número for par o bit de paridade será “0” caso contrario “1”, o receptor poderá identificar o erro da seguinte forma:

O transmissor envia a seguinte palavra. **Tabela 2.3**.

Tabela 2. 3 - Bits enviados pelo transmissor

b7	**b6**	**b5**	**b4**	**b3**	**b2**	**b1**	**b0**	**bp**
0	0	1	1	1	1	0	1	1

O receptor recebe a seguinte palavra. **Tabela 2.4**.

Tabela 2. 4 - Bits recebidos pelo receptor

b7	b6	b5	b4	b3	b2	b1	b0	bp
0	0	0	1	1	1	0	1	1

Assim, o receptor irá contar o número de ‘1’, se for par, conclui que a palavra foi correta, se for ímpar, conclui que houve um erro de transmissão. Método pouco eficaz pois não garante que o dado recebido foi realmente transmitido de forma correta, pois caso 2 bits estiverem errados, o método fracassou. Então este não constitui uma solução definitiva para detectar e corrigir erros nas transmissões, métodos mais modernos são estudados e utilizados sempre que necessário.

CAPITULO 3 – SOFTWARE PARA O CELULAR E APLICAÇÃO WEB

3.1 – Introdução

A aplicação para o aparelho celular foi desenvolvida utilizando-se a tecnologia J2ME (Java 2 Micro Edition – Edição Micro do Java 2), com suporte para muitos dispositivos, com memória, capacidade de vídeo e recursos limitados, portabilidade entre plataformas e com uma grande aceitação entre os fabricantes de dispositivos móveis.

3.2 – Software J2ME

Para permitir que muitos dispositivos móveis pudessem utilizar a tecnologia J2ME, a *Sun Microsystems*, empresa responsável pela tecnologia, introduziu o conceito de Configuração. Uma configuração permite que muitos dispositivos possam usufruir uma mesma plataforma Java. São duas as configurações que estão disponíveis nos diferentes dispositivos móveis (celulares, pager's ou PDA's):

A – Configuração de Dispositivo Conectado (**CDC**):

- Aparelho com 512 kbytes (no mínimo) de memória para executar o Java.
- 256 kbytes (no mínimo) de memória para alocação de memória em tempo de execução.
- Conectividade de rede, largura de banda alta.

B – Configuração de Dispositivo Conectado Limitado (**CLDC**):

- 128 kbytes de memória para executar o Java.
- 32 kbytes para alocação de memória em tempo de execução.

- Interface restrita com o usuário.
- Conectividade de rede, dispositivos sem fio com largura de banda baixa.

Os dispositivos móveis possuem uma vasta variação de recursos tecnológicos, como tamanho de tela e teclados diferentes, para tratar dessa diferenciação foi criado um tipo de extensão das configurações A e B citadas acima, chamada de perfil.

Um perfil fornecerá bibliotecas específicas para desenvolvimento para um tipo particular de dispositivo. Por exemplo, um celular pode seguir um perfil diferente de um Pager, porém, os dois podem pertencer à mesma configuração.

O desenvolvimento das Configurações e Perfis são definidos pelos grupos de trabalho utilizando a comunidade "*Java Community Process Program*" da Sun Microsystems, maiores informações podem ser verificadas em http://jcp.org. Na **Figura 3.1** é mostrada a arquitetura de software da tecnologia J2ME.

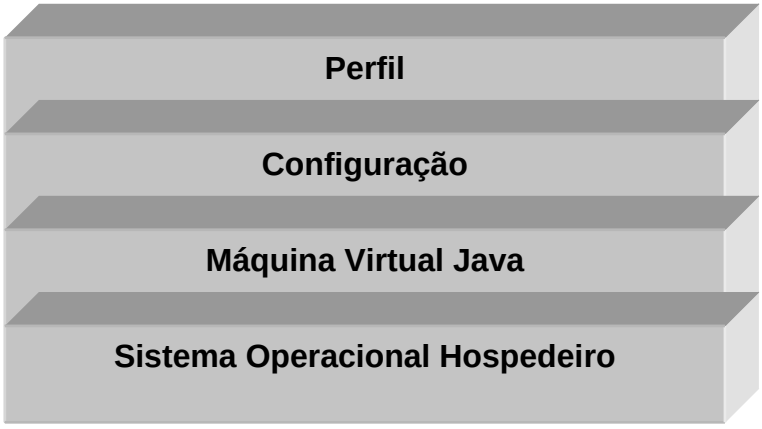


Figura 3. 1 - Arquitetura de software J2ME

Os aparelhos celulares normalmente suportam a configuração CLDC e o perfil chamado de MIDP (Móbile Information Device Profile, Perfil de Dispositivo de Informação Móvel). O MIDP, que é fornecido pela Sun, define APIs

(Application Program Interface, Interface de Programação de Aplicativos) para interligação em rede e tratamento de interface para o usuário, dentre outras.

Nos tipos de aparelhos celulares citados acima será executada uma máquina virtual diferente da tradicional utilizada pela tecnologia J2SE, conhecida como KVM (K Virtual Machine , Máquina Virtual K) escrita em linguagem C, sendo que a maior parte do código escrito será independente da plataforma. Exige de 40 a 80 kbytes de memória, 20 a 40 kbytes de memória dinâmica (heap) e processamento de 16 bits em freqüência de 25 MHz. Utiliza os arquivos .class (arquivos compilados em código de byte) e os transformam em código de máquina específico, sendo responsável por outras funções, como alocação e desalocação de memória. Na **Figura 3.2** é mostrada a arquitetura de software normalmente usada pelos aparelhos celulares. Na **Figura 3.3** são mostradas as várias edições do Java.

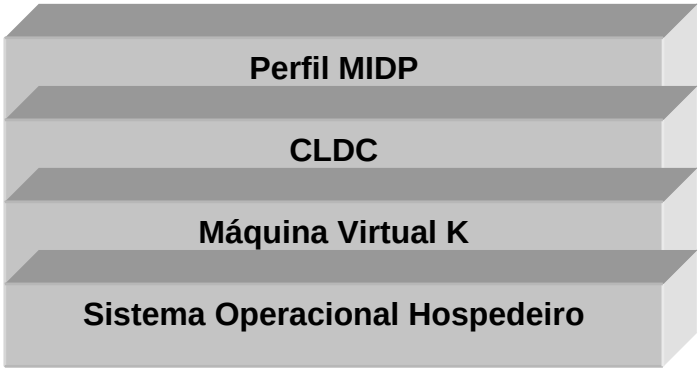


Figura 3. 2 - Arquitetura de software J2ME

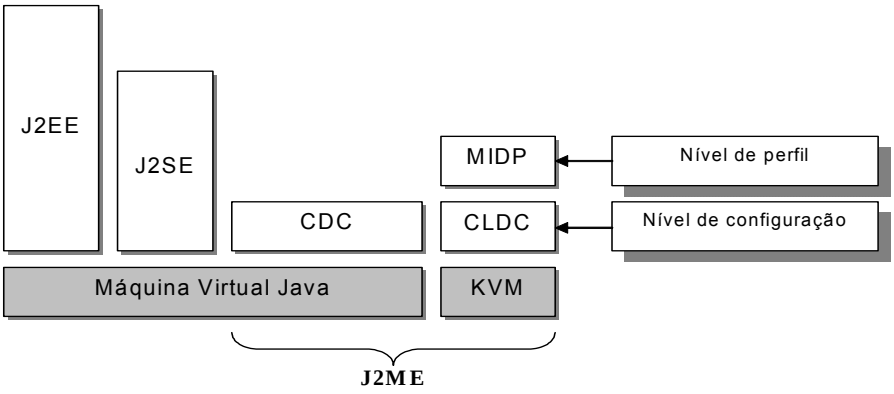


Figura 3. 3 - Edições do Java

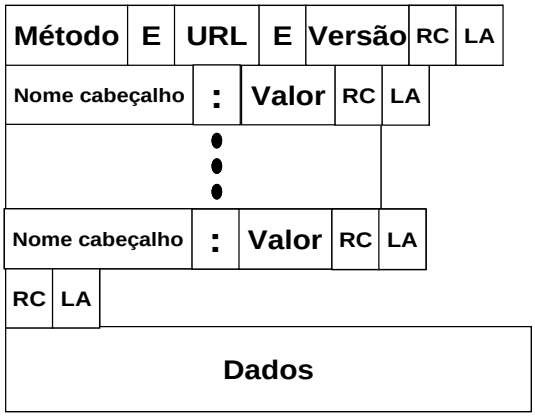
3.2.1 – Conexão HTTP

A implementação real do protocolo se dá em nível de perfil, no MIDP 2.0 o único protocolo implementado é o HTTP na versão 1.1 que tem sua especificação disponível no endereço: “http://www.ietf.org/rfc/rfc2616.txt”. Utilizando a classe HttpConection é possível se comunicar com um servidor Web ou com outro dispositivo remoto que suporte HTTP.

O HTTP é um protocolo de pedido e resposta. O cliente terá que iniciar um pedido, envia-o para um servidor usando um endereço especificado como URL (Uniform Resouce Locator – Endereço Virtual) e uma resposta é retornada do servidor. Usa o protocolo TCP como protocolo de transporte.

O cliente (celular ou página da web) envia uma mensagem para sua interface de porta, passando para o TCP que fornece ao HTTP um serviço confiável de transferência de dados. O servidor recebe e responde a requisição usando sua interface de porta. O HTTP/1.1 usa conexão do tipo persistente, que não são fechadas pelo servidor.

A mensagem de requisição é formada por linha de requisição, linha de cabeçalho e corpo de dados quando o tipo de método de pedido for POST. Ao enviar utilizando o método de envio GET o corpo de dados vai como parte da URL. Na especificação são definidos mais de 40 campos de cabeçalhos. A mensagem de resposta é formada por linha de status, linha de cabeçalho e o corpo de dados. Os formatos das mensagens são definidos conforme **Figura 3.4** para requisição e **Figura 3.5** para resposta.

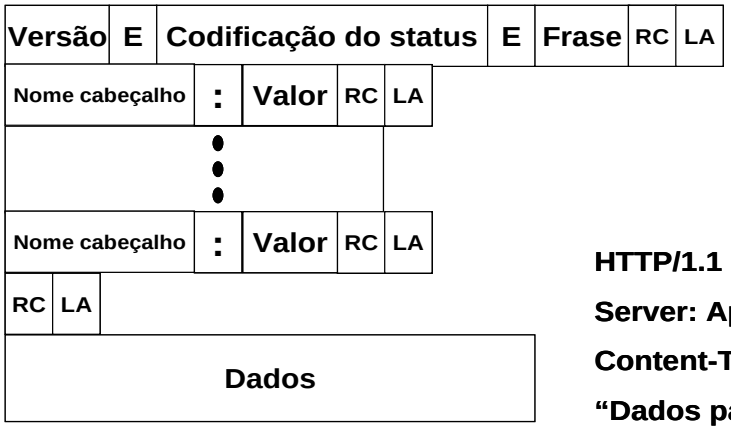


Exemplo real

POST 10.1.1.3/servlet HTTP/1.1
Host: 10.1.1.3
Accept-language: BR
"Dados para envio"

E – espaço em branco
RC – retorno do carro
LA – linha de alimentação

Figura 3. 4 - Formato de mensagem de requisição HTTP e exemplo



Exemplo real

HTTP/1.1 200 OK
Server: Apache/1.3.0 (Unix)
Content-Type: text/html
"Dados para resposta"

E – espaço em branco
RC – retorno do carro
LA – linha de alimentação

Figura 3. 5 - Formato de mensagem de resposta HTTP e exemplo

3.2.2 – Conectando-se ao Servidor Web

Para o cliente J2ME se conectar e enviar dados ao servidor web, são necessários três passos. Preencher corretamente a linha de requisição, linha de cabeçalho e criar um fluxo para saída dos dados. Para a linha de requisição e necessário que se faça conforme **Trecho de código 3.1**.

```
String url = “http://10.1.1.3:8080/nomeServlet”;
HttpConnection conexao = (HttpConection) Conector.open(url);
conexao.setRequestMethod(conexao.POST);
```

Trecho de Código 3. 1 - Criando conexão HTTP

É necessário que o método responsável pela conexão ao servidor Web seja chamado utilizando-se uma Thread separada, pois caso o servidor web não responda a aplicação poderá continuar seu fluxo normal. Exemplo de utilização de método em uma thread separada no **Trecho de código 3.2**.

```
Thread threadQualquer = new Thread()
{
  public void run()
  {
     conecta();
  }
};
threadQualquer.start();
```

Trecho de Código 3. 2 - Thread necessária para conexão

3.3 – Desenvolvimento da aplicação para o celular

A Midlet, como é usualmente chamada uma classe desenvolvida utilizando a tecnologia J2ME, foi criada com o nome de *ControleUsuarioMidlet.java*, uma aplicação compatível com a configuração CLDC 1.1 e perfil MIDP 2.0.

Na **Figura 3.6** pode-se observar a tela inicial de login, simulada pelo programa “*J2ME Wireless Toolkit 2.2*” fornecido gratuitamente pela Sun, em *www.sun.com*.

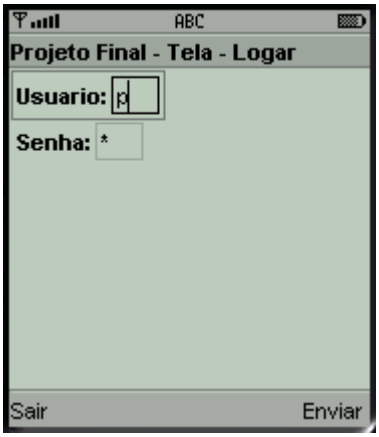


Figura 3. 6 - Tela Inicial para login

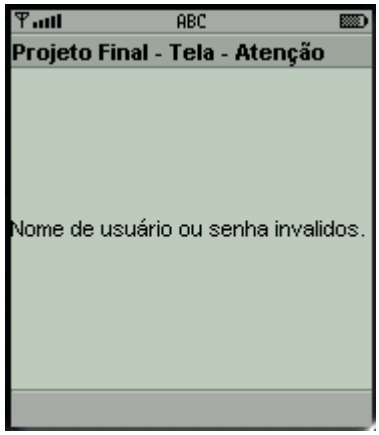


Figura 3. 7 - Tela de alerta para o login errado

A lógica para o processo de “login”, a qual o usuário com nome e senha, poderá ter acesso ao sistema, pode ser verificada no fluxograma mostrado na **Figura 3.12**. Na **Figura 3.7** a tela que é exibida quando o usuário não for reconhecido pela aplicação.

Ao se logar corretamente no sistema o usuário verá a tela apresentada na **Figura 3.8**. Esta, irá fornecer quatro opções ao usuário como: opção para se verificar o status do aparelho (ligado ou desligado e temperatura ambiente) e opção para se fornecer o Set-Point (entrada) para o modulo de controle. Ao entrar em Set-Point verá a tela apresentada na **Figura 3.9**, podendo fornecer valores de 18 a 30 ºC (definido na lógica do programa, podendo ser alterado). Nesta tela as opções serão vistas em um menu como apresentado na **Figura 3.10**. O usuário somente poderá digitar valores entre 18 a 30 ºC, caso não digite, será mostrada uma tela de alerta como apresentado na **Figura 3.11**.

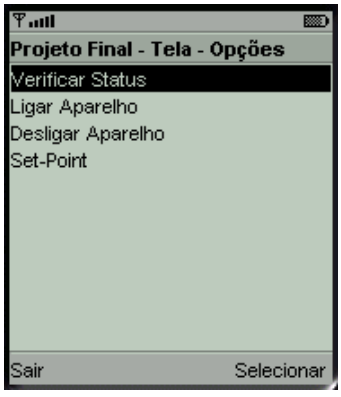


Figura 3. 8 - Tela de opções

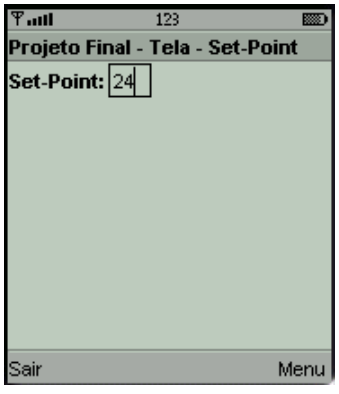


Figura 3. 9 - Tela para entrada de Set-Point

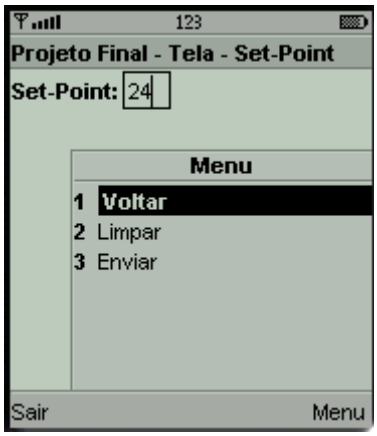


Figura 3. 10 - Opções fornecidas na tela de Set-Point

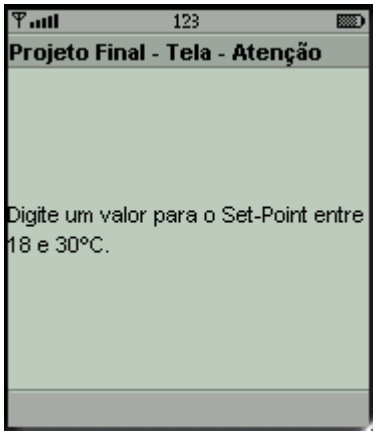


Figura 3. 11 - Tela de atenção

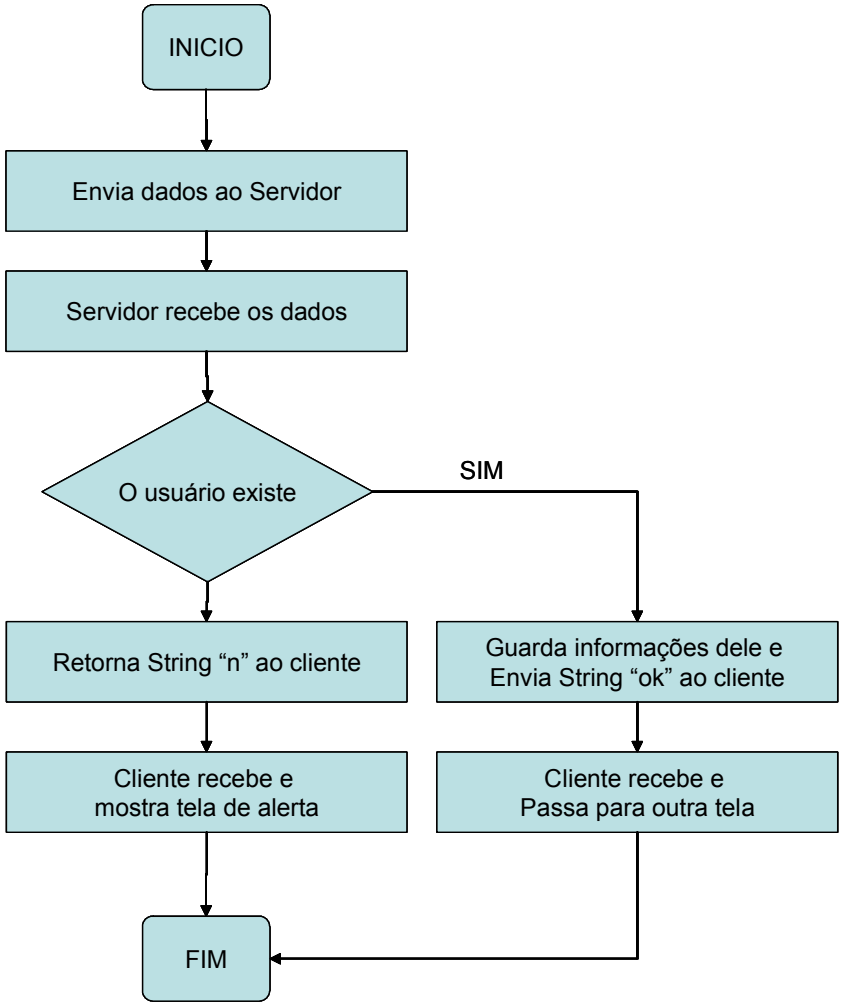


Figura 3. 12 - Fluxograma do processo de login

As telas de alerta são mostradas por um tempo de quatro segundos (tempo ajustável), juntamente com um aviso sonoro.

3.4 – Aplicação para a WEB em página dinâmica JSP

3.4.1 – Introdução

Neste tópico será mostrada a aplicação que pode ser acessada por browser como o Internet Explorer.

3.4.2 – JavaServer Pages (JSP)

Esta aplicação é executada a partir de um servidor Web. Esta tecnologia se baseia na tecnologia Java Servlet. Uma aplicação JSP contém código encontrados normalmente em paginas estatísticas HTML e também aceita trechos de código na linguagem Java.

Com este tipo de aplicação cria-se com facilidade uma aplicação cliente que é executada a partir de um browser.

3.4.3 – A aplicação cliente em JSP

Para o projeto foram criadas duas páginas utilizando a tecnologia JSP apresentadas nas **Figuras 3.13 e 3.14**. Vídeo em resolução de tela de 1024 por 768 pixels (aglutinação de Picture e Element, ou seja, elemento da imagem).

Na Figura 3.13 é apresentado à tela inicial de login e na Figura 3.14 é apresentado à tela de comandos. Esta última tela permite que com um click do mouse se ative um aparelho de ar-condicionado.

Ao realizar o login, o usuário não poderá fechar a janela apresentada na Figura 3.13, pois ao fazer isso terá que realizar um novo login. O processo de login é muito parecido com o processo da aplicação J2ME.

Na tela de comandos, as opções do usuário são auto-explicadas pelos botões de ação.

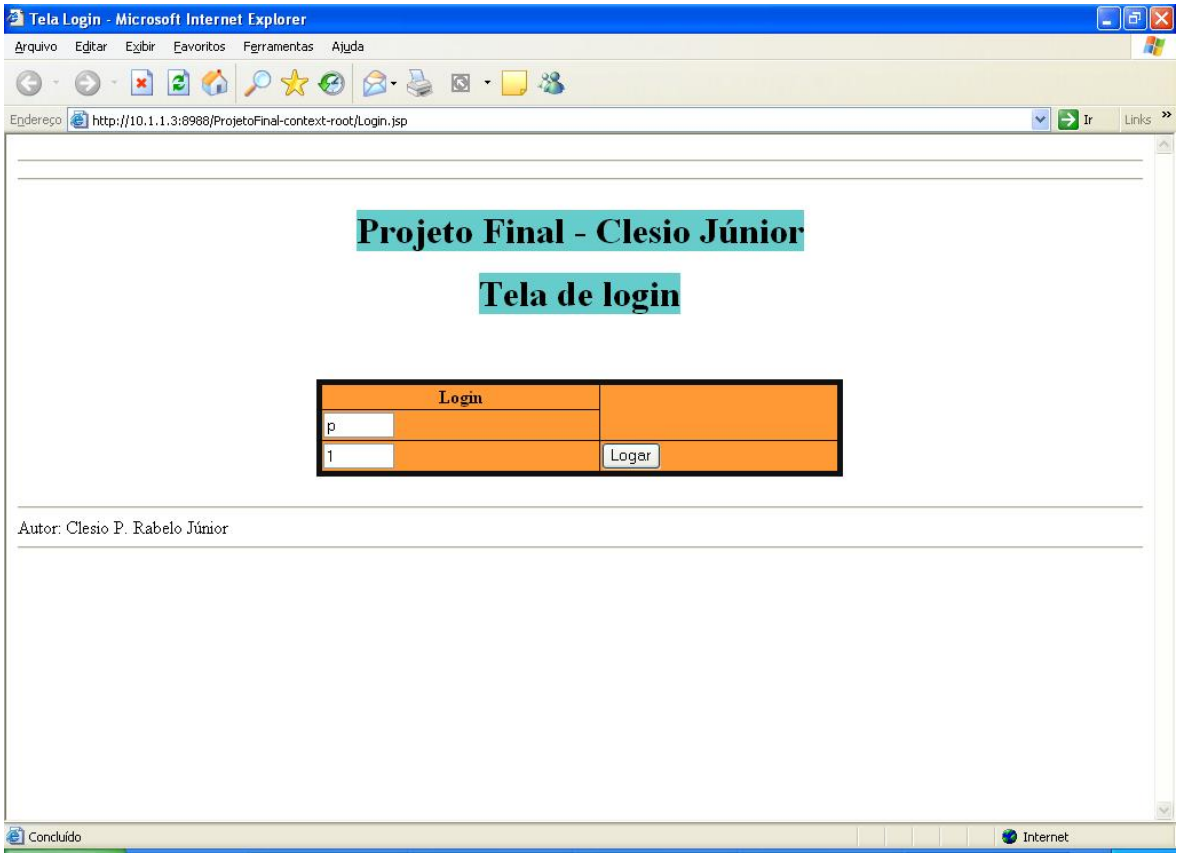


Figura 3. 13 - Tela de login

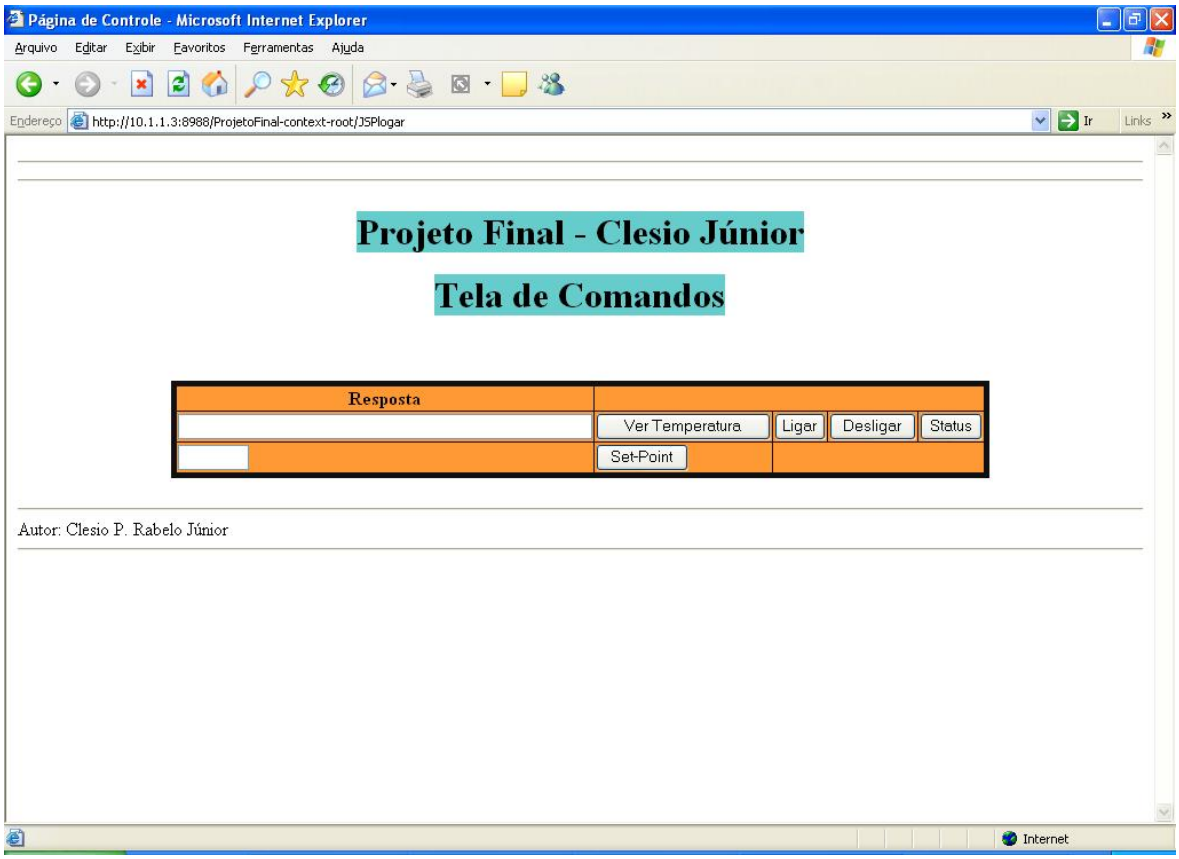


Figura 3. 14 - Tela de Comandos

CAPITULO 4 – SOFTWARE DO SERVIDOR

4.1 – Introdução

Para a comunicação da aplicação Midlet e da página Web com o microcontrolador foram criadas algumas aplicações (classes) utilizando a linguagem Java, a principal classe criada é chamada de Servlet. Também foi utilizada a API Java Communications para fornecer a comunicação pela porta serial do PC. A IDE (Integrated Development Environment. Um ambiente integrado para desenvolvimento de software) Oracle JDeveloper 10g foi utilizada para o desenvolvimento das aplicações Java, pois é de fácil utilização e possui versão grátis no site da oracle (http://www.oracle.com).

4.2 – A classe Servlet

A classe Servlet pode ser responsável por receber dados (parâmetros) vindos de um formulário HTML, ou de uma aplicação Midlet (J2ME) e enviar um comando ao microcontrolador, por exemplo. É um tipo de aplicação que roda do lado do servidor. As Servlet's necessitam de um servidor para que possam receber e transmitir requisições HTTP, a própria IDE *Oracle JDeveloper 10g* pode emular um servidor Web, sendo assim desnecessário a instalação e configuração de um servidor Web como o "*Apache TomCat*".

Ao "*colocar no ar*" o Servidor Web utilizando a IDE *Oracle JDeveloper 10g* , este poderá ser acessado pelo seu endereço IP da seguinte forma: "http://10.1.1.3:8988/ProjetoFinal-context-root/nomeServlet", onde "http" é o protocolo utilizado, "10.1.1.3" o endereço IP do Servidor, 8988 é a porta utilizada para a comunicação (diferente do padrão 80), cliente servidor, "*ProjetoFinal-*

context-root" é o "J2EE Web Context Root" configurado na IDE e "nomeServlet" é o nome que foi dado à classe Servlet no arquivo "web.xml", arquivo responsável pelo mapeamento da Servlet.

Toda Servlet, utilizando o recurso de herança fornecido pela programação orientada a objetos, será um tipo de HttpServlet, classe importada do pacote "javax.servlet.http.*", sendo necessário à implementação de pelo menos dois métodos: "doGet(HttpServletRequest request, HttpServletResponse response)" ou "doPost(HttpServletRequest request, HttpServletResponse response)". O método escolhido será o doPost(), responsável por receber os dados de formulários advindos do cliente Web (formulário) ou Midlet que enviam dados usando o método POST (método de envio utilizado). Este método possui dois parâmetros: "HttpServletRequest" para realizar o tratamento dos dados de entrada da Servlet, os dados recebidos, e "HttpServletResponse" para o tratamento dos dados de saída da Servlet, os dados que será retornados pela Servlet.

4.3 – Enviando e recebendo dados ao Microcontrolador

A aplicação cliente se comunicará com a Servlet enviando comandos em forma de números e a Servlet enviará "String's" como respostas. Por exemplo, para a aplicação cliente J2ME enviar o valor de Set-Point (Entrada) para o sistema de controle (Microcontrolador) esta deverá enviar o seguinte para a Servlet: "comando/valorSetPoint" ou "7/25", onde, o número do comando requisitado '7' separado pelo caractere '/' em seguida o valor do Set-Point '25'. Assim, a Servlet enviará o Set-Point ao microcontrolador e receberá o valor da temperatura ambiente do local monitorado, repassando via http o valor lido para a Midlet.

Para enviar o valor de Set-Point ao microcontrolador a Servlet terá que converter o dado enviado pela Midlet em um número que a lógica de controle entenda. Para isso, foi criada uma classe responsável por fornecer diferentes métodos utilizados em muitas partes do sistema, chamada de “Metodo.java”, onde pode ser encontrada a implementação dos métodos “String retornaSemPonto(String comPonto)” e “String retornaTemperatura(String strValor)” .

Os métodos citados “retornaSemPonto()” e “retornaTemperatura()”, têm uma forte ligação ao tipo de conversor A/D que foi utilizado (resolução), o primeiro “retornaSemPonto()” recebe como parâmetro um valor de Set-Point fornecido pelo cliente e converte-o utilizando a resolução do conversor A/D, em um número que é entendido pela lógica de controle, da seguinte forma: a resolução do conversor A/D (8 bits) é de (5/2^8) ou “0.01953125”, caso o parâmetro seja um valor de 25, representando a temperatura de 25ºC por exemplo, ao dividir pela resolução*10, isto é 0.1953125, chegaremos no número que a lógica de controle entenderá “128”.

Para que o usuário possa entender a temperatura envidada pelo microcontrolador o processo inverso terá que ser feito, para isso utiliza-se o método retornaTemperatura(), este recebe como parâmetro o valor de temperatura enviado pelo microcontrolador e o converte em um número que possa ser interpretado pelo o usuário, por exemplo, ao receber o valor de 128 (25ºC) faz, 128*[resolução(5/2^8)]*10, para obter o valor de 25ºC neste caso.

Toda a comunicação feita com o microcontrolador será por meio da interface serial do PC servidor, para isso utilizou-se a API Java Communications. Esta API define classes e métodos que possibilitam a comunicação por intermédio da porta serial com o microcontrolador.

4.4 – A API Java Communications

Diversos pacotes com classes e interfaces são fornecidas gratuitamente pela Sun para o desenvolvimento de diversos tipos de sistemas. Para a comunicação serial e ou paralela com outro PC ou microcontrolador, a Sun oferece a API Java Communications no endereço "http://java.sun.com/products/javacomm/index.jsp". Ao fazer o download do pacote Java Communications este fornecerá o arquivo "Win32Com.dll" que deve ser copiado para o diretório da JSDK utilizada pelo sistema, além dos arquivos "comm.jar" e "javax.comm.properties" que devem ser copiados para a pasta "BIN\LIB" do diretório da JSDK utilizada pelo sistema.

Para o projeto foi criado um pacote que fornecerá métodos para serem utilizados em qualquer parte do sistema, relacionado com a comunicação serial. Este pacote é composto de três classes que utilizam o recurso de herança oferecido pela programação orientada a objetos. Fazendo isso, tende-se a facilitar e simplificar o uso das classes responsáveis pela comunicação serial com o microcontrolador.

4.5 – O pacote responsável pela comunicação Serial

O diagrama de classe pode ser visto na **Figura 4.1**. A classe "PortaSerial.java" será utilizada para representar a interface serial com atributos como: nome da porta utilizada (COM1 ou COM2), a porta utilizada, id da porta utilizada, parâmetros da porta, fluxos de entrada e saída utilizada pela porta, dentre outros. Dessa forma, qualquer informação sobre a porta pode ser obtida utilizando-se esta classe.

Figura 4. 1 - Ligação das classes do pacote

A classe “PortaConfig.java” é a mais importante das classes deste pacote, estende a classe “PortaSerial.java” e é responsável por fornecer a conexão, o dado recebido e fechar a conexão com a porta serial. Esta classe implementa o método “public abstract void serialEvent(SerialPortEvent spe)”, da Interface “SerialPortEventListener.java” que vem no pacote Java Communications. Este método é chamado toda vez que o microcontrolador enviar algo ao PC servidor.

Para a utilização do pacote e de outros métodos necessários para a comunicação serial com o microcontrolador, foi criada a classe “PortaExecute.java” que estende a classe “PortaConfig.java”. Esta classe oferece métodos para enviar dados para a serial e para atribuir os parâmetros (velocidade, nome da porta utilizada, dentre outros) da porta serial. Ela será instanciada em qualquer parte do sistema e representará a porta serial, pois herda atributos e métodos da classe “PortaSerial.java”.

No próximo capítulo será abordado os conceitos referentes ao banco de dados e o seu software, que foi utilizado no projeto.

CAPITULO 5 – BANCO DE DADOS E SEU SOFTWARE

5.1 – Introdução

Neste capítulo é abordado o banco de dados utilizado e sua integração com o software em linguagem Java desenvolvido. Dessa forma, foi utilizado o banco de dados MySQL, o Drive “mysql-connector-odbc-3.51.12” para conectividade ODBC, o “MySQL Administrator” para a administração do banco e o “MySQL Query Browser” para criação e edição das tabelas e suas propriedades, todos gratuitos e que podem ser baixados no site “http://dev.mysql.com/downloads/” . O software em Java conta com a API JDBC que realiza a comunicação com uma fonte de dados ODBC[1] (Open Database Connectivity) e que já vem como pacote padrão do JDK da IDE Oracle JDeveloper 10 g, que é a IDE utilizada para o desenvolvimento do software Java.

O software da forma que foi desenvolvido possibilita a mudança de banco de dados para Oracle ou Access por exemplo. Sendo necessária a troca do drive para conectividade JDBC (Java Database Connectivity) ou ODBC instalado e configurado no sistema operacional do computador Servidor.

5.2 – O banco de dados

Para a utilização de um banco de dados e sua manipulação por meio de uma aplicação Java deve-se primeiramente criar o banco de dados.

Após a instalação e configuração do SGBD MySQL, foi criado o “Data Base” chamado de “ProjetoFinal”, com três tabelas chamadas de “usuario.frm”,

[1] ODBC, Open Data Base Connectivity, é uma tecnologia padrão de programação para o acesso a banco de dados por meio de uma biblioteca de funções pré-definida, criada pelo SQL Access Group. Basicamente, ODBC oferece uma interface padronizada de funções, uma API, ao programador, suportada por meio de um middleware apropriado. ODBC é independente de linguagem e baseado nas especificações de 'Call Level Interface do SQL. ODBC inspirou JDBC.

“acesso.frm” e “aparelho.frm” utilizando o software “MySQL Query Browser”. As tabelas “usuario" e “acesso” foram criadas para que fosse possível guardar dados dos usuários que podem acessar o sistema. Com a tabela “acesso” pode-se contar o número de acessos do usuário no sistema. As tabelas “acesso” e “usuário” estão relacionadas conforme tipo de relacionamento chamado de ”um para muitos”, pois um usuário poderá fazer muitos acessos. A tabela “aparelho” será útil para que se guardem dados da temperatura e a data atual por exemplo. Na **Figura 5.1** são mostradas as três tabelas e seus respectivos campos.

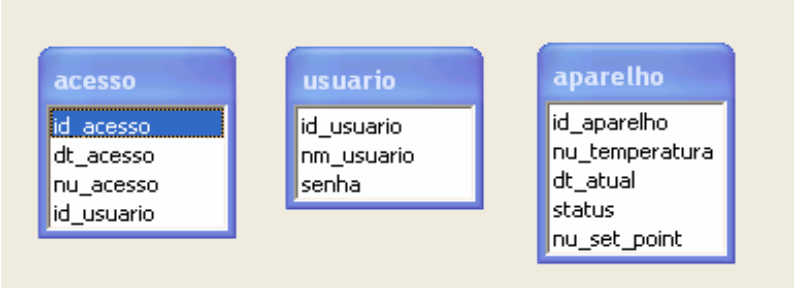


Figura 5. 1 - Tabelas criadas e seus campos

Para a utilização do sistema de banco de dados integrado com o sistema Java é necessário que se configure uma fonte de dados ODBC por meio do sistema operacional. Para o projeto foi utilizado o “Windows XP Profissional”. Essa fonte de dados será usada para estabelecer uma ponte de comunicação entre o banco de dados, no caso, MySQL e o Java.

A configuração da fonte de dados pode ser feita em: Painel de Controle → Ferramentas Administrativas → Fontes de dados (ODBC). Na **Figura 5.2** podem ser observadas as fontes de dados disponíveis no sistema. Para adicionar uma nova, basta clicar no botão “Adicionar” e assim aparecerá uma nova janela como na **Figura 5.3**, onde será apresentado vários tipos de Drivers instalados no sistema operacional.

Com o Drive “mysql-connector-odbc-3.51”, já instalado, basta selecionar a opção de Drive “MySQL ODBC 3.51 Driver” e clicar em concluir. Assim será mostrado a janela para configuração do Data Source criado anteriormente, como mostrado na **Figura 5.4**.

Nesta janela o usuário terá a opção de dar um nome ao “Data Source” a ser criado, dizer quem será o PC servidor, definindo o IP do servidor na caixa “Server”, caso seja o computador local, basta escrever “localhost” no campo e definir as opções de usuário, senha e Data Base a ser utilizado.

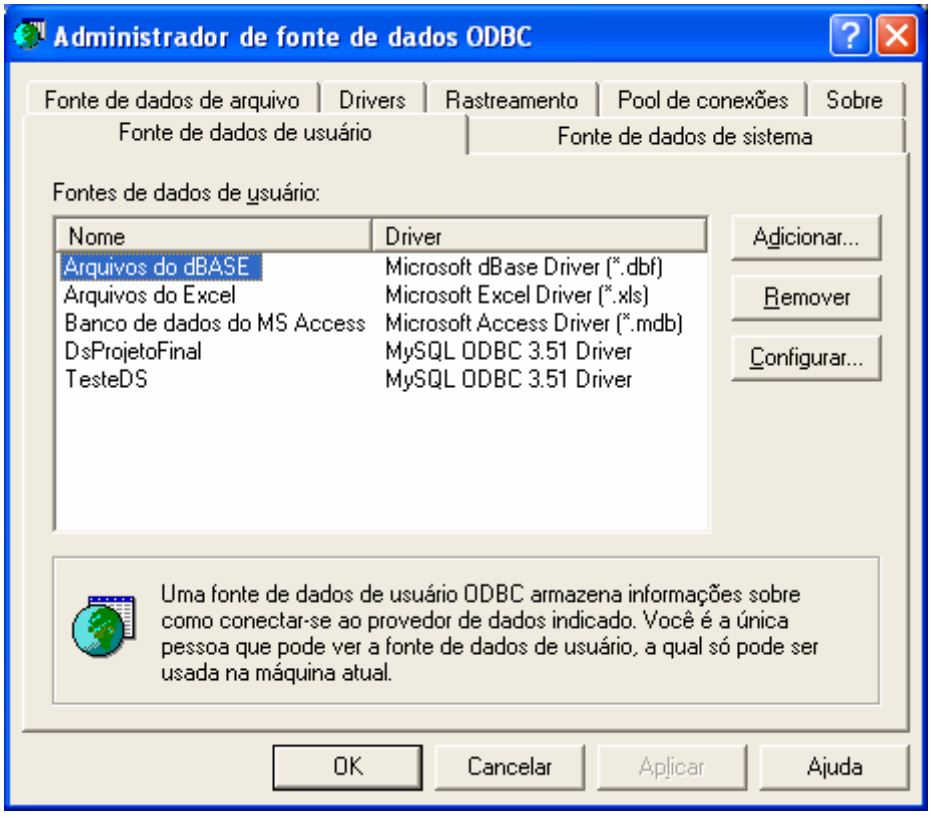


Figura 5. 2 - Fontes de dados disponíveis no sistema operacional

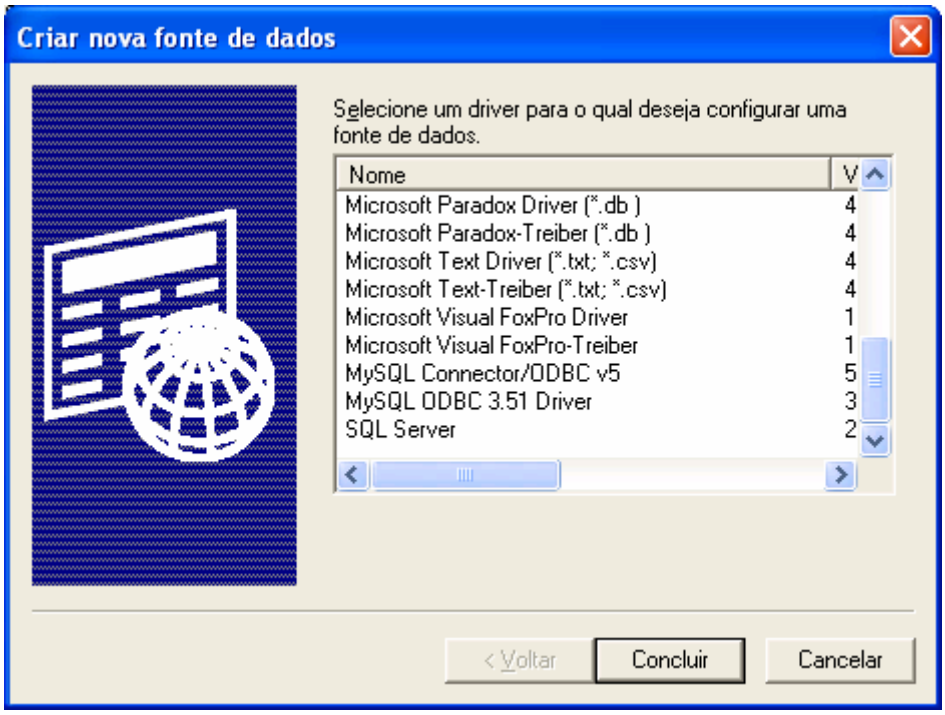


Figura 5. 3 - Drivers instalados no sistema operacional

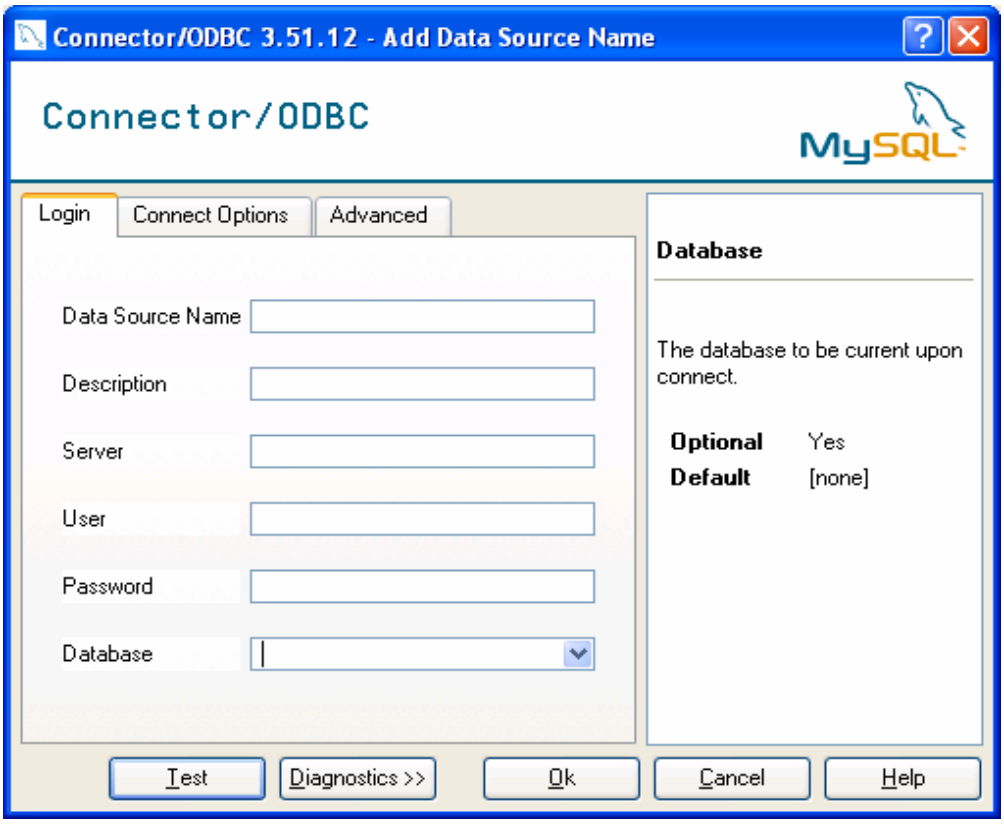


Figura 5. 4 - Criação do Data Source

5.3 – Aplicação Java para manipulação do Banco de Dados

Foi criado um pacote com três classes que poderá ser utilizado pela Servlet, pela aplicação (classe) "PegaDados.java", responsável por coletar os dados de temperatura em um tempo predeterminado, ou por outra parte do sistema.

As classes foram divididas em "Dados.java", "ConectaBD.java" e "DadosMySql.java". A classe "DadosMySql" "estende" a classe "ConectaBD", dessa forma "DadosMySql" se torna um tipo de "ConectaBD" e pode usufruir dos seus métodos por exemplo.

A classe "Dados" é utilizada para representar as tabelas criadas, pois terá atributos com o mesmo nome dos campos das três tabelas. Este tipo de classe se torna muito útil para facilitar a organização e manutenção do código fonte. Os Atributos nesta classe, serão acessados por meio dos métodos "get()" e "set()", sendo "set" para atribuir algum valor e "get" para retornar o valor do atributo selecionado.

A classe "ConectaBD" irá fornecer dois métodos, um que fornece a conexão ao bando de dados e o outro responsável por fechar a conexão ao banco de dados. Para realizar a conexão, é necessário carregar o drive que fará a comunicação com a fonte de dados e criará a conexão, para isso foi criado um método "Connection getConexao(Connection con)" que recebe como parâmetro uma conexão nula é retorna uma conexão estabelecida. O método pode ser visto no **Trecho de código 5.1**.

Para a conexão com outro banco de dados e necessário que se conheça os parâmetros como drive ("jdbc:odbc:NomeDoDataSource) nome de usuário e senha do banco, que serão utilizados conforme Trecho de código 5.1 (drive,user,senhaBanco).

```
public Connection getConexao(Connection con) throws SQLException
{
  if (con == null)
  {
    try
    {
      // Carrega o Drive
      Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
      // Cria a conexão chamada con
      con = DriverManager.getConnection(drive,user,senhaBanco);
    }
    catch(ClassNotFoundException ex)
    {
      // Retorna exceção para aplicação superior
      throw new SQLException("Erro, drive não encontrado! " + ex );
    }
    catch (SQLException e)
    {
      // Retorna exceção para aplicação superior
      throw new SQLException("Erro na SQL Conexao! " + e );
    }
  }
  // Retorna conexão estabelecida
  return con;
}
```

Trecho de Código 5. 1 - Método que fornece a conexão ao banco de dados

A classe “DadosMySql” será a classe utilizada (instanciada) na Servlet ou na classe “PegaDados” que fornecerá vários métodos para a manipulação do banco de dados. Os métodos são: “void contaEntradas(Dados dados)” , “void guardaTemperatura(Dados dados, Connection conn)” e “Dados logar(Dados dados)”.

O método “contaEntradas()” será utilizado para marcar o número de acesso do usuário. O método “logar()” será o método responsável pelo login do usuário. O método “quardaTemperatura()” é o método utilizado para guardar a temperatura lida em um determinado tempo.

5.4 – Aplicação responsável por coletar e guardar a temperatura

A aplicação responsável por coletar a temperatura em um tempo pré-definido terá um método, e fará a junção de vários métodos e classes utilizadas no sistema. O fluxograma mostrado na **Figura 5.5** explica este método.

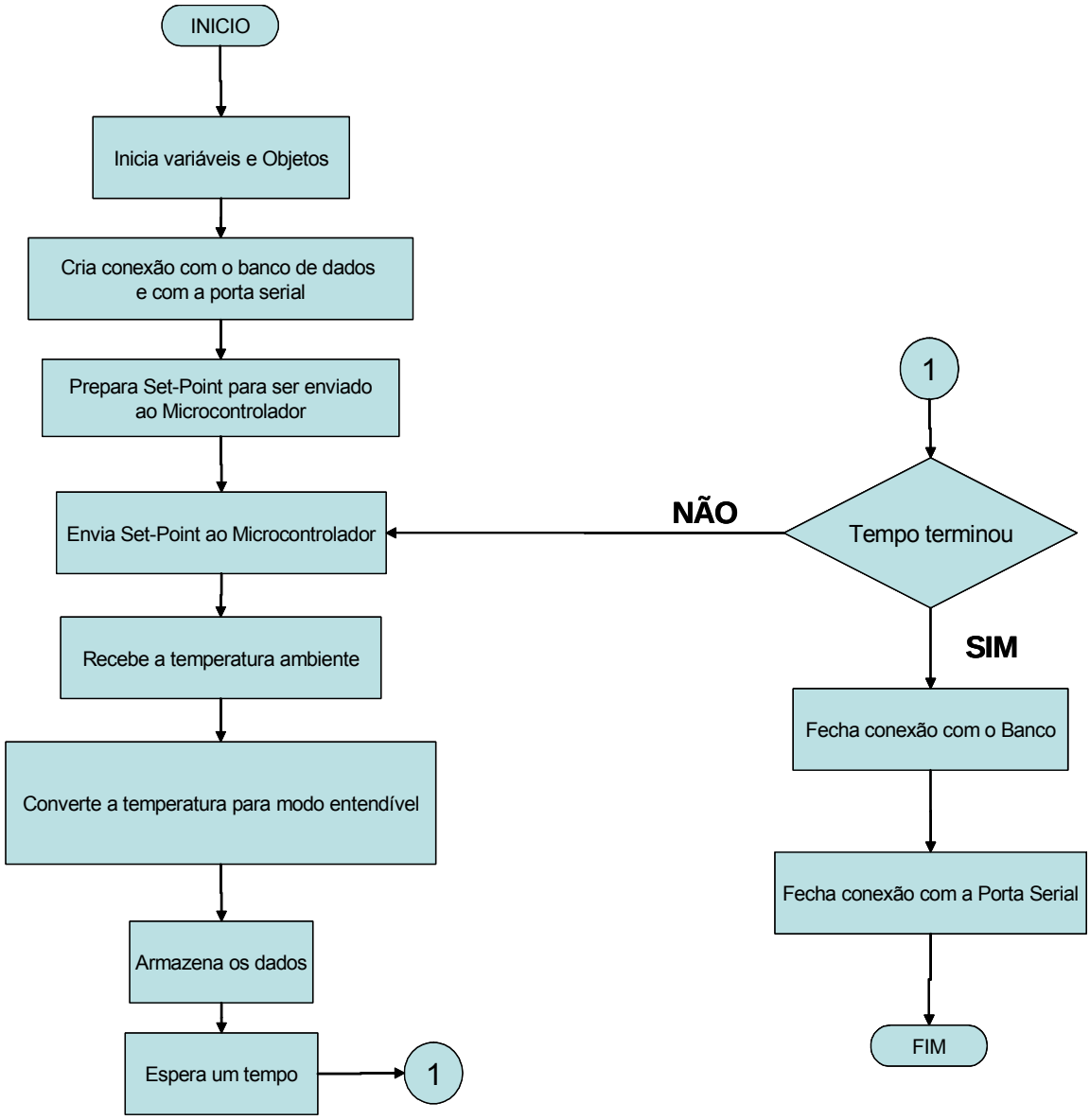


Figura 5. 5 - Fluxograma do método guardaDados()

Como apresentado no fluxograma, representado pela **Figura 5.5**, será iniciada uma conexão ao banco de dados e uma conexão com a porta serial. O método recebe como parâmetros o valor de set-point, o tempo total de espera para cada pesquisa e o número de vezes que o loop será efetuado. Caso o tempo escolhido seja de um minuto e o loop for executado durante sessenta vezes, serão feitas 60 pesquisas ao microcontrolador de minuto em minuto, isto é, será

feita uma amostra do sinal de temperatura lido do sensor. Por fim, com os dados guardados no banco pode-se utilizar o programa Excel para se criar um gráfico da temperatura coletada versus o tempo ou da temperatura ambiente e set-point versus o tempo.

CAPITULO 6 - SOFTWARE PARA MICROCONTROLADOR

6.1 – Introdução

O software para o microcontrolador será o responsável pelo controle do tipo liga e desliga do aparelho resfriador ou aquecedor. Este foi escrito em linguagem C e foi desenvolvido utilizando-se a IDE "*Jens File Editor*" (free em http://www.microcontroladores.com.br) que utiliza o compilador SDCC (Small Device C Compiler) para traduzir o código criado em C para uma linguagem de montagem suportada pelo AT89S8252 da "*Atmel*", que é o microcontrolador utilizado no projeto.

6.2 – Detalhes do software

São responsabilidades do software:

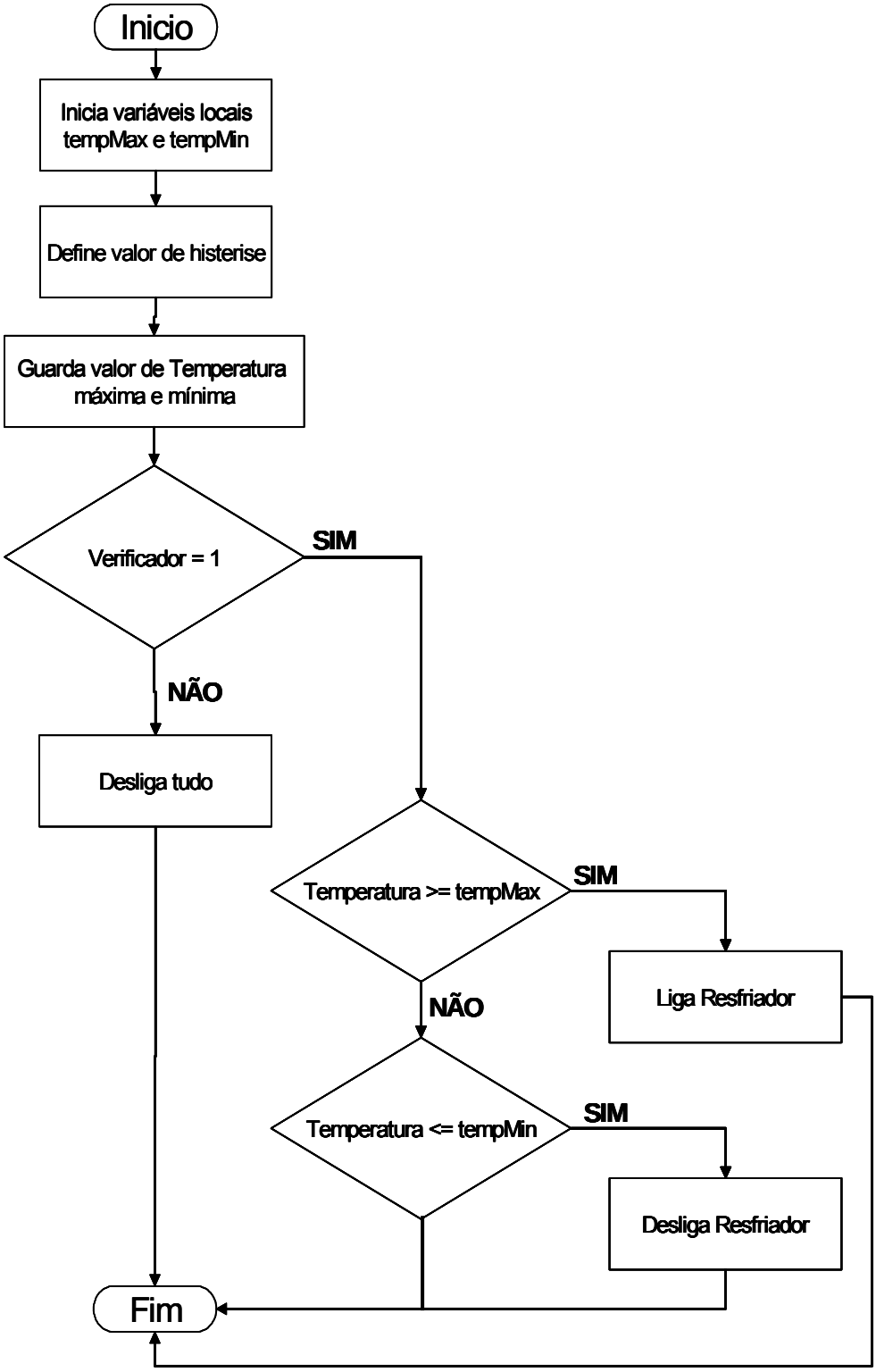
1. Estabelecer comunicação serial com o computador "Servidor Web";
2. Estabelecer comunicação paralela em 8 bits com o conversor A/D para obter dados digitalizados.
3. Realizar o controle do tipo ON/OFF do aparelho resfriador ou aquecedor. Enviando á interface de potência, nível lógico 1 para ligar e 0 para desligar o elemento resfriador ou aquecedor.

Estas responsabilidades e outros requisitos necessários ao software foram implementados em funções separadas, proporcionando uma grande modularidade ao código fonte. Como pode ser verificado na função principal main no **trecho de código 6.1**.

O código fonte da função de controle pode ser explicado pelo fluxograma apresentado na **Figura 6.1**. A lógica com um pequena mudança, pode ser usada tanto para o elemento resfriador como para o elemento aquecedor.

```
void main()
{
    // Configura comunicação serial
    confSerial();
    // Ativa todas as interrupções
    EA=1;
    // Habilita interrupção serial
    ES=1;
    // Coloca RD, CS e WR em zero
    P0=0;
    // Padrão com 25ºC
    setPoint=128;
    // Liga aparelho, se == 0 desliga aparelho
    verificador=1;
    while(1)
    {
        // Recebe valor analogico e guarda digitalizado
        lerADC();
        // Controla a temperatura
        controlaTemperatura();
    }
}
```

Trecho de Código 6. 1 - Função principal do software para o Microcontrolador



TempMax (temperatura máxima) = set-point+atraso;

TempMin (temperatura minima) = set-point – atraso;

Figura 6. 1 - Fluxograma da função controlaTemperatrura()

O fluxograma apresentado na **Figura 6.2** explica a função lerADC(), esta é responsável por guardar o dado digitalizado em variável global para ser utilizado pela função controlaTemperatura().

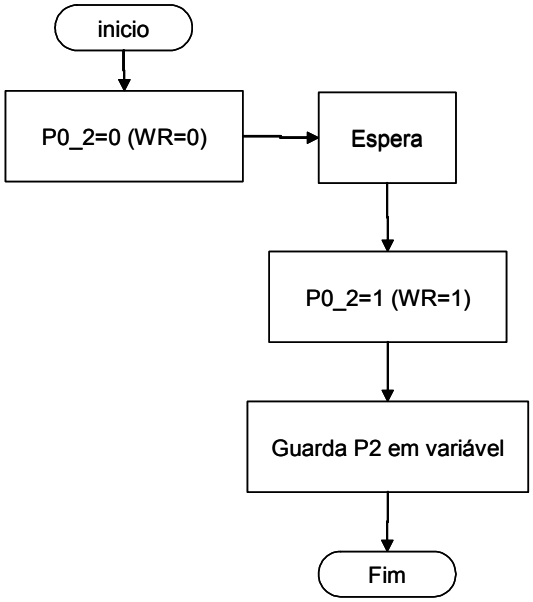


Figura 6. 2 - Fluxograma da função lerADC()

A função para configurar a porta serial apresentada no **trecho de código 6.2** é a responsável pela configuração da comunicação serial.

```
1. void confSerial()
2. {
3.   SCON=0x50;
4.   TCLK=1;
5.   RCLK=1;
6.   C_T2=0;
7.   RCAP2H=0xFF;
8.   RCAP2L=0xFD;
9.   TR2=1;
10. }
```

Trecho de Código 6. 2 - Função que configura a comunicação serial

Para que haja uma comunicação entre o computador e o microcontrolador de forma que não sejam perdidas informações entre o receptor e o transmissor, deve ser enviado e ou transmitidos os bits seguindo-se uma seqüência pré-

estabelecida. Dessa forma a função confSerial() será utilizada para criar essa seqüência.

Na linha 3 foi configurado o registrador **SCON** que é o registrador de controle do periférico serial para os modos síncrono e assíncrono. Formado por 8 bits, acessados bit a bit ou por byte. Na **Tabela 6.1** pode-se ver o registrador e seus bits com detalhes.

Tabela 6. 1 - Registrador SCON

Nome	SM0	SM1	SM2	REN	TB8	RB8	TI	RI
Posição	9Fh	9Eh	9Dh	9Ch	9Bh	9Ah	99h	98h

Configurado o byte com o valor 50 (em hexadecimal) teremos os seguinte valor em binário, **Tabela 6.2**.

Tabela 6. 2 - Configuração do SCON

Nome	SM0	SM1	SM2	REN	TB8	RB8	TI	RI
Valor	0	1	0	1	0	0	0	0

Dessa forma observando a **Tabela 6.3** pode-se verificar que a configuração escolhida é uma comunicação assíncrona com baute-rate gerado pelo timer 2 e com tamanho de 8 bits, *Modo 1*. O baute-rate é a freqüência com que os dados são armazenados ou enviados ou a taxa de transmissão é recepção.

Tabela 6. 3 - Modo de comunicação e detalhes

Modo	**SM0**	**SM1**	**Comunicação**	**Tamanho**	**Baute-rate**
0	0	0	Síncrona	8 bits	fclock/12
1	0	1	Assíncrona	8 bits	Gerado pelo Timer 1 / 2
2	1	0	Assíncrona	9 bits	fclock/32 ou fclock/64
3	1	1	Assíncrona	9 bits	Gerado pelo Timer 1 / 2

Os bits TCLK, RLCK, C_T2 e TR2 fazem parte do registrador **T2CON** que é destinado ao controle do timer/counter 2. Uma vantagem do microcontrolador AT89S8252 e a inclusão desse timer/counter para geração de baute-rate.

Com o bit TCLK em nível lógico 1 será gerado pelo timer 2 um baud-rate para recepção de dados. Com o bit RLCK em nível lógico 1 será gerado pelo timer 2 um baud-rate para transmissão de dados, com esses bits sendo colocados

em nível lógico 1 o timer 2 funcionará automaticamente em recarga automática de 16 bits sendo exclusivamente gerador de baud-rate. Com o bit C_T2 em nível lógico 0 será utilizada a função timer do timer/counter 2 e para habilitar o funcionamento do timer 2 deve se colocar o bit TR2 em nível lógico 1.

Os registradores RCAP2L (bits menos significativos) e RCAP2H (bits mais significativos) são registradores de recarga atribuídos por software. Responsáveis pela recarga dos registradores TL2 e TH2, que são os registradores de contagem e incrementado.

Conforme **Tabela 6.4** retirada do livro Microcontrolador 8051 Família AT89S8252 Atmel de Denys E. C. Nicolosi, pode-se ver vários valores para ser atribuído ao registrador RCAP2.

Tabela 6. 4 - Valores para RCAP2

Microcontrolador com Timer 2 com reload de 16 bits e cristal de 11,0592 MHz				
Taxas (bps)	**RCAP2**	**Prático**	**Taxa Obtida (bps)**	**%Erro**
300	64384	64384	300	0
600	64960	64960	600	0
1200	65248	65248	1200	0
2400	65392	65392	2400	0
4800	65464	65464	4800	0
9600	65500	65500	9600	0
14400	65512	65512	14400	0
19200	65518	65518	19200	0
28800	65524	65524	28800	0
38400	65527	65527	38400	0
57600	65530	65530	57600	0
115200	65533	65533	115200	0
230400	65534,5	65534	172800	25
460800	65535,25	65535	345600	25

Para encontrar o valor de **RCAP2** devemos usar as seguintes formulas. Segundo folha de dados fornecido pela Atmel.

$$Fot = Nbps * 16$$ **Equação 6. 1**

$$Nc = \frac{FcM}{(Fot * 2)}$$ **Equação 6. 2**

$$RCAP2 = 2^{16} - Nc$$ **Equação 6. 3**

$$BR = \frac{FcM}{32 \times [65536 - (RCAP2H, RCAP2L)]}$$ **Equação 6. 4**

onde:

$Fot \rightarrow$ freqüência de overflow do timer

$Nc \rightarrow$ número de contagens

$FcM \rightarrow$ freqüência do clock do microcontrolador

$RCAP2 = (RCAP2H, RCAP2L)$

Para produzir uma taxa de **115200** bits por segundo deve-se seguir os seguintes passos para obter o valor de RCAP2.

$$Fot = 115200 * 16 = 1.843200Hz$$ **Equação 6. 5**

$$Nc = \frac{11059200Hz}{3686400Hz} = 3$$ **Equação 6. 6**

$$RCAP2 = 65536 - 3 = 65533$$ **Equação 6. 7**

$$\%Erro = 100\% - \left(\frac{TaxaObtida * 100}{Taxa} \right)\%$$ **Equação 6. 8**

$$\%Erro = 100\% - \left(\frac{172800 * 100}{230400} \right)\% = 25\%$$ **Equação 6. 9**

CAPITULO 7 – CONSIDERAÇÕES FINAIS

7.1 – Dificuldades encontradas

A principal dificuldade encontrada para a realização do projeto foi a aquisição dos componentes elétricos e eletrônicos, como sensor de temperatura, conversor A/D, amplificador operacional, relé, dentre outros. Estes elementos não foram encontrados em lojas convencionais de elétrica ou eletrônica. Os produtos foram encontrados e adquiridos pelo site da loja virtual “Brasil Express” que pode ser acessada no endereço *http://www.brasilexpress.com.br*.

7.2 – Resultados obtidos

Todos os objetivos, estabelecidos inicialmente, foram atingidos:

1. Receber o valor de temperatura na tela do simulador de celular e na aplicação JSP;
2. Fornecer Set-Point como parâmetro de entrada do sistema de controle, pelo simulador de celular e aplicação JSP;
3. Coleta de dado referente à temperatura x tempo e armazenamento em banco de dados para criação de gráfico com o auxílio do Excel;
4. Verificação do sistema atuando automaticamente em máquina com alta ou baixa potência utilizando o relé e a contatora;
5. Verificação do controle automático da dinâmica do sistema térmico com auxilio de resistência elétrica (lâmpada de 12 volts ou de 220 volts 40 w) como aquecedora do ambiente próximo ao sensor de temperatura;

6. Verificação do controle automático da dinâmica do sistema térmico com auxilio de pequeno motor de tensão continua de 12 volts, agindo como ventilador no ambiente próximo ao sensor.

7.2.1 – Simulação do item 5

Para o processo de simulação foi utilizado um sistema de fácil acesso e baixo custo ao invés do aparelho de ar-condicionado, que é mais caro e complexo. Neste caso, a simulação não será feita utilizando a refrigeração do ambiente e sim o aquecimento do mesmo.

O sistema é formado por uma caixa de papelão com um elemento aquecedor (lâmpada) instalado em seu interior.

Inicialmente foi utilizado uma lâmpada de 12 volts com potência menor que 5 Wats e Set-Point estabelecido em 24 ºC, obteve-se o gráfico da **Figura 7.1**. A temperatura do dia estava em torno de 23 ºC.

Em segundo, utilizando a mesma configuração com o Set-Point de 26 Cº chegou-se ao gráfico da **Figura 7.2**. Analisando os dois primeiros gráficos, foi visto que ao subir o valor de Set-Point não se conseguiria chegar na temperatura desejada, pois a potência da lâmpada era pequena. Dessa forma, verificou-se que o controle da temperatura depende diretamente da potência do elemento aquecedor ou resfriador, no caso a potência da lâmpada.

Como solução, foi instalada uma lâmpada de 220 volts com 40 Wats de potência dentro da caixa e aumentou-se o Set-Point para 28 ºC, o resultado pode ser verificado na **Figura 7.3**.

Onde:

SP – Set-Point ;

TempMax – Temperatura Máxima;

TempMin – Temperatura Mínima.

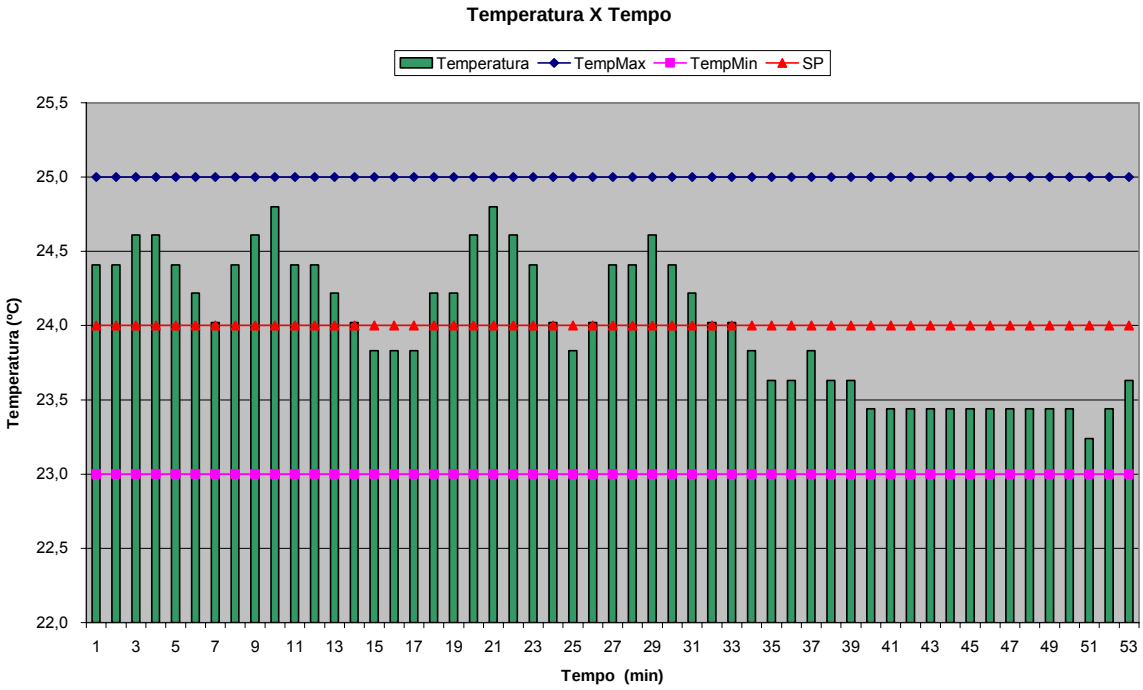


Figura 7. 1 - Set-Point de 24 °C

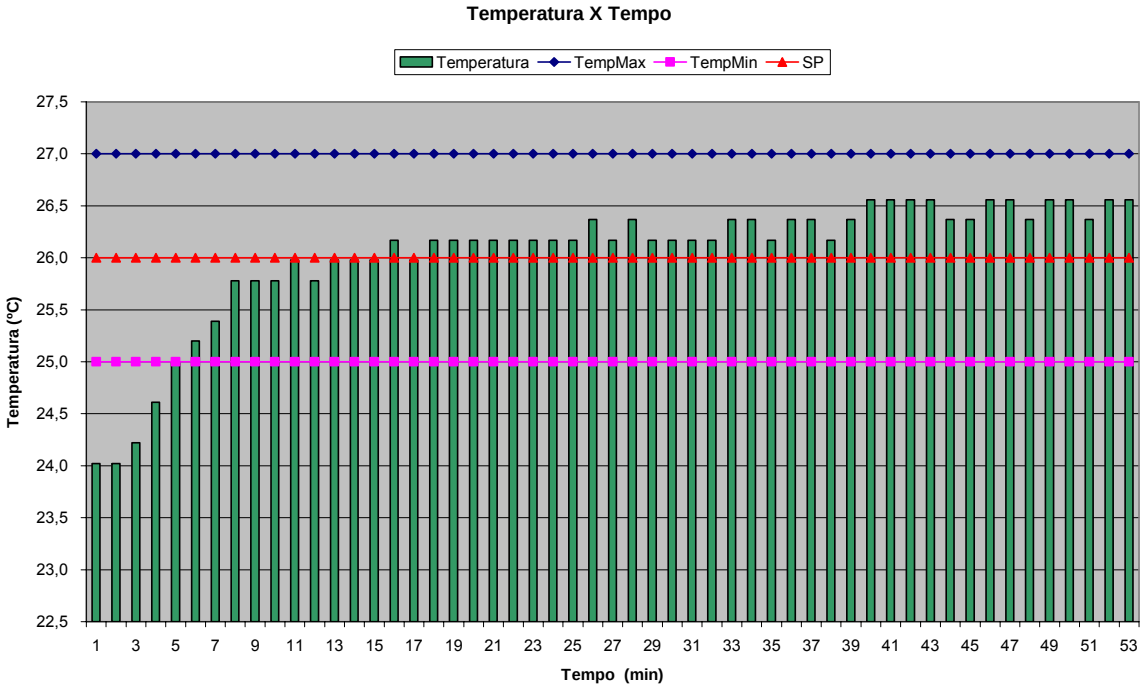


Figura 7. 2 - Set-Point de 26 °C

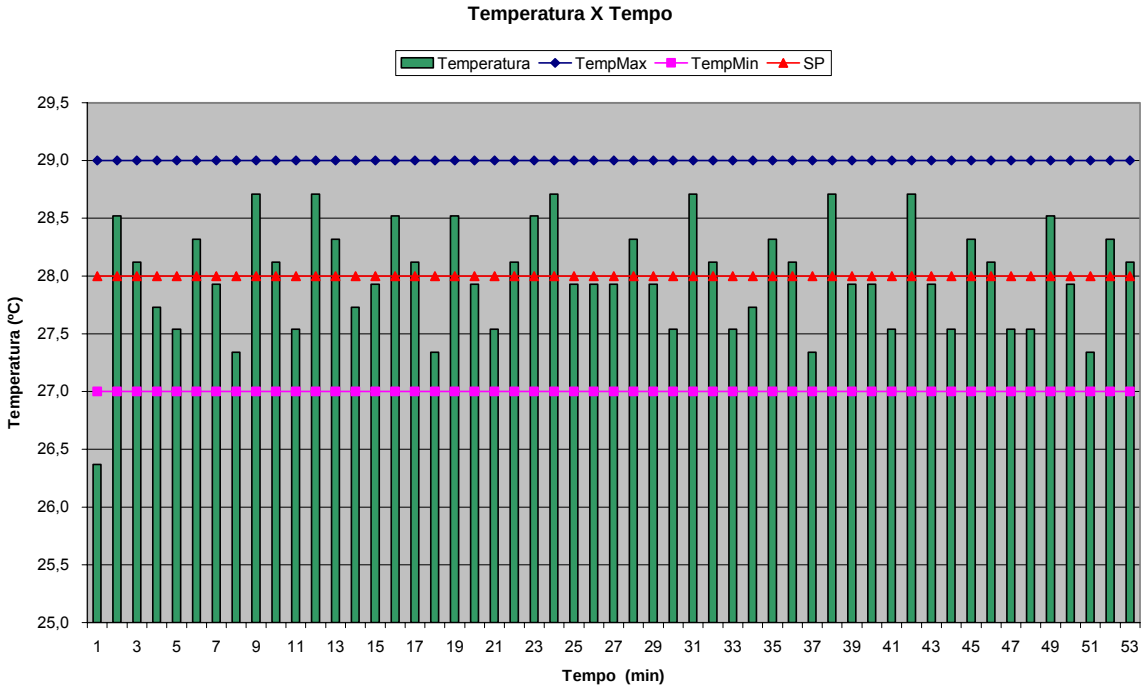


Figura 7. 3 - Set-Point de 28 °C

Observando-se os gráficos apresentados nas Figuras 7.1 à 7.3, pode-se observar que a partir da especificação de um valor de set-point, a temperatura do ambiente monitorado, tenderá a seguir valores de temperatura dentro da faixa estabelecida pelas temperaturas máxima e mínina.

7.3 – Conclusões

Com o sistema automático de controle de temperatura desenvolvido observou-se que com adequações específicas, o mesmo poderá ser utilizado em um outro tipo de automação. Controle de iluminação e de umidade, automação de prédios inteligentes e automação de carros, são exemplos de outros tipos de sistemas controláveis por este projeto.

A outra faceta do projeto, que é armazenar valores de temperatura durante um certo tempo automaticamente, mostrou-se muito útil para a análise da carga térmica em diferentes locais e horários do dia. Com essa função, prever-se várias

questões acerca do ambiente que será refrigerado ou aquecido, ajudando aos profissionais de mecânica de refrigeração, por exemplo, na verficação de levantamento de carga térmica do ambiente.

Por fim, uma importante característica verificada nesse tipo de projeto é a implementação de hardwares e de softwares. Foram aplicados vários conhecimentos adquiridos durante o curso de Engenharia da Computação, juntamente com as experiências prévias adquiridas em estágios e aulas práticas em laboratórios.

7.4 – Sugestões para trabalhos futuros

A partir desse projeto podem-se criar novas frentes de pesquisas dirigidas tanto para a área de software como para a de automação. São eles:

1. Adicionar outros sensores e ampliar os elementos controlados.
2. Criar sistema de alarme de incêndio que enviará avisos para o usuário, alertando do problema ocorrido.
3. Agregar ao software do servidor a possibilidade de conexão bluetooth com um celular próximo.
4. Inserir sistema de controle PID para válvula de água gelada, utilizada em grandes sistemas de refrigeração que necessitam da água como fluido refrigerante.
5. Criar aplicação que gere dinamicamente um gráfico baseando-se nos dados guardados pelo sistema de coleta de temperatura, substituindo o programa Excel.

REFERÊNCIAS BIBLIOGRÁFICAS

[1] Bernal, Paulo Sergio Milano, Comunicações Móveis, Tecnologia e Aplicações, SP: Erica, 2002.

[2] Furgeri, Sérgio, Java 2: Ensino didático: Desenvolvendo e Implementando Aplicações, SP: Érica, 2002.

[3] Harold, Elliotte Rusty, Java I/O, O'Reilly, 1999.

[4] Haykin, Simon, Sinais e Sistemas, Porto Alegre: Bookman, 2001.

[5] J. L. Martins de Carvalho, Sistemas de Controle Automático, RJ: LTC, 2000

[6] Kurose, James F. , Redes de Computadores e a Internet: uma nova abordagem, 1ª ed. – SP: Addison Wesley, 2003.

[7] Muchow, John W., Core J2ME: Tecnologia & MIDP, SP: Pearson Makron Books, 2004.

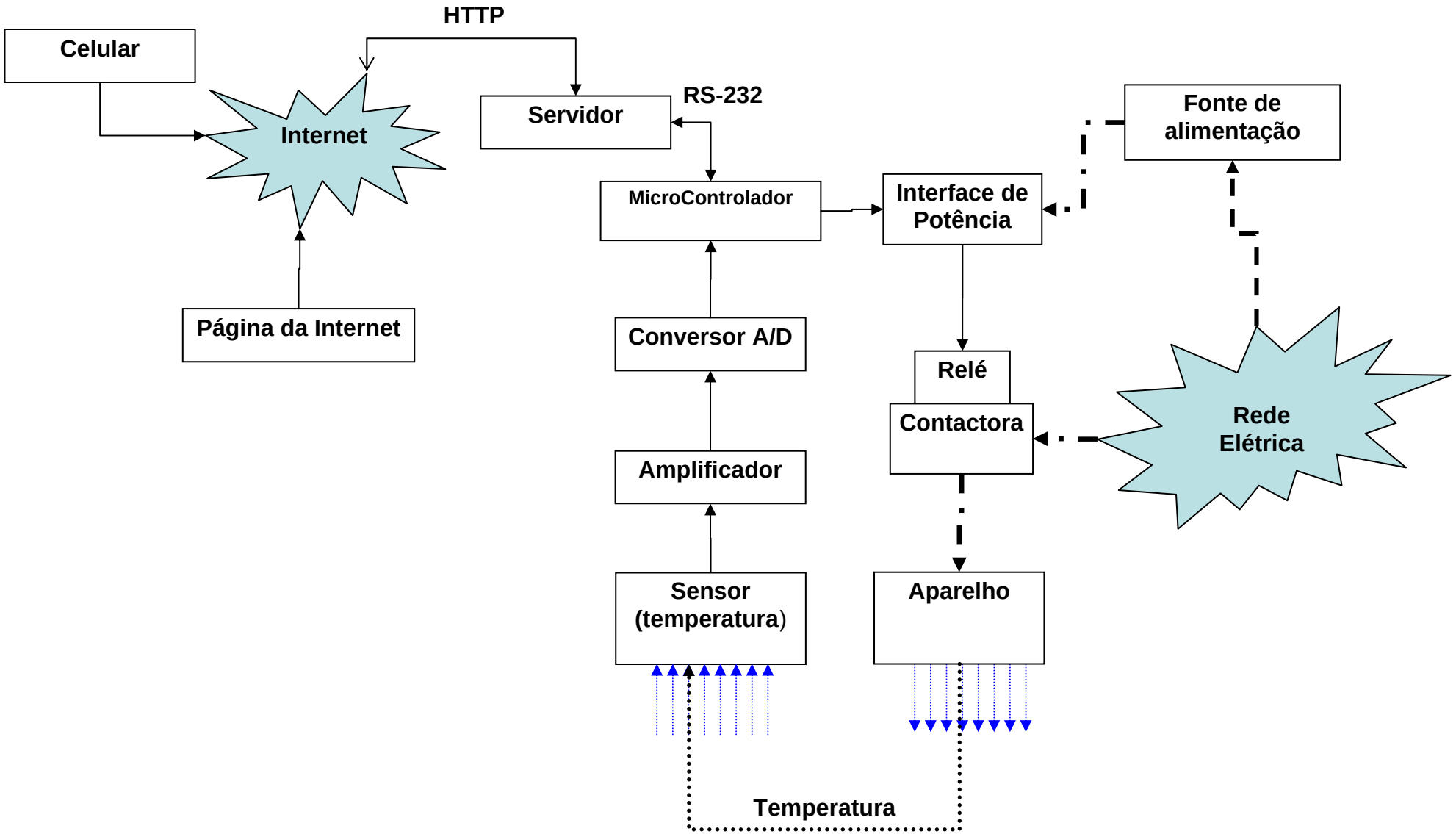
[8] Nicolosi, Denys Emílio Campio, Microcontrolador 8051 com linguagem C: prático e didático – família AT89S8252 Atmel, 1ª ed. SP: Érica, 2005.

[9] Nilsson W. James e Riedel A.. Susan, Circuitos Elétricos, 6ª ed. – RJ: LTC, 2003.

[10] Ogata, Katsutiko, Engenharia de Controle Moderno, 4ª ed.: Prentice Hall, 2003.

[11] Pazos, Fernando, Automação de Sistemas & Robótica, RJ: Axcel, 2002.

APÊNDICE A – DIAGRAMA EM BLOCOS DE TODO O SISTEMA



APÊNDICE B – CÓDIGO DA SERVLET (JAVA)

```java
/*
* Servlet que atende requisições HTTP
* Autor: Clesio Pinto Rabelo Júnior
* Junho de 2006
*/
package projetoFinal.controle;

import java.io.IOException;
import java.io.PrintWriter;

import java.sql.SQLException;

import javax.servlet.ServletConfig;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.HttpSession;

import projetoFinal.bancoDados.Dados;
import projetoFinal.bancoDados.DadosMySql;

import projetoFinal.comunicacaoSerial.PortaExecute;

import projetoFinal.metodos.Metodo;

public class ControlePortaSerial extends HttpServlet
{
  /**
   * Inicia a servlet
   * @param config
   * @throws javax.servlet.ServletException
   */
  public void init(ServletConfig config) throws ServletException
  {
    super.init(config);
  }

  /**
   * Recebe solicitações GET
   * @param request
   * @param response
   * @throws javax.servlet.ServletException
   * @throws java.io.IOException
   */
  public void doGet(HttpServletRequest request, HttpServletResponse response) throws
  ServletException, IOException
  {
    // Chama o metodo doPost
    doPost(request, response);
  }

  /**
   * Recebe solicitações POST
   * @param request
   * @param response
   * @throws javax.servlet.ServletException
   * @throws java.io.IOException
   */
  public void doPost(HttpServletRequest request, HttpServletResponse response) throws
  ServletException, IOException
  {
    String jspResposta = null;

    // Recebe o nome mapeado no momento para a Servlet
    String nome = request.getServletPath();
    // Retira o "/" que vem antes do nome dado à Servlet
    nome = nome.substring(1,nome.length());

    try
    {
      if ( nome.equals("verTemperatura") )
      {
        jspResposta = verificaSecao(request);
        if(jspResposta.equals("LerTemperatura.jsp"))
          verTemperaturaJSP(request);
```

```
      request.getRequestDispatcher(jspResposta).forward(request,response);
    }
    else if ( nome.equals("verStatus") )
    {
      jspResposta = verificaSecao(request);
      if(jspResposta.equals("LerTemperatura.jsp"))
        verStatusJSP(request);
      request.getRequestDispatcher(jspResposta).forward(request,response);
    }
    else if ( nome.equals("ligaMotor") )
    {
      jspResposta = verificaSecao(request);
      if(jspResposta.equals("LerTemperatura.jsp"))
        ligaJSP(request);
      request.getRequestDispatcher(jspResposta).forward(request,response);
    }
    else if ( nome.equals("desligaMotor") )
    {
      jspResposta = verificaSecao(request);
      if(jspResposta.equals("LerTemperatura.jsp"))
        desligaJSP(request);
      request.getRequestDispatcher(jspResposta).forward(request,response);
    }
    else if ( nome.equals("setPoint") )
    {
      jspResposta = verificaSecao(request);
      if(jspResposta.equals("LerTemperatura.jsp"))
        setPointJSP(request);
      request.getRequestDispatcher(jspResposta).forward(request,response);
    }
    else if ( nome.equals("logar") )
    {
      processarLoginJ2ME(request,response);
    }
    else if ( nome.equals("JSPlogar") )
    {
      iniciarSecao(request);
      jspResposta = processarLoginJSP(request);
      request.getRequestDispatcher(jspResposta).forward(request,response);
    }
    else if ( nome.equals("respostaJ2ME") )
    {
      processarRespostaJ2ME(request,response);
    }
  }
  catch (IOException ex)
  {
    ex.printStackTrace();
  }
}
/**
 * Método para inciar uma seção
 * @param request
 * @throws javax.servlet.ServletException
 * @throws java.io.IOException
 */
public void iniciarSecao(HttpServletRequest request) throws ServletException, IOException
{
  Metodo.log("ControlePortaSerial","secao(HttpServletRequest request)","Inicio");

  String idSecao = null;
  HttpSession secao;

  secao = request.getSession(true);
  idSecao = secao.getId();

  secao.setAttribute("idSecao",idSecao);

  Metodo.log("ControlePortaSerial","secao(HttpServletRequest request)","Fim");
}

/**
 * Método para verificar se a seção é nova
 * @param request
 * @throws javax.servlet.ServletException
 * @throws java.io.IOException
 */
public String verificaSecao(HttpServletRequest request) throws ServletException,
IOException
{
  Metodo.log("ControlePortaSerial","verificaSecao(HttpServletRequest request)","Inicio");
```

```
  String idSecao = null;
  HttpSession secao;

  secao = request.getSession(true);
  idSecao = (String) secao.getAttribute("idSecao");

  if(Metodo.nEstarVazio(idSecao))
  {
    Metodo.log("ControlePortaSerial","verificaSecao(HttpServletRequest request)","Fim");
    return "LerTemperatura.jsp";
  }
  else
  {
    Metodo.log("ControlePortaSerial","verificaSecao(HttpServletRequest request)","Fim");
    return "Login.jsp";
  }
}
/**
 * Método que retorna a temperatura para a aplicação JSP
 * @param request
 * @throws javax.servlet.ServletException
 * @throws java.io.IOException
 */
public void verTemperaturaJSP(HttpServletRequest request) throws ServletException,
IOException
{
   Metodo.log("ControlePortaSerial","verTemperaturaJSP(HttpServletRequest
   request)","Inicio");

  PortaExecute portaExecute = new PortaExecute();
  String comando = null;
  String resposta = null;
  comando = "4";
  try
  {
    portaExecute = portaExecute.configuraConectaSerial(portaExecute);
    portaExecute.setStrDadoEnviado(comando);
    portaExecute.enviaDadoParaSerial(portaExecute.getStrDadoEnviado());
    Metodo.recebeDado(portaExecute);
    resposta = portaExecute.getStrDadoRecebido();
    if ( Metodo.nEstarVazio(resposta) )
      resposta = Metodo.retornaTemperatura(resposta) + " ºC";
    request.setAttribute("resposta",resposta);
  }
  catch (Exception ex)
  {
    ex.printStackTrace();
  }
  finally
  {
     portaExecute.FecharCom();
     Metodo.log("ControlePortaSerial","verTemperaturaJSP(HttpServletRequest
     request)","Fim");
  }
}
/**
 * Método que retorna o status para a aplicação JSP
 * @param request
 * @throws javax.servlet.ServletException
 * @throws java.io.IOException
 */
public void verStatusJSP(HttpServletRequest request) throws ServletException, IOException
{
  PortaExecute portaExecute = new PortaExecute();
  String comando = null;
  String status = null;
  String temperatura = null;
  String setPointL = null;
  String resposta = null;
  comando = "3";
  try
  {
    portaExecute = portaExecute.configuraConectaSerial(portaExecute);
    portaExecute.setStrDadoRecebido(null);
    portaExecute.setStrDadoEnviado(comando);
    portaExecute.enviaDadoParaSerial(portaExecute.getStrDadoEnviado());
    Metodo.recebeDado(portaExecute);
    status = portaExecute.getStrDadoRecebido();
    if ( Metodo.nEstarVazio(status) )
    {
```

```
      if(status.equals("1"))
        status = "Tudo Ligado";
      else if(status.equals("2"))
        status = "Refrigerando";
      else if(status.equals("3"))
        status = "Aquecendo";
      else if(status.equals("4"))
        status = "Tudo Desligado";
      else if(status.equals("5"))
        status = "Aquecedor Desligado";
      else if(status.equals("6"))
        status = "Aquecedor Ligado";
    }
    comando = "4";
    portaExecute.setStrDadoRecebido(null);
    portaExecute.setStrDadoEnviado(comando);
    portaExecute.enviaDadoParaSerial(portaExecute.getStrDadoEnviado());
    Metodo.recebeDado(portaExecute);
    temperatura = portaExecute.getStrDadoRecebido();
    if ( Metodo.nEstarVazio(temperatura) )
      temperatura = Metodo.retornaTemperatura(temperatura) + " ºC";
    comando = "5";
    portaExecute.setStrDadoRecebido(null);
    portaExecute.setStrDadoEnviado(comando);
    portaExecute.enviaDadoParaSerial(portaExecute.getStrDadoEnviado());
    Metodo.recebeDado(portaExecute);
    setPointL = portaExecute.getStrDadoRecebido();
    if ( Metodo.nEstarVazio(setPointL) )
    {
      setPointL = Metodo.retornaTemperatura(setPointL) + " ºC";
      resposta = status + " à " + temperatura + " com Set-Point de: " + setPointL;
      request.setAttribute("resposta",resposta);
    }
  }
  catch (Exception ex)
  {
    ex.printStackTrace();
  }
  finally
  {
    portaExecute.FecharCom();
  }
}
/**
 * Método que liga o aparelho da aplicação JSP
 * @param request
 * @throws javax.servlet.ServletException
 * @throws java.io.IOException
 */
public void ligaJSP(HttpServletRequest request) throws ServletException, IOException
{
  PortaExecute portaExecute = new PortaExecute();
  String comando = "";
  String resposta = null;
  comando = "2";
  try
  {
    portaExecute = portaExecute.configuraConectaSerial(portaExecute);
    portaExecute.setStrDadoEnviado(comando);
    portaExecute.enviaDadoParaSerial(portaExecute.getStrDadoEnviado());
    Metodo.recebeDado(portaExecute);
    resposta = portaExecute.getStrDadoRecebido();
    if ( Metodo.nEstarVazio(resposta) )
    {
      resposta = "Aparelho Ligado";
    }
    request.setAttribute("resposta",resposta);
  }
  catch (Exception ex)
  {
    ex.printStackTrace();
  }
  finally
  {
    portaExecute.FecharCom();
  }
}
/**
 * Método que desliga o aparelho da aplicação JSP
 * @param request
 * @throws javax.servlet.ServletException
```

```
 * @throws java.io.IOException
 */
public void desligaJSP(HttpServletRequest request) throws ServletException, IOException
{
  PortaExecute portaExecute = new PortaExecute();
  String comando = "";
  String resposta = null;
  comando = "1";
  try
  {
    portaExecute = portaExecute.configuraConectaSerial(portaExecute);
    portaExecute.setStrDadoEnviado(comando);
    portaExecute.enviaDadoParaSerial(portaExecute.getStrDadoEnviado());
    Metodo.recebeDado(portaExecute);
    resposta = portaExecute.getStrDadoRecebido();
    if ( Metodo.nEstarVazio(resposta) )
    {
      resposta = "Aparelho desligado";
    }
    request.setAttribute("resposta",resposta);
  }
  catch (Exception ex)
  {
    ex.printStackTrace();
  }
  finally
  {
    portaExecute.FecharCom();
  }
}
/**
 * Método para setar o valor de set-point pela aplicação JSP
 * @param request
 * @throws javax.servlet.ServletException
 * @throws java.io.IOException
 */
public void setPointJSP(HttpServletRequest request) throws ServletException, IOException
{
  PortaExecute portaExecute = new PortaExecute();
  String setPoint = null;
  String setPointT = null;
  String resposta = null;
  try
  {
    setPoint = request.getParameter("inSetPoint");
    portaExecute = portaExecute.configuraConectaSerial(portaExecute);
    setPointT = Metodo.retornaSemPonto(setPoint);
    portaExecute.setStrDadoEnviado(setPointT);
    portaExecute.enviaDadoParaSerial(portaExecute.getStrDadoEnviado());
    Metodo.recebeDado(portaExecute);
    resposta = portaExecute.getStrDadoRecebido();
    if(Metodo.nEstarVazio(resposta))
      resposta = Metodo.retornaTemperatura(resposta) + " ºC";
    request.setAttribute("setPoint",setPoint);
    request.setAttribute("resposta",resposta);
  }
  catch (Exception ex)
  {
    ex.printStackTrace();
  }
  finally
  {
    portaExecute.FecharCom();
  }
}

/**
 * Método para que seja realizado login pelo JSP
 * @param request
 * @return
 * @throws javax.servlet.ServletException
 * @throws java.io.IOException
 */
public String processarLoginJSP(HttpServletRequest request) throws ServletException,
IOException
{
   Metodo.log("ControlePortaSerial","processarLoginJSP(HttpServletRequest
   request)","Inicio");

  String strNmUsuario = request.getParameter("nomeUsuario");
  String strSenha = request.getParameter("senhaUsuario");
```

```
  String respLogin = null;

  Dados usuario = new Dados();
  DadosMySql ms = new DadosMySql();

  usuario.setNmUsuario(strNmUsuario);
  usuario.setSenha(strSenha);

  try
  {
    usuario = ms.logar(usuario);
    if ( Metodo.nEstarVazio((Dados)usuario) )
      ms.contaEntradas(usuario);
  }
  catch (SQLException e)
  {
     throw new IOException("Erro ao processar o Login JSP |
     Classe: ControlePortaSerial | Método: processarLoginJSP(HttpServletRequest request)
     | Erro:  " + e);
  }

   Metodo.log("ControlePortaSerial","processarLoginJSP(HttpServletRequest
   request)","Fim");

  if(usuario != null && usuario.getIdUsuario() > 0 )
  {
    return "LerTemperatura.jsp";
  }
  else
  {
    respLogin = "N";
    request.setAttribute("respLogin",respLogin);
    return "Login.jsp";
  }
}

/**
 * Para que seja realizado login pelo Midlet -- celular
 * @param request
 * @param response
 * @throws javax.servlet.ServletException
 * @throws java.io.IOException
 */
public void processarLoginJ2ME(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException
{
  // Recupera a String vinda do J2ME com: nome de usuário e senha
  String strNmUsuario = request.getParameter("nomeUsado");
  String strSenha = null;
  // Cria fluxo para saída
  PrintWriter saida = null;
  // Pega o que estar antes e depois do caractere "/"
  String dados[] = strNmUsuario.split("/");

  // Cria objeto responsável por guardar dados que serão utilizados no banco
  Dados usuario = new Dados();

  // Seta / Preenche parâmetros
  usuario.setNmUsuario(dados[0]);
  usuario.setSenha(dados[1]);

  // Cria objeto responsável por interagir com o banco
  DadosMySql ms = new DadosMySql();

  try
  {
    usuario = ms.logar(usuario);
    if ( Metodo.nEstarVazio((Dados)usuario) )
      ms.contaEntradas(usuario);
  }
  catch (SQLException e)
  {
    throw new IOException("Erro em banco de dados: " + e);
  }

  if(usuario != null && usuario.getIdUsuario() > 0 )
  {
    response.setContentType("text/plain");
    saida = response.getWriter();
    saida.print("ok");
  }
```

```
  else
  {
    response.setContentType("text/plain");
    saida = response.getWriter();
    saida.print("n");
  }
  saida.close();
}

/**
 * Para processar as respostas para o J2ME -- celular -- Midlet
 * @param request
 * @param response
 * @throws javax.servlet.ServletException
 * @throws java.io.IOException
 */
public void processarRespostaJ2ME(HttpServletRequest request, HttpServletResponse
response)
throws ServletException, IOException
{
  String comando = request.getParameter("nomeCampo");
  String comandoSetPoint = request.getParameter("nomeUsado");

  PrintWriter saida = null;
  PortaExecute portaExecute = new PortaExecute();
  String status = null;
  String setPointL = null;
  String setPoint = null;
  String setPointT = null;
  String temperatura = null;
  String resposta = null;
  String dados[];

  if(Metodo.nEstarVazio(comandoSetPoint))
  {
    dados = comandoSetPoint.split("/");
    comando = dados[0];
    setPoint = dados[1];
  }

  try
  {
    if ( Metodo.nEstarVazio(comando) && comando.equals("3") )        // Pedindo o status
    {
      portaExecute = portaExecute.configuraConectaSerial(portaExecute);
      portaExecute.setStrDadoEnviado(comando);
      portaExecute.enviaDadoParaSerial(portaExecute.getStrDadoEnviado());
      Metodo.recebeDado(portaExecute);
      status = portaExecute.getStrDadoRecebido();
      if ( Metodo.nEstarVazio(status) )
      {
        if(status.equals("1"))
          status = "Tudo Ligado";
        else if(status.equals("2"))
          status = "Refrigerando";
        else if(status.equals("3"))
          status = "Aquecendo";
        else if(status.equals("4"))
          status = "Tudo Desligado";
        else if(status.equals("5"))
          status = "Aquecedor Desligado";
        else if(status.equals("6"))
          status = "Aquecedor Ligado";
      }
      comando = "4";
      portaExecute.setStrDadoEnviado(comando);
      portaExecute.enviaDadoParaSerial(portaExecute.getStrDadoEnviado());
      Metodo.recebeDado(portaExecute);
      temperatura = portaExecute.getStrDadoRecebido();
      if ( Metodo.nEstarVazio(temperatura) )
        temperatura = Metodo.retornaTemperatura(temperatura) + " ºC";
      comando = "5";
      portaExecute.setStrDadoEnviado(comando);
      portaExecute.enviaDadoParaSerial(portaExecute.getStrDadoEnviado());
      Metodo.recebeDado(portaExecute);
      setPointL = portaExecute.getStrDadoRecebido();
      if ( Metodo.nEstarVazio(setPointL) )
      {
        setPointL = Metodo.retornaTemperatura(setPointL) + " ºC";
        resposta = status + " à " + temperatura + " com Set-Point de: " + setPointL;
        response.setContentType("text/plain");
```

```
            saida = response.getWriter();
            saida.print(resposta);
          }
        }
        else if ( Metodo.nEstarVazio(comando) && comando.equals("2") )        // Ligar tudo
        {
          portaExecute = portaExecute.configuraConectaSerial(portaExecute);
          portaExecute.setStrDadoEnviado(comando);
          portaExecute.enviaDadoParaSerial(portaExecute.getStrDadoEnviado());
          Metodo.recebeDado(portaExecute);
          resposta = portaExecute.getStrDadoRecebido();
          if ( Metodo.nEstarVazio(resposta) )
            status = "Aparelho Ligado";
          comando = "4";
          portaExecute.setStrDadoEnviado(comando);
          portaExecute.enviaDadoParaSerial(portaExecute.getStrDadoEnviado());
          Metodo.recebeDado(portaExecute);
          temperatura = portaExecute.getStrDadoRecebido();
          if ( Metodo.nEstarVazio(temperatura) )
          {
            temperatura = Metodo.retornaTemperatura(temperatura) + " ºC";
            resposta = status + " à " + temperatura;
            response.setContentType("text/plain");
            saida = response.getWriter();
            saida.print(resposta);
          }
        }
        else if ( Metodo.nEstarVazio(comando) && comando.equals("1") )     // Desligar  tudo
        {
          portaExecute = portaExecute.configuraConectaSerial(portaExecute);
          portaExecute.setStrDadoEnviado(comando);
          portaExecute.enviaDadoParaSerial(portaExecute.getStrDadoEnviado());
          Metodo.recebeDado(portaExecute);
          resposta = portaExecute.getStrDadoRecebido();
          if ( Metodo.nEstarVazio(resposta) )
            status = "Aparelho Desligado";
          comando = "4";
          portaExecute.setStrDadoEnviado(comando);
          portaExecute.enviaDadoParaSerial(portaExecute.getStrDadoEnviado());
          Metodo.recebeDado(portaExecute);
          temperatura = portaExecute.getStrDadoRecebido();
          if ( Metodo.nEstarVazio(temperatura) )
          {
            temperatura = Metodo.retornaTemperatura(temperatura) + " ºC";
            resposta = status + " à " + temperatura;
            response.setContentType("text/plain");
            saida = response.getWriter();
            saida.print(resposta);
          }
        }
        else if ( Metodo.nEstarVazio(comando) && comando.equals("7") )  // Enviar Set-Point
        {
          portaExecute = portaExecute.configuraConectaSerial(portaExecute);
          setPointT = Metodo.retornaSemPonto(setPoint);
          portaExecute.setStrDadoEnviado(setPointT);
          portaExecute.enviaDadoParaSerial(portaExecute.getStrDadoEnviado());
          Metodo.recebeDado(portaExecute);
          resposta = portaExecute.getStrDadoRecebido();
          if(Metodo.nEstarVazio(resposta))
            resposta = Metodo.retornaTemperatura(resposta) + " ºC";
          response.setContentType("text/plain");
          saida = response.getWriter();
          saida.print(resposta);
        }
      }
      catch (Exception ex)
      {
        throw new IOException("Erro em servlet : " +ex);
      }
      finally
      {
        portaExecute.FecharCom();
        saida.close();
      }
    }
}
```

APÊNDICE C – CÓDIGO DO MICROCONTROLADOR (C)

```
/*
* Versão final do software para o controlador
* Autor: Clesio Pinto Rabelo Júnior
* Junho de 2006
*/
#include<at89s8252.h>

// Temperatura lida
int nuTemperatura;
// Set-Point definido
int setPoint;
// Dado recebido pelo servidor
int dadoRecebido;
// Flag para desligar aparelho
int verificador;
// Indica o status do aparelho
int status;
// Mesmo que Histerese
int atraso;

/*
 Função para configurar a comunicação serial
*/
void confSerial()
{
    // Modo 1 e Habilita Recepção de Dados
    SCON=0x50;
    // Seta bit de controle
    TCLK=1;
    // Seta bit de controle
    RCLK=1;
    // Clock como base
    C_T2=0;
    // 255 ---- 115200
    RCAP2H=0xFF;
    // 253 ---- 115200
    RCAP2L=0xFD;
    // Habilita operação do periférico
    TR2=1;
}
/*
 Função para enviar um inteiro pela serial
 Operação critica representado por “critical”
*/
void enviaInt(int *dado) critical
{
    // Configura a comunicação serial
    confSerial();
    // Recebe o valor contido no endereço apontado pelo ponteiro
    SBUF=*dado;
    // Inicia a transmissão automaticamente
    while(!TI);
    // Quando TI = 1 a transmissão terminou
    // Limpa TI para permitir nova transmissão
    TI=0;
}
/*
 Função que fornece um tempo de espera
*/
void espera()
{
    int i;
    // Espera tempo suficiente para conversão
    for(i=0;i<100;i++);
}

/*
 Função para fazer a leitura do sinal digitalizado
 Fornecido pelo conversor A/D
*/
void lerADC()
{
    // Clear bit (WR) para ler sinal analógico
    P0_2=0;
    // Espera a leitura
    espera();
    // Seta bit para iniciar a conversão
```

```
    P0_2=1;
    // Espera terminar a conversão
    espera();
    // Salva valor convertido da porta 2 para nuTemperatura
    nuTemperatura=P2;
}
/*
 Função que realiza o controle do tipo ON/OFF
*/
void controlaTemperatura()
{
    // Usadas no controle da temperatura
    int tempMax, tempMin;
    // Histerise entre 4 a 6 equivale a 1 ºC
    atraso=6;
    tempMax = (setPoint+atraso);
    tempMin = (setPoint-atraso);
    // Aparelho ligado
    if(verificador==1)
    {
        // Temperatura mais alta que Set-Point
        if(nuTemperatura>=(tempMax))
        {
            // Liga compressor
            P0_3=1;
            P0_4=1;
            // Aparelho ligado e compressor ligado
            status=1;
        }
        // Temperatura menor que Set-Point
        else if(nuTemperatura<=(tempMin))
        {
            // Também desliga compressor
            P0_3=0;
            P0_4=1;
            // Aparelho ligado e compressor desligado
            status=3;
        }
    }
    else
    {
        P0_3=0;
        P0_4=0;
        // Aparelho e compressor desligados
        status=4;
    }
}
/*
 Função que recebe os dados da serial
 Decide o que fazer com o dado recebido
 É Ativa por interrupção
*/
void recebeDado() interrupt 4 critical
{
    while(!RI);
    // Quando RI = 1 a recepção chegou ao fim
    // Salva o que foi recebido na variável
    dadoRecebido=SBUF;
    // Dado recebido ta entre 0 e 10
    if(dadoRecebido >= 0 && dadoRecebido <= 10)
    {
        // Comandos para controle
        switch(dadoRecebido)
        {
             // Desligar tudo
             case 1:
             {
                 verificador=0;
                 P0_3=0;
                 P0_4=0;
                 status=4;
                 enviaInt(&status);
            }break;
            // Ligar tudo

            case 2:
            {
                verificador=1;
                P0_3=1;
                P0_4=1;
                status=1;
```

```
                enviaInt(&status);
            }break;
            // Qual é o Status?

            case 3:
            {
                status=0;

                // Tudo ligado
                if(P0_3==1 && P0_4==1)
                    status=1;
                // Não vai acontecer
                if(P0_3==1 && P0_4==0)
                    status=2;
                // Compressor desligado
                if(P0_3==0 && P0_4==1)
                    status=3;
                // Tudo desligado
                if(P0_3==0 && P0_4==0)
                    status=4;
                // Envia o status
                enviaInt(&status);
            }break;

            // Envia a temperatura
            case 4:
            {
                enviaInt(&nuTemperatura);
            }break;

            // Envia o Set-Point
            case 5:
            {
                enviaInt(&setPoint);
            }break;

            // Envia dado recebido
            default : enviaInt(&dadoRecebido);
        }
    }
    // Entre 82 e 154 , estabelece Set-Point
    else if(dadoRecebido >= 82  && dadoRecebido <= 154)
    {
        // Dado recebido é um Set-Point
        setPoint=dadoRecebido;
        // Envia a temperatura ambiente
        enviaInt(&nuTemperatura);
    }
    // Limpa RI para receber um novo dado
    RI=0;
}
/*
 Função principal main, obrigatória
*/
void main()
{
    // Configura comunicação serial
    confSerial();
    // Ativa tidas as interrupções
    EA=1;
    // Habilita interrupção serial
    ES=1;
    // Coloca RD, CS e WR em zero
    P0=0;
    // Padrão com 25ºC
    setPoint=128;
    // Liga aparelho, se == 0 desliga aparelho
    verificador=1;
    while(1)
    {
        // Recebe valor analogico e guarda digitalizado
        lerADC();
        // Controla a temperatura
        controlaTemperatura();
    }
}
```

APÊNDICE D – CÓDIGO DA MIDLET (JAVA)

```
/*
 * ControleUsuarioMidlet.java classe para celular MIDP 2.0 e CLDC 1.0/1.1
*  Autor: Clesio Pinto Rabelo Júnior
*  Junho de 2006
*/

import java.io.ByteArrayOutputStream;
import java.io.IOException;
import java.io.InputStream;
import java.io.OutputStream;

import javax.microedition.io.Connector;
import javax.microedition.io.HttpConnection;
import javax.microedition.lcdui.Alert;
import javax.microedition.lcdui.AlertType;
import javax.microedition.lcdui.Choice;
import javax.microedition.lcdui.Command;
import javax.microedition.lcdui.CommandListener;
import javax.microedition.lcdui.Display;
import javax.microedition.lcdui.Displayable;
import javax.microedition.lcdui.Form;
import javax.microedition.lcdui.List;
import javax.microedition.lcdui.StringItem;
import javax.microedition.lcdui.TextField;
import javax.microedition.midlet.MIDlet;
import javax.microedition.midlet.MIDletStateChangeException;

/**
 * Extend MIDlet que é uma classe abstrata, possui métodos abstratos pronto
 * para serem implementados na sub-classe atual.
 * será utilizada a conexão HTTP
*/
public class ControleUsuarioMidlet extends MIDlet implements CommandListener
{
    private Display display;                    // Referência o objeto Display
    private Form fmForm;                        // Formulário principal
    private Form fmForm2;                       // Formulário Tela Set Point
    private Form fmForm3;                       // Formulário Tela Status Set Point
    private Form fmFormLogar;                   // Formulário Tela Para logar
    private Command cmSair;                     // Para sair do prog
    private Command cmVoltar;                   // Para voltar a tela anterior
    private Command cmVoltar2;                  // Para voltar a Tela Set-Point
    private Command cmSelecionar;               // Para selecionar o iten inicial
    private Command cmEnviar;                   // Para enviar algo ao servidor
    private Command cmEnviarLogin;              // Para enviar algo ao servidor
    private Command cmLimpar;                   // Para limpar o campo de texto
    private StringItem siTempAmbiente;          // Mostra a temperatura
    private StringItem siStatus;                // Mostra o Status
    private StringItem siStatusSetPoint;        // Mostra o Status
    private TextField tfSetPoint;               // Set point inserido
    private TextField tfSenha;                  // Senha
    private TextField tfUsuario;                // Nome de usuário
    private List listaOpcoes;                   // List opções
    private Alert alerta;                       // Quando o usuario digitar Set-Point errado
    private Alert alerta2;                      // Quando nm ou senha errados
    private boolean podeSair = false;           // Flag para verificar se pode sair
    private String msgRecebida = null;

    /**
     * Método Construtor
     */
    public ControleUsuarioMidlet()
    {
      System.out.println("Inicio Construtor");

      display = Display.getDisplay(this);

      // Cria comandos
      cmSair = new Command("Sair", Command.EXIT, 1);
      cmVoltar = new Command("Voltar", Command.SCREEN, 2);
      cmVoltar2 = new Command("Voltar", Command.SCREEN, 2);
      cmSelecionar = new Command("Selecionar", Command.OK, 3);
      cmEnviar = new Command("Enviar", Command.SCREEN, 3);
      cmLimpar = new Command("Limpar", Command.SCREEN, 3);
      cmEnviarLogin = new Command("Enviar", Command.SCREEN, 3);

      // Cria campo de texto
```

```
        tfSetPoint = new TextField("Set-Point:", "25", 2, TextField.NUMERIC);
        tfUsuario = new TextField("Usuario:", "p", 2, TextField.ANY);
        tfSenha = new TextField("Senha:", "1", 2, TextField.PASSWORD);

        // Cria label para receber texto
        siTempAmbiente = new StringItem("Temperatura: ", "");
        siStatus = new StringItem("Status: ", "");
        siStatusSetPoint = new StringItem("", "");

        // Cria um objeto Alert
        alerta = new Alert("Projeto Final - Tela - Atenção (Set-Point)",
                 "Digite um valor para o Set-Point entre 18 e 30°C.",null,AlertType.WARNING);
        alerta.setTimeout(4000);
        alerta2 = new Alert("Projeto Final - Tela - Atenção (Login)",
                            "Nome de usuário ou senha invalidos.",null,AlertType.WARNING);
        alerta2.setTimeout(4000);

        // Cria formulário
        fmForm = new Form("form1");
        fmForm2 = new Form("form2");
        fmForm3 = new Form("form3");
        fmFormLogar = new Form("Projeto Final - Tela - Logar");

        // Objeto List será visto sozinho na tela
        // Cria objeto List
        listaOpcoes = new List("Projeto Final - Tela - Opções", Choice.IMPLICIT);

        listaOpcoes.append("Verificar Status", null);
        listaOpcoes.append("Ligar Aparelho", null);
        listaOpcoes.append("Desligar Aparelho", null);
        listaOpcoes.append("Set-Point", null);

        listaOpcoes.addCommand(cmSelecionar);
        listaOpcoes.addCommand(cmSair);

        // Para tela de Logar
        fmFormLogar.addCommand(cmEnviarLogin);
        fmFormLogar.addCommand(cmSair);
        fmFormLogar.append(tfUsuario);
        fmFormLogar.append(tfSenha);

        // Para tela de Verificar Status ligar e desligar
        fmForm.addCommand(cmVoltar);
        fmForm.addCommand(cmSair);
        fmForm.append(siStatus);

        // Para a tela de Set Point
        fmForm2.addCommand(cmLimpar);
        fmForm2.addCommand(cmEnviar);
        fmForm2.addCommand(cmVoltar);
        fmForm2.addCommand(cmSair);
        fmForm2.append(tfSetPoint);

        // Para a tela de Status - Set-Point
        fmForm3.addCommand(cmVoltar2);
        fmForm3.addCommand(cmSair);
        fmForm3.append(siStatusSetPoint);
        fmForm3.append(siTempAmbiente);

        // Recebe os eventos do Form
        fmForm.setCommandListener(this);
        fmForm2.setCommandListener(this);
        fmForm3.setCommandListener(this);
        fmFormLogar.setCommandListener(this);

        // Recebe os eventos da List
        listaOpcoes.setCommandListener(this);

        System.out.println("Fim Construtor");
    }

    // Chamado pelo gerenciador de aplicações para inicializar a MIDlet
    public void startApp()
    {
        System.out.println("Inicio startApp");
        // Torna o form visível
        display.setCurrent(fmFormLogar);
        System.out.println("Fim startApp");
    }

    // Chamado pelo gerenciador de aplicações para pausar a MIDlet
```

```
 public void pauseApp()
 {
   System.out.println("Inicio pauseApp");
   System.out.println("Fim pauseApp");
 }

 // Chamado pelo gerenciador de aplicações antes de finalizar a MIDlet
 public void destroyApp(boolean condicao) throws MIDletStateChangeException
 {
   System.out.println("Inicio destroyApp");

   // Se não precisa sair agora
   if ( condicao == false )
   {
     System.out.println("Requisitando para não sair");
     throw new MIDletStateChangeException("Não desligue agora.");
   }
   System.out.println("Fim destroyApp");
 }

/**
* Implementação do método commandAction() da Interface receptora CommandListener
* Verifica qual evento ocorreu
* @param c
* @param s
*/
 public void commandAction(Command c, Displayable s)
 {
   if ( c == cmSair )
   {
     try
     {
       // Verifica se pode sair
       if ( podeSair == false )
       {
         destroyApp(false);
       }
       else
       {
         destroyApp(true);
         // Diz ao gerenciador de aplicativos que é seguro desligar
         notifyDestroyed();
       }
     }
     catch (MIDletStateChangeException ex)
     {
       podeSair = true;
       System.out.println(ex.getMessage());
       System.out.println("Voltando à ativo");
     }
   }
   else if ( c == cmSelecionar )
   {
     System.out.println("Inicio Comando Selecionar");
     int opcao = listaOpcoes.getSelectedIndex();

     switch(opcao)
     {
       case 0:
       {
         System.out.println("Inicio de comando Status");
         System.out.println("Opção escolhida: " + listaOpcoes.getString(0));
         msgRecebida = null;
         siStatus.setText(null);
         fmForm.setTitle("Projeto Final - Tela - Status");
         Thread tStatus = new Thread()
         {
           public void run()
           {
             System.out.println("Inicio da thread do comando Status");
             try
             {
               msgRecebida = conecta("3");
               siStatus.setText(msgRecebida);
               display.setCurrent(fmForm);
             } catch (Exception ex)
             {
               System.out.println("Erro no camando Status :" + ex );
             }
             System.out.println("Fim da thread do comando Status");
           }
```

```
      };
      tStatus.start();
      System.out.println("Fim de comando Status");
      break;
    }
    case 1:
    {
      System.out.println("Inicio de comando Ligar Aparelho");
      System.out.println("Opção escolhida: " + listaOpcoes.getString(1));
      msgRecebida = null;
      siStatus.setText(null);
      fmForm.setTitle("Projeto Final - Tela - Ligando Aparelho");
      Thread tStatusLigaAparelho = new Thread()
      {
        public void run()
        {
          System.out.println("Inicio da thread do comando Ligar Aparelho");
          try
          {
            fmForm.setTitle("Projeto Final - Tela - Ligando Aparelho");
            msgRecebida = conecta("2");
            siStatus.setText(msgRecebida);
            display.setCurrent(fmForm);
          }
          catch(Exception e)
          {
             System.out.println("Erro no camando Ligar Aparelho :" + e );
          }
          System.out.println("Fim da thread do comando Ligar Aparelho");
        }
      };
      tStatusLigaAparelho.start();
      System.out.println("Fim do comando Ligar Aparelho");
      break;
    }
    case 2:
    {
      System.out.println("Opção escolhida: " + listaOpcoes.getString(1));
      msgRecebida = null;
      siStatus.setText(null);
      fmForm.setTitle("Tela - Desligando Aparelho");
      Thread tStatusDesligaAparelho = new Thread()
      {
        public void run()
        {
          System.out.println("Inicio da thread do comando Desligar Aparelho");
          try
          {
            fmForm.setTitle("Projeto Final - Tela - Desligando Aparelho");
            msgRecebida = conecta("1");
            siStatus.setText(msgRecebida);
            display.setCurrent(fmForm);
          }
          catch(Exception e)
          {
             System.out.println("Erro no camando Desligar Aparelho :" + e );
          }
          System.out.println("Fim da thread do comando Desligar Aparelho");
        }
      };
      tStatusDesligaAparelho.start();
      System.out.println("Fim do comando Desligar Aparelho");
      break;
    }
    case 3:
    {
      System.out.println("Inicio do comando Set-Point");
      System.out.println("Opção escolhida: " + listaOpcoes.getString(3));
      fmForm2.setTitle("Projeto Final - Tela - Set-Point");
      display.setCurrent(fmForm2);
      System.out.println("Fim do comando Set-Point");
      break;
    }
  }
  System.out.println("Fim Comando Selecionar");
}
else if ( c == cmVoltar )
{
  System.out.println("Inicio Comando Voltar");
  display.setCurrent(listaOpcoes);
  System.out.println("Fim Comando Voltar");
```

```
}
else if ( c == cmVoltar2 )
{
  System.out.println("Inicio Comando Voltar2");
  display.setCurrent(fmForm2);
  System.out.println("Fim Comando Voltar2");
}
else if ( c == cmLimpar )
{
  System.out.println("Inicio Comando Limpar");
  tfSetPoint.setString("");
  System.out.println("Fim Comando Limpar");
}
else if ( c == cmEnviar )
{
  System.out.println("Inicio Comando Enviar");
  fmForm3.setTitle("Projeto Final - Tela - Set-Point - Resp");
  int setPoint;

  if( tfSetPoint.getString() != null && !tfSetPoint.getString().equals("") )
  {
    setPoint =  Integer.parseInt(tfSetPoint.getString());
    if ( setPoint >= 18 && setPoint <= 30 )
    {
      msgRecebida = null;
      siTempAmbiente.setText("");
      Thread threadSetPoint = new Thread()
      {
        public void run()
        {
          System.out.println("Inicio threadSetPoint");
          try
          {
            msgRecebida = conecta("7",tfSetPoint.getString().toString());
            siStatusSetPoint.setText("Set-Point de " +
                                     tfSetPoint.getString() + "ºC recebido.");
            siTempAmbiente.setText(msgRecebida);
            display.setCurrent(fmForm3);
          }
          catch(IOException e)
          {
            e.printStackTrace();
          }
          System.out.println("Fim threadSetPoint");
        }
      };
      threadSetPoint.start();
    }
    else
      display.setCurrent(alerta,fmForm2);
  }
  else
    display.setCurrent(alerta,fmForm2);

  System.out.println("Fim Comando Enviar");
}
else if ( c == cmEnviarLogin )
{
  System.out.println("Inicio Comando Enviar Login");
  fmFormLogar.setTitle("Projeto Final - Tela - Logar");
  Thread t1 = new Thread()
  {
    public void run()
    {
      System.out.println("Inicio Thread para login");
      try
      {
        msgRecebida = conecta(tfUsuario.getString().toString(),
                                                  tfSenha.getString().toString());
        System.out.println(msgRecebida);
        if( msgRecebida != null && msgRecebida.equals("ok"))
        {
          display.setCurrent(listaOpcoes);
        }
        else
        {
          display.setCurrent(alerta2,fmFormLogar);
        }
      }
      catch(IOException e)
      {
```

```
            e.printStackTrace();
          }
          System.out.println("Fim Thread para login");
        }
      };
      t1.start();
      System.out.println("Fim de Comando Enviar Login");
    }
  }

/**
* Conecta e envia uma mensagem para o servidor
* @param str1, str2
* @return
* @throws java.io.IOException
*/
  public String conecta(String str1, String str2) throws IOException
  {
    System.out.println("Inicio de Conecta");

    HttpConnection http = null;
    String url = null;
    InputStream is = null;
    OutputStream os = null;
    String msg = null;

    if(str1.equals("7"))
    {
      // Set-Point
      url = "http://10.1.1.3:8988/ProjetoFinal-context-root/respostaJ2ME";
    }
    else
    {
      // Login
      url = "http://10.1.1.3:8988/ProjetoFinal-context-root/logar";
    }

    try
    {
      // Cria conexão
      // Implementação da SUN do protocolo HTTP
      Class.forName("com.sun.midp.io.j2me.http.Protocol");
      http = (HttpConnection) Connector.open(url);

      // Configura o método do pedido
      // Exige que fique dentro de try
      // 3 partes (tipo de pedido, cabeçalho e corpo)
      http.setRequestMethod(http.POST);

      // Cabeçalho
      // Envia cabeçalho necessário para POST
      http.setRequestProperty("Content-Type","application/x-www-form-urlencoded");

      // Abre o fluxo
      os = http.openOutputStream();

      // Corpo
      // Não existe método definido em HttpConection para ler o corpo
      // Deve-se obter por meio de um fluxo

      // Dado para enviar

      byte dado[] = ("nomeUsado=" + str1).getBytes();

      os.write(dado);
      os.flush();
      dado = ("/" + str2).getBytes();

      // Grava para enviar ao servidor

      os.write(dado);

      // Recebe resposta em String do servidor
      msg = recebeReposta(http,is);

      System.out.println("Recebeu do servidor: " + msg);
    }
    catch (Exception e)
    {
      throw new IOException("Não foi possivel enviar a msg ao servidor. " + e);
    }
```

```
    finally
    {
      if ( is != null )
        is.close();
      if ( os != null )
        os.close();
      if ( http != null )
        http.close();

      System.out.println("Fim de Conecta");
    }
    return msg;
  }
/**
* Conecta e envia uma mensagem para o servidor
* @param msg
* @return
* @throws java.io.IOException
*/
  public String conecta(String msg) throws IOException
  {
    System.out.println("Inicio de Conecta");

    HttpConnection http = null;
    String url = null;
    InputStream is = null;
    OutputStream os = null;

    url = "http://10.1.1.3:8988/ProjetoFinal-context-root/respostaJ2ME";

    try
    {
      // Cria conexão
      http = (HttpConnection) Connector.open(url);

      http.setRequestMethod(http.POST);

      http.setRequestProperty("Content-Type","application/x-www-form-urlencoded");

      os = http.openOutputStream();

      System.out.println("Msg cliente: " +msg);

      byte dadoCliente[] = ("nomeCampo=" + msg).getBytes();

      os.write(dadoCliente);

      os.flush();

      msg = recebeReposta(http,is);

      System.out.println("Recebeu do servidor: " + msg);
    }
    catch (IOException e)
    {
      throw new IOException("Não foi possivel enviar a msg ao servidor. " + e);
    }
    finally
    {
      if ( is != null )
        is.close();
      if ( os != null )
        os.close();
      if ( http != null )
        http.close();

      System.out.println("Fim de Conecta");
    }
    return msg;
  }

/**
* Retorna a mensagem do servidor
* @param http
* @return
* @throws java.io.IOException
*/
  public String recebeReposta(HttpConnection http, InputStream is) throws IOException
  {
    System.out.println("Inicio recebeResposta");
```

```
        // Para recuperar a linha de status do servidor,
        // Pois reponde com linha de status, cabecalho e corpo
        String strLinhaStatus = http.getResponseMessage();
        int nuLinhaStatus = http.getResponseCode();
        String strDado = null;
        try
        {
          if ( nuLinhaStatus == http.HTTP_OK )
          {
            // 1 obtém linha de status já foi feito
            // 2 obtém cabeçalho dividido em campo, chave e conteúdo
            // Campo

            http.getHeaderField(0);
            // Chave
            http.getHeaderFieldKey(0);
            // Conteúdo
            http.getHeaderField("Tipo conteudo");

            // 3 obtém o dado
            // Abre fluxo de entrada
            is = http.openInputStream();
            // Verifica-se o comprimento do dado
            int length = (int) http.getLength();

            if (length != -1)
            {
              // Veio com o tamanho definido
              byte dadoServidor[] = new byte[length];
              is.read(dadoServidor);
              strDado = new String(dadoServidor);
            }
            else
            {
              // Não veio com o comprimento
              ByteArrayOutputStream bAos = new ByteArrayOutputStream();
              // Le dados um caracter por vez
              int caractere;
              while( (caractere = is.read()) != -1 )
              {
                bAos.write(caractere);
              }
              strDado = new String(bAos.toByteArray());
              bAos.close();
            }
          }
          else
          {
            strDado = new String(http.getResponseMessage());
          }
        }
        catch (IOException e)
        {
          throw new IOException("Não foi possivel receber do servidor. " + e);
        }
        System.out.println("Fim recebeResposta");
        return strDado;
    }
}
```

APÊNDICE E – CÓDIGO DA CLASSE DADOS (JAVA)

```
/*
* Classe que representa as tabelas no banco de dados
* Autor: Clesio Pinto Rabelo Júnior
* Junho de 2006
*/
package projetoFinal.bancoDados;

public class Dados
{
  private int idUsuario           = 0;
  private int idAcesso            = 0;
  private int idAparelho          = 0;
  private int nuAcesso            = 0;
  private String nmUsuario        = null;
  private String senha            = null;
  private String dtAcesso         = null;
  private String dtAtual          = null;
  private String nuTemperatura    = null;
  private String setPoint         = null;
  private String status           = null;

  public Dados()
  {
  }

  public void setIdUsuario(int idUsuario)
  {
    this.idUsuario = idUsuario;
  }

  public int getIdUsuario()
  {
    return idUsuario;
  }

  public void setIdAcesso(int idAcesso)
  {
    this.idAcesso = idAcesso;
  }

  public int getIdAcesso()
  {
    return idAcesso;
  }

  public void setIdAparelho(int idAparelho)
  {
    this.idAparelho = idAparelho;
  }

  public int getIdAparelho()
  {
    return idAparelho;
  }

  public void setNmUsuario(String nmUsuario)
  {
    this.nmUsuario = nmUsuario;
  }

  public String getNmUsuario()
  {
    return nmUsuario;
  }

  public void setSenha(String senha)
  {
    this.senha = senha;
  }

  public String getSenha()
  {
    return senha;
  }

  public void setDtAcesso(String dtAcesso)
  {
    this.dtAcesso = dtAcesso;
```

```
  }

  public String getDtAcesso()
  {
    return dtAcesso;
  }

  public void setNuAcesso(int nuAcesso)
  {
    this.nuAcesso = nuAcesso;
  }

  public int getNuAcesso()
  {
    return nuAcesso;
  }

  public void setDtAtual(String dtAtual)
  {
    this.dtAtual = dtAtual;
  }

  public String getDtAtual()
  {
    return dtAtual;
  }

  public void setStatus(String status)
  {
    this.status = status;
  }

  public String getStatus()
  {
    return status;
  }

  public void setNuTemperatura(String nuTemperatura)
  {
    this.nuTemperatura = nuTemperatura;
  }

  public String getNuTemperatura()
  {
    return nuTemperatura;
  }

  public void setSetPoint(String setPoint)
  {
    this.setPoint = setPoint;
  }

  public String getSetPoint()
  {
    return setPoint;
  }
}
```

APÊNDICE F – CÓDIGO DA CLASSE CONECTABD (JAVA)

```
/*
* Classe que manipula as conexões com o banco de dados
* Autor: Clesio Pinto Rabelo Júnior
* Junho de 2006
*/
package projetoFinal.bancoDados;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import projetoFinal.metodos.Metodo;

public class ConectaBD
{
  private static final String drive="jdbc:odbc:DsProjetoFinal";
  private static final String user="root";
  private static final String senhaBanco="123456";

  /**
  * Método que fornece a conexão ao banco de dados
  * @param con
  * @return
  * @throws java.sql.SQLException
  */
  public Connection getConexao(Connection con) throws SQLException
  {
    Metodo.log("ConectaBD","getConexao(Connection con)","Inicio");
    if (con == null)
    {
      try
      {
        Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
        con = DriverManager.getConnection(drive,user,senhaBanco);
      }
      catch(ClassNotFoundException ex)
      {
        throw new SQLException("Erro, drive não encontrado | Classe: ConectaBD |
                                    Método: getConexao(Connection con) | Erro:  " + ex );
      }
      catch (Exception e)
      {
        throw new SQLException("Erro na SQL Conexao | Classe: ConectaBD |
                                      Método: getConexao(Connection con) | Erro: " + e );
      }
    }
    Metodo.log("ConectaBD","getConexao(Connection con)","Fim");
    return con;
  }
  /**
   * Método que tenta fechar as conexões do bando de dados
   * @param conexao
   * @param preparedStatement
   * @param resultSet
   * @throws java.sql.SQLException
   */
  public void FecharBanco(Connection conexao, PreparedStatement preparedStatement,
  ResultSet resultSet) throws SQLException
  {
    Metodo.log("ConectaBD","FecharBanco(Connection conexao, PreparedStatement
                                      preparedStatement, ResultSet resultSet)","Inicio");
    try
    {
      if(conexao != null)
      {
        conexao.close();
        conexao=null;
      }
      if(preparedStatement != null)
      {
        preparedStatement.close();
        preparedStatement = null;
      }
      if(resultSet != null)
```

```
      {
        resultSet = null;
      }
    }
    catch (Exception ex)
    {
      throw new SQLException("Erro ao tentar fechar o banco | Classe: ConectaBD |
      Método: FecharBanco(Connection conexao, PreparedStatement
      preparedStatement, ResultSet resultSet) | Erro: " + ex);
    }
    Metodo.log("ConectaBD","FecharBanco(Connection conexao, PreparedStatement
                                        preparedStatement, ResultSet resultSet)","Fim");
  }
}
```

APÊNDICE G – CÓDIGO DA CLASSE DADOSMYSQL

```
/*
* Classe responsável pela interação com as tabelas do banco de dados
* Autor: Clesio Pinto Rabelo Júnior
* Junho de 2006
*/
package projetoFinal.bancoDados;

import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Types;

import projetoFinal.metodos.Metodo;

public class DadosMySql extends ConectaBD
{

  /**
   * Método que determina se o usuario pode logar
   * @param Dados dados
   * @return Dados dados
   * @throws java.lang.Exception
   */
  public Dados logar(Dados dados) throws SQLException
  {
    Metodo.log("DadosMySql","logar(Dados dados)","Inicio");

    String sql            = null;
    Connection conn       = null;
    PreparedStatement ps  = null;
    ResultSet rs          = null;
    try
    {
      sql = " select id_usuario, nm_usuario, senha from usuario " +
            " where nm_usuario=? and senha=? ";

      conn  = getConexao(conn);
      ps = conn.prepareStatement(sql);

      ps.setString(1,dados.getNmUsuario());
      ps.setString(2,dados.getSenha());

      rs = ps.executeQuery();

      if ( rs != null && rs.next() )
      {
        Dados usuario = new Dados();
        usuario.setIdUsuario(rs.getInt("id_usuario"));
        usuario.setNmUsuario(rs.getString("nm_usuario"));
        usuario.setSenha(rs.getString("senha"));
        return usuario;
      }
      else
        return null;
    }
    catch(SQLException e)
    {
      FecharBanco(conn,ps,rs);
      throw new SQLException("Erro ao tentar logar | Classe: DadosMySql |
                                          Método: logar(Dados dados) | Erro: " + e);
    }
    finally
    {
      FecharBanco(conn,ps,rs);
      Metodo.log("DadosMySql","logar(Dados dados)","Fim");
    }
  }

  /**
   * Método que guarda no banco os valores de temperatura
   * Abre sempre uma nova conexão ao banco
   * @param dados
   * @throws java.lang.Exception
   */
  public void guardaTemperatura(Dados dados) throws SQLException
  {
```

```
  Metodo.log("DadosMySql","guardaTemperatura(Dados dados)","Inicio");

  String sql            = null;
  Connection conn       = null;
  PreparedStatement ps  = null;
  String data           = Metodo.retornaData();
  dados.setDtAtual(data);
  try
  {
    sql = " insert into aparelho(nu_temperatura, nu_set_point, status, dt_atual) " +
          " values (?,?,?,?) ";

    conn = getConexao(conn);
    ps = conn.prepareStatement(sql);

    ps.setString(1,dados.getNuTemperatura());
    ps.setString(2,dados.getSetPoint());
    ps.setString(3,dados.getStatus());
    ps.setString(4,dados.getDtAtual());

    ps.executeUpdate();
  }
  catch(SQLException e)
  {
    FecharBanco(conn,ps,null);
    throw new SQLException("Erro em guardaTemperatura(dados): " + e);
  }
  finally
  {
    FecharBanco(conn,ps,null);
    Metodo.log("DadosMySql","guardaTemperatura(Dados dados)","Fim");
  }
}

/**
 * Método que guarda no banco os valores de temperatura
 * Usa uma conexão já aberta
 * @param dados
 * @throws java.lang.Exception
 */
public void guardaTemperatura(Dados dados, Connection conn) throws SQLException
{
  Metodo.log("DadosMySql","guardaTemperatura(Dados dados, Connection conn)","Inicio");

  String sql            = null;
  PreparedStatement ps  = null;
  String data           = Metodo.retornaData();
  dados.setDtAtual(data);
  try
  {
    sql = " insert into aparelho(nu_temperatura, nu_set_point, status, dt_atual) " +
          " values (?,?,?,?) ";

    ps = conn.prepareStatement(sql);

    if(Metodo.nEstarVazio(dados.getNuTemperatura()))
      ps.setString(1,dados.getNuTemperatura());
    else
      ps.setNull(1,Types.VARCHAR);

    if(Metodo.nEstarVazio(dados.getSetPoint()))
      ps.setString(2,dados.getSetPoint());
    else
      ps.setNull(2,Types.VARCHAR);

    if(Metodo.nEstarVazio(dados.getStatus()))
      ps.setString(3,dados.getStatus());
    else
      ps.setNull(3,Types.VARCHAR);

    if(Metodo.nEstarVazio(dados.getDtAtual()))
      ps.setString(4,dados.getDtAtual());
    else
      ps.setNull(4,Types.VARCHAR);

    ps.execute();
  }
  catch(SQLException e)
  {
    FecharBanco(conn,ps,null);
    throw new SQLException("Erro em guardaTemperatura(dados,conn): " + e);
```

```
    }
    finally
    {
      // Não fecha a conexão aqui
      FecharBanco(null,ps,null);
      Metodo.log("DadosMySql","guardaTemperatura(Dados dados, Connection conn)","Fim");
    }
  }

  /**
   * Método para marcar o número de acessos no sistema de cada usuário
   * @param dados
   * @throws java.lang.Exception
   */
  public void contaEntradas(Dados dados) throws SQLException
  {
    Metodo.log("DadosMySql","contaEntradas(Dados dados)","Inicio");

    String sql            = null;
    String sqlConsulta    = null;
    Connection conn       = null;
    PreparedStatement ps  = null;
    ResultSet rs          = null;
    String data           = Metodo.retornaData();
    int nuAcesso          = 0;
    dados.setDtAcesso(data);
    try
    {
      sql = " insert into acesso(dt_acesso,nu_acesso,id_usuario) " +
            " values (?,?,?) ";

      sqlConsulta = " select nu_acesso from acesso where id_usuario=? ";

      conn = getConexao(conn);

      //TYPE_SCROLL_SENSITIVE=1005 -> navegação nos dois sentidos
      //CONCUR_UPDATABLE=1008 -> leitura e atualização

      ps = conn.prepareStatement(sqlConsulta,1005,1008);
      ps.setInt(1,dados.getIdUsuario());

      rs = ps.executeQuery();

      if(rs != null && rs.last())
      {
        nuAcesso = (int) rs.getInt("nu_acesso");
      }

      if(Metodo.nEstarVazio(nuAcesso))
      {
        nuAcesso = nuAcesso + 1;
        dados.setNuAcesso(nuAcesso);
      }
      else
      {
        nuAcesso = 1;
        dados.setNuAcesso(nuAcesso);
      }

      ps = null;
      ps = conn.prepareStatement(sql);

      ps.setString(1,dados.getDtAcesso());
      ps.setInt(2,dados.getNuAcesso());
      ps.setInt(3,dados.getIdUsuario());

      ps.executeUpdate();
    }
    catch(SQLException e)
    {
      FecharBanco(conn,ps,rs);
      throw new SQLException("Não foi possível contar as entradas: " + e);
    }
    finally
    {
      FecharBanco(conn,ps,rs);
      Metodo.log("DadosMySql","contaEntradas(Dados dados)","Fim");
    }
  }
}
```

APÊNDICE H – CÓDIGO DA CLASSE PEGADADOS (JAVA)

```
/*
* Classe responsável por guardar a temperatura ambiente, Set-Point o status do aparelho
* e a data em que foi realizado a leitura.
* Será usada para a criação de gráfico
* Autor: Clesio Pinto Rabelo Júnior
* Junho de 2006
*/
package projetoFinal.bancoDados;

import java.sql.Connection;

import projetoFinal.comunicacaoSerial.PortaExecute;

import projetoFinal.metodos.Metodo;

public class PegaDados
{
  public PegaDados()
  {
  }
  /**
   * Método para coletar a temperatura em tempo definido
   * @param setPoint
   * @throws java.io.IOException
   */
  public static void guardarDados(String setPoint, int tempo, int vezes)
                                                                    throws Exception
  {
    String resposta = null;
    DadosMySql dadosMySql = new DadosMySql();
    Dados dados = new Dados();
    Connection conexao = null;
    PortaExecute portaExecute = new PortaExecute();

    try
    {
      conexao = dadosMySql.getConexao(conexao);
      portaExecute = portaExecute.configuraConectaSerial(portaExecute);
      dados.setSetPoint(setPoint);
      setPoint = Metodo.retornaSemPonto(setPoint);
      do
      {
        portaExecute.setStrDadoEnviado(setPoint);

        portaExecute.setStrDadoRecebido(null);

        portaExecute.enviaDadoParaSerial(portaExecute.getStrDadoEnviado());

        Metodo.recebeDado(portaExecute);

        resposta = portaExecute.getStrDadoRecebido();

        if(Metodo.nEstarVazio(resposta))
        {
          resposta = Metodo.retornaTemperatura(resposta);

          dados.setNuTemperatura(resposta);

          System.out.println("Temperatura lida: " + resposta);

          dadosMySql.guardaTemperatura(dados,conexao);

          Metodo.espera(tempo);
        }
        dados.setNuTemperatura(null);
        vezes--;
      } while(vezes>0);
    }
    catch (Exception ex)
    {
      throw new Exception("Erro no método guardaDados(): " + ex);
    }
    finally
    {
      portaExecute.FecharCom();
```

```
      dadosMySql.FecharBanco(conexao,null,null);
    }
  }
/**
* Método principal necessário
*
*/
  public static void main(String a[]) throws Exception
  {
    try
    {
      PegaDados.guardarDados("25",1,60);
    }
    catch (Exception ex)
    {
      ex.printStackTrace();
    }
  }
}
```

APÊNDICE I – CÓDIGO DA CLASSE PORTASERIAL (JAVA)

```
/*
* Classe que servirá para guardar dados da porta serial
* Autor: Clesio Pinto Rabelo Júnior
* Junho de 2006
*/

package projetoFinal.comunicacaoSerial;

import java.io.InputStream;
import java.io.OutputStream;

import javax.comm.CommPortIdentifier;
import javax.comm.SerialPort;

public class PortaSerial
{
  private CommPortIdentifier idPorta   = null;
  private SerialPort pSerial           = null;
  private String nmPorta               = null;
  private String strDadoRecebido       = null;
  private String strDadoEnviado        = null;
  private InputStream entrada          = null;
  private OutputStream saida           = null;
  private int nuDadoRecebido           = 0;
  private int nuBaudRate               = 0;
  private int nuTimeOut                = 0;
  private int nuParidade               = 0;
  private int nuBitsDados              = 0;
  private int nuBitsParada             = 0;
  private int nuControleFluxo          = 0;
  private boolean dadoChegou           = false;
  private boolean portaAberta          = false;

  public PortaSerial()
  {
  }

  // Criação dos métodos set e get
  // set é para colocar algo dentro da variável
  // get é para recuperar o que tem dentro da variável

  public void setNmPorta(String nmPorta)
  {
    this.nmPorta = nmPorta;
  }

  public String getNmPorta()
  {
    return nmPorta;
  }

  public void setNuBaudRate(int nuBaudRate)
  {
    this.nuBaudRate = nuBaudRate;
  }

  public int getNuBaudRate()
  {
    return nuBaudRate;
  }

  public void setNuTimeOut(int nuTimeOut)
  {
    this.nuTimeOut = nuTimeOut;
  }

  public int getNuTimeOut()
  {
    return nuTimeOut;
  }

  public void setNuParidade(int nuParidade)
  {
    this.nuParidade = nuParidade;
  }
```

```
public int getNuParidade()
{
  return nuParidade;
}

public void setNuBitsDados(int nuBitsDados)
{
  this.nuBitsDados = nuBitsDados;
}

public int getNuBitsDados()
{
  return nuBitsDados;
}

public void setNuBitsParada(int nuBitsParada)
{
  this.nuBitsParada = nuBitsParada;
}

public int getNuBitsParada()
{
  return nuBitsParada;
}

public void setNuControleFluxo(int nuControleFluxo)
{
  this.nuControleFluxo = nuControleFluxo;
}

public int getNuControleFluxo()
{
  return nuControleFluxo;
}

public void setNuDadoRecebido(int nuDadoRecebido)
{
  this.nuDadoRecebido = nuDadoRecebido;
}

public int getNuDadoRecebido()
{
  return nuDadoRecebido;
}

public void setIdPorta(CommPortIdentifier idPorta)
{
  this.idPorta = idPorta;
}

public CommPortIdentifier getIdPorta()
{
  return idPorta;
}

public void setPSerial(SerialPort pSerial)
{
  this.pSerial = pSerial;
}

public SerialPort getPSerial()
{
  return pSerial;
}

public void setEntrada(InputStream entrada)
{
  this.entrada = entrada;
}

public InputStream getEntrada()
{
  return entrada;
}

public void setSaida(OutputStream saida)
{
  this.saida = saida;
}
```

```
  public OutputStream getSaida()
  {
    return saida;
  }

  public void setPortaAberta(boolean portaAberta)
  {
    this.portaAberta = portaAberta;
  }

  public boolean isPortaAberta()
  {
    return portaAberta;
  }

  public void setStrDadoRecebido(String strDadoRecebido)
  {
    this.strDadoRecebido = strDadoRecebido;
  }

  public String getStrDadoRecebido()
  {
    return strDadoRecebido;
  }

  public void setStrDadoEnviado(String strDadoEnviado)
  {
    this.strDadoEnviado = strDadoEnviado;
  }

  public String getStrDadoEnviado()
  {
    return strDadoEnviado;
  }

  public void setDadoChegou(boolean dadoChegou)
  {
    this.dadoChegou = dadoChegou;
  }

  public boolean isDadoChegou()
  {
    return dadoChegou;
  }

}
```

APÊNDICE J – CÓDIGO DA CLASSE PORTACONFIG (JAVA)

```
/*
* Classe para configurar a porta serial
* precisa do pacote javax.com
* Esta classe implementa um método de SerialPortEventListener
* Autor: Clesio Pinto Rabelo Júnior
* Junho de 2006
*/

package projetoFinal.comunicacaoSerial;

import java.io.IOException;

import java.util.Enumeration;

import javax.comm.CommPortIdentifier;
import javax.comm.SerialPort;
import javax.comm.SerialPortEvent;
import javax.comm.SerialPortEventListener;
import projetoFinal.metodos.Metodo;

public class PortaConfig extends PortaSerial implements SerialPortEventListener
{
  private Enumeration listaDePortas   = null;
  private boolean OK                  = false;
  private boolean ocupado             = false;

  public PortaConfig()
  {
  }

  /**
   * Método responsável por fornecer a conexão à porta serial
   * @throws java.lang.Exception
   */
  public void conecta() throws Exception
  {
    Metodo.log("PortaConfig","conecta()","Inicio");
    if(PortaExiste(getNmPorta()))
    {
      try
      {
        if (isPortaAberta()==false)
        {
          // Pega id da porta serial e abre conexão
          setIdPorta( CommPortIdentifier.getPortIdentifier( getNmPorta() ) );
          setPSerial( (SerialPort) getIdPorta().open( "PortaExecute", getNuTimeOut() ) );
          // Configura serial com os parâmetros escolhidos
          getPSerial().setSerialPortParams( getNuBaudRate(), getNuBitsDados() ,
                                                  getNuBitsParada(), getNuParidade() );
          getPSerial().setFlowControlMode( getNuControleFluxo() );
          getPSerial().addEventListener( this );
          // Abre fluxos de entrada e saída
          setEntrada( getPSerial().getInputStream() );
          setSaida( getPSerial().getOutputStream() );
          // Configura notificações automáticas
          getPSerial().notifyOnDataAvailable( true );
          getPSerial().notifyOnBreakInterrupt( true );
        }
        OK = true;
      }
      catch (Exception e)
      {
        FecharCom();
        Metodo.log("PortaConfig","conecta()","Erro");
        throw new IOException("Erro ao tentar se conectar: conecta() da
                                                          classe PotaConfig : " + e);
      }
    }
    if(OK)
      setPortaAberta(true);
    else
      setPortaAberta(false);

    Metodo.log("PortaConfig","conecta()","Fim");
  }
```

```
/**
* Método que fecha a conexão com a porta serial
*/
public void FecharCom()
{
  Metodo.log("PortaConfig","FecharCom()","Inicio");

  if ( !isPortaAberta() )
    return;

  if ( getPSerial() != null )
  {
    try
    {
      getSaida().close();
      getEntrada().close();
    }
    catch (IOException e)
    {
      Metodo.log("PortaConfig","eFecharCom()","Erro");
    }
    getPSerial().close();
  }
  setPortaAberta(false);

  Metodo.log("PortaConfig","FecharCom()","Fim");
}

/**
 * Método que deve ser implementado ao se extender a interface SerialPortEventListener:
 * public interface SerialPortEventListener extends EventListener
 * {
 *    public abstract void serialEvent(SerialPortEvent spe)
 * }
 *
 * @param SerialEvet e
 */
public void serialEvent(SerialPortEvent e)
{
  Metodo.log("PortaConfig","serialEvent(SerialPortEvent e)","Inicio");

  switch (e.getEventType())
  {
    case SerialPortEvent.DATA_AVAILABLE:
    {
      lerDadoPorta();
    }
    break;

    case SerialPortEvent.BI:
      setStrDadoRecebido("\n--- Recpeção Encerrada ---\n");
  }
 Metodo.log("PortaConfig","serialEvent(SerialPortEvent e)","Fim");
}
/**
 * Método para ler o dado que estiver na porta serial
 */
private void lerDadoPorta()
{
  Metodo.log("PortaConfig","lerDadoPorta()","Inicio");

  StringBuffer inputBuffer = new StringBuffer();
  int tamanhoDado = 1;
  int dadoRecebido = 0;
  while ( tamanhoDado > 0 )
  {
    try
    {
      tamanhoDado = getEntrada().available();
      if(tamanhoDado>0)
      {
        dadoRecebido = getEntrada().read();
        System.out.println("Dado recebido: " + dadoRecebido );
        inputBuffer.append(dadoRecebido);
        setStrDadoRecebido(new String(inputBuffer));
        System.out.println("Dado guardado em String");
        System.out.println("OK acabou de receber pela serial.");
        setDadoChegou(true);
      }
    }
    catch (IOException ex)
```

```
      {
        Metodo.log("PortaConfig","lerDadoPorta()","Erro");
      }
    }
    Metodo.log("PortaConfig","lerDadoPorta()","Fim");
  }
   /**
   * Pesquisa se a porta existe
   * @param nmPorta
   * @return boolean e
   */
  private boolean PortaExiste(String nmPorta)
  {
    Metodo.log("PortaConfig","PortaExiste(String nmPorta)","Inicio");

    String temp;
    boolean e = false;
    listaDePortas = getIdPorta().getPortIdentifiers();
    if ( nmPorta != null )
    {
      while (listaDePortas.hasMoreElements())
      {
        setIdPorta((CommPortIdentifier) listaDePortas.nextElement());
        temp = getIdPorta().getName();
        if (temp.equals(nmPorta) == true)
        {
          e = true;
          setIdPorta(null);
          break;
        }
      }
    }
    else
      System.out.println("Nome da porta não foi passado.");

    Metodo.log("PortaConfig","PortaExiste(String nmPorta)","Fim");

    return e;
  }
}
```

APÊNDICE L – CÓDIGO DA CLASSE PORTAEXECUTE (JAVA)

```
/*
* Classe para interagir com a porta serial
* Extend a classe PortaConfig então é também um tipo de PortaSerial
* Autor: Clesio Pinto Rabelo Júnior
* Junho de 2006
*/
package projetoFinal.comunicacaoSerial;

import java.io.IOException;
import projetoFinal.metodos.Metodo;

public class PortaExecute extends PortaConfig
{
  public PortaExecute()
  {
  }
  /**
   * Método que envia um inteiro para a porta serial
   * @param String msg
   */
  public void enviaDadoParaSerial( String msg ) throws IOException
  {
    Metodo.log("PortaExecute","enviaDadoParaSerial( String msg )","Inicio");

    try
    {
      setSaida( getPSerial().getOutputStream() );
      System.out.println("Fluxo OK!");
      System.out.println("Enviando um byte para " + getNmPorta());
      System.out.println("Enviando: " + msg);
      int dadoServidor = Integer.parseInt(msg);
      getSaida().write(dadoServidor);
      getSaida().flush();
    }
    catch (Exception ex)
    {
      Metodo.log("PortaExecute","enviaDadoParaSerial( String msg )","Erro");
      throw new IOException("Houve um erro durante o envio! STATUS " + ex);
    }

    Metodo.log("PortaExecute","enviaDadoParaSerial( String msg )","Fim");
  }
  /**
   * Método para configurar os parâmetros da porta serial e conectar
   * @return PortaExecute portaExecute
   */
  public PortaExecute configuraConectaSerial(PortaExecute portaExecute) throws IOException
  {
    Metodo.log("PortaExecute","configuraConectaSerial(PortaExecute
                                                        portaExecute)","Inicio");

    portaExecute.setNmPorta("COM2");
    portaExecute.setNuBaudRate(115200);
    portaExecute.setNuBitsDados(8);
    portaExecute.setNuBitsParada(1);
    portaExecute.setNuControleFluxo(0);
    portaExecute.setNuParidade(1);
    portaExecute.setNuTimeOut(3000);

    try
    {
      conecta();
    }
    catch(Exception e)
    {
      FecharCom();
      Metodo.log("PortaExecute","configuraConectaSerial(PortaExecute
                                                          portaExecute)","Erro");
      throw new IOException("Erro em configurar e conectar a porta serial: " + e);
    }
    Metodo.log("PortaExecute","configuraConectaSerial(PortaExecute portaExecute)","Fim");
    return portaExecute;
  }
}
```

APÊNDICE M – CÓDIGO DA CLASSE METODO (JAVA)

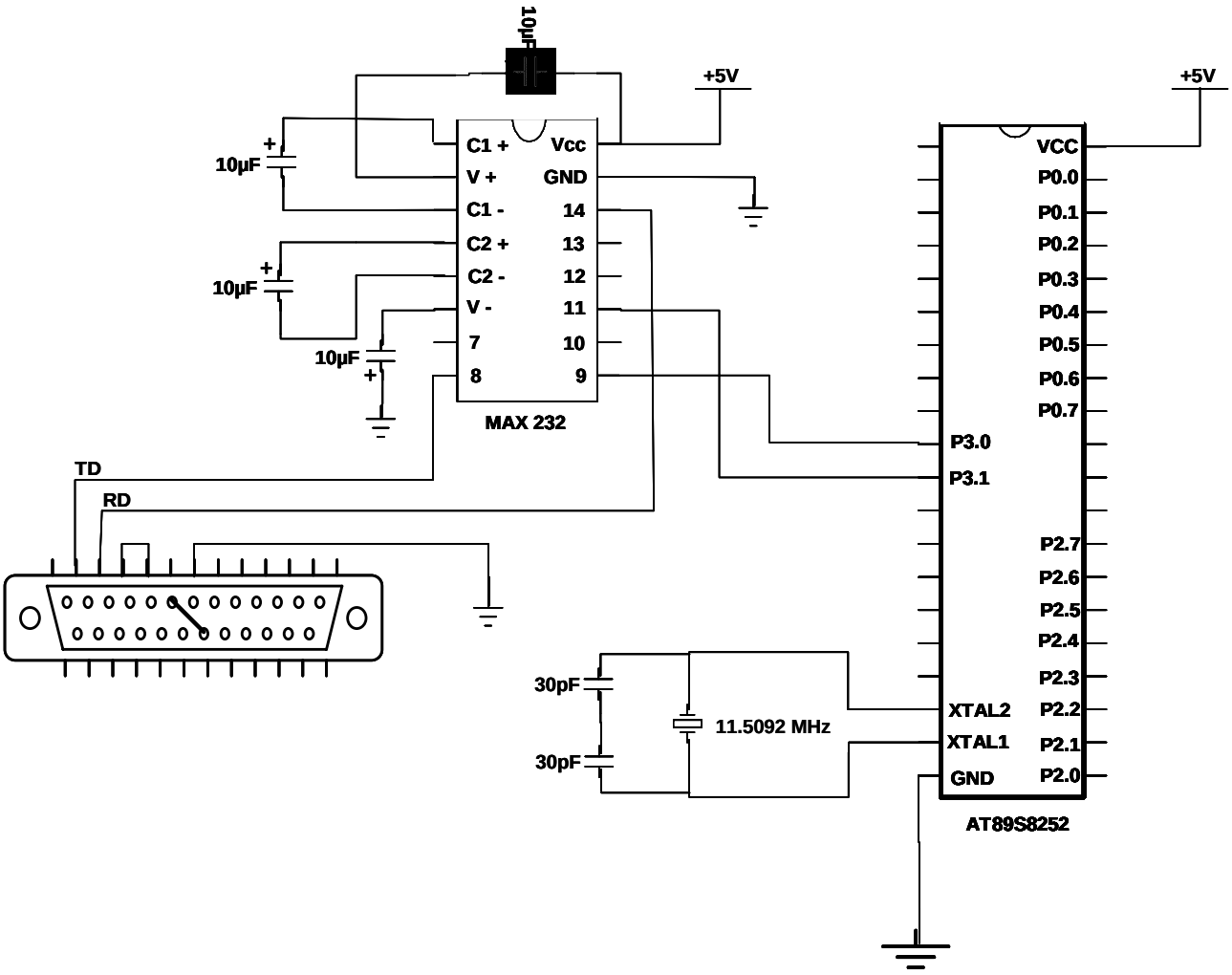
```
/*
 * Essa classe disponibilizará vários métodos para diferentes classes do projeto
 * Autor: Clesio Pinto Rabelo Júnior
 * Junho de 2006
 */
package projetoFinal.metodos;

import java.text.DecimalFormat;

import java.util.Date;

import projetoFinal.comunicacaoSerial.PortaExecute;
/*
 * Pode ser usada da seguinte forma
 * Exemplo:
 * ..................................
 * if(Metodo.nEstarVazio())
 *  System.out.println("String cheia");
 * else
 *  System.out.println("String vazia");
 * ..................................
 */
public class Metodo
{
  /**
   * Método que retornar a data atual do servidor
   * @return Data
   */
  public static String retornaData()
  {
    log("Metodo","retornaData()","Inicio");

    Date dataAtual = new Date();
    String dataInteira = null;

    int dia       = dataAtual.getDate();
    int mes       = dataAtual.getMonth();
    int ano       = dataAtual.getYear();
    int hora      = dataAtual.getHours();
    int minutos   = dataAtual.getMinutes();
    int segundos  = dataAtual.getSeconds();

    log("Metodo","retornaData()","Fim");

    return dataInteira = dia + "-" + (mes+1) + "-" + (ano+1900) + " " + hora + ":" +
                                                              minutos + ":" + segundos;
  }

  /**
   * Método para identificar se o inteiro estar vazio
   * @param nuValor
   * @return boolean
   */
  public static boolean nEstarVazio(int nuValor)
  {
    if(nuValor != 0 && nuValor > 0)
      return true;
    else
      return false;
  }
  /**
   * Método para identificar se a String estar vazia
   * @param String strValor
   * @return boolean
   */
  public static boolean nEstarVazio(String strValor)
  {
    if( strValor != null && !strValor.equals("0") && !strValor.equals("null")  &&
                                                                  !strValor.equals("") )
      return true;
    else
      return false;
  }
  /**
   * Método para identificar se o obejeto estar vazio
```

```
 * @param String strValor
 * @return boolean
 */
public static boolean nEstarVazio(Object strValor)
{
  if( strValor != null && !strValor.equals("0") && !strValor.equals("null"))
    return true;
  else
    return false;
}
/**
 * Método que pode ser usado nos campos do JSP
 * @param String strValor
 * @return ""
 */
public static String campoVazio(String strValor)
{
  log("Metodo","campoVazio(String strValor)","Inicio");

  if( strValor != null && !strValor.equals("null") )
  {
    log("Metodo","campoVazio(String strValor)","Fim");
    return strValor;
  }
  else
  {
    log("Metodo","campoVazio(String strValor)","Fim");
    return "";
  }
}
/**
 * Método para retornar de um "float" uma String sem virgula
 * @param String comVirgula
 * @return String
 */
public static String retornaSemVirgula(String comVirgula)
{
  log("Metodo","retornaSemVirgula(String comVirgula)","Inicio");

  String semVirgula = new String();
  StringBuffer sb = new StringBuffer();

  for(int i=0 ; i<comVirgula.length(); i++)
  {
    char c = (char)comVirgula.charAt(i);
    if(c==',')
    {
      System.out.println("Chegou na virgula");
      break;
    }
    else
      sb.append(c);
  }
  semVirgula = new String(sb);

  log("Metodo","retornaSemVirgula(String comVirgula)","Fim");

  return semVirgula;
}
/**
 * Método para retornar uma String sem ponto separador
 * @param String comPonto
 * @return String
 */
public static String retornaSemPonto(String comPonto)
{
  log("Metodo","retornaSemPonto(String comPonto)","Inicio");

  StringBuffer sb = new StringBuffer();
  String setPoinConvertido = new String();
  final double resolucao = 0.1953125;
  double setPoint = Double.parseDouble(comPonto);
  setPoint = setPoint/resolucao;
  double v = Math.ceil(setPoint);
  comPonto = v + " ";
  for(int i=0 ; i<comPonto.length(); i++)
  {
    char c = (char)comPonto.charAt(i);
    if(c=='.')
    {
      System.out.println("Chegou no ponto");
```

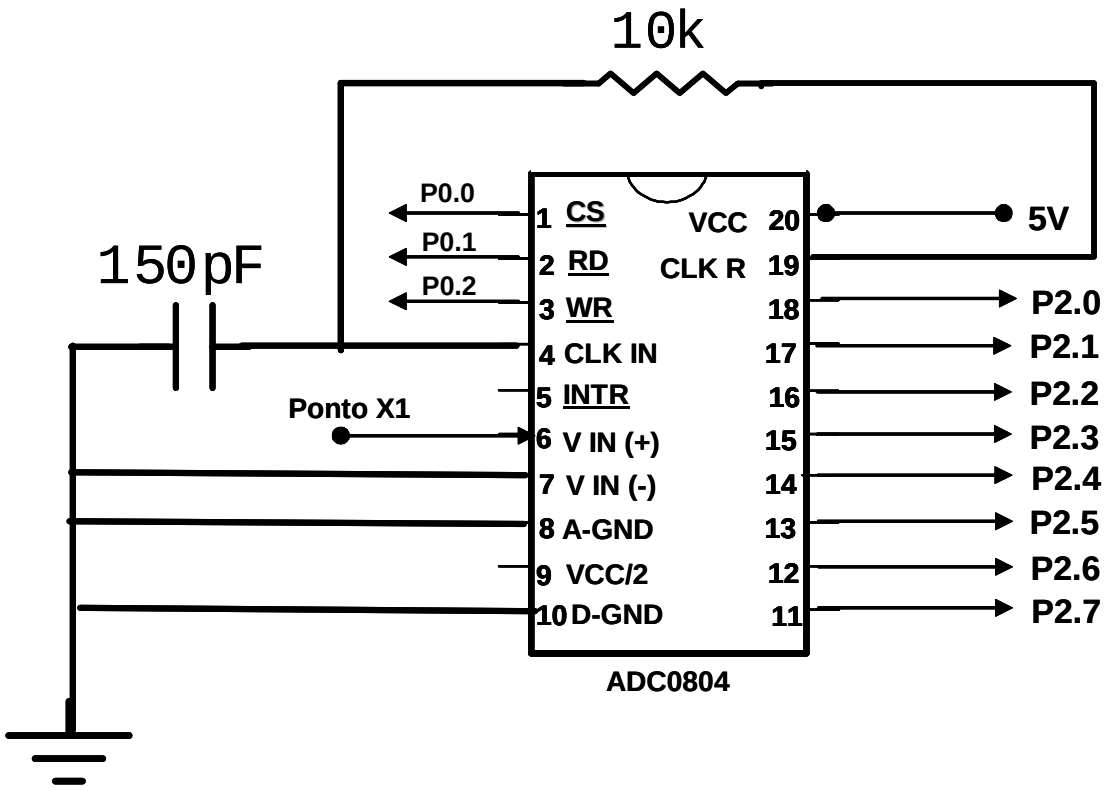
```
        break;
      }
      else
        sb.append(c);
    }
    setPoinConvertido = new String(sb);

    log("Metodo","retornaSemPonto(String comPonto)","Fim");

    return setPoinConvertido;
  }
  /**
   * Método para converter o valor lido do A/D em temperatura (ºC)
   * @param String strValor
   * @return String
   */
  public static String retornaTemperatura(String strValor)
  {
    log("Metodo","retornaTemperatura(String strValor)","Inicio");

    final double resolucao = 0.1953125;
    double temp = Double.parseDouble(strValor);
    temp = temp*resolucao;

    // Número de casas decimais. No caso 2.
    DecimalFormat casas = new DecimalFormat ("00.00");
    strValor = casas.format(temp);

    log("Metodo","retornaTemperatura(String strValor)","Fim");

    return strValor;
  }
  /**
   * Método para identificar se chegou algum dado pela serial
   * @param PortaExecute portaExecute
   */
  public static void recebeDado(PortaExecute portaExecute)
  {
    log("Metodo","recebeDado(PortaExecute portaExecute)","Inicio");

    Date tempo2 = new Date();
    int t2 = tempo2.getMinutes();
    int t = 0;
    int t1 = 0;
    do
    {
      Date tempo1 = new Date();
      t1 = tempo1.getMinutes();
      t = t1-t2;
      if ( t == 1 )
      {
        System.out.println("passou 1 minuto");
        portaExecute.setDadoChegou(true);
        portaExecute.setStrDadoRecebido(null);
      }
    }
    while(portaExecute.isDadoChegou()==false);
    portaExecute.setDadoChegou(false);

    log("Metodo","recebeDado(PortaExecute portaExecute)","Fim");
  }
  /**
   * Método para gerar um tempo de espera
   * @param int temp
   */
  public static void espera(int temp)
  {
    log("Metodo","espera(int temp)","Inicio");

    boolean v = false;
    Date tempo1 = new Date();
    int tMinutos1 = tempo1.getMinutes();
    int tMinutos2 = 0;
    int tempoTotal = 0;
    do
    {
      Date tempo2 = new Date();
      tMinutos2 = tempo2.getMinutes();
      tempoTotal = tMinutos2-tMinutos1;
      if ( tempoTotal == temp || tempoTotal < 0 )
      {
```

```
        System.out.println("passou: " + temp + " minutos" );
        v = true;
      }
    }
    while(v==false);
    log("Metodo","espera(int temp)","Fim");
  }
  /**
   * Método para imprimir as mensagens
   * @param msg
   */
  public static void log(String strClasse, String strMetodo, String v)
  {
    if(nEstarVazio(strClasse) || nEstarVazio(strMetodo))
      System.out.println(v + " da Classe:" + strClasse + " | Método:" + strMetodo);
    else
      System.out.print(" Método e/ou Classe não identificado.");
  }
}
```

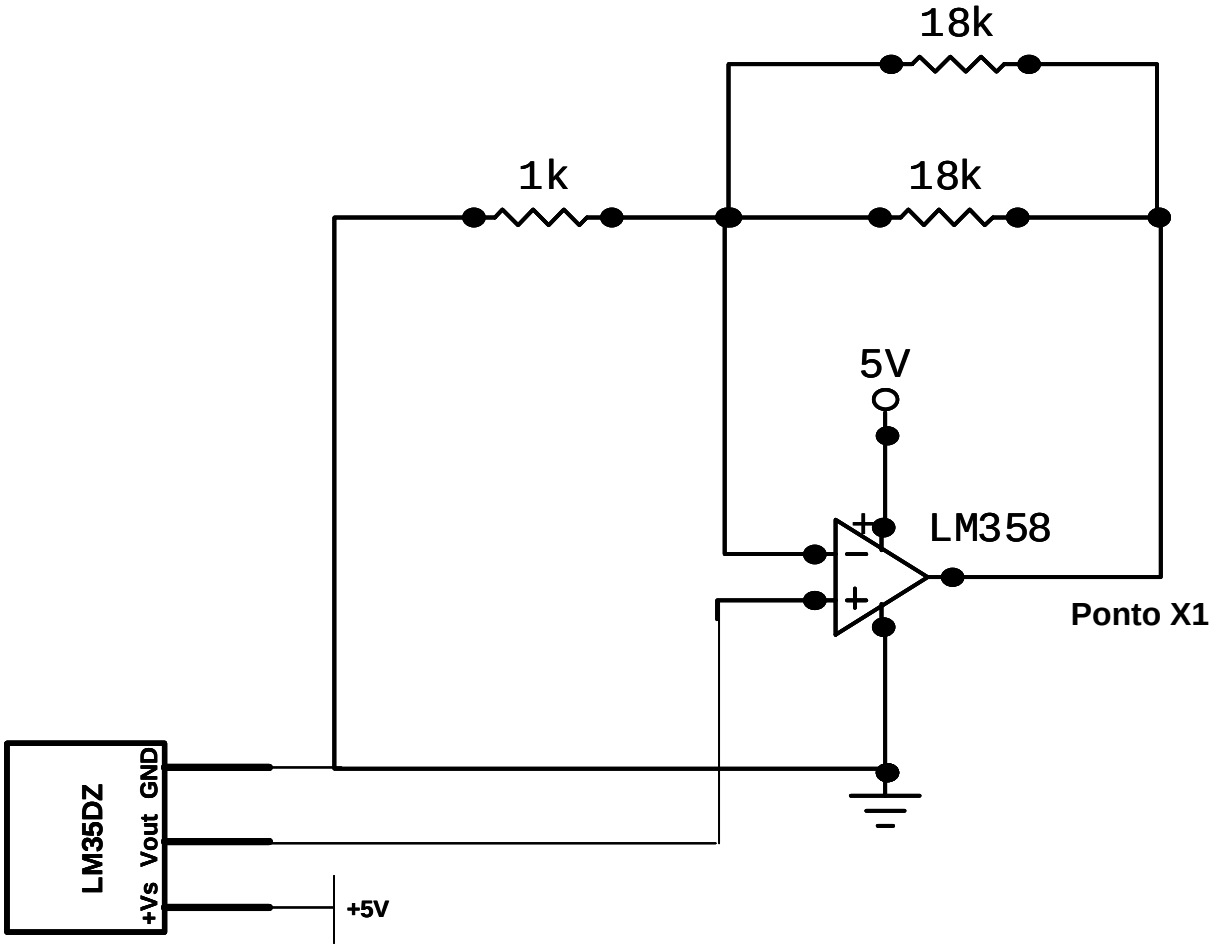
APÊNDICE N – LIGAÇÃO DO MICROCONTROLADOR À INTERFACE SERIAL

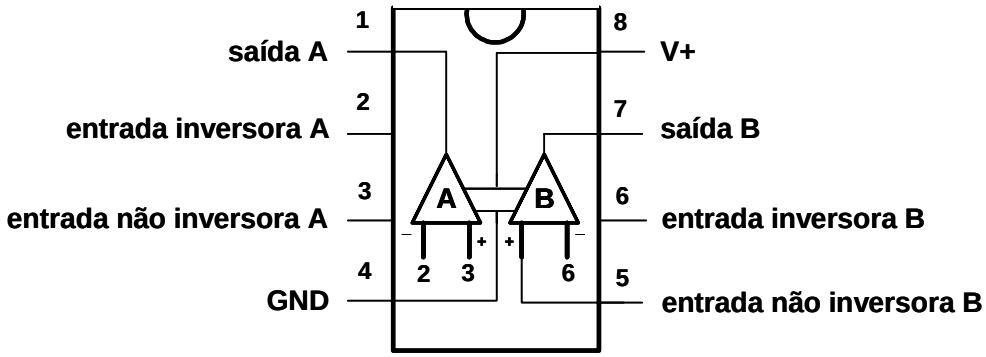


APÊNDICE O – LIGAÇÃO DA INTERFACE DE ENTRADA AO MICROCONTROLADOR



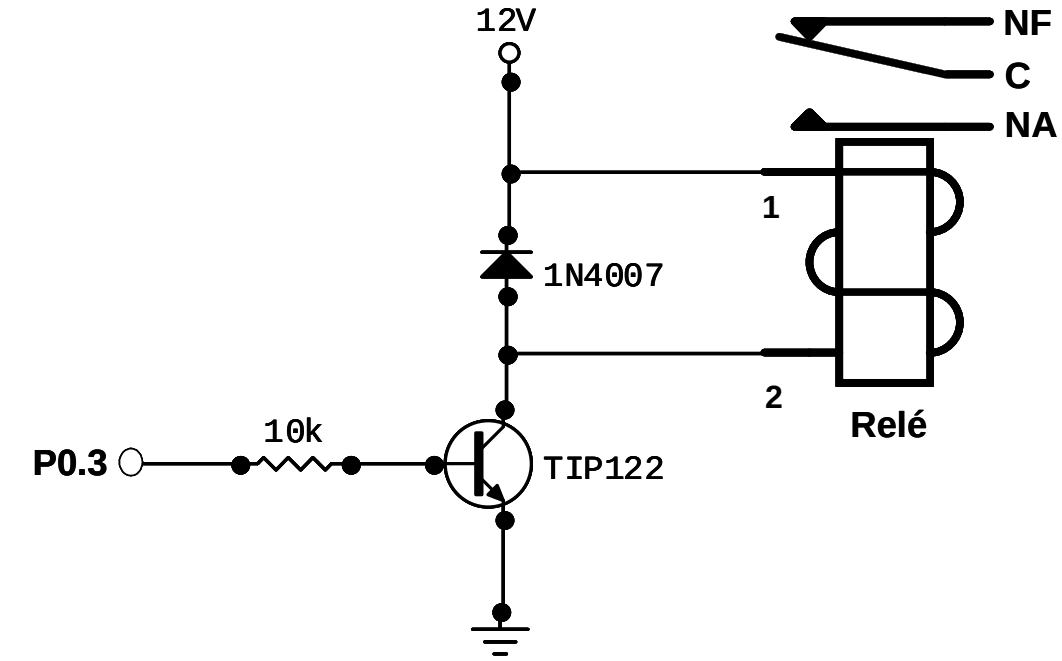
APÊNDICE P – LIGAÇÃO DA INTERFACE CONDICIONADORA E AMPLIFICADORA DE SINAL E SENSOR DE TEMPERATURA À INTERFACE DE ENTRADA

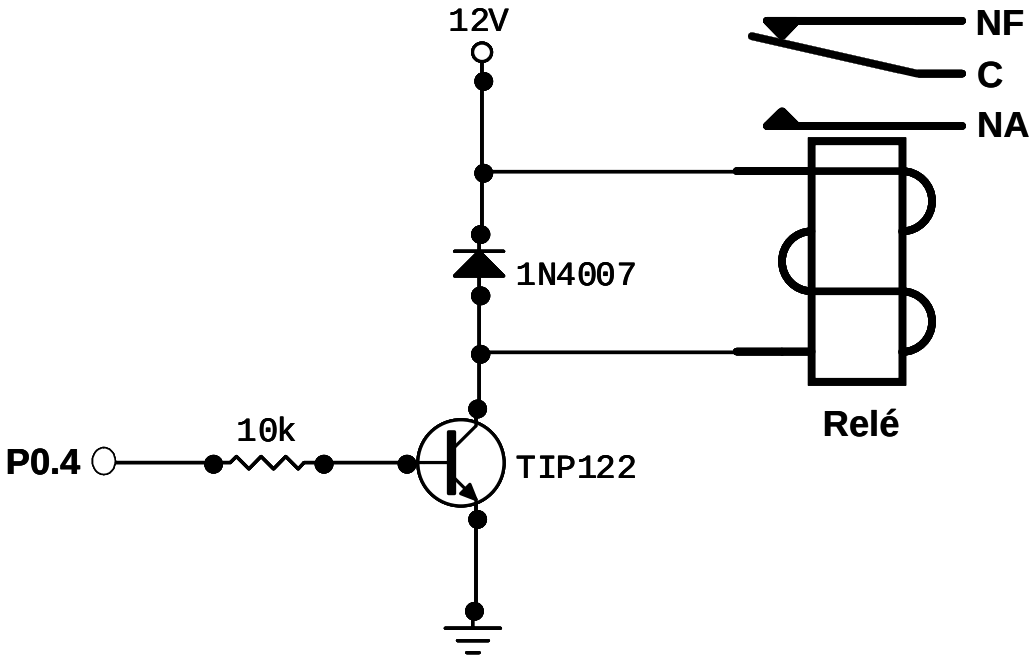




Circuito Integrado LM358 da National Semiconductor

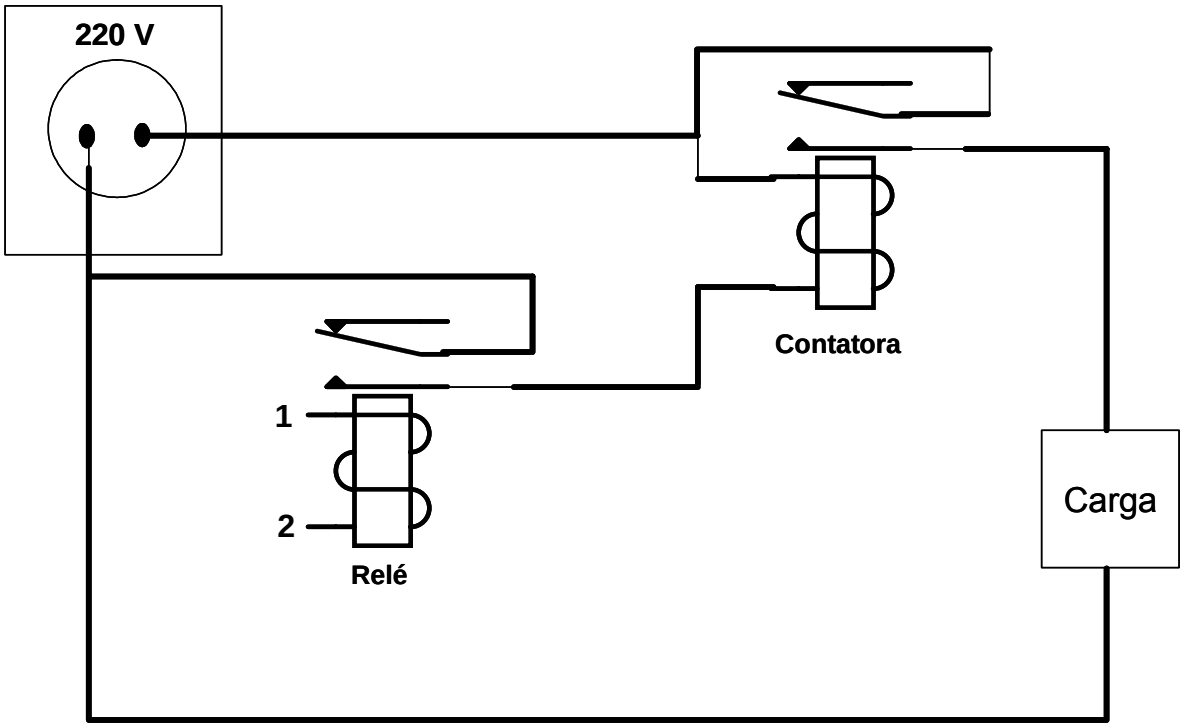
APÊNDICE Q – LIGAÇÃO DAS INTERFACES DE POTÊNCIA AO MICROCONTROLADOR E AO RELÉ





NF – Normalmente fechado
C – Comum
NA – Normalmente aberto

APÊNDICE R – LIGAÇÃO DO RELÉ À CONTATORA E A REDE ELÉTRICA



APÊNDICE S – PARTES IMPORTANTES DO DATASHEETS DO SENSOR DE TEMPERATURA LM35DZ DA NATIONAL SEMICONDUCTOR

Typical Applications

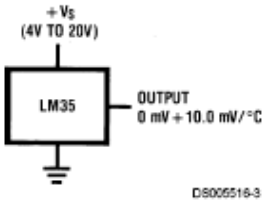
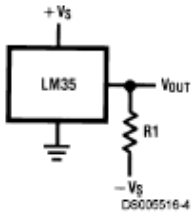


FIGURE 1. Basic Centigrade Temperature Sensor
(+2°C to +150°C)

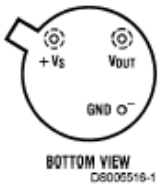


Choose $R_1 = -V_S/50\ \mu A$
V_{OUT}=+1,500 mV at +150°C
= +250 mV at +25°C
= −550 mV at −55°C

FIGURE 2. Full-Range Centigrade Temperature Sensor

Connection Diagrams

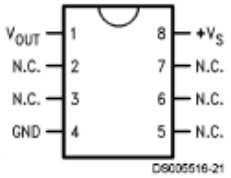
TO-46
Metal Can Package*



*Case is connected to negative pin (GND)

Order Number LM35H, LM35AH, LM35CH, LM35CAH or LM35DH
See NS Package Number H03H

SO-8
Small Outline Molded Package



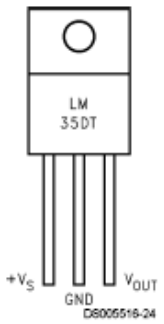
N.C. = No Connection

Top View
Order Number LM35DM
See NS Package Number M08A

TO-92
Plastic Package

Order Number LM35CZ,
LM35CAZ or LM35DZ
See NS Package Number Z03A

TO-220
Plastic Package*



*Tab is connected to the negative pin (GND).
Note: The LM35DT pinout is different than the discontinued LM35DP.

Order Number LM35DT
See NS Package Number TA03F

Absolute Maximum Ratings (Note 10)

If Military/Aerospace specified devices are required, please contact the National Semiconductor Sales Office/ Distributors for availability and specifications.

Supply Voltage	+35V to −0.2V
Output Voltage	+6V to −1.0V
Output Current	10 mA
Storage Temp.;	
TO-46 Package,	−60°C to +180°C
TO-92 Package,	−60°C to +150°C
SO-8 Package,	−65°C to +150°C
TO-220 Package,	−65°C to +150°C
Lead Temp.:	
TO-46 Package, (Soldering, 10 seconds)	300°C
TO-92 and TO-220 Package, (Soldering, 10 seconds)	260°C
SO Package (Note 12)	
Vapor Phase (60 seconds)	215°C
Infrared (15 seconds)	220°C
ESD Susceptibility (Note 11)	2500V
Specified Operating Temperature Range: T_{MIN} to T_{MAX} (Note 2)	
LM35, LM35A	−55°C to +150°C
LM35C, LM35CA	−40°C to +110°C
LM35D	0°C to +100°C

Electrical Characteristics

(Notes 1, 6)

Parameter	Conditions	LM35A			LM35CA			Units (Max.)
		Typical	Tested Limit (Note 4)	Design Limit (Note 5)	Typical	Tested Limit (Note 4)	Design Limit (Note 5)	
Accuracy	T_A=+25°C	±0.2	±0.5		±0.2	±0.5		°C
(Note 7)	T_A=−10°C	±0.3			±0.3		±1.0	°C
	T_A=T_{MAX}	±0.4	±1.0		±0.4	±1.0		°C
	T_A=T_{MIN}	±0.4	±1.0		±0.4		±1.5	°C
Nonlinearity	T_{MIN}≤T_A≤T_{MAX}	**±0.18**		**±0.35**	**±0.15**		**±0.3**	°C
(Note 8)								
Sensor Gain	T_{MIN}≤T_A≤T_{MAX}	**+10.0**	**+9.9, +10.1**		**+10.0**		**+9.9, +10.1**	mV/°C
(Average Slope)								
Load Regulation	T_A=+25°C	±0.4	±1.0		±0.4	±1.0		mV/mA
(Note 3) 0≤I_L≤1 mA	T_{MIN}≤T_A≤T_{MAX}	**±0.5**		**±3.0**	**±0.5**		**±3.0**	mV/mA
Line Regulation	T_A=+25°C	±0.01	±0.05		±0.01	±0.05		mV/V
(Note 3)	4V≤V_S≤30V	**±0.02**		**±0.1**	**±0.02**		**±0.1**	mV/V
Quiescent Current	V_S=+5V, +25°C	56	67		56	67		µA
(Note 9)	V_S=+5V	**105**		**131**	**91**		**114**	µA
	V_S=+30V, +25°C	56.2	68		56.2	68		µA
	V_S=+30V	**105.5**		**133**	**91.5**		**116**	µA
Change of	4V≤V_S≤30V, +25°C	0.2	1.0		0.2	1.0		µA
Quiescent Current	4V≤V_S≤30V	**0.5**		**2.0**	**0.5**		**2.0**	µA
(Note 3)								
Temperature Coefficient of Quiescent Current		**+0.39**		**+0.5**	**+0.39**		**+0.5**	µA/°C
Minimum Temperature for Rated Accuracy	In circuit of *Figure 1*, I_L=0	+1.5		+2.0	+1.5		+2.0	°C
Long Term Stability	T_J=T_{MAX}, for 1000 hours	±0.08			±0.08			°C

Electrical Characteristics

(Notes 1, 6)

Parameter	Conditions	LM35			LM35C, LM35D			Units (Max.)
		Typical	Tested Limit (Note 4)	Design Limit (Note 5)	Typical	Tested Limit (Note 4)	Design Limit (Note 5)	
Accuracy,	T_A=+25°C	±0.4	±1.0		±0.4	±1.0		°C
LM35, LM35C	T_A=−10°C	±0.5			±0.5		±1.5	°C
(Note 7)	T_A=T_{MAX}	±0.8	±1.5		±0.8		±1.5	°C
	T_A=T_{MIN}	±0.8		±1.5	±0.8		±2.0	°C
Accuracy, LM35D	T_A=+25°C				±0.6	±1.5		°C
(Note 7)	T_A=T_{MAX}				±0.9		±2.0	°C
	T_A=T_{MIN}				±0.9		±2.0	°C
Nonlinearity	T_{MIN}≤T_A≤T_{MAX}	**±0.3**		**±0.5**	**±0.2**		**±0.5**	°C
(Note 8)								
Sensor Gain	T_{MIN}≤T_A≤T_{MAX}	**+10.0**	**+9.8, +10.2**		**+10.0**		**+9.8, +10.2**	mV/°C
(Average Slope)								
Load Regulation	T_A=+25°C	±0.4	±2.0		±0.4	±2.0		mV/mA
(Note 3) 0≤I_L≤1 mA	T_{MIN}≤T_A≤T_{MAX}	**±0.5**		**±5.0**	**±0.5**		**±5.0**	mV/mA
Line Regulation	T_A=+25°C	±0.01	±0.1		±0.01	±0.1		mV/V
(Note 3)	4V≤V_S≤30V	**±0.02**		**±0.2**	**±0.02**		**±0.2**	mV/V
Quiescent Current	V_S=+5V, +25°C	56	80		56	80		µA
(Note 9)	V_S=+5V	**105**		**158**	**91**		**138**	µA
	V_S=+30V, +25°C	56.2	82		56.2	82		µA
	V_S=+30V	**105.5**		**161**	**91.5**		**141**	µA
Change of	4V≤V_S≤30V, +25°C	0.2	2.0		0.2	2.0		µA
Quiescent Current	4V≤V_S≤30V	**0.5**		**3.0**	**0.5**		**3.0**	µA
(Note 3)								
Temperature		**+0.39**		**+0.7**	**+0.39**		**+0.7**	µA/°C
Coefficient of								
Quiescent Current								
Minimum Temperature	In circuit of	+1.5		+2.0	+1.5		+2.0	°C
for Rated Accuracy	*Figure 1*, I_L=0							
Long Term Stability	T_J=T_{MAX}, for	±0.08			±0.08			°C
	1000 hours							

Note 1: Unless otherwise noted, these specifications apply: −55°C≤T_J≤+150°C for the LM35 and LM35A; −40°≤T_J≤+110°C for the LM35C and LM35CA; and 0°≤T_J≤+100°C for the LM35D. V_S=+5Vdc and I_{LOAD}=50 µA, in the circuit of *Figure 2*. These specifications also apply from +2°C to T_{MAX} in the circuit of *Figure 1*. Specifications in **boldface** apply over the full rated temperature range.

Note 2: Thermal resistance of the TO-46 package is 400°C/W, junction to ambient, and 24°C/W junction to case. Thermal resistance of the TO-92 package is 180°C/W junction to ambient. Thermal resistance of the small outline molded package is 220°C/W junction to ambient. Thermal resistance of the TO-220 package is 90°C/W junction to ambient. For additional thermal resistance information see table in the Applications section.

Note 3: Regulation is measured at constant junction temperature, using pulse testing with a low duty cycle. Changes in output due to heating effects can be computed by multiplying the internal dissipation by the thermal resistance.

Note 4: Tested Limits are guaranteed and 100% tested in production.

Note 5: Design Limits are guaranteed (but not 100% production tested) over the indicated temperature and supply voltage ranges. These limits are not used to calculate outgoing quality levels.

Note 6: Specifications in **boldface** apply over the full rated temperature range.

Note 7: Accuracy is defined as the error between the output voltage and 10mv/°C times the device's case temperature, at specified conditions of voltage, current, and temperature (expressed in °C).

Note 8: Nonlinearity is defined as the deviation of the output-voltage-versus-temperature curve from the best-fit straight line, over the device's rated temperature range.

Note 9: Quiescent current is defined in the circuit of *Figure 1*.

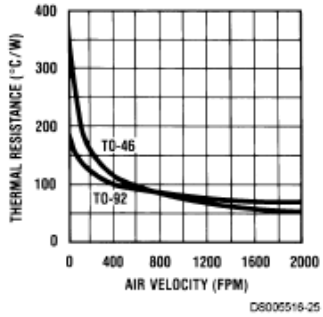
Note 10: Absolute Maximum Ratings indicate limits beyond which damage to the device may occur. DC and AC electrical specifications do not apply when operating the device beyond its rated operating conditions. See Note 1.

Note 11: Human body model, 100 pF discharged through a 1.5 kΩ resistor.

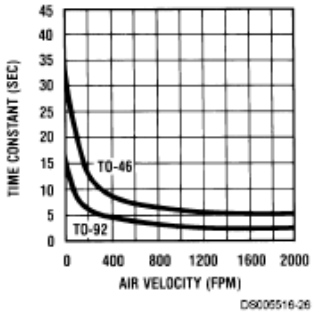
Note 12: See AN-450 "Surface Mounting Methods and Their Effect on Product Reliability" or the section titled "Surface Mount" found in a current National Semiconductor Linear Data Book for other methods of soldering surface mount devices.

Typical Performance Characteristics

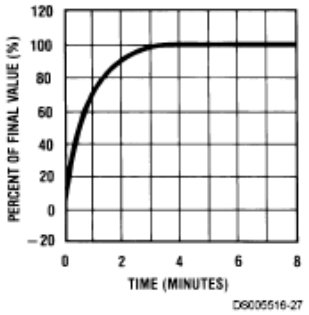
Thermal Resistance
Junction to Air



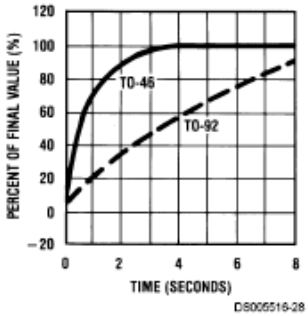
Thermal Time Constant



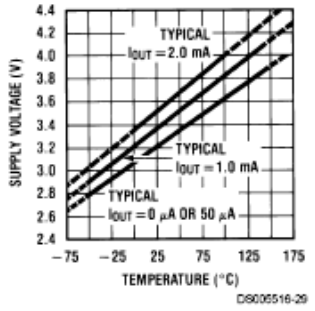
Thermal Response
in Still Air



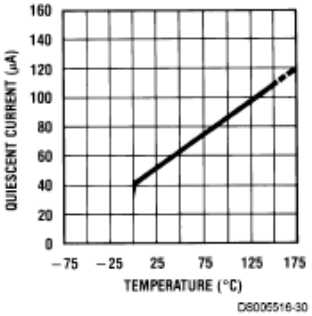
Thermal Response in
Stirred Oil Bath



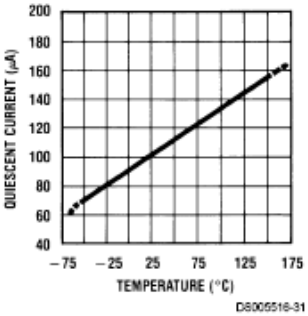
Minimum Supply
Voltage vs. Temperature



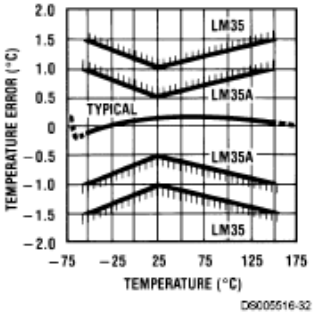
Quiescent Current
vs. Temperature
(In Circuit of *Figure 1*.)



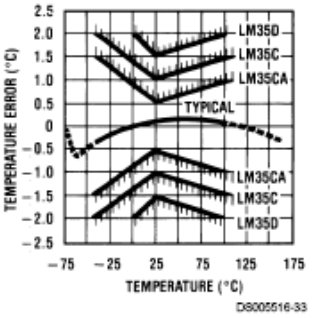
Quiescent Current
vs. Temperature
(In Circuit of *Figure 2*.)



Accuracy vs. Temperature
(Guaranteed)

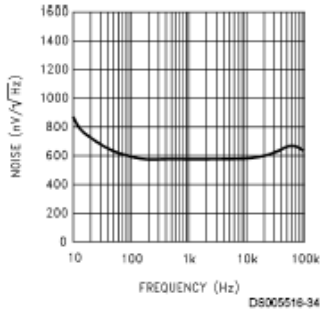


Accuracy vs. Temperature
(Guaranteed)

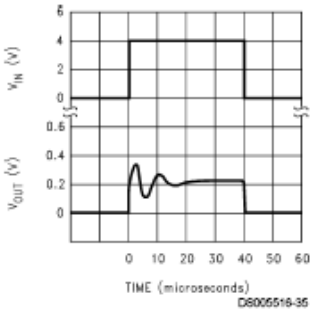


Typical Performance Characteristics (Continued)

Noise Voltage



Start-Up Response



Applications

The LM35 can be applied easily in the same way as other integrated-circuit temperature sensors. It can be glued or cemented to a surface and its temperature will be within about 0.01°C of the surface temperature.

This presumes that the ambient air temperature is almost the same as the surface temperature; if the air temperature were much higher or lower than the surface temperature, the actual temperature of the LM35 die would be at an intermediate temperature between the surface temperature and the air temperature. This is expecially true for the TO-92 plastic package, where the copper leads are the principal thermal path to carry heat into the device, so its temperature might be closer to the air temperature than to the surface temperature.

To minimize this problem, be sure that the wiring to the LM35, as it leaves the device, is held at the same temperature as the surface of interest. The easiest way to do this is to cover up these wires with a bead of epoxy which will insure that the leads and wires are all at the same temperature as the surface, and that the LM35 die's temperature will not be affected by the air temperature.

The TO-46 metal package can also be soldered to a metal surface or pipe without damage. Of course, in that case the V− terminal of the circuit will be grounded to that metal. Alternatively, the LM35 can be mounted inside a sealed-end metal tube, and can then be dipped into a bath or screwed into a threaded hole in a tank. As with any IC, the LM35 and accompanying wiring and circuits must be kept insulated and dry, to avoid leakage and corrosion. This is especially true if the circuit may operate at cold temperatures where condensation can occur. Printed-circuit coatings and varnishes such as Humiseal and epoxy paints or dips are often used to insure that moisture cannot corrode the LM35 or its connections.

These devices are sometimes soldered to a small light-weight heat fin, to decrease the thermal time constant and speed up the response in slowly-moving air. On the other hand, a small thermal mass may be added to the sensor, to give the steadiest reading despite small deviations in the air temperature.

Temperature Rise of LM35 Due To Self-heating (Thermal Resistance, θ_{JA})

	TO-46, no heat sink	TO-46*, small heat fin	TO-92, no heat sink	TO-92**, small heat fin	SO-8 no heat sink	SO-8** small heat fin	TO-220 no heat sink
Still air	400°C/W	100°C/W	180°C/W	140°C/W	220°C/W	110°C/W	90°C/W
Moving air	100°C/W	40°C/W	90°C/W	70°C/W	105°C/W	90°C/W	26°C/W
Still oil	100°C/W	40°C/W	90°C/W	70°C/W			
Stirred oil	50°C/W	30°C/W	45°C/W	40°C/W			
(Clamped to metal, Infinite heat sink)	(24°C/W)				(55°C/W)		

*Wakefield type 201, or 1" disc of 0.020" sheet brass, soldered to case, or similar.
**TO-92 and SO-8 packages glued and leads soldered to 1" square of 1/16" printed circuit board with 2 oz. foil or similar.