UNIVERSIDADE LUTERANA DO BRASIL

PRÓ-REITORIA DE GRADUAÇÃO

DEPARTAMENTO DE ENGENHARIA ELÉTRICA

ADILSON R. DOS SANTOS

SISTEMA DE PESAGEM COM INTERFACE WEB EM PLATAFORMA LINUX EMBARCADA

Canoas, Julho de 2010

ADILSON R. DOS SANTOS

SISTEMA DE PESAGEM COM INTERFACE WEB EM PLATAFORMA LINUX EMBARCA

Trabalho de Conclusão de Curso apresentado ao Departamento de Engenharia Elétrica da ULBRA como um dos requisitos obrigatórios para a obtenção do grau de Engenheiro Eletricista

Departamento:

Engenharia Elétrica

Área de Concentração

Processadores

Professor Orientador:

MSc. Eng. Eletr. Augusto Alexandre Durgante de Mattos – CREA-RS: 088003-D

Canoas

2010

FOLHA DE APROVAÇÃO

Nome do Autor: Adilson R. dos Santos

Matrícula: 021010659-0

Título: Sistema de pesagem com interface *WEB* em plataforma linux embarcada.

Trabalho de Conclusão de Curso apresentado ao Departamento de Engenharia Elétrica da ULBRA como um dos requisitos obrigatórios para a obtenção do grau de Engenheiro Eletricista

Professor Orientador:

MSc. Eng. Eletr. Augusto Alexandre Durgante de Mattos

CREA-RS: 088003-D

Banca Avaliadora:

MSc. Eng. Eletr. Dalton Luiz Rech Vidor

CREA-RS: 79.005-D

Conceito Atribuído (A-B-C-D):

MSc. Eng. Eletr. André Luis Bianchi

CREA-RS: 089197

Conceito Atribuído (A-B-C-D):

Assinaturas:

Autor Adilson Ribeiro dos Santos	Orientador Augusto A. Durgante. de Mattos
Avaliador Dalton Luiz Rech Vidor	Avaliador André Luis Bianchi

Relatório Aprovado em:

DEDICATÓRIA

Dedico para minha mãe e para o meu pai.

Dedico a minha esposa e ao meu filho.

AGRADECIMENTOS

Para viabilidade de qualquer conquista em nossas vidas, passamos pela vontade de Deus e assim fica o primeiro agradecimento e o mais importante.

Agradeço muito as pessoas mais importantes em minha vida, minha mãe Norci e ao meu pai Claudio, por compartilharem as mesmas horas que dedico na busca de um sonho.

A Gisele, minha esposa, por acreditar que é possível vencer toda e qualquer dificuldade para alcançar os objetivos.

Ao meu filho Vicente, que está prestes a nascer e que fez revigorar as forças para o andamento dos meus projetos de vida.

Ao meu irmão, Régis, pelo incentivo, apoio e exemplo.

Ao colega Fábio Júnior, por estar junto nas mesmas angústias, anseios, desafios e vitórias nesse período acadêmico.

EPÍGRAFE

Nunca esquecerei,

os dias que passei...

RESUMO

SANTOS, Adilson Ribeiro dos. Sistema de Pesagem com Interface *Web* em Plataforma Linux Embarcada. 80f. Trabalho de Conclusão de Curso em Engenharia Elétrica - Departamento de Engenharia Elétrica. Universidade Luterana do Brasil. Canoas, RS. Ano do Trabalho.

O trabalho descreve o uso de uma plataforma com sistema operacional embarcado para aplicações eletrônicas. Ter uma interface na *web*, que possa interagir com um protótipo, permitindo a calibração e coleta de medidas de peso é o objetivo principal desse projeto. O sistema efetua medições e permite via navegador de internet à parametrização e visualização gráfica dos valores obtidos, os testes comprovaram isso. Conclui-se que o fato de ter toda a informação junto com o *hardware* torna-se um diferencial importante em aplicações eletrônicas, pois no computador externo não há necessidade de instalação de *software* de controle, pois qualquer computador em rede consegue acessar o equipamento e é dentro dele que está a interface disponibilizada na *web*.

Palavras chave: Processador. Linux. *WEB*. Célula de Carga.

ABSTRACT

SANTOS, Adilson Ribeiro dos. Weighing System with *Web* Interface in Embedded Linux. 80 p. Work of Conclusion of Course in Electrical Engineering - Electrical Engineering Department. Lutheran University of Brazil. Canoas, RS. 2010.

The paper describes the use of an embedded operating system platform for electronic applications. Having a web interface, which allows one to interact with a prototype, allowing the calibration and weight measure collecting is the main objective of this project. The system effectuates measurement and allows through the internet navigator the parameterization and graphic visualization of the values obtained, the tests prove that. It is concluded that the fact that having all the information that the fact of having all the information with the hardware becomes an important differential on electronic applications, as in the external computer there is no need of installing controlling software, since any network computer is accessible the equipment and the available interface in the web is inside it.

Key Words: Processor, Linux, Web, Charging cell.

LISTA DE ILUSTRAÇÕES

FIGURA 2.1 – CÉLULA DE CARGA TIPO VIGA....4
FIGURA 2.2 – STRAIN GAGE....5
FIGURA 2.3 – PONTE DE WHEATSTONE....7
FIGURA 2.4 – MÓDULOS BÁSICOS DE UM SISTEMA DE INSTRUMENTAÇÃO....9
FIGURA 2.5 – APLICAÇÃO DO CONVERSOR ANALÓGICO PARA DIGITAL....9
FIGURA 2.6 – PLATAFORMA DE PROCESSAMENTO....10
FIGURA 2.7 – ARQUITETURA DO PROCESSADO POWERPC MPC850 [7]....11
FIGURA 2.8 – DIAGRAMA DE BLOCOS DA SERIAL DA POWERPC....13
FIGURA 2.9 – SINAIS DISPONÍVEIS NOS CONECTORES DA POWERPC....13
FIGURA 2.9 – O.S. EMBARCADOS MAIS UTILIZADOS X TENDÊNCIA DE UTILIZAÇÃO DO LINUX....14
FIGURA 3.1 – VISÃO GERAL DO SISTEMA....20
FIGURA 3.2 – DETALHES DA POWERPC....22
FIGURA 3.3 – REGULADOR DE TENSÃO 3,3 VOLTS....23
FIGURA 3.4 – CIRCUITO MAX232 PARA COMUNICAÇÃO SERIAL....23
FIGURA 3.5 – TERMINAL DE OPERAÇÃO DA POWERPC....24
FIGURA 3.6 – CIRCUITO DE MEDIÇÃO DO PESO....26
FIGURA 3.7 – ALGORITMO DE CONVERSÃO SERIAL PARA SPI....27
FIGURA 3.8 – SERIAL SPI CLOCK E DADO....28
FIGURA 3.9 – INICIALIZAÇÃO DO CONVERSOR AD....30
FIGURA 3.10 – COMPILAÇÃO DO WEBSERVER....31
FIGURA 3.11 – PASSOS PARA UMA PÁGINA DE TESTE....32
FIGURA 3.12 – CICLO DE TROCA DE INFORMAÇÕES....33
FIGURA 3.13 – NAVEGADOR COM BOTÃO PARA CHAMAR SCRIPT....34
FIGURA 3.14 – SCRIPT COM A DECLARAÇÃO DAS VARIÁVEIS DE AMBIENTE....34
FIGURA 3.15 – RESULTADO DO COMANDO DE TESTE....35
FIGURA 4.1 – COMANDOS BÁSICOS DE CONEXÃO COM A REDE....36
FIGURA 4.2 – BLOCO DE MEDIÇÃO....37
FIGURA 4.3 – TEMPOS DE ACIONAMENTO POWERPC E 89C2051....37
FIGURA 4.4 – COMANDO TELNET PARA ACESSO AO EQUIPAMENTO....38
FIGURA 4.5 – ETAPA DE INICIO E FIM DE UMA CALIBRAÇÃO....38
FIGURA 4.6 – PROCEDIMENTO DE CALIBRAÇÃO....39
FIGURA 4.7 – PESO PADRÃO TESTADO TAMBÉM EM UMA BALANÇA COMERCIAL....40

FIGURA 4.8 – CONDIÇÃO DE TESTE.....40
FIGURA 4.9 – PERCENTUAL DE ERRO MÉDIO ENTRE DEZ MEDIDAS.....41
FIGURA 4.10 – TELA DE ENTRADA.....42
FIGURA 4.11 – TELA UM DO SISTEMA WEB.....43
FIGURA 4.12 – TELA DOIS DE INTERFACE WEB.....43
FIGURA 4.13 – TELA TRÊS DE INTERFACE WEB.....44
FIGURA 4.14 – TELA QUATRO DE INTERFACE WEB.....44
FIGURA 4.15 – TELA CINCO DE INTERFACE WEB.....45
FIGURA 4.16 – TELA SEIS DE INTERFACE WEB.....45
FIGURA 4.17 – TELA SETE DE INTERFACE WEB.....46
FIGURA 4.18 – SISTEMA COMPLETO FUNCIONANDO.....46

LISTA DE TABELAS

TABELA 2-1 – SINAIS DEFINIDOS PELO PADRÃO RS – 232.....12

TABELA 3-2 – COMANDO ACEITO PELO CONVERSOR SERIAL.....28

TABELA 3-3 – COMANDO LEITURA DO AD.....28

TABELA 3-4 – COMANDO DE RETORNO CONTENDO A LEITURA DO AD.....29

TABELA 4-5 – COLETA DE DEZ MEDIDAS DE CADA PESO E CÁLCULO DO ERRO EM PERCENTUAL.....41

LISTA DE ABREVIATURAS E SIGLAS

AD – Analógico / Digital

UART – Universal Asynchronous Receiver/Transmitter

MPC850 – Processado PowerPC da Motorola

TXD – Sinal de transmissão serial

RXD – Sinal de recepção serial

RS-232 – Padrão para troca de informações seriais

TTL – Lógica Transistor-Transistor

SPI – Serial Peripheral Interface

I/O – Entrada/saída

GCC – GNU Compiler Collection

RJ-45 – Conector modular utilizado em redes

NFS – Network File System

RFC – Request for Comments

HTTP – Hypertext Transfer Protocol

DNS – Domain Name System

LISTA DE SÍMBOLOS

Hz - Hertz
Mb - Megabit
A - Ampère
V – Volts
W – Watts

SUMÁRIO

1. INTRODUÇÃO.....1
2. REFERENCIAL TEÓRICO.....4
2.1. Células de carga.....4
2.2. Condicionamento de sinais.....8
2.3. Conversor de leitura da tensão da célula de carga.....9
2.4. Processador PowerPC.....10
2.5. Sistemas linux embarcado.....14
2.6. TCP/IP.....16
2.7. Web/Webserver.....18
3. MATERIAIS E MÉTODOS.....20
3.1. Descrição Geral do Sistema.....20
3.2. Descrição dos Sistemas Eletroeletrônicos.....21
3.3. Descrição do bloco de medição.....25
3.4. Descrição dos Sistemas Computacionais.....26
3.5. Interface WEB.....30
4. APRESENTAÇÃO E DISCUSSÃO DOS RESULTADOS.....36
4.1. PowerPC.....36
4.2. Bloco de medição.....36
4.3. Aplicação preliminar acessada por telnet.....37
4.4. Aplicação principal e interface web.....41
4.5. Visão completa do sistema.....46
5. CONSIDERAÇÕES FINAIS.....47
6. REFERÊNCIAS.....49
OBRAS CONSULTADAS.....50
GLOSSÁRIO.....51
APÊNDICE A – FIRMWARE APLICAÇÃO WEB.....52
ANEXO A – CONCEITOS BÁSICOS DE LINUX.....62

1. INTRODUÇÃO

Os dispositivos de automação na indústria evoluem rapidamente na direção da maximização das funcionalidades. Atualmente, o crescimento de dispositivos embarcados está remodelando as aplicações dos projetos eletrônicos. Hoje, até mesmo nas residências já é possível encontrar dispositivos com processamento que realizam as mesmas funcionalidades de um computador de mesa. Em 2009 o mercado cresceu 10% em relação a 2008 e este crescimento chega a representar o dobro do que se comercializava em 2004 [1].

Para complementar e tornar mais poderoso o desenvolvimento de projetos eletrônicos a tecnologia embarcada hoje faz uso de um sistema operacional instalado em sua central de processamento de dados (CPU). Para essa finalidade os sistemas têm características reduzidas, mas com as principais funcionalidades ativadas.

O uso do sistema operacional linux para ser esse sistema nos dispositivos embarcados vem cada vez mais conquistando adeptos. Segundo a projeção do site **linuxdevices**, em 2010 o linux estará rodando em 60% dos sistemas embarcados produzidos mundialmente. Além disto, existe uma estimativa que o crescimento do uso em linux em sistemas embarcados **cresça 278%** em relação aos projetos do passado.

A grande tendência de migração para o *software* livre é evidente, todas as características que o acompanham junto da evolução do sistema nos últimos anos tornam interessante esse processo.

Outro mecanismo auxiliar em projetos eletrônicos é a monitoração do sistema por meio de um servidor *web*. A interatividade através de sistemas *web* revolucionou o mundo, o fato de poder conectar as pessoas em uma rede interligada de dados, impulsiona as tecnologias de comunicação, divulgação ou busca de informações de interesse. Através de tecnologias *web*, qualquer empresa pode aparecer para o mundo, consultar e compilar informações sobre o

mercado, divulgar ou buscar produtos e serviços. O chamado mundo virtual veio para ficar e junto com ele, a demanda de serviços nesta área.

O objetivo do trabalho é fazer uso destas tecnologias para ter uma interface de controle na *web*, que possa interagir com um protótipo eletrônico, permitindo a calibração e coleta de medidas de peso. O projeto busca alternativas para obter através de uma rede comum de computadores as informações medidas, sendo possível efetuar o diagnóstico das grandezas envolvidas. Hoje sistemas de medição em sua maioria são engessados no sentido de haver necessidades de registros ou intervenções manuais que podem inserir o erro humano.

Os sistemas de medidas de peso em processos industriais são importantíssimos no que diz respeito a característica final do produto. Em processos automatizados medições erradas em dosagens de quantidade de material são traduzidas em prejuízo para empresa.

No comércio, existe a relação direta com o consumidor onde problemas de medida provocam o descontentamento do cliente. Até mesmo o empresário e ou gestor podem deixar passar despercebido detalhes nas medições, no funcionamento ou obter funcionalidades que poderiam lhe auxiliar no controle desse setor na empresa.

O projeto busca compartilhar as medições obtidas no sistema para os computadores da mesma rede, através de um navegador de internet, permitindo que as ocorrências possam ser constantemente monitoradas.

Tradicionalmente, trabalhar com sistemas embarcados envolve desenvolver todo o projeto baseando-se em linguagem de máquina ou na linguagem C. Para a maioria dos projetos não há o uso de um sistema operacional, quase sempre algum sistema é fornecido pelo fabricante como compiladores e etc.

O mercado a cada dia demanda produtos com mais funcionalidades e acompanhar essa demanda requer um ambiente mais produtivo de desenvolvimento onde o uso de tecnologias difundidas como USB, *Pendrive*, *Wireless*, Rede e GPRS são facilitadas quando se tem a possibilidade de trabalhar junto com um sistema operacional.

O desenvolvimento e estudo deste trabalho estão estruturados em quatro capítulos. O capítulo um introduz o leitor nas propostas do projeto. O capítulo

dois apresenta fundamentação teórica necessária para a realização do projeto, disposto da seguinte maneira: Células de carga; Condicionamento de sinais; Referência zero absoluto; PowerPC; Sistemas embarcados linux; Sistemas *web*.

No capítulo três são apresentados os detalhes do protótipo, condicionamento de sinal utilizado, estrutura de *hardware* de *firmware* e de *software*. No capítulo quatro são expostos os resultados e discussão. Por fim, no capítulo cinco são apresentadas às conclusões e sugestões para extensão do protótipo.

2. REFERENCIAL TEÓRICO

2.1. Células de carga

As células de carga são sensores de alta precisão que através das deformações estruturais, são capazes de mensurar grandezas como massa, pressão, força, torque, aceleração, etc. Sendo assim, podem ser classificadas como transdutores, mais precisamente, transdutor eletro-mecânico, transformando a grandeza física medida em tensão elétrica.

Seu funcionamento baseia-se na variação da resistência de um material metálico condutor (extensômetro) que é colado sobre a estrutura. Ao sofrer uma tensão mecânica, a estrutura em análise tende a se deformar dentro do regime elástico. Essa deformação é transformada em variação de resistência e que por fim é convertida em variação de tensão [2].

A figura 2.1 ilustra um modelo de célula de carga muito utilizado em diversas aplicações na indústria. Ex.: medidas de peso em silos de estocagem de arroz. Sua fabricação geralmente é de alumínio maciço, esse modelo é chamado de célula de carga tipo viga, com rasgos projetados para auxiliar na deformação e pontos de fixação de acordo com a aplicação.

Figura 2.1 – Célula de carga tipo viga.

2.1.1. Extensômetro – Strain Gage

Desde a invenção dos extensômetros também conhecidos como *strain gage*, há mais de 60 anos atrás, esta tecnologia vem sendo continuamente melhorada. Começando com o *strain gage* de fio único, uma enorme diversidade de tipos, para as mais variadas tarefas, foi e vem sendo desenvolvida..

A figura 2.2 mostra *strain gage* em seu formato mais comum, esse elemento resistivo, afixado sobre um corpo sólido através de convenientes técnicas, de tal modo que a resistência do elemento irá variar assim que a superfície ao qual está fixado, deformar.

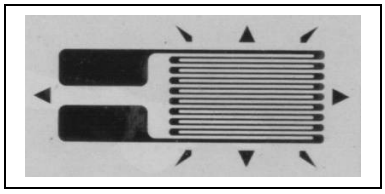

Figura 2.2 – Strain Gage

O extensômetro, portanto, responde à deformação superficial exercida na estrutura e em usos normais, uma conveniente estrutura elástica deve ser de material cuidadosamente escolhido [3].

2.1.2. Fatores favoráveis

- Pequenas dimensões e pouca massa aliadas a uma elevada rigidez do material utilizado na confecção da célula, proporcionam uma elevada frequência de ressonância, permitindo operação numa larga faixa de frequência.
- Excelente linearidade sobre uma larga faixa de tensão. Em geral a linearidade é limitada pelas características do material em que está aplicado e da geometria do transdutor.
- Altamente estável com o tempo. Desde que protegidos contra a agressividade do meio, sua calibração permanece inalterável por muito tempo.
- Custo relativamente baixo. Por isso um dos principais fatores de sua utilização.
- Simplicidade do circuito de saída. Tais extensômetros podem ser utilizados em sistemas tanto de excitação em corrente alternada, quanto de excitação em corrente contínua, pois são sensíveis à frequência da tensão de alimentação. Em geral operam na configuração de ponte de *Wheatstone*.

- Medições dinâmicas podem ser facilmente executadas, sendo que a frequência máxima que pode ser medida geralmente é determinada antes pelos equipamentos utilizados na célula de carga [3].

2.1.3. Fatores limitantes

Degradação térmica. Ao se empregar materiais orgânicos, tais como isolantes, adesivos e demais revestimentos de proteção, apesar da praticidade e embora facilmente utilizáveis, as aplicações em altas temperaturas ficam limitadas. Isto pelas próprias características destes materiais empregados, a máxima temperatura é em torno de 260ºC, que pode depender das especificações do material empregado, esquema de operação, tempo de exposição e etc.

Baixos sinais de saída. Devido aos baixos sinais de saída, as características de amplificação de sinal estável, controle das voltagens de excitação, ausência de ruídos, compensação de efeitos térmicos na variação de resistência dos *strain gages*, são essenciais para uma precisa leitura de saída.

Cuidados na instalação e utilização. Os *strain gages* são materiais muito sensíveis, portanto, sua instalação, manuseio e uso devem ser cuidadosos, seguindo-se um rigoroso controle.

Os *strain gages* devem ser bem protegidos, pois em caso contrário sua vida útil é tremendamente limitada. Sendo atingidos pela água são destruídos, oxidados ou degradados na sua performance, por está razão o transdutor deve ser hermeticamente selado [3].

2.1.4. Ligação dos extensômetros

Em 1843, o físico Sir Charles Wheatstone descobriu uma ponte para medição de resistências elétricas. Esta ponte, hoje conhecida como ponte de Wheatstone, é ideal para a medição de pequenas variações de resistência, como é o caso da variação de resistência de um extensômetro. Como já foi dito, um extensômetro transforma uma deformação, numa variação proporcional da sua resistência elétrica. A relação entre a deformação aplicada $(\varepsilon = \Delta L / L0)$ e a variação de resistência de um extensômetro é obtida pela equação abaixo:

$$\frac{\Delta R}{R0} = K\varepsilon$$

Onde: ε - Valor de medida da extensão; ΔL - Variação da distância; $L0$ - Distância entre os pontos a serem medidos; ΔR - Variação da resistência por deformação; $R0$ - Resistência inicial do extensômetro; K - Fator do extensômetro, calculado experimentalmente.

A ponte de Wheatstone está representada esquematicamente na Figura 2.3.

Os quatro braços da ponte contêm quatro resistências, onde cada uma é constituída por um extensômetro, *strain gage*, (R1) à (R4). Se os nós (2) e (3) forem ligados a uma fonte com tensão conhecida (VE), aparecerá uma outra diferença de potencial (VA), entre os nós (1) e (4). O valor de (VA) depende dos quocientes entre resistências (R1; R2) e (R3; R4). Tem-se então a equação abaixo:

$$\frac{VA}{Ve} = \frac{R1}{R1 + R2} - \frac{R4}{R3 + R4} = \frac{R1R3 - R2R4}{(R1 + R2)(R3 + R4)}$$

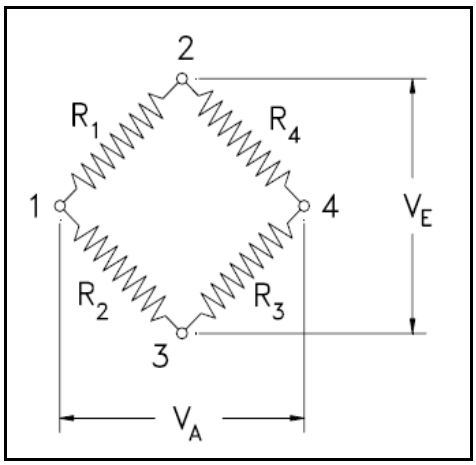

Figura 2.3 – Ponte de Wheatstone

A ponte de Wheatstone diz-se equilibrada quando se tem: $VA / VE = 0$.

Para a ponte ser equilibrada é necessário que se verifique:

$$R1 = R2 = R3 = R4 \text{ ou então } \frac{R1}{R2} = \frac{R4}{R3}$$

Partindo então do princípio que uma dada ponte de Wheatstone está equilibrada, qualquer variação de resistência em uma ou mais resistências da ponte, provocará uma diferença de potencial (VA) diferente de zero. Se assumir, além disso, que a variação de resistência (Δ Ri) é muito inferior à própria resistência (Ri), o que em geral é sempre válido, temos a seguinte relação [4].

$$\frac{\mathrm{VA}}{\mathrm{VE}} = \frac{1}{4}\left(\frac{\Delta R1}{R1} - \frac{\Delta R2}{R2} + \frac{\Delta R3}{R3} - \frac{\Delta R4}{R4}\right)$$

Ou em outra forma:

$$\frac{\mathrm{VA}}{\mathrm{VE}} = \frac{\mathrm{K}}{4}\left(\varepsilon 1 - \varepsilon 2 + \varepsilon 3 - \varepsilon 4\right)$$

2.1.5. Sensibilidade e precisão para as células de cargas

Alguns critérios devem ser utilizados na escolha de uma célula de carga, por exemplo, a capacidade nominal. A força máxima que ela deverá medir (OS fatores de segurança, 50% de sobrecarga contra danos de funcionamento e 300% para a ruptura, são intrínsecos a própria célula).

Sensibilidade

A sensibilidade na medição do desbalanceamento da ponte de Wheatstone é feita através da variação da tensão de saída em função da tensão de excitação aplicada na entrada da ponte. Quando a célula de carga esta carregada, este valor é dado em milivolt por volt aplicado e, normalmente, entre 2 e 3 mV/V. Isto significa que uma célula de carga de 30kg de capacidade nominal e 2mV/V de sensibilidade, com uma tensão de excitação na entrada de 10 V, quando sujeita a uma força de 30Kg apresentará na saída uma variação de tensão de 20mV.

Precisão

A precisão é o erro máximo admissível relacionado em divisões da capacidade nominal. As células de carga neste caso podem ser divididas em:

Baixa precisão: até 1.000 divisões (ou 0,1% da capacidade nominal)

Média precisão: de 3.000 a 5.000 divisões (ou 0,03 a 0,02% da capacidade nominal)

Alta precisão: 10.000 divisões (ou 0,01% da capacidade nominal)

2.2. Condicionamento de sinais

Os módulos básicos compreendem o transdutor ou sensor, a unidade de tratamento de sinal e o dispositivo mostrador, figura 2.4. O transdutor é o módulo que gera um sinal de medição geralmente proporcional ao valor do mensurando. O sensor é o primeiro estágio do transdutor, é a parte deste que é diretamente afetada pelo mensurando. A unidade de tratamento de sinais

processa o sinal de medição do transdutor/sensor e amplifica, atenua, filtra, digitaliza, armazena, ou seja, trata o sinal para que o mesmo possa ser mais bem apresentado à etapa seguinte. O dispositivo mostrador é o módulo que torna o sinal de medição perceptível ao usuário [5].

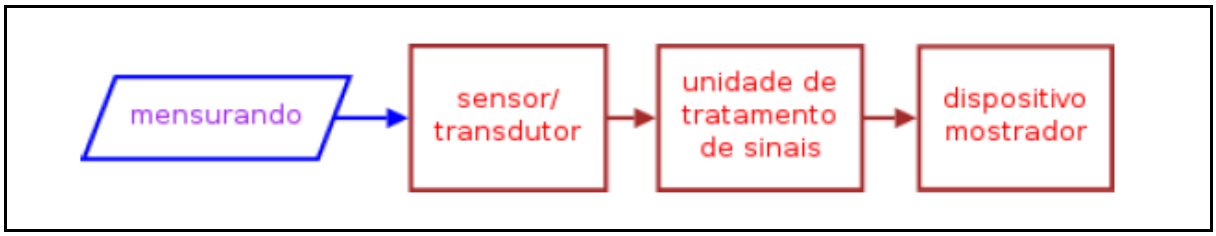

Figura 2.4 – Módulos básicos de um sistema de instrumentação

2.3. Conversor de leitura da tensão da célula de carga

Para interpretar a tensão lida da célula de carga, foi utilizado o conversor analógico para digital CS5531, suas características são apropriadas para esse tipo de aplicação possuindo dezesseis bits para converter o valor analógico lido em uma das suas duas entradas.

Trata-se um de conversor AD que se comunica com o processador através de uma saída serial SPI (*Clock*/Dado) como ilustram os pinos de 11 à 14 da figura 2.5. Necessita de um cristal de 4.9152 Mhz e pode ser alimentado de 2,5 à 5 volts, maiores detalhes para sua configuração e modo de trabalho serão abordados no capítulo quatro ou podem ser vistos no datasheet do conversor na página do fabricante *Cirrus Lógic* [6].

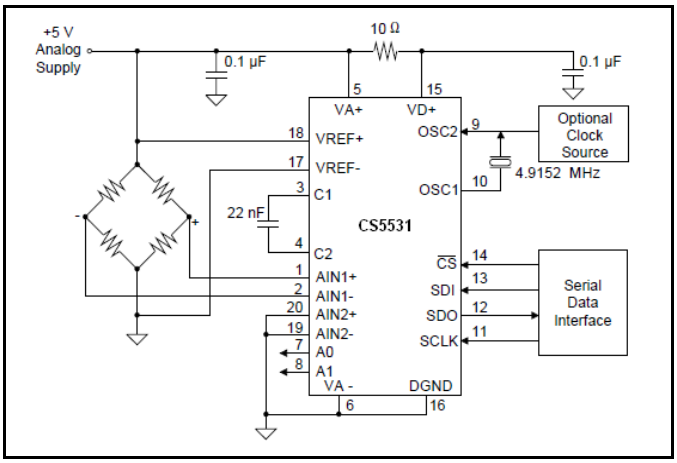

Figura 2.5 – Aplicação do conversor analógico para digital

2.4. Processador PowerPC

O sistema contempla o processador PowerPC MPC850 da Motorola. Com tecnologia RISC, ele opera a uma freqüência de 48MHz. Por ser versátil, pode ser usado em uma variedade de aplicações de controle, destacando-se em particular no domínio das comunicações e produtos que necessitem de suporte a rede para acessos remotos [7].

Esse processador faz parte de uma plataforma desenvolvida pela empresa digicon S/A e tem características de *hardware* específicas onde é encapsulado em uma só placa o processador, memória e os pinos de entrada e saída com níveis de tensão de 3,3 volts, Figura 2.6 – Plataforma de processamento.

Esse conjunto é responsável pelo processamento, sendo nele que o sistema operacional pode ser executado, porém se resume a essa função se analisado em separado. Essa unidade não é capaz de efetuar nenhum acionamento ou interagir com o mundo externo sem que se agregue a interface eletrônica para o fim que se deseja trabalhar.

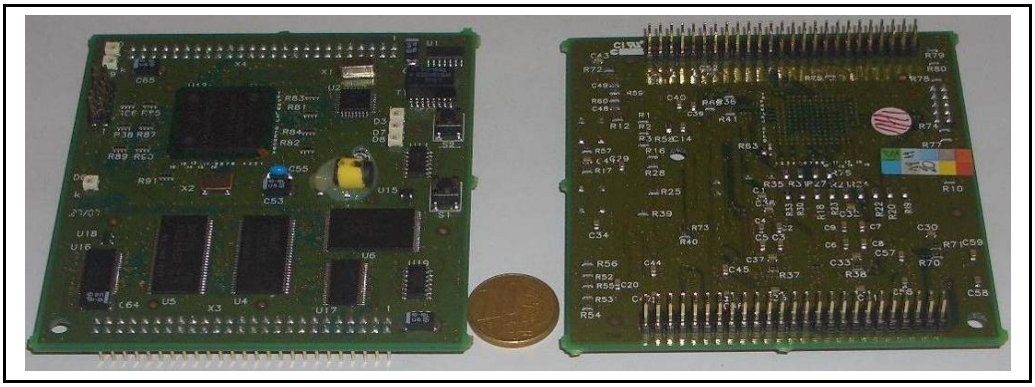

Figura 2.6 – Plataforma de processamento

Essa plataforma faz uso de dois tipos de memória, a do tipo *flash* e do tipo SDRAM. A memória *flash* serve para armazenar os códigos do *bootloader* e do sistema operacional, bem como informações de configuração e eventos, com uma capacidade de 8Mb. A memória SDRAM, com capacidade de 32Mb é utilizada para executar o código, armazenar dados e acomodar a pilha do sistema operacional e da aplicação existentes. Na figura 2.7 é possível visualizar como é formada a arquitetura interna do processador e algumas características particulares [8].

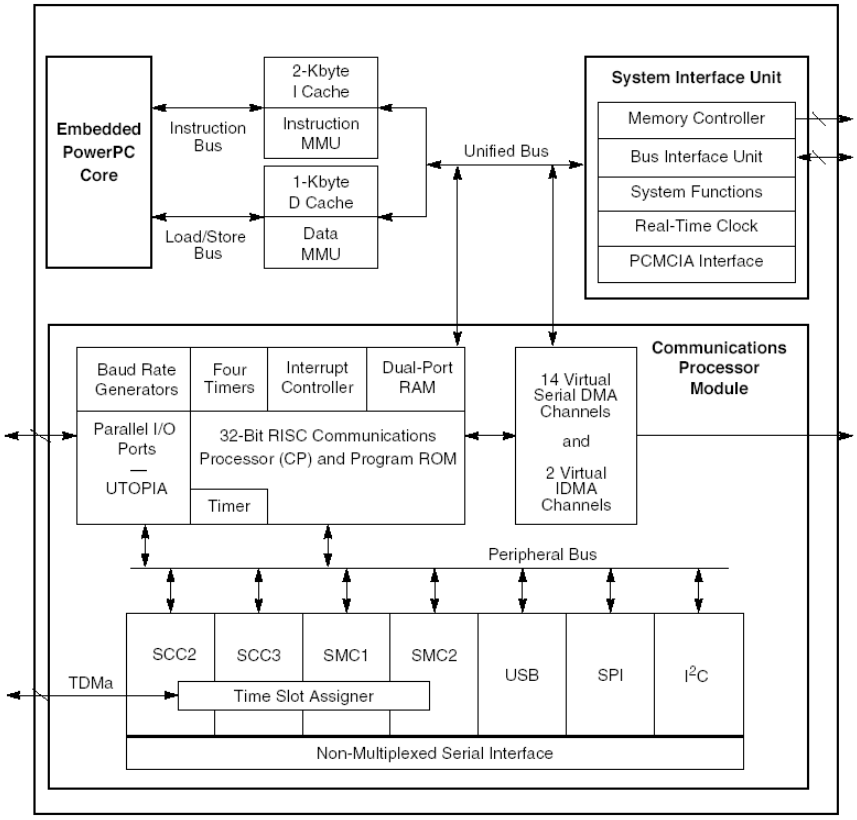

Figura 2.7 – Arquitetura do processado PowerPC MPC850 [7]

Um grande diferencial da PowerPC é uma porta de comunicação *Ethernet,* especialmente considerando-se a utilização de sistemas operacionais elaborados como o Linux. Ela pode ser implementada com o SCC2 do processador MPC850 junto de um regulador e um transformador de isolação. O conector RJ-45, necessário para a conexão física com os cabos é implementado em separado.

Devido às limitações do SCC do processador, a velocidade da porta *Ethernet* está limitada a 10Mbps. Entretanto a comunicação em rede *Ethernet* 10/100Mbps é plenamente possível quando se usar o cabo de rede com a ligação *crossover*.

Outro modo de comunicação são as portas seriais, uma UART simples pode ser implementada com o SMC2 do processador MPC850. Os SMC são controladores seriais simples por definição, e que implementam apenas os protocolos UART e transparente. Pinos de transmissão de dados (TXD) e de recepção de dados (RXD) existem para cada SMC e a operação no modo UART é *full-duplex*. Os sinais de TXD e RXD referentes ao SMC2 são disponibilizados no barramento de expansão em nível lógico 3.3V e sem compatibilidade com o

padrão TTL de 5V. Isto implica na obrigatoriedade do uso de reguladores de 3.3V na placa mãe [8].

O padrão RS-232 foi criado pela EIA (*Electronic Industries Association*) nos anos 60 para padronizar a comunicação entre computador e modem, tornando-se um padrão de fato para a comunicação serial assíncrona. Este padrão nomeia o computador como DTE (*Data Terminal Equipment* – Equipamento Terminal de Dados) e o modem como DCE (*Data Communications Equipment* – Equipamento de Comunicação de Dados). Apesar de não definidos pela norma, os conectores DB-9 e DB-25 foram consagrados pela utilização. Os sinais definidos pelo padrão RS-232, seus respectivos nomes e sentidos são mostrados na Tabela 2 –1.

Tabela 2-1 – Sinais definidos pelo padrão RS – 232.

Sinal	**Direção (DTE)**	**Direção (DCE)**	**Pino DB-9**	**Pino DB-25**	**Nome do Sinal**
DCD	Entrada	Saída	1	8	*Carrier Detect*
RxD	Entrada	Saída	2	3	*Receive Data*
TxD	Saída	Entrada	3	2	*Transmit Data*
DTR	Saída	Entrada	4	20	*Data Terminal Ready*
GND	-	-	5	7	*Ground*
DSR	Entrada	Saída	6	6	*Data Set Ready*
RTS	Saída	Entrada	7	4	*Request To Send*
CTS	Entrada	Saída	8	5	*Clear To Send*

O *driver* de interface serial do Linux contém funções de configuração dos registradores e manipulação dos dados da interface física. O mapa de registradores do controlador de periféricos é acessado através de uma estrutura contendo funções de inicialização, configuração e acesso aos dados comunicados. Esse driver está localizado em <linux>/arch/ppc/8xx_io/uart.c.

As interfaces seriais da plataforma podem ser acessadas pelo sistema Linux usando os comandos de manipulação de arquivos como open(), read(), write() e close(). [8]. A figura 2.8 ilustra por blocos a serial junto com a PowerPC.

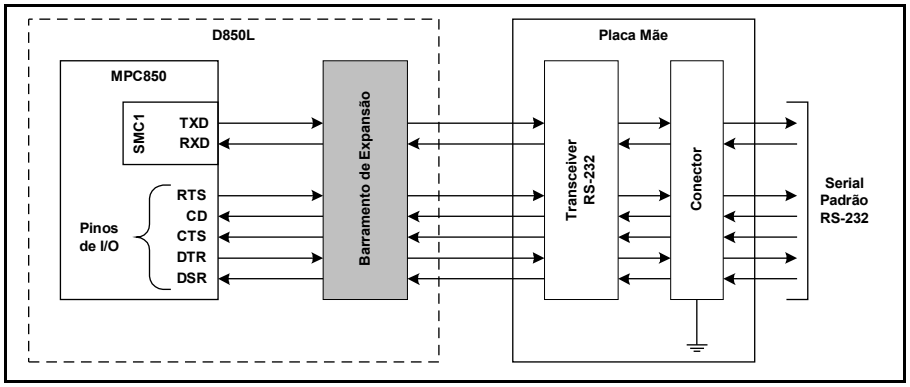

Figura 2.8 – Diagrama de blocos da serial da PowerPC.

Todos os pinos disponibilizados no barramento de expansão podem ser utilizados também como pinos de I/O. O uso de um pino com função dedicada a um periférico ou como I/O de uso geral é programável via registradores internos do MPC850. Do ponto de vista lógico, o MPC850 disponibiliza 4 portas de I/O: Porta A de 10 bits, Porta B de 14 bits, Porta C de 12 bits e Porta D de 13 bits. A porta D está reservada para I/O e as demais são multiplexadas com os periféricos integrados.

Toda a alimentação necessária deverá ser provida pela placa mãe via conector do barramento de expansão. A única tensão requerida é 3.3V, com consumo máximo de 1.5A, levando a uma potência máxima de 5W. A tolerância na alimentação é de ±5%.

Para conectar-se a placa mãe, a plataforma disponibiliza dois conectores tipo barra de pinos, com 50 pinos cada, organizados em 25 pinos por 2 fileiras. Um diagrama esquemático destes conectores é mostrado na Figura 2.9.

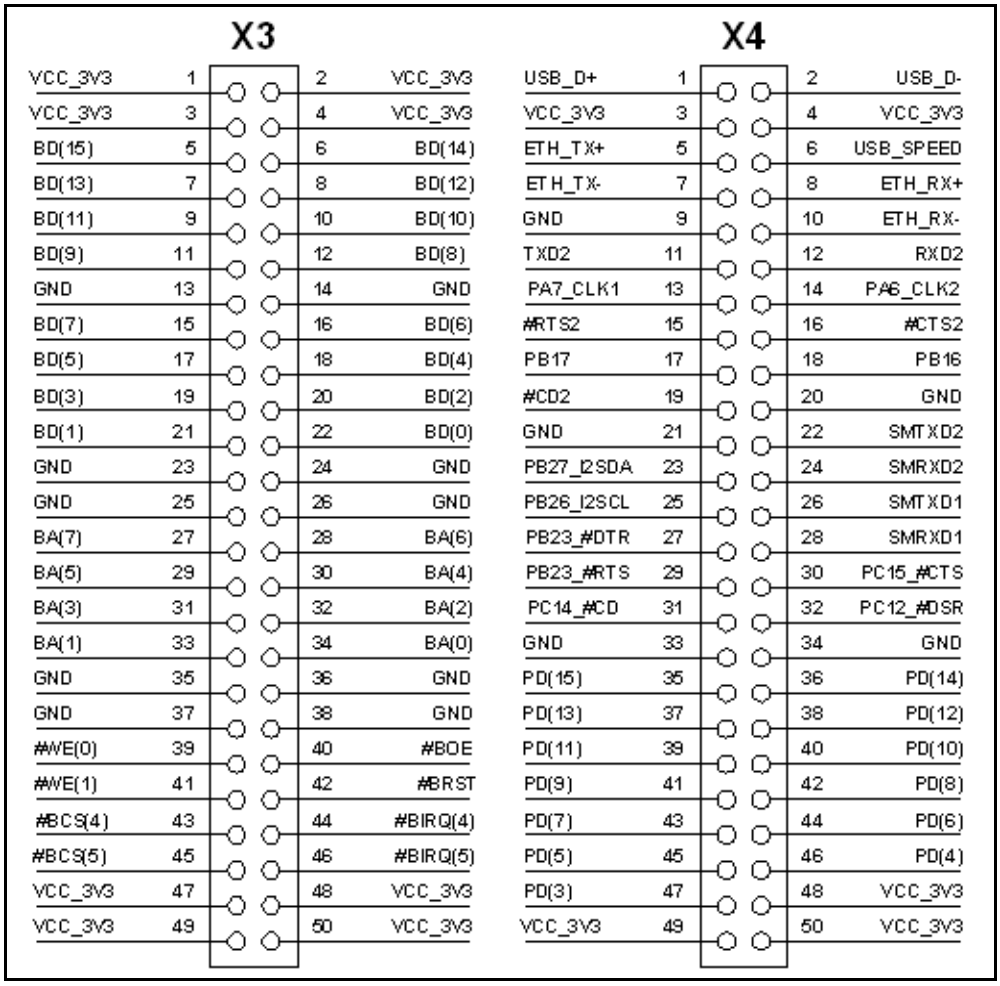

Figura 2.9 – Sinais disponíveis nos conectores da PowerPC

Para trabalhar com a PowerPC existe um *firmware* de *boot* que é capaz de carregar e gravar na memória *flash* uma imagem do sistema operacional Linux. O *firmware* utilizado para esta função, baseado em *software* livre, é o ppcboot desenvolvido pela *Denx Software Engineering* da Alemanha [8].

O *ppcboot* é distribuído na forma de código fonte, e precisa ser adaptado e compilado novamente para qualquer novo módulo de *hardware*. Vários exemplos, relativos a placas PowerPC disponíveis no mercado, são distribuídos junto com o código fonte do *ppcboot*, permitindo que os desenvolvedores de novas plataformas possam gerar novas versões deste programa para atender a seus projetos. A versão em uso do *ppcboot*, liberada em 1º de novembro de 2003, é a 2.0.0.

O compilador utilizado para gerar uma imagem do PPCBoot é o mesmo utilizado para compilar o *kernel* Linux e as aplicações para PowerPC, ou seja, um GCC para powerpc-linux. A Denx disponibiliza uma versão binária deste tipo de compilador, incluindo bibliotecas e diversos aplicativos já portados, que executa em máquinas hospedeiras Linux x86 e que gera código para powerpc-linux. O pacote chama-se ELDK e pode ser encontrado na página da Denx na Internet. [8].

2.5. Sistemas linux embarcado

No mercado exterior o Linux já está bem consolidado, porém no mercado brasileiro ainda está começando a ser alvo de interesse das empresas. Grandes fabricantes de microprocessadores para sistemas embarcados estão incentivando a utilização do Linux nestes sistemas no Brasil [9].

Na figura 2.10 está uma demonstração de utilização do linux no mundo até 2007 e o estudo de tendência de utilização desse sistema para o futuro. Essa pesquisa foi desenvolvida pela *linuxdevices.com*.

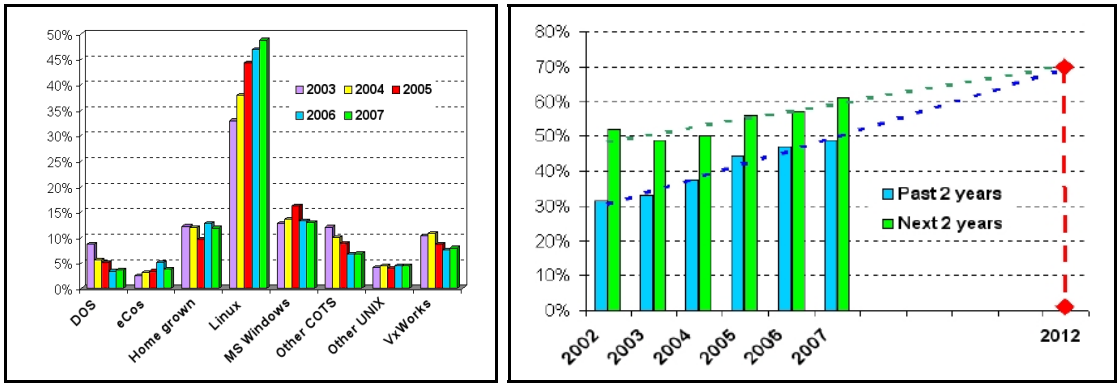

Figura 2.9 – O.S. embarcados mais utilizados x Tendência de utilização do Linux

2.5.1. O que é o linux?

O Linux foi criado inicialmente como um *hobby*, por Linus Torvalds, para aumentar as funcionalidades de seu computador 80386 PC. Em 5 de outubro de 1991 circulava pela Internet a seguinte mensagem assinada por Linus: "*...Como eu mencionei há um mês, estou trabalhando em uma versão free de um sistema semelhante ao Minix para computadores AT-386. Ele já alcançou o estágio de ser usável (embora possa não ser, dependendo do que você quer fazer), e pretendo distribuir o código fonte. É apenas a versão 0.02..., mas já consegui rodar o bash, gcc, gnu-make, gnu-sed, compress, etc., nele*". A partir daí seu código se espalhou rapidamente pela Internet, caindo na mão de várias pessoas que se interessaram e deram incentivos ao projeto. Provavelmente Linus jamais imaginou que em menos de 10 anos depois atingiria 10 milhões de seguidores [9]. Uma abordagem mais detalhada sobre esse sistema operacional e sua estrutura básica pode ser visto no anexo A.

2.5.2. Linux Embarcado X Linux Comum

A narrativa de GARBELLINI (2006) diz que, embora o Linux que roda em um PC seja essencialmente o mesmo usado em um sistema embarcado, este último não poderá utilizar as mesmas funcionalidades de um sistema de propósito comum que esbanja recursos. Muitos serviços e aplicativos acabam sendo desnecessários para o propósito esperado e devem ser removidos.

Na interface com o usuário de um sistema comum, são disponibilizados diversos aplicativos de acesso a Internet, ferramentas de escritório, servidores *web*, manipulação de imagens, multimídia, etc. Já em sistemas embarcados o sistema provavelmente terá uma interface personalizada para atender as especificações do projeto.

O *kernel* do Linux é um ponto fundamental a ser destacado. Em um sistema comum o *kernel* é muito completo, tem suporte a inúmeras funcionalidades, pode carregar *drivers* da maioria dos dispositivos de *hardware* existentes, características que são desnecessárias em um sistema embarcado. Para estes sistemas ele deve ser configurado para se tornar bem "leve", mantendo apenas as características necessárias e os *drivers* dos dispositivos que estão presentes no equipamento. Assim, se pode partir de um *kernel* comum e configurá-lo com as opções necessárias antes da compilação ou admitir um *kernel* específico para sistemas restritos como o uClinux, que já está bem próximo do que deseja o desenvolvedor [9].

2.6. TCP/IP

De maneira resumida, neste item é apresentada uma visão de modelos de comunicação TCP/IP para que seja fácil relacionar o que será abordado durante o desenvolvimento do próximo capítulo. Hoje, quando se menciona TCP/IP, vem imediata a associação com a internet, ocorrendo de modo idêntico o inverso: a internet está diretamente relacionada à arquitetura TCP/IP [10].

2.6.1. Endereçamento

O endereçamento de datagramas no modelo TCP/IP é implementado pela camada de rede (IP). Uma das informações de controle do datagrama é o endereço IP do destinatário e do emitente.

O endereço IP é formado por um número de 32 bits no formato nnn.nnn.nnn.nnn onde cada nnn pode variar de 0 até 255 (1 octeto = 8 bits). Os endereços possuem uma classificação que varia de acordo com o número de sub-redes e de *hosts*. Tal classificação tem por finalidade otimizar o roteamento de mensagens na rede.

Os endereços são fornecidos por uma entidade central: NIC (Network Information Center) e devem ser únicos para cada estação (*host*). Para o usuário dos serviços de rede, há uma forma mais simples de endereçamento onde cada computador irá receber um IP válido do servidor [10].

2.6.2. TCP (Transmission Control Protocol)

É o protocolo TCP que faz a comunicação fim-a-fim da rede. É orientado à conexão e altamente confiável independente da qualidade de serviços das sub-redes que servem de caminho. Para a confiabilidade de transmissão, garante a entrega das informações na seqüência em que lhe foi fornecida, sem perda nem duplicação.

Principais funções:

a. Transferência de dados — Através de mensagens de tamanho variável em full-duplex;

b. Transferência de dados urgentes — Informações de controle, por exemplo;

c. Estabelecimento e liberação de conexão — Antes e depois das transferências de dados, através de um mecanismo chamado three-way-handshake;

D. Multiplexação — As mensagens de cada aplicação simultânea são multiplexadas para repasse ao IP. Ao chegar ao destino, o TCP demultiplexa as mensagens para as aplicações destinatárias;

e. Segmentação — Quando o tamanho do pacote IP não suporta o tamanho do dado a ser transmitido, o TCP segmenta (mantendo a ordem) para posterior remontagem na máquina destinatária;

f. Controle do fluxo — Através de um sistema de buferização denominada janela deslizante, o TCP envia uma série de pacotes sem aguardar o reconhecimento de cada um deles. Na medida em que recebe o reconhecimento de cada bloco enviado, atualiza o buffer (caso reconhecimento positivo) ou reenvia (caso reconhecimento negativo ou não reconhecimento após um timeout);

g. Controle de erros — Além da numeração dos segmentos transmitidos, vai junto com o header uma soma verificadora dos dados transmitidos (checksum), assim o destinatário verifica a soma com o cálculo dos dados recebidos.

h. Precedência e segurança — Os níveis de segurança e precedência são utilizados para tratamento de dados durante a transmissão [10].

2.6.3. IP (Internet Protocolo)

A função básica do protocolo IP é o transporte dos blocos de dados por entre as sub-redes até chegar ao destinatário. Durante o tráfego pelas sub-redes, existem componentes denominados *gateways*, que desviam o datagrama IP para outras sub-redes ou para o destinatário, se este fizer parte da sub-rede a que o *gateway* está conectado.

Por limitação tecnológica, algumas sub-redes tem capacidade apenas para trafegar pacotes menores (volume de dados menor). Assim, o roteador fragmenta o datagrama original em datagramas menores, que serão restabelecidos futuramente quando possível [10].

2.6.4. Aplicações do protocolo TCP/IP

As aplicações, no modelo TCP/IP, não possuem uma padronização comum. Cada uma possui um RFC próprio. O endereçamento das aplicações é feito através de portas (chamadas padronizadas a serviços dos protocolos TCP), por onde são passadas as mensagens [10].

1. TELNET (Terminal Virtual)

É um protocolo que permite a operação em um sistema remoto através de uma sessão de terminal. Com isso, a aplicação servidora recebe as teclas acionadas no terminal remoto como se fosse local. Utiliza a porta 23 do TCP.

O TELNET oferece três serviços: Definição de um terminal virtual de rede, Negociação de opções (modo de operação, eco, etc.) e Transferência de dados.

2. NFS (Network File System)

O NFS supre uma deficiência do FTP que não efetua acesso on-line aos arquivos da rede.

O NSF cria uma extensão do sistema de arquivos local, transparente para o usuário, e possibilita várias funções como as seguintes:

a. Criação e modificação de atributos dos arquivos;

b. Criação, leitura, gravação, renomeação e eliminação de arquivos;

c. Criação, leitura e eliminação de diretórios;

d. Pesquisa de arquivos em diretórios;

e. Leitura dos atributos do sistema de arquivos.

Um dos problemas do NFS é que não suporta acesso compartilhado aos arquivos, portanto tais preocupações devem estar a cargo da aplicação.

O NFS utiliza várias rotinas de segurança para suprir a deficiência do protocolo.

2.7. Web/Webserver

Web pode ser caracterizado pelo conjunto dos hipertextos publicados na Internet acessíveis através do protocolo HTTP. A *Web* é basicamente constituída pelas páginas e ligações entre elas. O termo é usado tanto para designar o programa que disponibiliza conteúdos na *web* como o computador que o aloja.

Muitos sistemas embarcados conectados em rede de computadores funcionam como servidor *web*. As páginas de sistemas embarcados frequentemente precisam mostrar conteúdo dinâmico que podem mudar a cada vez que a página é acessada. Exemplos de conteúdos dinâmicos incluem monitoramento de sensores, informação de data e hora ou contador de visitas à página. Algumas páginas podem permitir que o visitante forneça entradas que retornem dados na própria página *Web*. Há várias maneiras para implementar conteúdo dinâmico em sistemas embarcados e o mais popular é o CGI (Commom Gateway Interface) [11].

Um navegador *Web* tal como o FireFox do Linux é uma aplicação cliente que usa HTTP para requisitar página *Web* para um servidor na Internet ou em uma rede local. Mesmo um pequeno dispositivo de sistema embarcado com memória bastante limitada pode hospedar uma página de texto e imagens, incluindo páginas que mostram dados em tempo real e interagem com o usuário.

Um navegador tem uma interface com o usuário para requisitar e mostrar páginas. O computador que requisita a página *Web* normalmente usa uma exibição em tela inteira. Para algumas aplicações, um sistema embarcado com capacidades limitadas de exibição de dados pode usar a função de cliente http para aproveitar os recursos do navegador. Para uma página simples até mesmo um texto de poucas linhas pode ser o suficiente.

Quando conectado à Internet, um servidor *Web* pode responder a qualquer navegador de qualquer computador também conectado ou mesmo ser programado para responder a um endereço IP específico. Um servidor *Web* em uma rede local pode responder a grupo de trabalhos ou qualquer computador da rede local.

3. MATERIAIS E MÉTODOS

As características para integrar o sistema eletrônico de medição de peso para que tenha um controle virtual é ilustrada neste capítulo. A figura 3.1 demonstra os requisitos para execução do sistema, onde no bloco de medição está contido o conversor AD e o conversor da serial SPI para RS-232 permitido a comunicação com a PowerPC. Na PowerPC estará o *firmware* e o *webserver* boa e juntos permitiram a troca de informações com uma página de internet. Entre a PowerPC e um computador externo haverá a necessidade de comunicação em rede TCP/IP permitindo que esse computador possa manipular o sistema via *web*.

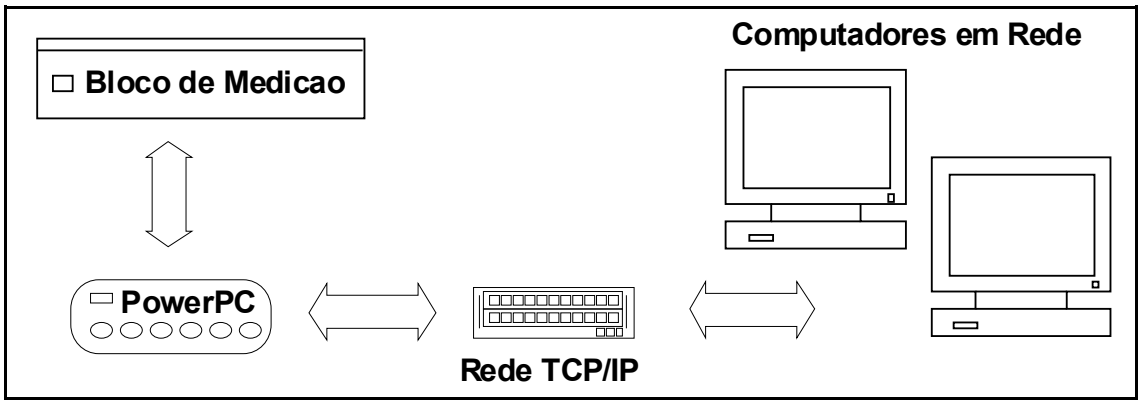

Figura 3.1 – Visão geral do sistema.

3.1. Descrição Geral do Sistema

O desenvolvimento da solução ocorre na seguinte ordem: No sistema de medição eletrônica são avaliados os níveis de tensão através de um conversor de sinais analógicos para digitais. A célula de carga combina a variação de tensão devido ao seu peso, modificando os valores informados a esse conversor. Um algoritmo avalia a condição inicial da balança e faz as considerações para chegar ao zeramento do *off-set* de calibração com peso padrão, tara e etc.

As informações de ajuste e parametrização do sistema de medida são armazenadas na PowerPC em formato de arquivos, o interpretador de comando do *browser,* que está na PowerPC, recebe a nova parametrização via protocolo

http e endereço de IP na rede, o sistema valida a informação oriunda da *web* de acordo com o que foi permitido ser alterado e também com regras do algoritmo, garantido parâmetros pertinentes para o funcionamento do sistema. Essa mesma interface serve para efetuar leituras de peso remotamente.

3.2. Descrição dos Sistemas Eletroeletrônicos

A célula de carga do projeto tem a capacidade de avaliar um mensurando de 0 a 50 kg, com um formato mecânico apropriado para aplicações genéricas, pode ser alimentada de 5 a 15 volts e sua sensibilidade é de 10 gramas.

O conversor analógico CS5531 com 16 bits de conversão digital serve para medição de pequenos sinais e ou pequenas variações de tensão. Interligados esses dois conjuntos, se define o bloco de medição. A célula de carga percebe variação mecânica modificando a resistência, por consequência a tensão, que o conversor irá ler.

Já o conversor precisa ser devidamente parametrizado para que funcione de acordo com as características do sistema, ou seja, por se tratar de um AD que responde serialmente a conversão obtida, não basta apenas inserir em uma porta a grandeza analógica e em seguida verificar a combinação binária em oito ou dez pinos como é normal para a maioria dos AD’s. Para este conversor à que se respeitar uma sequência de comando iniciais, como um protocolo, efetuando a configuração do modo de trabalho.

Com características particulares, principalmente pelo ajuste de ganho por *firmware*, torna-se um importante componente para o sistema. O modulo composto pelo processador PowerPC da motorola, memória *flash* de 8Mb e memória RAM de 32Mb viabiliza a proposta do trabalho e algumas de suas características, citadas no referencial teórico para um primeiro contato, são aplicadas e melhor detalhadas a seguir.

3.2.1. Hardware da plataforma PowerPC

A plataforma de processamento tem características bem definidas e simplificando a abordagem sobre esse módulo, pode-se compará-lo a uma CPU de computador. Das características do *hardware*, os pinos de entrada e saída disponíveis para controle por *firmware* não dispõem de resistores de garantia de nível de sinal, como *pull-up*, sendo necessário a utilização para sensores de contato seco.

O processador é alimentado com 3,3 volts e utiliza um cristal com frequência de 48Mhz determinando a velocidade de processamento. Outra característica importante são as duas portas seriais nativas, sendo uma delas direcionada em definitivo para acesso ao sistema como um terminal de comandos. Porém essas saídas de comunicação têm a mesma tensão do processador, sendo necessário nesse projeto o conversor de nível MAX232 e sua ligação correspondente para permitir a comunicação com um computador a ser utilizado como terminal.

Uma grande vantagem desse *hardware* é a possibilidade de ter uma porta ethernet para comunicação com a rede no formato TCP/IP, padrão esse mundialmente difundido na utilização da internet. Esse conector para uso em rede não está disponível na plataforma, precisando ser previsto e ligado externamente na placa de interface com a plataforma de processamento.

Sua memória flash permite armazenar informações pertinentes ao sistema, arquivos de configuração, aplicativo executável, assim como também tem capacidade de armazenar o sistema operacional dedicado. A figura 3.2, localiza os principais componentes da plataforma. O que acontece, resumidamente no processo de inicialização do sistema é a descompactação do *kernel* da memória flah para a memória RAM permitindo a execução do linux.

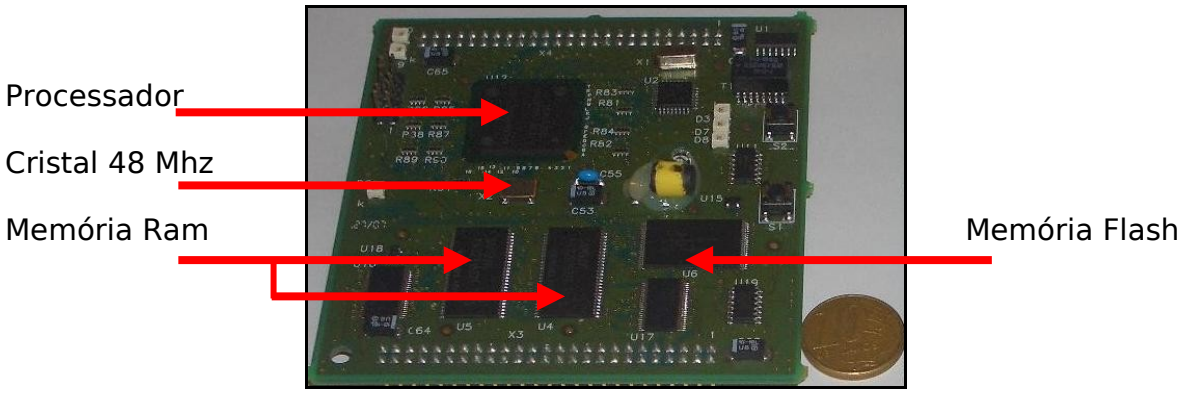

Figura 3.2 – Detalhes da PowerPC

Alimentação

Para alimentar a placa com 3,3 volts se utilizou um LM317 com circuito típico do seu manual. Na figura 3.3, mostra o transistor BC327 para controle do TIP122 e com essa configuração consegue-se uma maior capacidade de fornecimento de corrente obtendo com segurança 1,5A necessários para o funcionamento da placa.

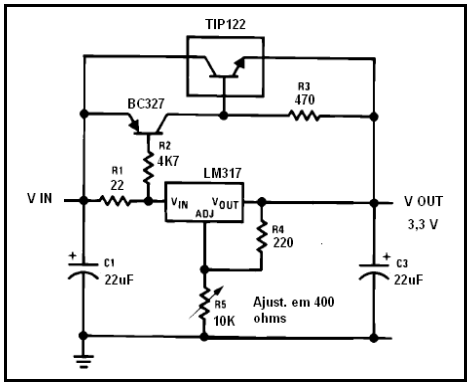

Figura 3.3 – Regulador de tensão 3,3 volts

Serial para debug

Para utilizar a serial de debug da CPU, foi utilizado o circuito integrado MAX232 para converter os níveis de tensão de TTL para o padrão RS232 tomando o devido cuidado de converter a saída TX do MAX232 em nível de tensão 3,3 volts. Para isso, como mostra a figura 3.4, utilizou-se o diodo zener 1N746A, na posição D1, tornando o sinal compatível com a PowerPC.

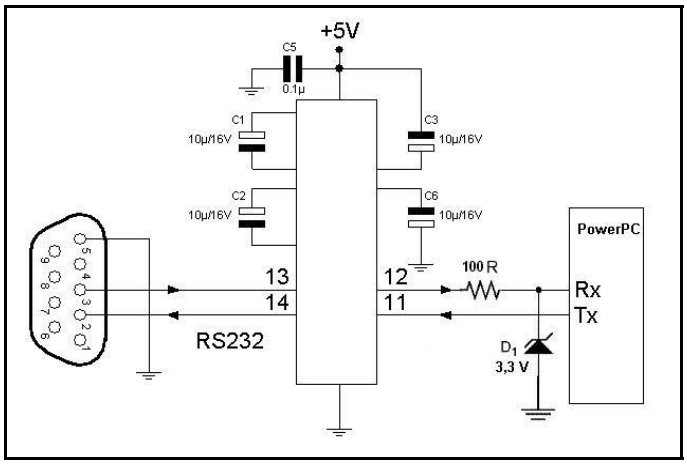

Figura 3.4 – Circuito MAX232 para comunicação serial

3.2.2. Linux embarcado

Um recurso de grande valor em projetos com rede e protocolos TCP/IP é o fato de poder trabalhar diretamente com sistema operacional, sendo assim muito do que se precisa já está embutido na compilação de *kernel* que roda de maneira embarcada. Exemplificando, da mesma maneira que se configura um IP no sistema linux usual se repete para o linux embarcado.

Para acessar o sistema operacional da plataforma há necessidade de uma comunicação serial, essa serial precisa ser devidamente convertida em termos de níveis de tensão para o padrão 232. Através de um terminal com taxa de

115200bps, 8 bits de dados, paridade nenhuma, 1 bit de parada e controle de fluxo nenhum, então com essa configuração se pode interagir diretamente no linux.

Logo que o sistema inicia o terminal registra informações de tudo que está ocorrendo na plataforma, exemplo na figura 3.5.

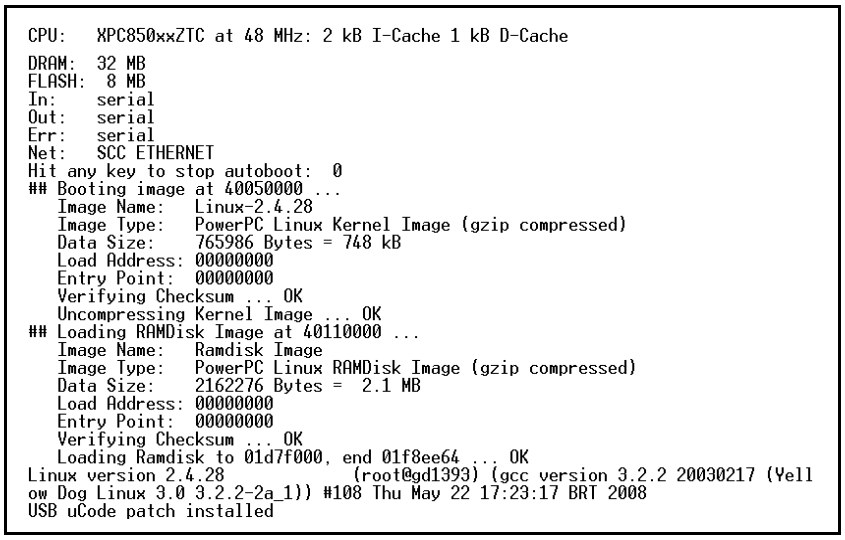

Figura 3.5 – Terminal de operação da PowerPC

Neste terminal é onde tudo pode ser acessado e configurado. Um dos primeiros procedimentos no inicio dos trabalhos é o de determinar um IP e uma máscara de rede válida para a rede onde se deseja trabalhar. Ou seja, das características de redes TCP/IP uma máquina pode comunicar-se com outra através da rede desde que estejam na mesma faixa de IP.

Configurando um computador com um IP:192.168.1.10 para que essa máquina possa interagir por rede com o *hardware*, a CPU deverá estar com um IP 192.168.1.11. Esse truncamento que restringe exatamente qual endereço é permitido ter interação de um ponto à outro se dá pela mascara de rede. A máscara de rede é usada para determinar que parte do IP seja o endereço da rede e qual parte é o endereço pontual do *hardware* na rede. Neste trabalho a máscara de rede é 255.255.0.0. tanto para o computador quanto para a PowerPC, assim fica determinado que para haver comunicação entre elas obrigatoriamente devam estar na faixa 192.168.X.X.

3.2.3. Compilação e execução do firmware na PowerPC

Para gerar um código executável há necessidade da instalação de um pacote dedicado para o processador PowerPC no computador onde vai ser

desenvolvido o *firmware*. Nesse pacote existe a estrutura de bibliotecas necessárias para a geração da aplicação. No pacote ELDK encontra-se o compilador GCC apropriado para a arquitetura desse processador. O código é escrito e gerado fora da PowerPC e posteriormente atualizado através das ferramentas para conexão em rede.

Logo que o equipamento encontra-se configurado na mesma rede da estação de trabalho, pode-se fazer uso da conexão para acesso ao *hardware* por comando *telnet*, o que a partir deste momento torna-se muito importante. Esse comando vem a ter o mesmo efeito da serial direcionada para o terminal, ou seja, uma vez estabelecida a conexão *telnet* é possível acesso ao *hardware* através de um terminal de rede fugindo das limitações físicas que tem uma comunicação serial.

Essa característica somada a mais uma ferramenta de rede, o compartilhamento NFS, permite o comando de execução do *firmware* em desenvolvimento e ou troca da aplicação pela rede de maneira muito eficaz e ágil.

3.3. Descrição do bloco de medição

O bloco de medição foi concebido de duas maneiras. Inicialmente foi feito a tentativa de acionamento direto dos pinos de entrada e saída da plataforma de processamento. O conversor AD interage através de uma serial SPI, conhecida como clock/dado. Apesar da PowerPC ter um bloco específico para esse controle SPI internamente, a plataforma não disponibiliza esse controle para acesso externo.

Houve a tentativa de aquisição direta nos pinos de I/O, efetuando o controle por firmware dos acionamentos, porém o que se observou em testes com osciloscópio é que a PowerPC não mantém a prioridade para esses acionamentos. Existem blocos de maior prioridade como a porta ethernet, a porta serial, acesso as memória e etc. Mesmo nas tentativas de melhorar a prioridade da execução da aplicação não se conseguiu o resultado esperado.

Montou-se uma segunda estrutura, agora com conversão da serial padrão da PowerPC para uma serial SPI. Para essa conversão se utilizou o controlador 89C2051. O controlador recebe o comando serial por interrupção da PowerPC e o valida através do protocolo desenvolvido, a seguir, o controlador converte na sequência clock/dado para que haja comunicação com o conversor AD.

A figura 3.6 mostra os pinos do 3.2, 3.3, 3.4, do controlador 89C2051, ligados diretamente no conversor AD que é onde ocorre a comunicação SPI e os pinos 3.0 e 3.1 serão ligados ao CN1 disponibilizando a conversão no formato serial TTL para a PowerPC.

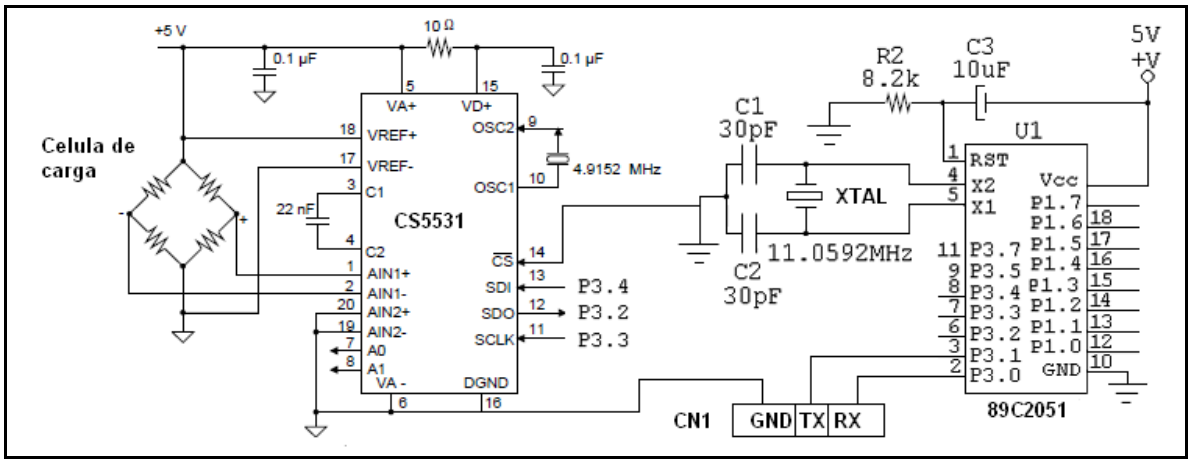

Figura 3.6 – Circuito de medição do peso

3.4. Descrição dos Sistemas Computacionais

Para desenvolver o projeto se fez uso do sistema operacional Linux Ubuntu 9.10. Neste ambiente foi instalado o compilador GCC para a PowerPC. Foi utilizado o *software* NetBeans IDE 6.8 como editor para escrever o código na linguagem C. Nos testes de comunicação serial utilizamos kermit 8.0 e o pacote do linux wine, versão 1.0.1, para permitir executar programas que só funcionariam no sistema operacional windows. Para gerar o código HTML o programa escolhido foi o Dreamweaver 8.0. Todas essas ferramentas de *software* são livres e podem ser baixadas facilmente da internet.

3.4.1. Firmware do conversor serial 89C2051

Para efetuar a conversão dos bytes recebidos serialmente da PowerPC em um padrão *clock/dado*. Desenvolveu-se um pequeno protocolo para garantir uma sequência válida. O *firmware* foi planejado e desenvolvido como mostra a figura 3.7. onde contém uma máquina de estados para cada byte que o controlador recebe. Quando o primeiro byte é recebido o *firmware* vai para o estado zero e o byte é avaliado, caso seja o comando previsto segue o fluxograma, caso contrário o comando é rejeitado.

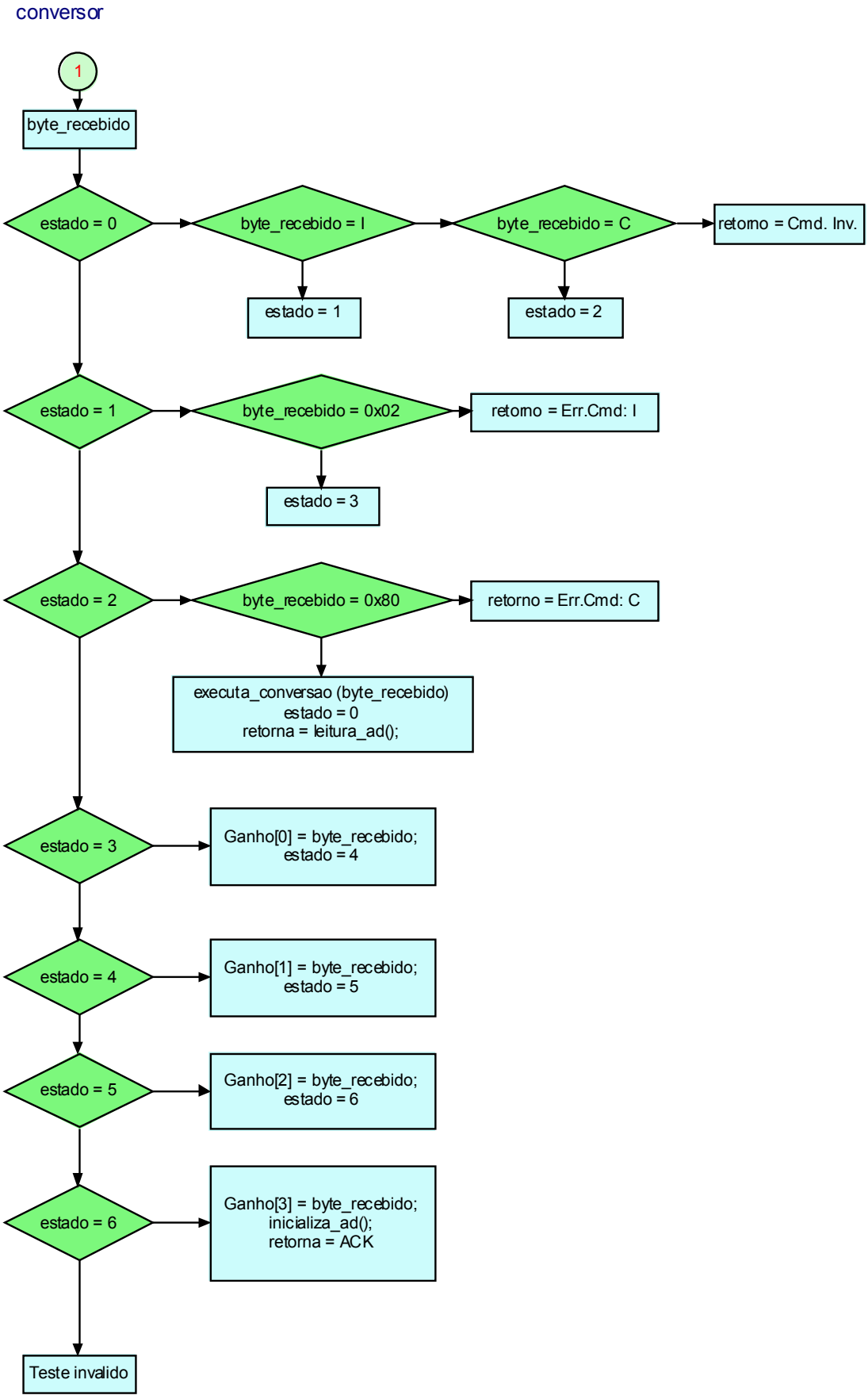

Figura 3.7 – Algoritmo de conversão serial para SPI.

Foi criada a possibilidade de receber dois comandos. O comando “I” e o comando “C”. O comando “I” que corresponde em hexadecimal no valor 0x49, indica o processo de inicialização, esse comando deve ser seguido de um byte verificador 0x02 e na sequência do comando deve conter quatro bytes que correspondem ao ganho inicial que o conversor AD deve trabalhar. Quando o conversor serial recebe da PowerPC a sequência correta conforme a tabela 3.1 o conversor AD CS5531 passa operar com os valores de ganho contidos nos bytes de 0 a 3.

Tabela 3-2 – Comando aceito pelo conversor serial.

Comando	Verificação	Byte_0	Byte_1	Byte_2	Byte_3
0x49	0x02	x	x	x	*x*

Após receber todo o comando, o algoritmo do conversor serial separa os bytes que contém o ganho e envia para o para o conversor AD bit a bit essa informação, efetuando um pulso de clock para cada bit a ser transmitido. A figura 3.8 – Serial SPI clock e dado ilustra o processo de uma serial SPI.

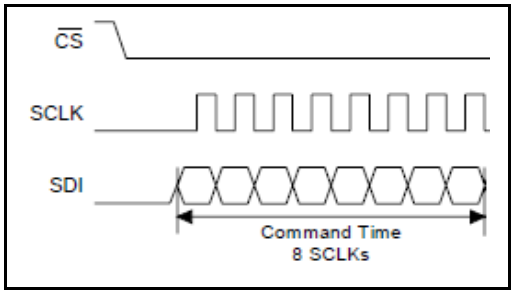

Figura 3.8 – Serial SPI clock e dado.

O comando “C” que corresponde a 0x43 em hexadecimal é responsável por uma solicitação de leitura do conversor AD. Esse comando deve ser seguido por um sub-comando 0x80 formando a sequência da tabela 3.2 – Comando leitura do AD.

Tabela 3-3 – Comando leitura do AD.

Comando	Sub-comando
0x43	0x80

O algoritmo do conversor serial separa o sub-comando 0x80 e repete o procedimento de envio bit a bit. Logo após o algoritmo monta um comando de retorno colocando na primeira posição um sinalizador ACK e nos próximos quatro

bytes, lendo bit a bit a informação passada pelo AD, conclui a resposta que será dada para a PowerPC. Tabela 3.3 – Comando de retorno contendo a leitura do AD.

Tabela 3-4 – Comando de retorno contendo a leitura do AD.

Confirmação	**Byte_0**	**Byte_1**	**Byte_2**	**Byte_3**
0x06	x	x	0	0

3.4.2. Inicialização do conversor AD

Para trabalhar com o conversor, o primeiro passo é um processo de inicialização, ilustrado na figura 3.9 onde se observa o entendimento obtido do manual do circuito integrado CS5531. Foi criada uma rotina de espera de 30 milissegundos. Coloca-se o pino de entrada de dados do AD, o pino SDI, em nível 1 efetuando 127 pulsos de clock, o que vem a corresponder a quantidade de bits de 16bytes menos um, pois o último bit deve ser zero. Então, logo se observa mais um pulso de clock completando um procedimento de sincronismo. A seguir estão escritos os cinco procedimentos:

O procedimento [A] da figura 3.9 junto com o [B], corresponde ao reset e zeramento do chip, a etapa [C] opta dentre os dois canais possíveis de monitorar a tensão da célula de carga, pelo canal 1, do mesmo modo que zera ganho de *offset* e *span.* O item [D] configura o canal 1 com ganho 3, nos testes de desenvolvimento foi observado que o valor lido pela balança vazia e o valor lido com o peso padrão de 1 kg sempre apontam uma diferença aproximada de 3 vezes. Então configurando esse valor nos aproximaremos da calibração com menos interações de *firmware*, isso será mais bem detalhado no processo de calibração. A etapa [E], a última, estabelece *offset* para o canal 1 igual a zero.

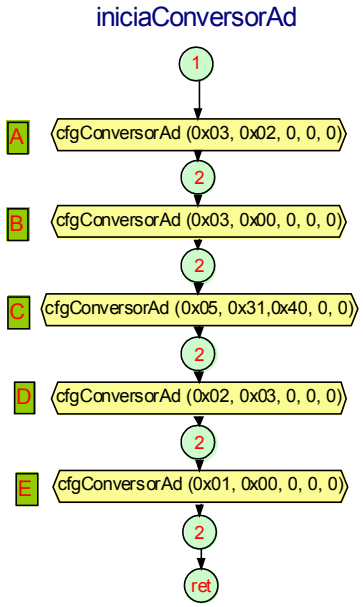

Figura 3.9 – Inicialização do conversor AD

3.5. *Interface WEB*

Para se chegar até uma primeira tela, foi necessário inserir um programa de *webserver* dentro da PowerPC. Foi pesquisado um *webserver* reduzido para ser compatível com os tamanho de memória que disponível.

A melhor opção foi *webserver boa*. Devido suas características práticas, o *boa* é uma ferramenta flexível e compacta, além se ser um *software* livre. Outro estudo necessário foi sobre a interface CGI, pois será através dela que será possível receber os comando da página e atualizar com os novos parâmetros, ou retornar os parâmetros que o sistema está monitorando.

3.5.1. *Web*server Boa

O *boa* é um *webserver* genérico, podendo ser utilizado em qualquer plataforma, porém para tanto, tem que se compilar o seu código fonte corretamente para o *hardware* de destino, esse é o fato principal.

Os arquivos fontes, disponíveis no site http://www.boa.org/, acompanham uma estrutura de outros arquivos que auxiliam na geração do código executável dessa aplicação. A figura 3.10 demonstra as etapas que tiveram que ser pesquisadas para gerar o executável do *webserver boa*. Em um primeiro momento se utiliza um arquivo *script* para o linux, chamado configure, esse arquivo tem por finalidade gerar o *makefile* com a plataforma correta.

O arquivo *makefile* chama o compilador. Neste arquivo estão contidos os parâmetros de escolha de qual compilador será utilizado, porém ao executar o *script* o compilador que fica configurado é o GCC padrão do linux ubuntu e não o da PowerPC. Como a necessidade é gerar um executável para a PowerPC, tem que substituir o conteúdo do arquivo onde consta essa informação, pelo compilador GCC correto.

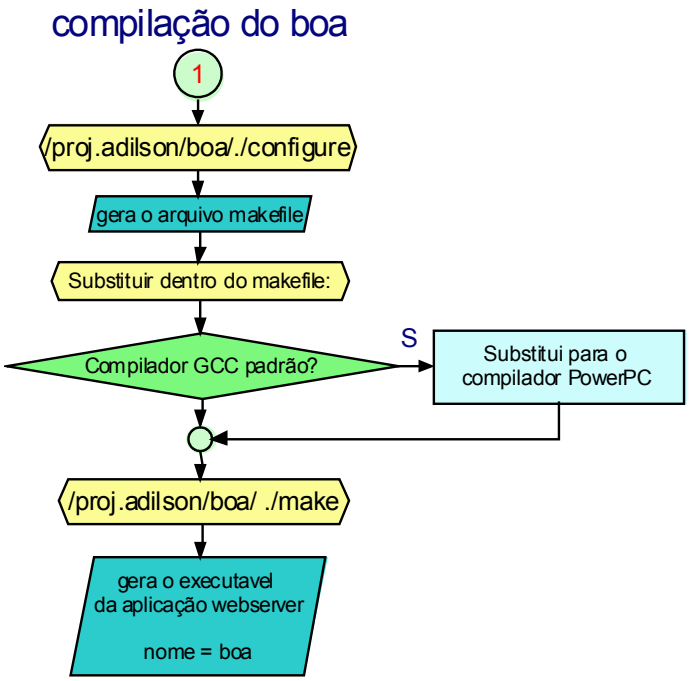

Figura 3.10 – Compilação do *web*server

3.5.2. Entendimento do *web*server boa

Para trabalhar com *webserver* boa já na PowerPC, será necessário liberar as permissões de leitura, escrita e execução na pasta de trabalho. Existe um arquivo chamado *boa.conf* onde serão passados os parâmetros de trabalho para o aplicativo boa.

Então há a necessidade de abrir esse arquivo e escrever em modo texto as opções de funcionamento. A primeira parametrização chama-se *DocumentRoot* onde será passado nesse parâmetro o caminho da pasta de trabalho. Será nesse caminho onde se encontrará a estrutura de arquivos mostrados na página, assim como o *firmware* chamado pela navegador para interagir com o *hardware*.

O segundo parâmetro a ser registrado é o nome do primeiro arquivo que deve surgir na página quando o endereço eletrônico for inserido no navegador.

Então o aplicativo trata de encontrar esse arquivo no caminho que está registrado, (passo anterior) para exibir na tela.

Existem outros tantos parâmetros possíveis neste arquivo, inclusive no que diz respeito a tempos de acesso, segurança, armazenamento de informações que não serão abordados neste trabalho.

A figura 3.11 demonstra os passos para executar o *webserver* na PowerPC em que apresenta as etapas desde a configuração necessária dentro do arquivo base do *boa* até a necessidade de informar o caminho onde ele se encontra na hora da execução. O procedimento para acessar uma página simbólica que foi criada também é descrito.

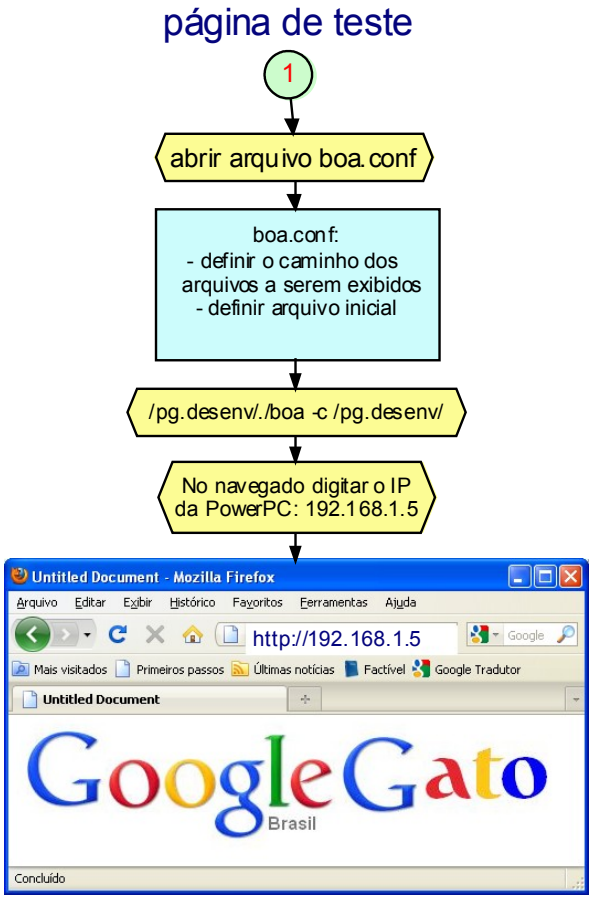

Figura 3.11 – Passos para uma página de teste

3.5.3. Método CGI

O CGI - *Common gateway Interface* é uma padronização de comunicação entre o *webserver* e o sistema operacional. Ele pode ser considerado o caminho necessário para que o *webserver* execute as ações passadas pela página. Este modelo de comunicação é aplicado na maioria dos servidores.

Os arquivos para serem executados por uma chamada do navegador, necessariamente precisam conter a extensão CGI. Esse arquivo pode ser um script padrão do linux ou um *firmware*. A informação de quais "arquivos.CGI" devem ser chamados estão contidos nos botões ou em links do código HTML exibido pelo navegador.

O CGI tem como principal função acessar as variáveis de ambiente do sistema operacional, atualizando essas variáveis com informações passadas pela página. O "*firmware*.CGI" irá trabalhar igualmente com essas variáveis retornando valores de acordo com seu algoritmo para realimentar a página.

A figura 3.12 ilustra, de maneira simplificada, a comunicação entre o navegador e o *webserver* BOA.

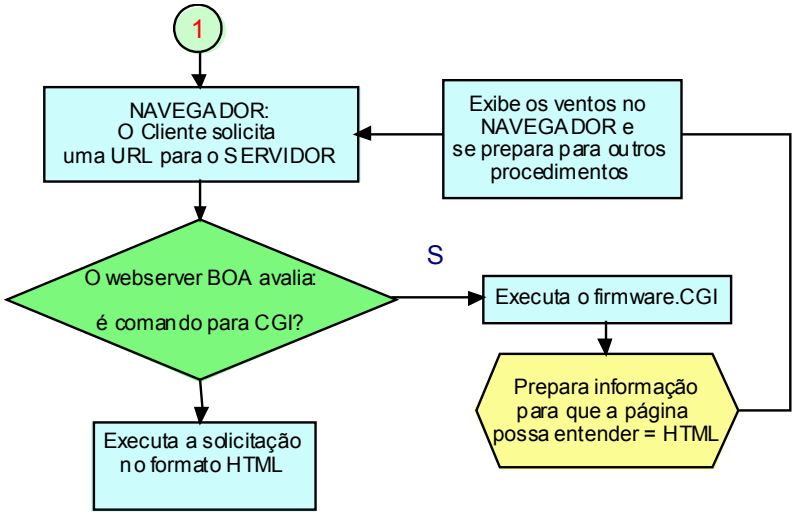

Figura 3.12 – Ciclo de troca de informações

3.5.4. Comunicação CGI com a página *web*

Seguindo interpretação do Manual de CGI, uma excelente referência do assunto, foi possível criar um "script.CGI" para os primeiros passos na manipulação das variáveis de ambiente. Na figura 3.13 é demonstrado o comando executado pelo navegador, que chama o arquivo *script* da figura 3.14 previamente criado.

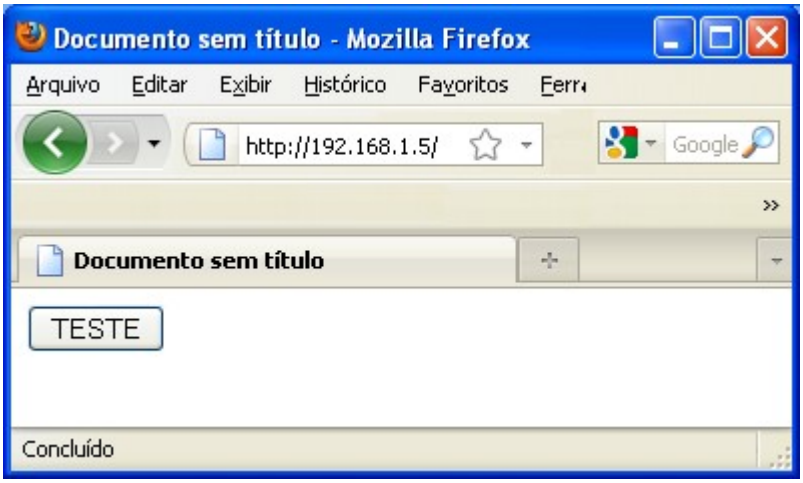

Figura 3.13 – Navegador com botão para chamar script

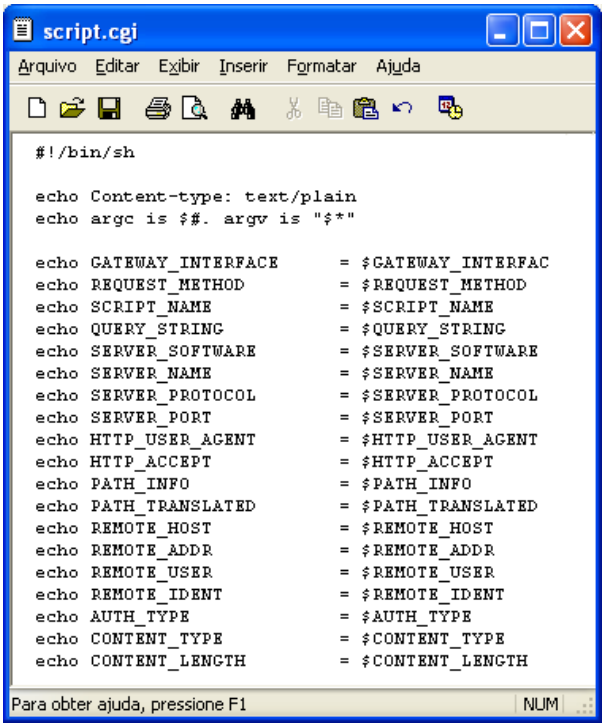

Figura 3.14 – Script com a declaração das variáveis de ambiente

Na figura 3.15, apresenta o retorno do comando de teste e estão as variáveis que o sistema operacional conseguiu avaliar. A principal variável a ser utilizada para o desenvolvimento desse projeto será QUERY_STRING, será nela que o as principais informações irão trafegar.

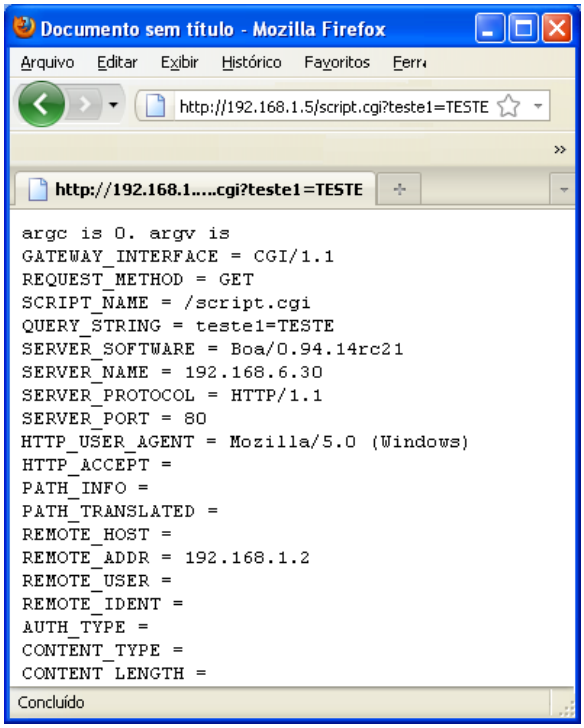

Figura 3.15 – Resultado do comando de teste

4. APRESENTAÇÃO E DISCUSSÃO DOS RESULTADOS

Os resultados obtidos no projeto estão descritos em etapas, os testes feitos junto com o desenvolvimento avaliam o funcionamento de todo o sistema. A figura 3.1 Visão geral do sistema, do capítulo anterior, serve como apoio para ilustrar as etapas.

4.1. PowerPC

A PowerPC passou por vária procedimentos de testes para o entendimento das suas características, a primeira etapa foi as comunicação via serial para as primeiras interações e configurações. Os testes básicos de rede vieram logo a seguir com o comando ping para identificar a estação de trabalho, a figura 4.1 mostra o acesso à plataforma através do comando *telnet* efetuado da estação de trabalho.

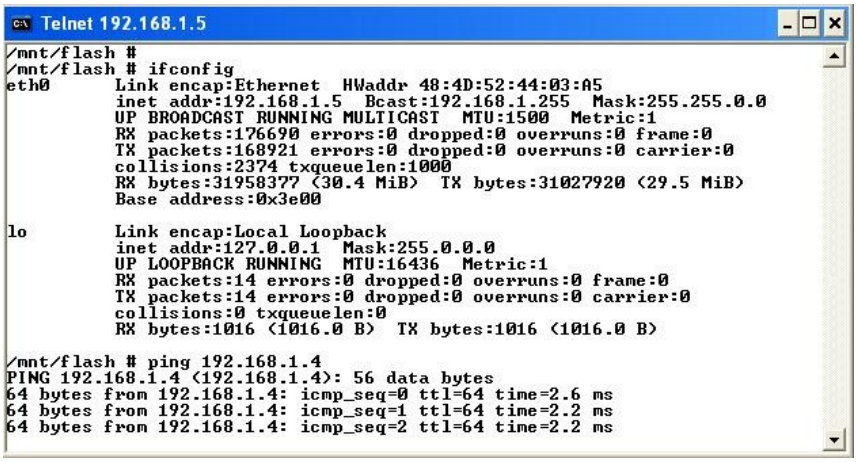

Figura 4.1 – Comandos básicos de conexão com a rede

4.2. Bloco de medição

O bloco de medição, da figura 4.1, precisou ser modificado e no decorrer do projeto foi inserido um controlador para converter os sinais recebidos no padrão serial da PowerPC para o formato SPI utilizado pelo AD. A figura 4.2 apresenta como ficou a montagem do bloco de medição onde, bem à esquerda,

está o conector para a célula de carga, no centro o conversor AD e à direita da placa o controlador 89C2051 junto da saída serial que vai para a PowerPC.

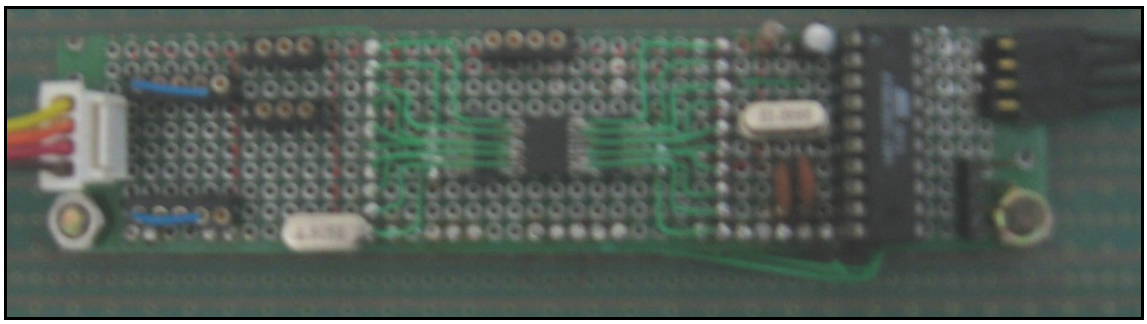

Figura 4.2 – Bloco de medição

Nos testes, pode-se avaliar que os pinos comuns de entrada e saída da PowerPC não podem ser utilizados em tarefas onde há necessidade de controle de tempo com precisão de intervalos. No desenvolvimento, para diagnosticar o problema, foi criado um código fonte exatamente igual para a PowerPC e o 89C2051 com diretivas de processamento para escolher o *hardware* de teste.

Na figura a 4.3 é possível visualizar a ocorrência dos diferentes períodos de acionamento dos pinos de saída. Na figura fica visível que o acionamento da PowerPC se comporta diferente do controlador dedicado.

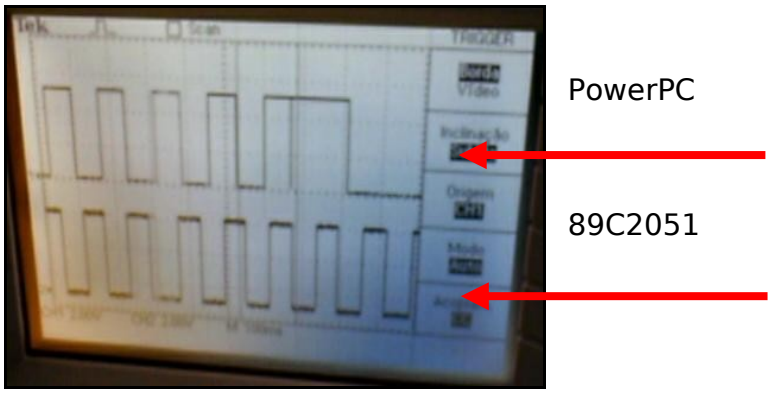

Figura 4.3 – Tempos de acionamento PowerPC e 89C2051

4.3. Aplicação preliminar acessada por telnet

Trabalhando em conjunto o bloco de medição e a PowerPC chegou-se aos testes da aplicação preliminar. Essa aplicação se caracteriza em operar por acesso em rede utilizando o comando telnet, não permitindo ainda acesso pelo navegador *web*. Nesse *firmware* pode-se efetuar os testes de desenvolvimento, calibrar o equipamento com uma medida padrão, validando e simplificando a etapa posterior que seria a comunicação com a *web*.

4.3.1. Executando o *firmware* por telnet

Para acessar o equipamento pela rede, utilizou-se o comando de telnet, a figura 4.4 mostra o procedimento de conexão, o qual informa o IP da máquina que se deseja atuar e logo em seguida o usuário e a senha.

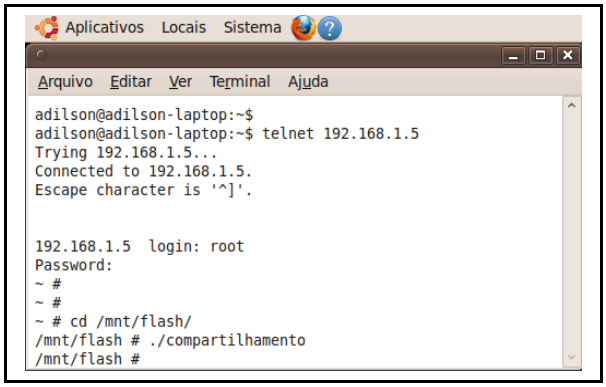

Figura 4.4 – Comando telnet para acesso ao equipamento.

O teste completo inicia pela etapa de calibração chegando até a coleta dos valores de peso. Uma vez que o *firmware* é executado no terminal, as mensagens dizem o que deve ser feito, quando colocar o peso padrão até a calibração ficar completa. A figura 4.5 no quadro à esquerda mostra o comando de execução do *firmware* de teste e no quadro a direita às mensagens ao final de uma calibração.

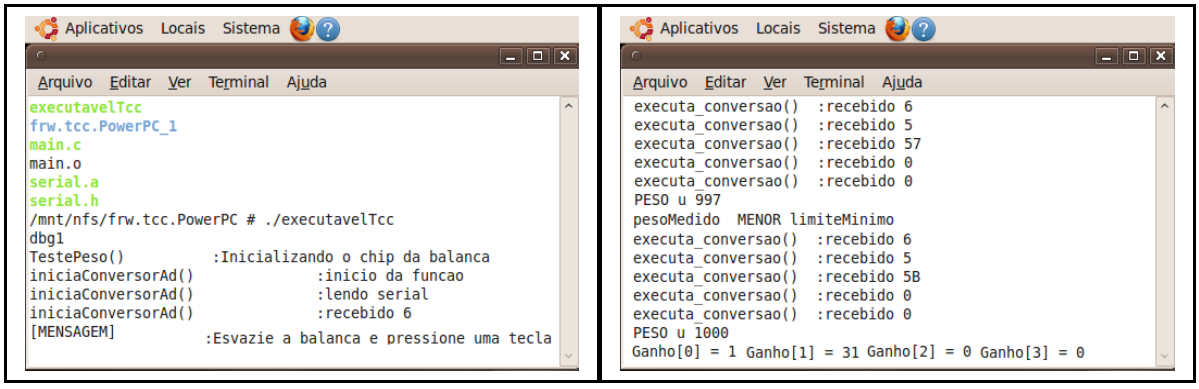

Figura 4.5 – Etapa de inicio e fim de uma calibração

4.3.2. O processo de calibração

Para calibrar o sistema foi desenvolvido um algoritmo de interação, comparando o valor da balança em vazio com o valor da balança com o peso padrão. O *firmware* segue ajustando o ganho até que se chegue ao valor determinado de 1 kg. A figura 4.6 é o resultado do que foi feito para efetuar a calibração na balança.

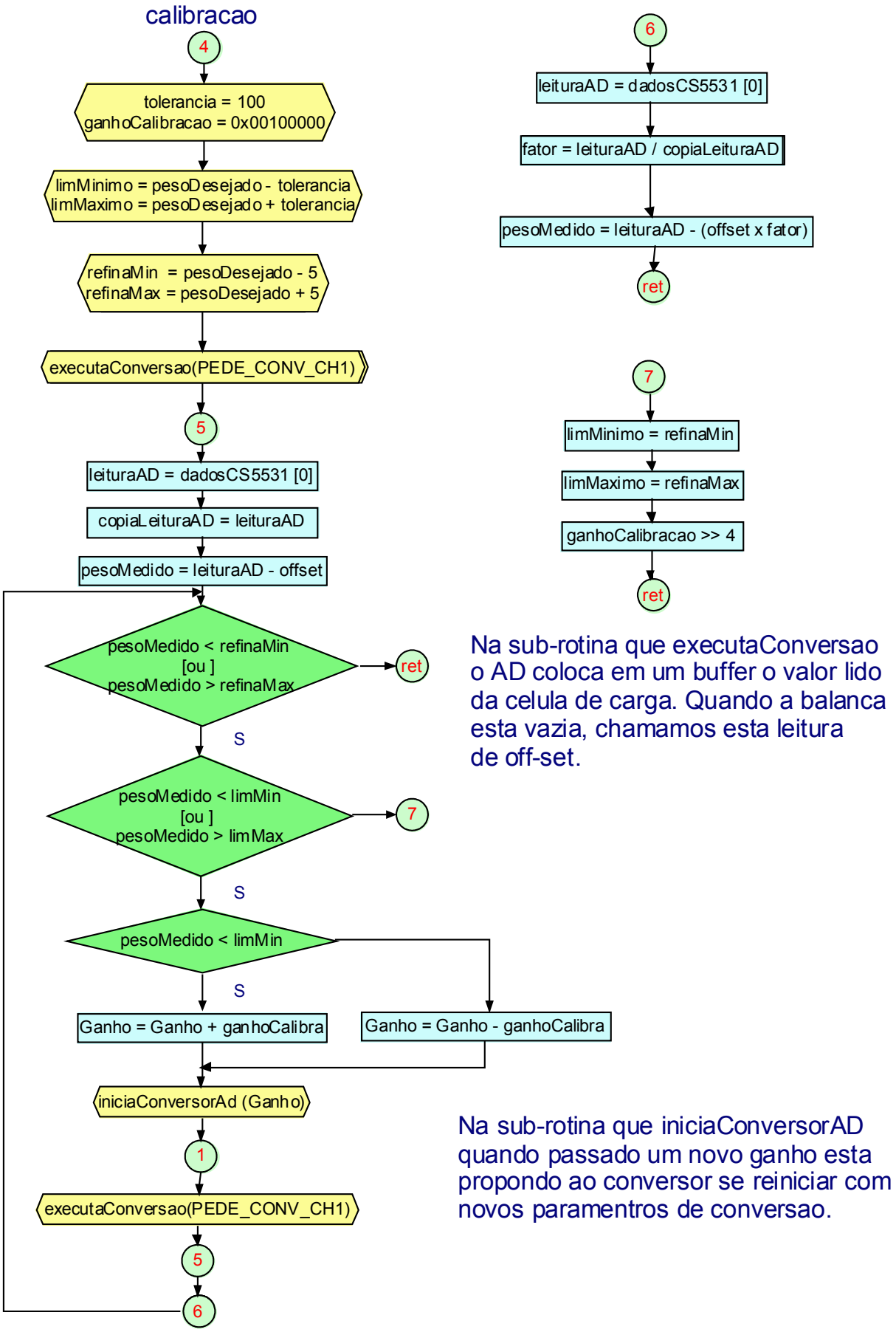

Figura 4.6 – Procedimento de calibração

4.3.3. Testes de leitura de peso

Algumas informações foram levadas em conta para qualificar esse sistema. Uma sequência de levantamentos é demonstrado a seguir. A figura 4.7 compara o um peso padrão de um quilograma no sistema desenvolvido versus a balança PRIX4 do fabricante Toledo. No primeiro quadro há o peso padrão sobre a bandeja do sistema e no quadro que está ao meio é demonstrado as três medidas feitas pelo sistema. No quadro a direita o valor obtido em uma balança comercial.

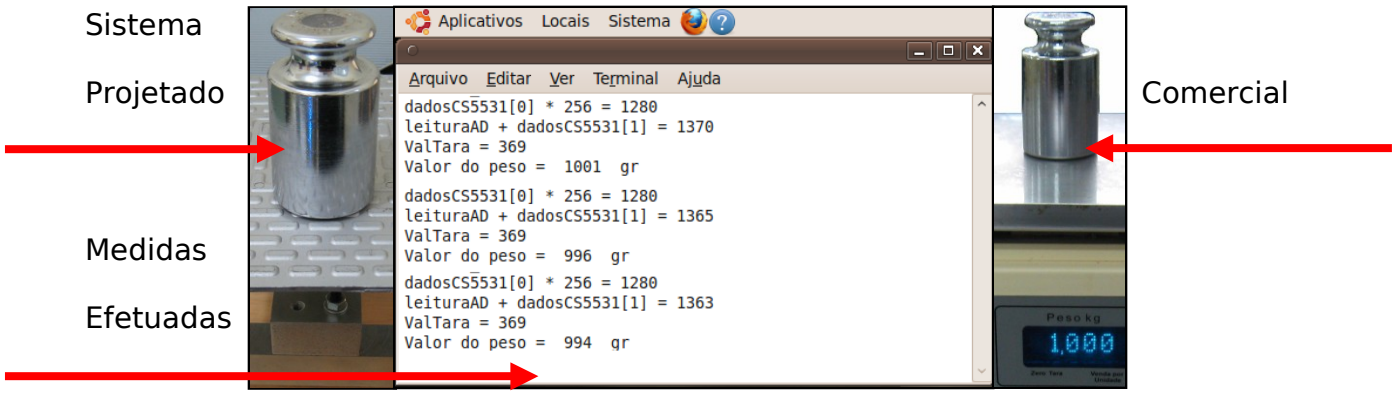

Figura 4.7 – Peso padrão testado também em uma balança comercial.

Uma segunda medida foi levantada, agora com mais amostras de peso padrão para um mesmo teste. A figura 4.8 ilustra o *hardware* conectado célula de carga junto com a estrutura mecânica e uma sequência de pesos com 20 g, 50 g, 100 g, 200 g, 500 g e 1000 g. conforme a marcação de cada um.

Analisado o conjunto de amostras em separado, observou-se uma diminuição do erro à medida que o peso aumentava. Coletando dez medidas do peso máximo padrão disponível, como mostra a figura 4.8, no qual seu somatório mede 2870 gramas, o erro diminui ainda mais, nunca ultrapassando o 0,1%.

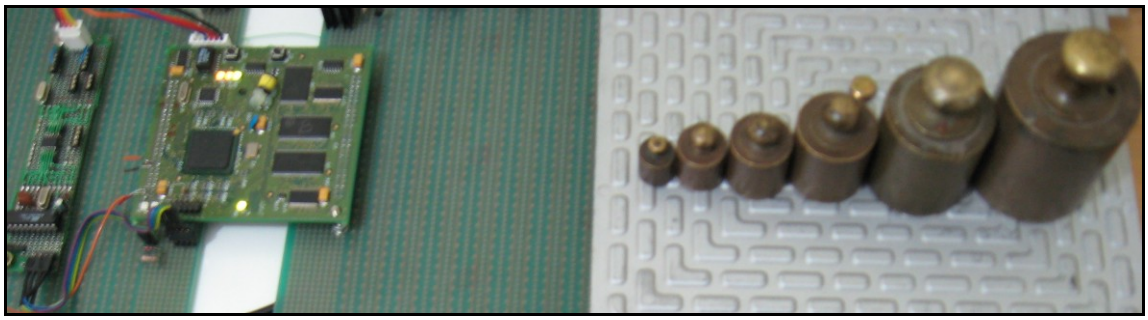

Figura 4.8 – Condição de teste.

A tabela a seguir ilustra dez medidas de cada peso padrão amostrado individualmente e na sequência um gráfico para o erro médio das dez amostras.

Tabela 4-5 – Coleta de dez medidas de cada peso e cálculo do erro em percentual.

	20g	**% e**	**50 g**	**% e**	**100 g**	**% e**	**200 g**	**% e**	**500 g**	**% e**	**1000 g**	**% e**
1	14	30	43	14	93	7	197	1,5	494	1,2	1001	0,1
2	16	20	42	16	91	9	204	2	494	1,2	1002	0,2
3	12	40	43	14	90	10	198	1	497	0,6	999	0,1
4	15	25	44	12	89	11	200	0	496	0,8	1002	0,2
5	13	30	43	14	91	9	211	5,5	501	0,2	996	0,4
6	14	30	42	16	91	9	196	2	497	0,6	994	0,6
7	14	30	43	14	93	7	195	2,5	495	1	998	0,2
8	15	25	42	16	92	8	197	1,5	494	1,2	994	0,6
9	13	35	40	20	94	6	199	0,5	499	0,2	998	0,2
10	16	20	44	12	92	8	196	2	498	0,4	997	0,3

A figura 4.9 ilustra o erro após as dez coletas de peso do sistema, demonstrando que para valores menores que 500g o condicionamento deve ser reavaliado.

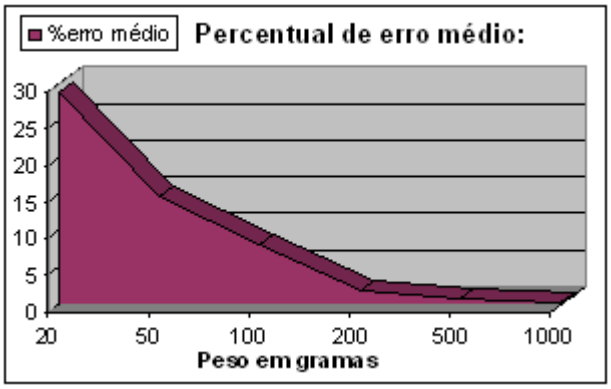

Figura 4.9 – Percentual de erro médio entre dez medidas.

4.4. Aplicação principal e interface web

Para trabalhar com a monitoração por uma página em um navegador de internet, a aplicação preliminar teve que ser toda remodelada, principalmente no que diz respeito à comunicação. O que acontece é que à medida que a página solicita alguma informação do *firmware*, esse não pode fica trancado em nenhuma rotina sem atualizar as mensagens no navegador, para não fornecer a falsa sensação de falha.

O *firmware* que trabalha junto com a página HTML, teve que a partir desse momento, dividir tarefas que antes ocorriam em uma única interação.

Algumas variáveis tiveram que ser salvas em arquivos para que os procedimentos subsequentes continuassem sem erro.

4.4.1. Testes de controle do sistema pelo navegador

A sequência a seguir ilustra a interface que foi criada em HTML para comunicar com o *firmware* no formato CGI e também todo um processo de calibração até a coleta das medidas de peso.

A figura 4.10 foi criada com base na capa deste trabalho, devido a facilidade de converter um documento do word em código html. Com o título do projeto é a primeira mensagem que surge ao acessar o equipamento através do navegador digitando o IP: 192.168.1.5.

Essa tela fica exposta aguardando um comando para prosseguir, o arquivo que contém essa tela está dentro da PowerPC. O Resultado disso é que todas as interações ocorrem diretamente dentro do equipamento e o navegador que está sendo utilizado é apenas uma ferramenta de acesso.

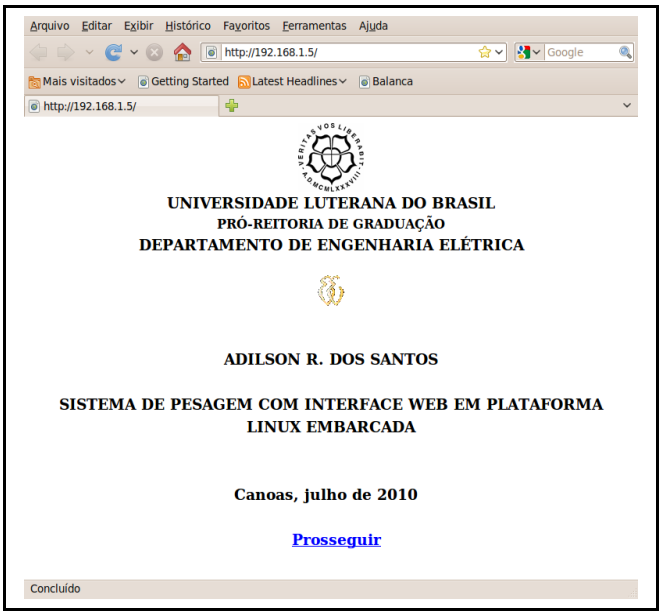

Figura 4.10 – Tela de entrada

A figura 4.11 mostra a segunda tela em sequência fornecida pelo sistema, essa tela ainda é formada por um arquivo que é carregado pelo navegador, porém necessita a inserção de uma senha. A partir desse momento o código HTML está dentro do *firmware* e através dele que irão surgir as próximas interfaces.

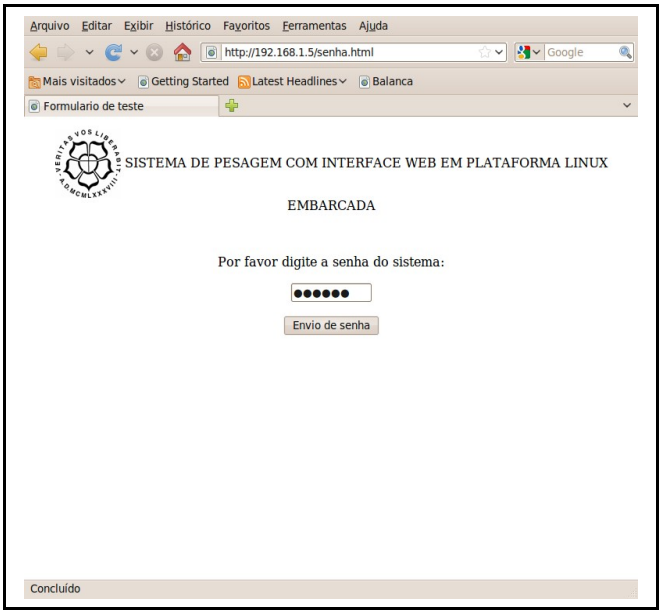

Figura 4.11 – Tela um do sistema *web*

A figura 4.12 inicia o processo de calibração do sistema e alguns comandos por intermédio do navegador são necessários para finalizar o processo.

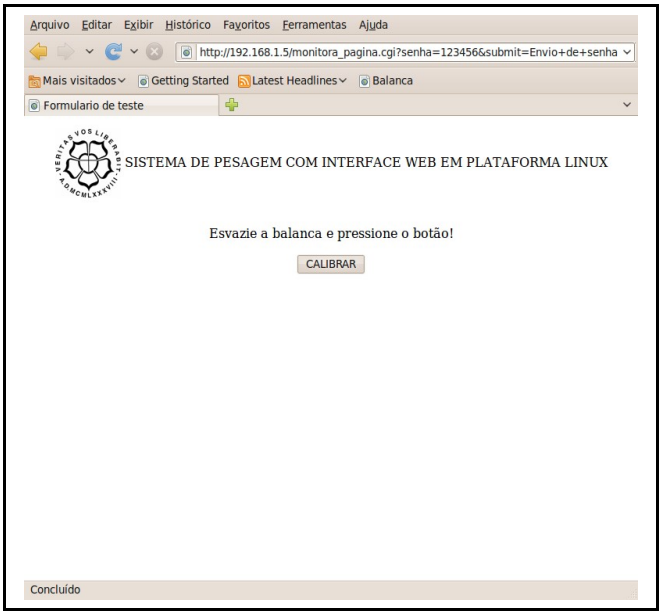

Figura 4.12 – Tela dois de interface *web*

A figura 4.13, mostra o retorno do *firmware* na primeira interação com o *hardware* após o comando do navegador. É possível observar nas indicações da figura, que o navegador fica aguardando a resposta do *firmware* antes de atualizar as mensagens na tela. É importante que o *firmware* tenha uma interação rápida para não passar uma sensação de falha.

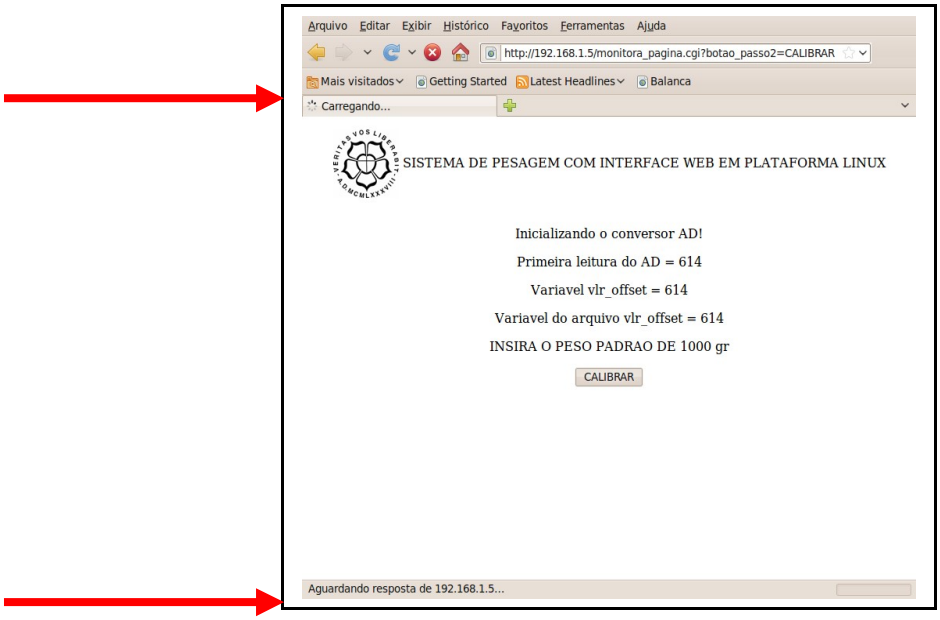

Figura 4.13 – Tela três de interface *web*

A figura 4.14 conclui a etapa de avaliar o que foi lido pelo conversor AD quando a balança estava vazia e ajustar conforme a leitura efetuada do peso padrão de um quilograma.

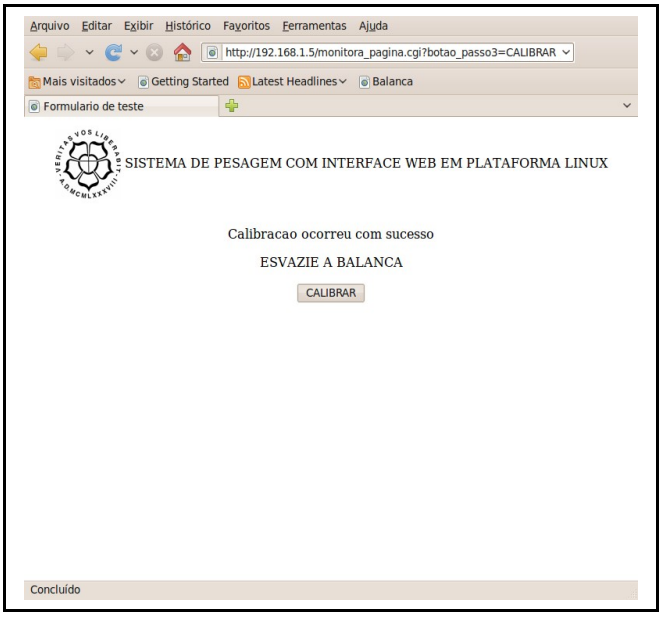

Figura 4.14 – Tela quatro de interface *Web*

A figura 4.15 mostra a gravação da variável de tara da balança em um arquivo. Uma vez que o *firmware* não pode ficar trancado em uma rotina, algumas variáveis são armazenadas dessa maneira permitindo que a cada interação feita pelo sistema esses valores possam ser recuperados.

Figura 4.15 – Tela cinco de interface *web*

A figura 4.16 ilustra uma sequência de inserções de peso apenas para demonstrar a variação gráfica que ocorre a cada medição. Dentro do *firmware* existe um comando passado para o navegador que faz com que ocorra uma leitura do sistema a cada segundo.

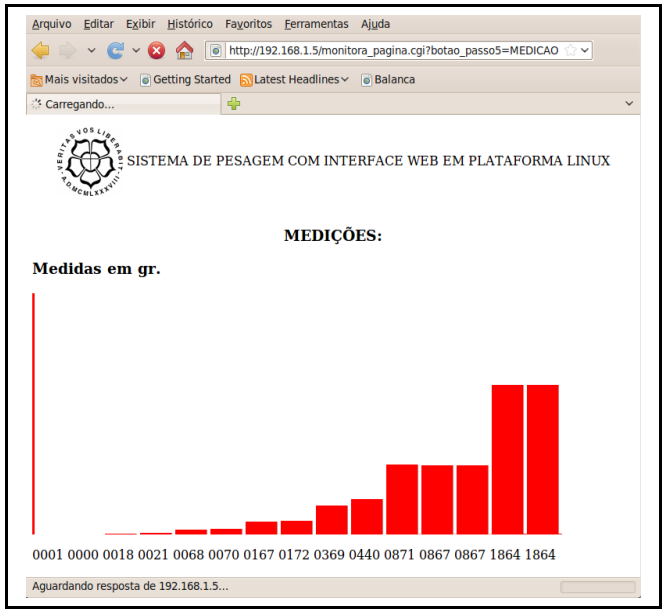

Figura 4.16 – Tela seis de interface *web*

A figura 4.17 ilustra uma sequência de dez medidas do peso padrão de um quilograma.

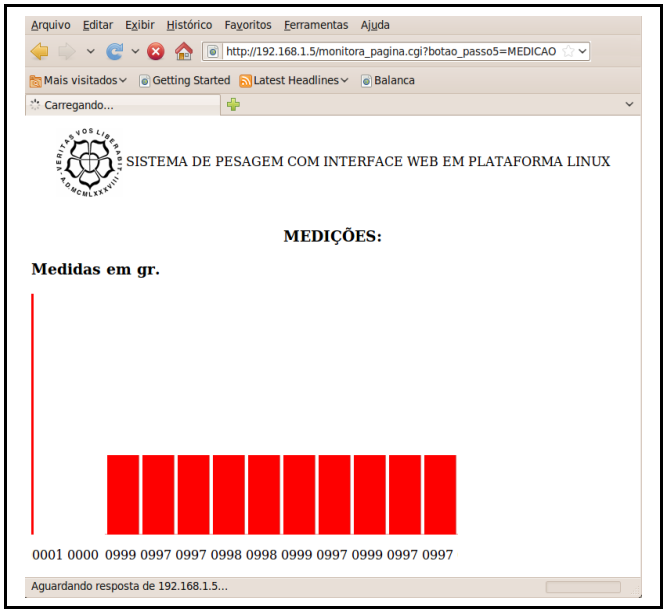

Figura 4.17 – Tela sete de interface *web*

4.5. *Visão completa do sistema*

Na figura 4.18 o resultado final do sistema. Um computador externo estabelece a comunicação com a CPU para obter as medições de peso.

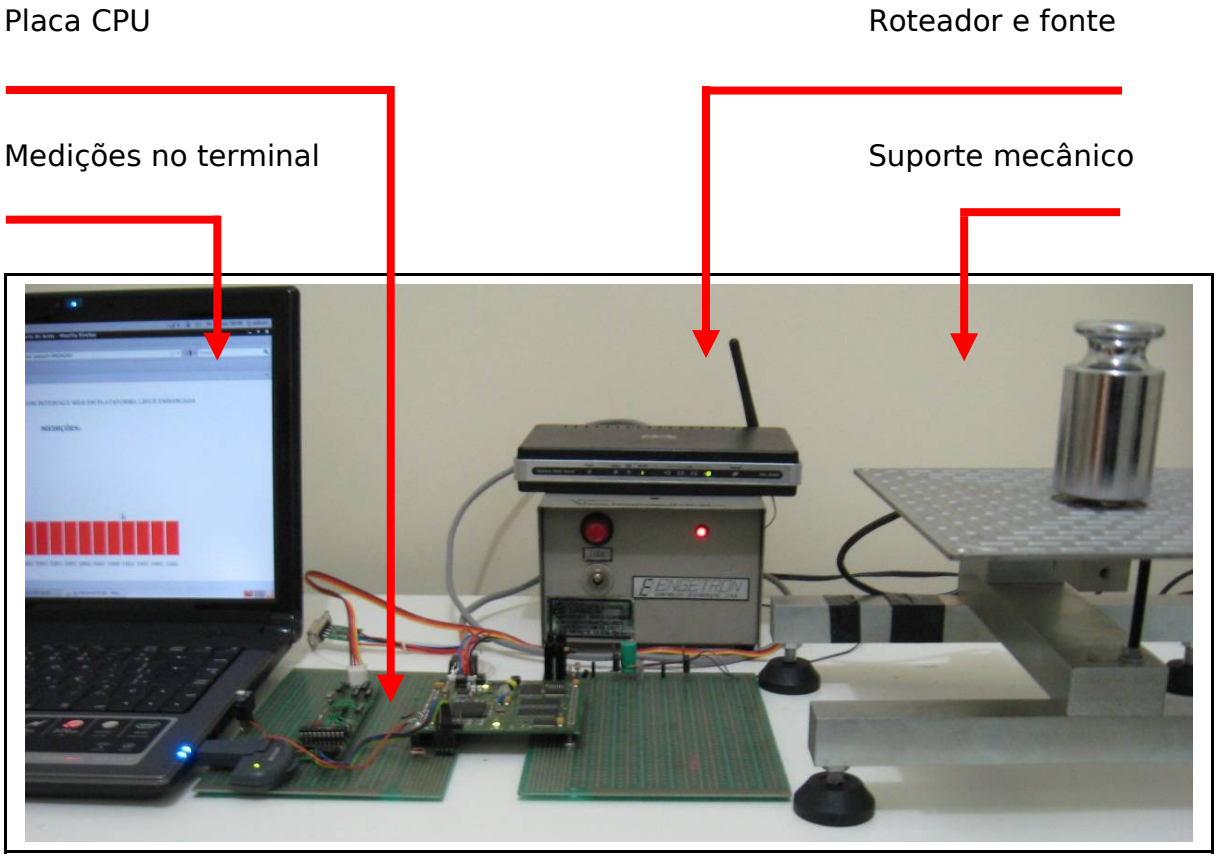

Figura 4.18 – Sistema completo funcionando

5. CONSIDERAÇÕES FINAIS

Foi apresentado um sistema de monitoramento de peso permitindo interação remota. Uma quantidade de variáveis pontuais foi abordada e permitiu o andamento do trabalho. Essas variáveis dizem respeito às escolhas feitas no andamento do projeto e junto com elas algumas dificuldades foram encontradas.

As ferramentas de *software* livre, assim como a inovação que traz para as áreas tecnológicas, também nos remetem ao universo de muita dedicação para o entendimento. Muitas informações ou documentações da internet são distorcidas e testes aprofundados sempre precisam ser feitos.

No decorrer do projeto estiveram às características de *hardware* para medição, *firmware* de controle do *hardware* e a comunicação do equipamento com a *Web*. As definições para a criação de cada uma dessas partes demandaram estudos e precisaram ir amadurecendo durante o desenvolvimento.

O objetivo principal de monitorar as medidas de peso e ainda permitir um processo de calibração utilizando uma página *Web* foi alcançado e testes iniciais se mostraram satisfatórios. Um rápido levantamento aponta para um erro aceitável nas medições próximas de um quilo grama.

Testes em outros navegadores, assim como melhorias na interface para o usuário leigo poderiam ser desenvolvidas. Muitos profissionais da área técnica dizem que um projeto nunca acaba ficando apenas funcional. Talvez seja essa uma boa definição para este sistema. Ele hoje, nessa versão, está funcional e atende objetivo proposto.

As características que foram abordadas no decorrer desse projeto buscando interação remota de um equipamento, permitindo comandos de qualquer outro computador que esteja em rede qualificam o desenvolvimento. O fato de que na maquina utilizada para obter o acesso ao sistema não ser necessário à instalação de *software* específico indicam uma tendência para novos projetos.

Há sugestões de *hardware* de *firmware* e *software* da *web* para implementações futuras. No *hardware* devido ao potencial, pensando em um produto para automação comercial, sensores poderiam ser inseridos na bandeja da balança para auto-detecção de um objeto a ser pesado, como o equipamento estaria em rede, um alerta remoto poderia ser passado ao funcionário para se dirigir a balança e efetuar determinado procedimento evitando o tempo de espera do cliente.

Ainda no *hardware* um sistema integrado de *wi-fi* tornaria o sistema muito atrativo, evitando a necessidade de infra-estrutura de rede para novas implantações. No *firmware* novas funcionalidades de segurança e tratamento de erros específicos poderiam ser agregadas. No desenvolvimento HTML, garantir a segurança dos dados trafegados implementando criptografias para a validação de operações.

6. REFERÊNCIAS

[1] RUTZIG, Mateus Beck. Gerenciamento Automático de Recursos Reconfiguráveis Visando a Redução de Área e do Consumo de Potência em Dispositivos Embarcados 2008.

[2] SPINARDI, Marcelo Augusto. Projeto Otimizado de Células de Carga Extensométricas, 2006.

[3] BECK, João Carlos Pinheiro. Projeto, Construção e Análise de Células de Carga de Placa e de Anel, 1983.

[4] PORTELA A. e SILVA A. Mecânica dos Materiais, 1996.

[5] MEDEIROS, Carolina Brum. Apostila de Extensometria, 2008.

[6] Cirrus Logic, CS5531 – Datasheet, 2008.

[7] Motorola, MPC850 – Datasheet, 2002.

[8] BONATO, Alexsandro Cristóvão. Projeto VoIP – Sistema de Comunicação de Voz Sobre Internet Disponível em: <**http://www.lapsi.eletro.ufrgs.br/~bonatto/** voip/node4.html >. Acessado em: 7 janeiro 2010.

[9] GARBELLINI, Vitor Neves. Criando Dispositivos Embarcados com Linux, 2003.

[10] AGUIAR, André Luis Santos. Arquitetura TCP/IP, 2008.

[11] ANDREANI, Alexandre Cassimiro. Sistemas embarcados: estudo através de um servidor HTTP, 2007.

OBRAS CONSULTADAS

WILLIAM – Manual de CGI. Editora Makron Books. São Paulo, 1988.

SCHILDT, Herbert. C Total e Completo (3a. Edição). Editora Pearson Education doBrasil. São Paulo, 2002.

GLOSSÁRIO

Boot - Em computação é o termo para definir o processo de iniciação do computador

Ppcboot - É o boot da PowerPC

Gateway - É uma máquina intermediária geralmente destinada a interligar redes.

Kernel - Componente central do sistema operativo da maioria dos computadores.

Kernel - Componente central do sistema operativo da maioria dos computadores

Browser - Navegador

APÊNDICE A – *FIRMWARE* APLICAÇÃO *WEB*

```
#include <stdio.h>
#include <string.h>
#include <sys/stat.h>
#include <sys/mman.h>
#include <unistd.h>
#include <stdlib.h>
#include <asm/8xx_immap.h>
#include <asm/mpc8xx.h>
#include <asm/io.h>
#include <sys/ioctl.h>
#include <termios.h>
#include <term.h>
#include <sys/time.h>
#include <fcntl.h>

int main(int argc, char *argv[])
{
   char strtemp[30];
   char *query_string = getenv("QUERY_STRING");

   iniciaPowerPC();

   vstrSerial.BaudRate = B9600;

   //inicializa a serial para ser utilizada
   memset(uc_recebe_serial,0,sizeof(uc_comando));
   memset(uc_comando,0,sizeof(uc_comando));

   abre_serial(&vstrSerial);

   //Fundamental para funcionar e não pode ter nenhum printf antes
   printf("Content-type: text/html\r\n\r\n");

   //Testa os parametros passados pela página na avriável QUERY_STRING
   if (localiza_string("senha", strtemp, 20,&query_string[0]))
   {
      if(strcmp(strtemp,"123456")==0)
      {
         calibracao_passo1();
      }
      else
      {
         senha_incorreta();
      }
   }
   else if (localiza_string("botao_passo2", strtemp, 20,&query_string[0]))
   {
      if(strcmp(strtemp,"CALIBRAR")==0)
```

```
        {
            calibracao_passo2();
        }
    }

    else if (localiza_string("botao_passo3", strtemp, 20,&query_string[0]))
    {
            if(strcmp(strtemp,"CALIBRAR")==0)
        {
            calibracao_passo3();
        }

    }
    else if (localiza_string("botao_passo4", strtemp, 20,&query_string[0]))
    {
            if(strcmp(strtemp,"CALIBRAR")==0)
        {
            calibracao_passo4();
        }

    }
    else if (localiza_string("botao_passo5", strtemp, 20,&query_string[0]))
    {
            if(strcmp(strtemp,"MEDICAO")==0)
        {
            monitora_peso();
        }

    }
    exit(0);
}

void calibracao_passo1 (void)
{
//Imprime na tela o título do trabalh e o logotipo da ulbra
cabecalho_ulbra();

printf ("<p align=\"center\">Esvazie a balanca e pressione o bot&atilde;o!");
printf ("<form name=\"form1\" method=\"get\" action=\"monitora_pagina.cgi\">");
printf ("    <div align=\"center\">");
printf ("      <input type=\"submit\" name=\"botao_passo2\" value=\"CALIBRAR\">");
printf ("    </div>");
printf ("</form>");
printf ("</p>");
printf ("</body>");
printf ("</html>");
return;
}

void senha_incorreta(void)
{
//Imprime na tela o título do trabalh e o logotipo da ulbra
cabecalho_ulbra();

//Mensagens no formato HTML para o erro na senha
printf ("<form method=\"get\" action=\"monitora_pagina.cgi\">");
printf ("<div align=\"center\" class=\"style1\">SENHA INCORRETA ! </div>");
printf ("<p align=\"center\">");
printf ("\n");
printf ("Por favor digite a senha do sistema:");
printf ("<p align=\"center\"></BR>");
```

```
printf ("  <input type=\"password\" size=10 name=\"senha\"/>");
printf ("\n");
printf ("<p align=\"center\">");
printf ("  <input name=\"submit\" type=\"submit\" value=\"Envio de senha\" />");
printf ("</p>");
printf ("</body>");
printf ("</html>");
printf ("\n");

return;
}

void calibracao_passo2 (void)
{
   unsigned int leituraAD;

   cabecalho_ulbra();

   //Implementar apagar o arquivo com o valor da tara

   printf ("<p align=\"center\">Inicializando o conversor AD!");
   if (iniciaConversorAd() != OK)
   {
      printf ("<p align=\"center\">Inicializando o conversor AD = ERRO");
   }

   //guarda valor inicial de peso
   if (executa_conversao(PEDE_CONVERSAO_CH1) != OK)
   {
      printf ("<p align=\"center\">executa_conversao(PEDE_CONVERSAO_CH1) != OK");
      return (NOK);
    }

    leituraAD = dadosCS5531[0] * 256;
    leituraAD = (unsigned int) leituraAD + dadosCS5531[1];
    printf ("<p align=\"center\">Primeira leitura do AD = %u",leituraAD);

   vlr_offset=(unsigned int) leituraAD;
   printf ("<p align=\"center\">Variavel vlr_offset = %u",vlr_offset);

   fp_offset = fopen("offset","w+");
   fwrite(&vlr_offset,sizeof(vlr_offset),1,fp_offset);
   fclose(fp_offset);

   vlr_offset = 0;

   fp_offset = fopen("offset","r+");
   fread(&vlr_offset,sizeof(vlr_offset),1,fp_offset);
   fclose(fp_offset);

   printf ("<p align=\"center\">Variavel do arquivo vlr_offset = %u ",vlr_offset);

   printf ("<p align=\"center\"> INSIRA O PESO PADRAO DE 1000 gr");

printf ("<form name=\"form1\" method=\"get\" action=\"monitora_pagina.cgi\">");
printf ("    <div align=\"center\">");
printf ("      <input type=\"submit\" name=\"botao_passo3\" value=\"CALIBRAR\">");
printf ("    </div>");
printf ("</form>");
printf ("</p>");
printf ("</body>");
```

```
printf ("</html>");

exit(0);
}

void calibracao_passo3 (void)
{
   cabecalho_ulbra();

   //Testa para a calibração um peso padrão de 1 kilo
   pesoPadrao = 1000;

   fp_offset = fopen("offset","r+");
   fread(&vlr_offset,sizeof(vlr_offset),1,fp_offset);
   fclose(fp_offset);

   if(Command(CMD_CALIB) != OK)
   {
      printf("Erro! Calibracao \n");
      return(NOK);
    }

printf ("<p align=\"center\">Calibracao ocorreu com sucesso");
printf ("<p align=\"center\">ESVAZIE A BALANCA");

printf ("<form name=\"form1\" method=\"get\" action=\"monitora_pagina.cgi\">");
printf ("    <div align=\"center\">");
printf ("      <input type=\"submit\" name=\"botao_passo4\" value=\"CALIBRAR\">");
printf ("    </div>");
printf ("</form>");
printf ("</p>");
printf ("</body>");
printf ("</html>");
}

void calibracao_passo4 (void)
{
   cabecalho_ulbra();

   printf ("<p align=\"center\">Ajustando a tara e salvando em arquivo");
   if(Command(CMD_TARAON) != OK)
   {
      printf("Erro ao executar tara");
      msleep(1000);
    }

   ValTara = 0;

   fp_tara = fopen("tara","r+");
   fread(&ValTara,sizeof(ValTara),1,fp_tara);
   fclose(fp_tara);

printf ("<p align=\"center\">Variavel do arquivo ValTara = %u ",ValTara);


printf ("<form name=\"form1\" method=\"get\" action=\"monitora_pagina.cgi\">");
printf ("    <div align=\"center\">");
printf ("      <input type=\"submit\" name=\"botao_passo5\" value=\"MEDICAO\">");
printf ("    </div>");
printf ("</form>");
printf ("</p>");
```

```
printf ("</body>");
printf ("</html>");
}

int executa_conversao(unsigned char setup)
{
   emset(uc_recebe_serial,0,sizeof(uc_comando));
   memset(uc_comando,0,sizeof(uc_comando));

   uc_comando[0]          = 'C';
   uc_comando[1]          = setup;

   write(vstrSerial.i_descriptor,uc_comando,2);

   msleep (200);

   if(ler_serial(vstrSerial.i_descriptor) == OK)
   {
      if (uc_recebe_serial[0] != ACK)
      {
         return (NOK);
      }
   }

   dadosCS5531[0] = uc_recebe_serial[1];
   dadosCS5531[1] = uc_recebe_serial[2];
   dadosCS5531[2] = uc_recebe_serial[3];
   dadosCS5531[3] = uc_recebe_serial[4];

   return(OK);
}

int iniciaConversorAd( void )
{
   memset(uc_recebe_serial,0,sizeof(uc_comando));
   memset(uc_comando,0,sizeof(uc_comando));

   //Monta o comando a ser enviado pela serial
   uc_comando[0]          = 0x49;
   uc_comando[1]          = 0x02;

   uc_comando[2]          = Ganho[0];
   uc_comando[3]          = Ganho[1];
   uc_comando[4]          = Ganho[2];
   uc_comando[5]          = Ganho[3];


   write(vstrSerial.i_descriptor,uc_comando,6);

   if(ler_serial(vstrSerial.i_descriptor) == OK)
   {
      if (uc_recebe_serial[0] != ACK){
      return (NOK);
   }
}
return(OK);
}

int Command (unsigned char tipo)
{
   unsigned int leituraAD;
```

```
   switch(tipo)
   {
      case CMD_TARAOFF:
         ValTara=0;
         return(OK);
         break;

      case CMD_TARAON:
         executa_conversao(PEDE_CONVERSAO_CH1);
         leituraAD = dadosCS5531[0] * 256;
         leituraAD = (unsigned int) leituraAD + dadosCS5531[1];
         ValTara=leituraAD;

         //cria o arquivo para backup da ValTara
            fp_tara = fopen("tara","w+");
            fwrite(&ValTara,sizeof(ValTara),1,fp_tara);
            fclose(fp_tara);
            return(OK);
            break;

      case CMD_CALIB:
         if (calibracao() != OK)
         {
            return(NOK);
         }
         break;
   }
   return (OK);
}
int calibracao(void)
{
   unsigned char  tolerancia;

   long  ganhoCalibracao;
   long  ganhoFinal;
   float fator;

   unsigned int  ajusteMinimoAceitavel;
   unsigned int  ajusteMaximoAceitavel;

   unsigned int  limiteMinimo;
   unsigned int  limiteMaximo;

   unsigned int  pesoMedido = 0;
   unsigned int  copiaLeituraAd;

   unsigned int leituraAD;

   tolerancia=100;

   //Aqui é o valor inicial que eu somo com o 03 qu coloquei no buffer ganho
   ganhoCalibracao=0x00100000;

   limiteMinimo=pesoPadrao-tolerancia;
   limiteMaximo=pesoPadrao+tolerancia;

   ajusteMinimoAceitavel=pesoPadrao-1;
   ajusteMaximoAceitavel=pesoPadrao+1;

   executa_conversao(PEDE_CONVERSAO_CH1);
```

```
   leituraAD = dadosCS5531[0] * 256;
   leituraAD = (unsigned int) leituraAD + dadosCS5531[1];

   pesoMedido=(uint)leituraAD-vlr_offset;

   copiaLeituraAd=leituraAD;

   //determina o ajuste para a balanca
   //Inicialmente o peso vai ser sempre maior ou bem menor que o peso padrão + ou - 5gr
   while((pesoMedido<=(ajusteMinimoAceitavel)) || (pesoMedido>=(ajusteMaximoAceitavel)))
   {
      //Aqui é apena para iniciar a calibracao, e faz o auste
      //mais grosso até que o peso seja + ou - 100gr
      while((pesoMedido<=(limiteMinimo)) || (pesoMedido>=(limiteMaximo)))
      {
         if(pesoMedido<=(limiteMinimo))
         {
         //carrega o buffer com valores intermediários de ganho
          (*((long *) Ganho)) = (*((long *) Ganho)) + ganhoCalibracao;
         }
         else
         {
            if(pesoMedido>=(limiteMaximo))
            {
               //carrega o buffer com valores intermediários de ganho
                ((long *) Ganho)) = (*((long *) Ganho)) - ganhoCalibracao;
            }
         }

         //atualiza o AD com novos valores de gannho
         iniciaConversorAd();

         executa_conversao(PEDE_CONVERSAO_CH1);

         leituraAD = dadosCS5531[0] * 256;
         leituraAD = (unsigned int) leituraAD + dadosCS5531[1];

         fator = (((float)leituraAD) / ((float)copiaLeituraAd));

         pesoMedido = (uint)(((float)leituraAD) - (((float)vlr_offset) * fator));

      }

      ganhoCalibracao>>=4;
      ganhoFinal = (*((long *) Ganho));

      imiteMinimo=ajusteMinimoAceitavel;
      limiteMaximo=ajusteMaximoAceitavel;

   }
   return (OK);
}

void monitora_peso (void)
{
   unsigned int leituraAD;
   unsigned int bkp_leituraAD;
   unsigned int contador = 0;

   cabecalho_ulbra();
```

```
executa_conversao(PEDE_CONVERSAO_CH1);

leituraAD = dadosCS5531[0] * 256;
leituraAD = (unsigned int) leituraAD + dadosCS5531[1];

fp_tara = fopen("tara","r+");
fread(&ValTara,sizeof(ValTara),1,fp_tara);
fclose(fp_tara);

leituraAD = leituraAD - ValTara;

if ((leituraAD < 0) || (leituraAD > 3000))
{
   leituraAD = 0;

}

fp_leituraAD = fopen("leituraAD","a+");
fwrite(&leituraAD,sizeof(leituraAD),1,fp_leituraAD);
fclose(fp_leituraAD);


bkp_leituraAD = leituraAD / 10;


fp_medicoes = fopen("medicoes","a+");
fwrite(&bkp_leituraAD,sizeof(bkp_leituraAD),1,fp_medicoes);
fclose(fp_medicoes);

printf("<h3 align=\"center\"> MEDI&Ccedil;&Otilde;ES: </h3>");
printf("<h3 align=\"left\">Medidas em gramas. </h3>");
printf("<img src=\"figuras/barra.jpg\" width=\"3\" height=\"%d\" />",300);

fp_medicoes = fopen("medicoes","r+");
while(!feof(fp_medicoes))
{

   fread(&bkp_leituraAD,sizeof(bkp_leituraAD),1,fp_medicoes);
   printf("<img src=\"figuras/barra.jpg\" width=\"45\" height=\"%d\" />",bkp_leituraAD);

}
fclose(fp_medicoes);
printf("</body>");
printf("<p>");

fp_leituraAD = fopen("leituraAD","r+");
while(!feof(fp_leituraAD))
{
   fread(&leituraAD,sizeof(leituraAD),1,fp_leituraAD);
   printf("  %04d  ",leituraAD);
   contador = contador + 1;
   if(contador >= 25)
   {
      break;
   }
}
fclose(fp_leituraAD);
if(contador >= 25)
{
   system("rm -f /mnt/nfs/web.pagina/pg.desenvolvida/medicoes");
   system("rm -f /mnt/nfs/web.pagina/pg.desenvolvida/leituraAD");
```

```
   }
printf("<HEAD>");
printf("<META HTTP-EQUIV=REFRESH CONTENT=1>");
printf("</HEAD>");
printf("</html>");
   return;
}

int localiza_string(char *name, char *value, int maxlen, char *entrada)
{
   char *pos1, *pos2;
   char *query_string = entrada;
   int success = 0;

   if(query_string)
   {
      pos1 = strstr(query_string, name);
      if(pos1)
      {
         pos2 = strstr(pos1, "&");

         unsigned char temp=0;
         unsigned char temp2=0;
         char str[500];

         while(pos1[temp++]!='=');

         while(pos1[temp]!='&'&&pos1[temp]!='\0')
         {
            str[temp2++] = pos1[temp];
            temp++;
         }
         str[temp2]='\0';
         strcpy(value, str);
         success = 1;
      }
   }

   return success;
}

void cabecalho_ulbra (void)
{
printf ("<!DOCTYPE html PUBLIC \"-//W3C//DTD XHTML 1.0
Transitional//EN\" \"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd\">");
printf ("<html xmlns=\"http://www.w3.org/1999/xhtml\">");
printf ("<head>");
printf ("<meta http-equiv=\"Content-Type\" content=\"text/html; charset=windows-1252\" />");
printf ("<title>Formulario de teste</title>");
printf ("<style type=\"text/css\">");
printf ("<!--");
printf (".style1 {color: #FF0000}");
printf ("-->");
printf ("</style>");
printf ("</head>");
printf ("<body>");
//printf ("<p> </p>");
printf ("<p align=\"center\"><span class=\"Centralizado\">");
printf ("<img src=\"figuras/image003.gif\" width=\"84\" height=\"85\" align=\"absmiddle\"
v:shapes=\"_x0000_i1026\" /></span><span class=\"Capa\"> SISTEMA DE PESAGEM COM INTERFACE
WEB");
```

```
printf ("    <st1:PersonName");
printf ("ProductID=\"EM PLATAFORMA LINUX EMBARCADA\" w:st=\"on\">EM PLATAFORMA
LINUX");
printf ("      EMBARCADA</st1:PersonName>");
printf ("</span></p>");
printf ("\n");

return;
}
```

ANEXO A – CONCEITOS BÁSICOS DE LINUX

Inicialização do sistema linux

Muitas pessoas, ao ligarem um computador, nem imaginam que existem várias etapas a serem seguidas até o sistema operacional estar realmente carregado na memória e assim estar pronto para a utilização do usuário. Será dada uma pequena introdução de como isso se procede com os principais conceitos necessários para o entendimento. A abordagem a seguir terá sua base um PC comum, porém o processo é basicamente o mesmo em todas as plataformas [9].

O processo de boot envolverá uma série de passos e conceitos, listados a seguir, como ilustra a Figura.

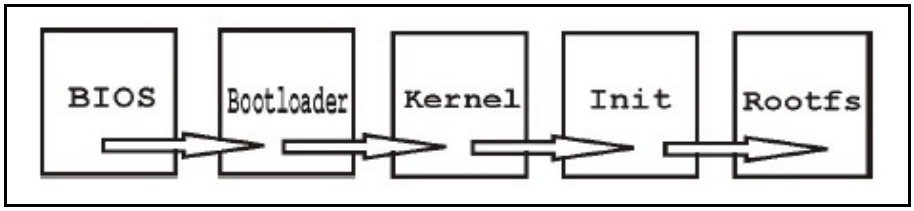

Figura – Passos de inicialização de um sistema Linux

Quando o usuário liga o sistema, a primeira etapa a ser executada é um teste de *hardware* chamado "*Power-On Self-Test*", onde o sistema verifica a integridade de seus componentes e faz uma comparação com as configurações do último uso, armazenadas na BIOS. A BIOS é um programa que fica armazenado em memória ROM, ela contém rotinas básicas de entrada e saída e executa funções necessárias para inicialização de *hardware* [9]. Se tudo ocorrer bem, o sistema prossegue com o processo verificando na BIOS qual a sua primeira fonte de inicialização (também conhecido como *boot*).

Caso o sistema admita, por exemplo, um *Hard Disk* como sua primeira fonte para o boot, ele irá acessar a primeira trilha do disco (cilindro 0, cabeça 0, setor1) chamada de MBR (*Master Boot Record*), onde encontrará todas as

informações necessárias para dar continuidade ao processo através de um *bootloader* [9].

Bootloader ou gerenciador de *boot*, é o nome que se dá ao código localizado na MBR responsável em dar continuidade a inicialização do sistema, carregando o *kernel*. Neste caso, este código reside nos primeiros 446 bytes da MBR (de um total de 512 bytes). Quando carregado, lista todas as partições existentes através de um arquivo de configuração [9].

Os *bootloaders* mais conhecidos que suportam a plataforma x86 são o LILO (*LInux Load*) e o GRUB (*GRand Unified Bootloader*), cada um tem uma forma diferente de proceder com a inicialização e fazer o acesso ao *kernel*. Podem estarem configurados para inicializar vários sistemas operacionais, ou até um mesmo sistema operacional com versões de *kernel* diferentes.

No entanto, existem diversos outros *bootloaders*, capazes de inicializar plataformas diferentes. Como são voltados para sistemas embarcados, acabam agregando funcionalidades específicas como gravação/limpeza de *flash*, *boot* remoto via nfs ou tftp, consoles seriais para acesso, etc. [9].

Alguns exemplos de *bootloaders*:

u-boot: Suportas as plataformas ARM, PowerPC e MIPS. O u-boot é utilizado em sistemas Linux.

MicroMonitor: Suporta as plataformas ARM, SH2, ColdFire e XScale. É utilizado para carregar vários sistemas operacionais, dentre eles: Linux, eCos, pSOS, etc.

Redboot: Suporta boot remoto (tftp), seu código fonte é aberto. Muito utilizado pela Intel.

LinuxBIOS: Suporta as arquiteturas x86 e PowerPC, é regido sob licença GPL. É inteiramente programado em C.

Kernel Linux

A palavra *kernel* foi muito difundida com o surgimento do sistema Linux, mas este está presente em todos os sistemas operacionais. É o núcleo do sistema operacional, representa a camada mais baixa de interface com o *hardware*, sendo responsável por gerenciar os recursos do sistema computacional como um todo. É nele que estão definidas funções para operação com os periféricos do sistema, gerenciamento de memória (MMU), entre outros.

Podemos definir que o *kernel* é um *software* que fornece aos aplicativos do usuário uma interface para que utilize de forma correta os recursos do sistema [9]. São geralmente desenvolvidos em linguagens de baixo nível como C ou *Assembly*.

O *kernel* Linux surgiu em 1991, quando Linus Torvalds iniciou a idéia de *software* livre. O *kernel* do Linux é disponibilizado livremente sob licença GPL2, no seu site próprio (http://*kernel*.org/). Como está sempre em evolução e a busca por melhorias é constante, suas versões são fornecidas sempre com três algarismos de identificação. O primeiro e o segundo indicam a série do *kernel*, enquanto o terceiro indica a versão da série. O segundo algarismo é usado para diferenciar entre versões de desenvolvimento e de produção. Se for um número ímpar significa que é uma série ainda em desenvolvimento, ou seja, uma versão instável em fase de testes e aperfeiçoamento. Caso seja par indica uma série estável e pronta para o uso. Estes números vão sendo incrementados conforme novos recursos são adicionados ou alterados, salientando que nem sempre é preciso rodar sua última versão, a não ser que nela tenha algum suporte específico necessário para sua aplicação ou *bugs* críticos solucionados [9].

Alguns exemplos de versões do *kernel*:

- 2.6.2 - Versão estável, série par (6).
- 2.4.5 - Versão estável, série par (4).
- 2.5.19 - Versão instável, série ímpar (5).

A Tabela 2.2 mostra como o *kernel* Linux, ou o sistema operacional Linux, evoluiu desde a sua criação em 1991 até os dias de hoje, salientando algumas de suas versões que ficaram conhecidas por suas novas e marcantes inovações no sistema [9].

Tabela – Evolução do *Kernel*

Ano	***Kernel***	**Características**
1991	0.02	Versão pública oficial
1994	1.0	Suporte a uma única arquitetura (i386)
1995	1.2	Outras arquiteturas: Alpha, Mips, Sparc
1996	2.0	Modularidade e Multiprocessamento(2)
1999	2.2	USB, ISDN, Masc.IP, Vídeo4Linux
2001	2.4	IPTables, Multiproc.(16), Sup.64MB RAM
2003	2.6	+Preempitível, Melhor Escalabil., uClinux

Como o Linux é livre, seu *kernel* é disponibilizado junto de seus arquivos fontes e podemos configurá-lo antes de compilá-lo, ou seja, é possível definir um *kernel* Linux específico para sua utilização, o que é uma grande vantagem especialmente quando se fala de sistemas embarcados.

initrd

O initrd é um pequeno sistema de arquivos montado na partida pelo *kernel* Linux. É usado para auxiliá-lo na montagem do sistema raiz (*rootfs*). Através do initrd é possível tratar a diversidade de plataformas existentes de maneira mais geral.

O initrd pode, por exemplo, ser capaz de fazer uma busca do sistema raiz nas diversas fontes de armazenamento do sistema, como discos, *pendrives*, CDs, etc. O initrd pode também, por exemplo, permitir que um *rootfs* compactado seja corretamente acessado. Em geral, este tipo de funcionalidade específica não é suportada pelo *kernel* Linux diretamente.

Uma outra utilidade é a de ter a capacidade de carregar um determinado módulo, dependendo da configuração de *hardware* no momento da inicialização, deixando o sistema mais genérico, mais simples e consequentemente mais rápido.

Pode-se figurar como uma etapa na inicialização do sistema onde podem ser executados vários *scripts* com funções diversas, como por exemplo, fazer uma atualização no sistema com o intuito de encontrar alguma alteração dos dispositivos de *hardware* para passar estas informações ao *kernel*.

No caso dos sistemas embarcados o initrd pode não ter muita utilidade, uma vez que o *kernel* destes dispositivos já é configurado especificadamente para prover o suporte ideal de acordo com o *hardware* do sistema, dificilmente alterado ao longo de sua vida útil.

No entanto, caso o sistema embarcado tenha o sistema raiz fornecido remotamente, pode ser interessante considerar uma abordagem de partida com o uso do initrd.

rootfs

O *rootfs* é o conjunto das aplicações e serviços de um sistema baseado em Unix, é carregado pelo *kernel* na inicialização do sistema para assim poder ser executado e estar à disposição do usuário.

Cada sistema tem apenas um *rootfs*, ele pode estar comprimido em apenas um arquivo (imagem), comum em sistemas embarcados, pode ser uma partição (sistemas comuns), pode estar em uma rede ou mesmo em um disco SCSI, Sata ou IDE.

Após subir o *rootfs* para a memória do sistema, o *kernel* executa o *script* **/sbin/init** que contém instruções de como proceder com a inicialização do sistema. O init é responsável por finalizar o processo de inicialização do sistema. Ele requisita o arquivo **/etc/inittab**, que é constituído de uma tabela que mantém passo a passo de como isso dever ser feito [9].

O init é o serviço pai que o sistema executa na inicialização (*bootup*), sendo responsável pela inicialização de outros serviços requeridos como por exemplo sh, telnet, ftp, *web* server e etc.

Na Tabela a seguir é apresentada uma estrutura padrão de um *rootfs*. Cada um de seus diretórios tem uma função especifica e alguns apenas são necessários em sistemas de multiusuários e/ou servidores. Como a maioria dos sistemas embarcados com Linux não são multiusuários, esta árvore pode ser bem mais simples do que um *rootfs* de um sistema comum.

Tabela – Estrutura básica de diretórios do sistema operacional.

Diretório	**Conteúdo**
bin	Binários essenciais
boot	Arquivos estáticos utilizados pelo bootloader
dev	Arquivos especiais e dos dispositivos
etc	Arquivos de configuração do sistema
home	Diretórios do usuário
lib	Bibliotecas essenciais, incluindo C e módulos do *kernel*
mnt	Ponto de montagem dos sists. de arquivos temporários
opt	Pacotes de *softwares* extras
proc	Sist. de arquivos virtual, para o *kernel* e outros procs.
root	Diretório do administrador
sbin	Binários de administração do sistema
tmp	Arquivos temporários
usr	Contém a maioria das aplicações do usuário
var	Armazenamento de valores de variáveis

Na Tabela abaixo alguns exemplos de funcionalidades típicas que acompanham um sistema mesmo que bem reduzido

Tabela – Estrutura básica de serviços do sistema operacional.

Serviço	**Nome Programas.**
Arquivo	cp, ls, mv, rm, rmdir, chmod, sync
Editor	cat, more, vi, find, grep
Rede	ping, ifconfig, netstat, route
Processo	kill, ps, insmod, lmmod, rmmode