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

AUTOMAÇÃO DE LEITURA DE MEDIDORES DE CONSUMO DE ENERGIA

Autor: Bruno Mesquita Santana

Orientador: Professor M.C. José Julimá Bezerra Júnior

Brasília

2007

BRUNO MESQUITA SANTANA

AUTOMAÇÃO DE LEITURA DE MEDIDORES DE CONSUMO DE ENERGIA

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

Orientador: Prof. M.C. José Julimá Bezerra Júnior

Brasília

2007

AGRADECIMENTOS

À Deus.

À minha mãe, meu pai, meu irmão e minhas irmãs, as pessoas mais importantes na minha vida, que sempre me apoiaram e estiveram presentes. À minha avó "Tê", sem a qual não teria feito este trabalho o mais esmerado possível. Aos meus avós Santana e Magnólia que mesmo à distancia sempre se preocuparam com meus estudos, saúde e bem estar. Ao meu avô Ildeu, quem sempre esteve interessado em saber dos meus projetos e planos.

Aos tios e tias, especialmente à tia Selma, tia Ana, tio Marcelo, tio Getúlio e tia "maryloo", pessoas muito queridas que participaram ativamente do meu desenvolvimento pessoal e também intelectual. Aos meus primos e primas, companheiros de sempre: Rafa, Gabi, Dani, Didi e Carol. Ao tio Macedo que me ajudou a conseguir informações a respeito dos medidores de energia e da concessionária local.

À Roberta que presenciou o início deste trabalho e me ajudou em momentos cruciais da vida. Aos meus amigos, peças fundamentais na qualidade de vida: "Carlins", "Fabinho", Fernando, Alex, Ivan, Leonardo, Marquete, Alessandro, Bianca.

Aos professores do UniCEUB que me ajudaram e passaram o mais importante: conhecimento! Ao meu orientador que depositou confiança no meu trabalho. Ao UniCEUB, que proporcionou um curso de qualidade e realização pessoal.

À Isabella que me auxiliou e presenciou os últimos momentos desta realização sempre com muito carinho, segurança e bom humor.

Obrigado a todos que não citei mas que têm ciência da importância que exerceram neste processo único da minha vida.

RESUMO

Atualmente, a leitura do consumo de energia elétrica é realizada presencialmente por técnicos das concessionárias de energia. Esta medição é encaminhada à empresa para processar as informações e, posteriormente, enviada por correio juntamente com a fatura para que os consumidores realizem o pagamento. Para reduzir os custos e o tempo levados nesta operação foi proposta, neste projeto, uma solução para realização da leitura remota dos medidores de consumo de energia empregados. Para executar esta tarefa foram projetados um acoplador óptico, um circuito (contador e transmissor de informações) e os softwares de recebimento e apresentação da leitura.

Palavras-chave: automação, consumo de energia, medidor de energia, microcontrolador, acoplador óptico, foto-sensor, *LASER* linguagem de programação C e PHP, banco de dados, MySQL, Apache.

ABSTRACT

Nowadays, the measure of the eletrical energy consumption of the customers is executed presently by the electricity company employees. This measurement is sent to the company to process the data and after that mailed to the customers with the bill, so they can pay it. To reduce the costs and time taken in this process a solution to do the remote reading of the electrical measures used was proposed in this project. To execute this job the author designed an optical acoplator, a data and counter circuit and the softwares of receving and reporting the measurement.

Keywords: automation, energy consumption, energy measurement, microcontroller, optical acoplator, photosensor, *LASER*, C and PHP programming language, MySQL database, Apache web server.

SUMÁRIO

ÍNDICE DE FIGURAS

ÍNDICE DE TABELAS

LISTA DE ABREVIAÇÕES

ABNT – Associação Brasileira de Normas Técnicas

ANEEL – Agência Nacional de Energia Elétrica

ANSI – Instituto Nacional de Padrões Americanos

BIT – Binary Digit

BPS – Bits Per Second

CEB – Companhia Energética de Brasília

CENELEC – Comité Européen de Normalisation Electrotechnique

Conversor A/D – Conversor Analógico/Digital

DB9 – Conector utilizado em comunicação serial RS232 de 9 pinos

DCE – Data Circuit-terminating Equipament

DTE – Data Terminal Equipament

EIA – Eletronic Industries Association

ETSI – European Telecommunications Standards Institute

GND – Pólo "terra" de um circuito eletrônico

GSM – Global System for Mobile Communications

Hayes Command Set, Comandos AT – abreviação de *attention*

HTTP – HyperText Transfer Protocol

IEC – International Electrotechnical Commission

ISO – International Organization for Standardization

LASER – Light Amplification by Stimulated Emission of Radiation

LED – Light Emission Diode

Oscilador RC – Resistor-Capacitor

PCI – Protocol Control Information

PDU – Protocol Data Unit

Perl – Pratical Extraction and Report Language

PHP – Personal Home Page

PIC – Programmeable Integrated Circuits

RISC – Reduced Instrunction Set Computer

RM-OSI – Reference Model – Open Systems Interconnection

RS232 – Padrão de comunicação serial EIA (EIA232F)

RX – Via de recepção de dados na comunicação serial

SDU – Service Data Unit

SMS – Short Message System

TX – Via de transmissão de dados na comunicação serial

USART – Universal Synchronous Receiver Transmitter

USB – Universal Serial Bus

LISTA DE SÍMBOLOS

p	Representação de 10^{-12}
m	Representação de 10^{-3}
k	Representação de 10^{3}
M	Representação de 10^{6}
m	Metro
W	Potência elétrica
V	Tensão elétrica em *Volts*
I	Corrente elétrica
A	Corrente elétrica em *Ámperes*
Hz	Freqüência em *Heartz*
F	Capacitância em *Faraday*
h	Hora
Ω	Resistência elétrica em *Ohms*

1 INTRODUÇÃO

Este capítulo tem como objetivo apresentar as idéias principais e a estrutura organizacional do projeto, facilitando assim o entendimento de todo o trabalho desenvolvido ao longo da pesquisa.

1.1 Objetivos

Pesquisa e implementação com vistas à solução para leitura remota do consumo de energia elétrica onde há uso de medidores eletromecânicos, seja devido ao tempo que estão instalados ou a outras necessidades de natureza financeira, regional etc. Realizou-se os estudos identificando as dificuldades conseqüentes dos já existentes medidores eletromecânicos e ainda, propondo meio eficaz, prático e dinâmico de leitura dos mesmos. Tais estudos não implicaram, necessariamente em vínculos com nenhuma concessionária.

Deu-se continuidade e finalizou-se estudos iniciados em 2004 no Projeto de Iniciação Científica – PIC/UniCEUB interrompidos por motivos pessoais.

1.1.1 Objetivos específicos

Projetou-se e construiu-se um circuito leitor, utilizando os medidores eletromecânicos e comunicação sem fio (*wireless*).

Entende-se por inteligente um equipamento capaz de:

I – Realizar conversão analógica para digital dos medidores já instalados nos consumidores;

II – Transmitir os dados lidos;

A transmissão dos dados é realizada via *wireless* (por *Short Message System* – SMS, em português Sistema de Mensagem Curta) utilizando telefonia celular, controlada pelo equipamento construído e é recebida por um servidor, que trata a informação e provê interface com os usuários do sistema. O envio por SMS se deu pelo uso de infra-estrutura já

existente de empresa de telecomunicação que atende à tecnologia SMS.[1]

O projeto procurou minimizar o erro de leitura, atendendo a um erro relativo de no máximo 3%, atendento a um nível de qualidade comercial de instrumentação.

1.2 Motivação

O fornecimento de energia elétrica à população é tarefa das concessionárias de energia e é, também, considerada uma indústria de interesse público, pois representa a estrutura material através da qual se realiza o serviço publico. Essa atribuição, que é regulamentada pela Agência Nacional de Energia Elétrica – ANEEL, muitas vezes encontra obstáculos para ser realizada em locais de difícil acesso, mesmo em centros urbanos.

Além do fornecimento, as concessionárias enfrentam dificuldades periódicas para executar medição de seus serviços prestados ao consumidor. Essa medição é essencial para a correta cobrança e manutenção do fornecimento de energia.

Neste caso, o estudo de uma maneira mais eficaz e a prática de realização deste tipo de medição faz-se economicamente interessante, uma vez que equipamentos em locais como esse, são mais baratos, embora conseqüentemente ultrapassados, como os medidores de consumo eletromecânicos, comumente chamados *relógios de luz*.

---

[1] Esta pesquisa não abrange estudos inerentes ao funcionamento destas empresas.

1.3 Visão Geral

Na Figura 1 é possível visualizar e entender as várias partes do projeto desenvolvido:

Figura 1 – Visão geral do projeto.

O medidor eletromecânico é composto por um disco que gira proporcionalmente ao consumo de energia.

Nele é acoplado um emissor *LASER* (*Light Amplification by Stimulated Emission of Radiation*) e um foto-sensor (Acoplador Óptico). Com o uso deste acoplador pode-se contar a quantidade de voltas que o disco realiza, pois neste disco existe um "marco zero", representado por uma faixa preta na sua borda.

A luz do *LASER* é refletida com menos intensidade quando passa por esta faixa preta. O sensor capta todas as variações de luz, inclusive quando passa pela faixa preta do disco. O sinal de saída do sensor é conectado ao microcontrolador que converte o sinal elétrico em código binário através de conversor A/D (Analógico/Digital).

O microcontrolador armazena a contagem de voltas que o disco realiza e periodicamente transmite os dados ao celular que, por sua vez, envia um SMS ao servidor.

Este servidor faz o tratamento dos dados recebidos e os disponibiliza para consulta pelo consumidor.

1.4 Organização da Monografia

A monografia está separada em cinco capítulos. Um breve epítome destes

capítulos é realizado a seguir:

O Capítulo 1 apresenta as idéias e a estrutura organizacional do projeto.

No Capítulo 2 são levantados os embasamentos teóricos necessários para o desenvolvimento do projeto, os conceitos apresentados pela estrutura específica, bem como os estudos advindos das pesquisas recentes, porém, sem a preocupação de aprofundar estes conhecimentos.

No terceiro capítulo são mostrados os passos, os materiais usados e a forma como o projeto foi implementado. Para realizar a montagem, o autor fez uso de muitos equipamentos que por si só não tinham como escopo a função que exercem, e por isso algumas dificuldades tiveram que ser transpostas, as quais estão citadas no capítulo seguinte.

No Capítulo 4 são apresentados os resultados obtidos nos testes com o projeto implementado, ressaltando as dificuldades encontradas durante o processo de trabalho.

Finalmente, no Capítulo 5 são relatadas as conclusões, os resultados finais, os efeitos práticos derradeiros do desenvolvimento do projeto, demonstrando como um sistema de leitura e comunicação *wireless* (via telefonia celular) pode substituir os métodos atuais de leitura.

Os documentos utilizados como referência, os códigos fontes, as imagens que enriquecem o trabalho e ainda, qualquer declaração escrita que agregou valor ao projeto foram anexados à monografia.

2 EMBASAMENTO TEÓRICO

O embasamento teórico contempla os conhecimentos necessários para desenvolvimento deste projeto. Alguns dos conteúdos aqui expostos podem ter maior aprofundamento em seu teor, porém, isso não foi necessário para apoiar o estudo desenvolvido neste projeto, especificamente.

2.1 Medição de Energia Elétrica

A importância em mensurar a energia[2] elétrica consumida está em realizar um faturamento coerente com o gasto de cada cliente. Executar esta medição portanto é uma tarefa realizada pela concessionária de energia, a qual utiliza medidores eletromecânicos do tipo indução, por serem de simples manuseio, robustos, apresentando exatidão e desempenho ao longo dos anos (MEDEIROS, 1997, p.167).

De modo simplificado, a energia elétrica pode ser entendida como uma mercadoria comercializada pelas concessionárias de energia. Porém os medidores são instalados na localidade em que se encontra o cliente, tendo portanto, a concessionária que realizar uma leitura periódica no local onde o medidor está instalado.

2.1.1 Medidores de energia elétrica

Os medidores de energia elétrica, como foi dito, somam a potência elétrica consumida ao longo do tempo e disponibilizam estes dados de forma inteligível às pessoas. Para facilitar esta leitura, a unidade utilizada pelos medidores é o kWh (lê-se "quilo-watt-hora"), onde:

$$kWh \tag{1}$$

$$W = \text{Potência} = \text{Tensão em volts x Corrente em apères } \therefore W = V \times I \tag{2}$$

---

[2] **"**Energia é tudo aquilo capaz de produzir calor, trabalho mecânico, luz, radiação, etc. Em sentido geral, poderia ser definida como substrato básico de todas as coisas, responsável por todos os processos de transformação, propagação e interação que ocorrem no universo. A energia elétrica é um tipo especial de energia através da qual podemos obter os efeitos acima; ela é usada para transmitir e transformar a energia primária da fonte produtora que aciona os geradores em outros tipos de energia que usamos em nossas residências." (CREDER, 2000)

$$W = \text{hora} \tag{3}$$

2.1.1.1 Princípios de funcionamento

Na Figura 2, pode-se ver a bobina de tensão (ou bobina de potencial e que é altamente indutiva com grande número de espiras de fio fino) ligada em paralelo com a carga, a bobina de corrente (com poucas espiras de fio grosso) ligada em série com a carga e o disco de alumínio, formando o conjunto móvel. A energia elétrica ao passar por uma bobina, gera um campo magnético que interage com o campo magnético gerado pela outra bobina. Isto é chamado de fenômeno da interação eletromagnética. Este fenômeno gera uma força que faz com que o disco gire em torno de seu eixo. Cada volta que o disco realiza equivale a um consumo de energia, informado pelo fabricante do disco com constado do disco (*kd*). O eixo, por sua vez faz girar registradores que fornecem a leitura. No disco de alumínio são observadas algumas marcações numéricas de 0 a 100 e no marco zero é gravada uma faixa de aproximadamente 1,5 cm preta (CREDER, 2000, p.30).

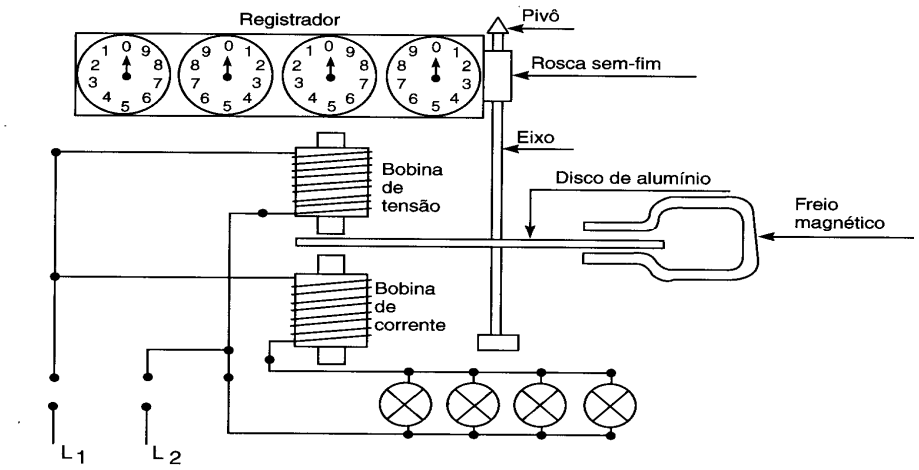

Figura 2 – Funcionamento interno do medidor eletromecânico (CREDER, 2000, p.30).

2.2 O Acoplador Óptico

O acoplador óptico é a parte do projeto responsável por realizar a conversão de uma leitura física e visual do medidor de energia em um sinal elétrico, que permita ser tratado pelo microcontrolador. O acoplador óptico é composto pelos dispositivos que realizam

esta conversão: o emissor *LASER* e o sensor óptico.

2.2.1 O emissor LASER

O emissor *LASER – Light Amplification by Stimulated Emission of Radiation* – dentre muitos materiais, pode ser um componente eletrônico do tipo LED – *Light Emission Diode*. Ou seja, um dispositivo que usa a emissão estimulada de ondas eletromagnéticas na região do visível (entre o infravermelho e o ultravioleta) para produzir um feixe intenso de fótons coerentes. [3]

O funcionamento deste diodo especial consiste em um bloco semicondutor (junção PN-GaAlAS), que por intermédio de uma baixa corrente produzirá oscilações nesta junção. Estas oscilações gerarão colisões e recombinam elétrons e lacunas, emitindo fótons ou elementos de luz. Por se mostrar o mais econômico, estável, com poucas dimensões e boa durabilidade, tornou-se o modelo mais popular para as aplicações técnicas em leitura de dados (MALVINO, 1995, p. 161).

2.2.1.1 Classificação dos produtos LASER

Em todo mundo, existem um grande número de órgãos com diferentes classificações para os produtos *LASER*s, desde o IEC – *International Electrotechnical Commission* ou o *Comité Européen de Normalisation Electrotechnique* – CENELEC ou ainda os órgãos do Japão, Austrália e Estados Unidos. No Brasil, o órgão responsável pela padronização é a ABNT – Associação Brasileira de Normas Técnicas, que ainda não tem uma classificação muito ampla, ou seja, que atenda à ampla possibilidade de uso dos *LASERs*. As duas únicas normas relacionadas à tecnologia *LASER* e constantes do acervo da ABNT são: 1) NBR 14588 de 09/2000, que prescreve a determinação do raio de encurvamento em fibras ópticas pelos métodos por vista lateral e por reflexão de feixe *LASER*; 2) NBR IEC 60601-2-22 de 10/1997, norma internacional adotada no Brasil e que prescreve os requisitos particulares para a segurança de equipamentos *LASER* utilizados em diagnósticos e terapias. Como se pode perceber, não se regula o uso do *LASER* em procedimentos que não sejam médicos. O que se tem e pode ser associado é, que baseado na Tabela I a seguir (ver fonte em

---

[3] GEODESIA ONLINE. Disponível em <http://geodesia.ufsc.br/Geodesia-online/arquivo/cobrac_2002/010/010.htm>. Acesso em 29/05/07.

anexo), que relaciona classes e padrões internacionais, que podem ser consultados nas tabelas em anexo a este documento.

Tabela I – Classificação dos *LASERs*.

Classe	IEC:1993	IEC:2001	CDRH1	ANSI
1	*	*	*	*
1M		*		
2	*	*	*	*
2A			*	
2M		*		
3A	*		*	*
3R		*		
3B	*	*	*	*
4	*	*	*	*

2.2.2 O sensor

Um sensor é um dispositivo tecnológico responsável por detectar um sinal ou condição física[4]. Um foto-sensor ou foto-detector, portanto, é o dispositivo que detecta luz, convertendo-a num sinal elétrico (REZENDE, 2004, p. 323).

2.2.2.1 Fototransistor

Um transistor é um dispositivo semicondutor capaz de amplificar sinais (MALVINO, 1995, p. 201). Este dispositivo pode ser utilizado de diversas maneiras em um circuito eletrônico, dentre eles como um fototransistor, que consiste num transistor com a base aberta. Neste dispositivo o ganho de corrente no emissor, varia de acordo com a intensidade de luz na junção emissor-base (REZENDE, 2004, p. 335).

2.3 Microcontroladores

O microcontrolador, também chamado de microcomputador de um só chip[5],

---

[4] SENSOR, WIKIPEDIA A ENCICLOPÉDIA ONLINE. Disponível em <http://pt.wikipedia.org/wiki/Sensor>. Acesso em 01/06/2007.

[5] *Chip* é um sinônimo de circuito integrado (INTEGRATED CIRCUIT. WIKIPEDIA THE ONLINE ENCICLOPEDIA.Disponível em <http://en.wikipedia.org/wiki/Integrated_circuit>. Acessado em

está revolucionando o projeto de sistemas eletrônicos digitais, devido a sua enorme versatilidade de hardware e seu poderoso software (SILVA JÚNIOR, 1998, p. 6-10).

Um microcontrolador é um componente eletrônico que já tem incorporado em seu invólucro, várias funcionalidades periféricas a um microprocessador, permitindo dessa forma, a construção de sistemas compactos tão poderosos quanto os baseados em microprocessadores (SILVA JÚNIOR, 1998, p. 34).

O que diferencia os diversos tipos de microcontroladores são: a quantidade de memória interna para armazenar dados e as instruções de programas (memória de programa e memória de dados), a velocidade de processamento, a quantidade de pinos de I/O, a forma de alimentação, os tipos e as quantidades de periféricos, a arquitetura e o conjunto de instruções disponibilizado nos circuitos internos (SILVA JÚNIOR, 1998, p. 15).

2.3.1 O PIC

PIC ou *Programmeable Integrated Circuits* é um microcontrolador desenvolvido pela *MICROCHIP Technology Inc*, empresa precursora no uso de tecnologia RISC (*Reduced Instrunction Set Computer*) em microcontroladores. Este microcontrolador é vantajoso porque apresenta pequeno tamanho físico, facilidade de programação e de manuseio, baixo consumo, além de flexibilidade (SILVA JÚNIOR, 1998, p.13).

No que diz respeito à facilidade de programação, os microcontroladores da família PIC, tem uma grande vantagem: o uso da linguagem C, utilizada neste projeto (PEREIRA, 2003, p. 18).

Quanto aos periféricos contemplados por este microcontrolador, os que são importantes para a efetivação do projeto são: conversor A/D, pinos de I/O, uso de *clock* externo e o uso comunicação serial.

2.3.1.1 Conversor A/D

Um sinal de tensão elétrica para ser tratado digitalmente exige sua transformação, ou seja, a conversão em sinal digital. O microcontrolador PIC realiza

---

01/06/2007).

conversão A/D[6] internamente.

Todo conversor A/D tem uma limitação (em *bits*[7]) quanto à resolução do seu conversor A/D, ou seja, num conversor A/D de 10 bits a resolução é (SOUZA e LAVÍNIA, 2005, p. 115):

$$resolução = \frac{V_{ref}}{2^n} \tag{4}$$

Em que V_{ref} é a tensão referência e *n* o numero de *bits* do CAD.

Cada um dos *n* bits que formam o dado digital simboliza uma parcela do valor da tensão analógica a ser convertida, de tal forma que a soma de todas as contribuições de cada um dos *n* bits forma a tensão de entrada do conversor A/D. Portanto, a parcela de tensão proporcional ao bit m do conversor A/D é dada por (SOUZA e LAVÍNIA, 2005, p. 115):

$$V_{entrada} = \frac{b_m \times 2^{(m-1)}}{2^n} \times V_{ref} \tag{5}$$

Onde b_m é o valor do bit *m*, ou seja, zero ou um.

É importante salientar os recursos do PIC que se deseja. No caso do PIC 16F877A, as características do conversor A/D são:

- Conversor de 10 bits;
- Até oito canais de conversão;
- Quatro tipos de referência: V_{dd} (interna), V_{ss} (interna), V_{ref+} (externa), V_{ref-} (externa);
- Freqüência de conversão baseada no *clock* da máquina ou em *Resistor-*

[6] Conversor Analógico/Digital
[7] Um **bit** (***B**inary Digi**t***) é a menor unidade da informática.

Capacitor – RC – dedicado, possibilitando o modo *sleep*;

- Três ajustes de freqüência (divisores) para uso do *clock* de máquina;
- Dois tipos de justificação para o resultado da conversão: direita e esquerda;
- Um interrupção para o término da conversão.

Um fator para se atentar é o tempo de conversão que o microcontrolador leva para concluí-la. Para que o sistema de conversão funcione corretamente, um *clock* deve ser aplicado a ele. Cada período de *clock* é chamado de T_{ad} e é equivalente ao tempo de conversão de 1 *bit*. O tempo total para conversão de 10 *bits*, portanto, é $10\,T_{ad} + 2\,T_{ad}$, onde a soma de $+\,2\,T_{ad}$ ocorre para que o sistema adeque o início da conversão (SOUZA e LAVÍNIA, 2005, p. 121).

2.3.1.2 Pinos de i/o

Os pinos de entrada e saída de um microcontrolador são partes fundamentais para a escolha do modelo desejado. Isto porque estes pinos servem para capturar dado (no caso de pinos configurados como entrada) de periféricos como teclados, sensores etc., ou externar informações (pinos configurados como saída) para displays, interface serial, ou mesmo um computador pessoal – PC. Sendo assim, um projeto deve levar em consideração a quantidade de pinos que vai utilizar de forma a obter a melhor relação custo benefício do hardware adquirido.

Em alguns casos particulares, os pinos de entrada/saída podem ter mais de uma função atribuída a eles, sendo este tipo de atribuição realizada via programação. Além disso, nestes casos particulares, os pinos devem obedecer aos requisitos do fabricante, o que pode ser observado no *datasheet*[8] do produto. Neste projeto, o *datasheet* do microcontrolador utilizado foi inserido como anexo.

---

[8] *Datasheet* é o manual do equipamento provido pelo fabricante. Nele se encontram as características físicas, elétricas e, em alguns casos, a melhores práticas de uso do equipamento.

2.3.1.3 Clock externo

O *clock* responde pelo sincronismo de todas as operações de um microcontrolador. Isto é conseguido com o uso de um dispositivo externo, como um cristal chamado de oscilador. Em outras palavras, o oscilador externo é responsável por temporizar os ciclos de máquina. Os programas (softwares) gravados no microcontrolador executam em ciclos de máquina, ou seja, uma sub-rotina de um programa necessita de tantos ciclos quantos forem necessários para executarem por completo. Portanto, sendo o oscilador responsável por temporizar os ciclos, é ele também o responsável pelo tempo que uma sub-rotina leva para executar por completo (SOUZA e LAVÍNIA, 2005, p. 26).

No caso do oscilador, existem quatro modelos diferentes de osciladores:

- LP – Cristal de baixa potência;
- XT – Cristal / Ressonador;
- HS – Cristal / Ressonador de alta freqüência;
- RC – Oscilador RC (*Resistor-Capacitor*).

Especificamente, para o uso de cristal do tipo XT, devem ser colocados dois capacitores cerâmicos ligados com o "terra" (representado pela sigla GND). E ainda, para o cristal com freqüência de 4MHz, a capacitância do capacitor deve ser de 15pF (SOUZA e LAVÍNIA, 2005, p. 273)..

2.3.1.4 Comunicação serial do microcontrolador

As técnicas de comunicação serial têm avançado sensivelmente nos últimos tempos, sendo utilizada em protocolos como o USB – *Universal Serial Bus*, Serial ATA etc. No caso de comunicação serial com o microcontrolador, uma grande vantagem é a utilização do USART interno. USART ou *Universal Synchronous Receiver Transmitter* é um padrão universal para comunicação em dois diferentes modos de trabalho: o sincronizado e o não-sincronizado.

No modo assíncrono, a comunicação é realizada por meio de duas vias de tráfego de dados, sendo uma de transmissão (TX) e outra de recepção (RX). A transmissão e

recepção podem ocorrer simultaneamente, recurso conhecido como *Full Duplex* (PEREIRA, 2003, p.262).

Como apenas duas vias são utilizadas para transmissão de dados, a forma que se realiza o sincronismo dos dados é conhecido como *Baud Rate* (ou velocidade de transmissão). O *Baud Rate* é um valor que deve ser adotado igualmente de ambos os lados da comunicação (receptor e transmissor) e é descrito como a quantidade de *bits* enviados por segundo (representado pela sigla *bps*). Sendo assim, é possível calcular o tempo de duração de cada *bit* (SOUZA e LAVÍNIA, 2005, p. 243 a 255):

$$T_{bit} = \frac{1}{BaudRate} \tag{6}$$

Para conseguir um sincronismo na transferência de dados, este tipo de comunicação utiliza também um *bit* de início, chamado de *start bit*. Como o estado padrão das vias de comunicação é em nível alto, conhecido como *stand-by*, para iniciar a comunicação, o canal transmissor, força seu TX para nível baixo, mantendo-o baixo por um tempo T_{bit} . O receptor reconhece este sinal em nível baixo como o *start bit*. Em seguida, o transmissor enviará os 8 *bits* de dados com tempo de duração T_{bit} . O receptor, tendo conhecimento do início da comunicação, após ter recebido o *start bit*, fará a leitura dos bits de dados, lendo-os aproximadamente no meio do tempo de cada *bit*, $T_{bit}/2$. Finalmente, o transmissor envia um *bit* de finalização, *stop bit*, que é um *bit* com duração T_{bit} em nível alto, garantindo o modo *stand-by* da via de comunicação. Caso o receptor receba o *stop bit* em nível baixo, sabe-se então que ocorreu erro na comunicação de dados (SOUZA e LAVÍNIA, 2005, p. 243 a 255).

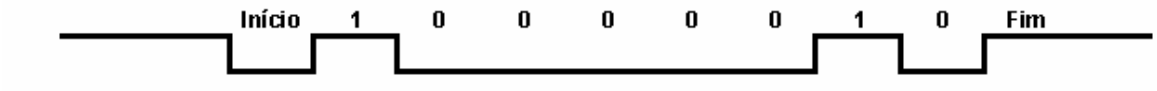

Figura 3 – Comunicação serial.

No caso de comunicações padronizadas, o *Baud Rate* obedece a valores pré-

estabelecidos como 9.600, 19.200 bps (SOUZA e LAVÍNIA, 2005, p. 243 a 255).

Outra característica da comunicação serial assíncrona é o *bit* de paridade que é a confirmação matemática dos 8 *bits* de dados, porém o *bit* de paridade é opcional.

Para o microcontrolador, algumas diretivas podem ser definidas (em linguagem C, descrita mais adiante neste trabalho) de forma a atender a comunicação serial assíncrona:

Tabela II – Parâmetros de comunicação serial em linguagem C (PEREIRA,2003, p. 173)

Opção	Descrição
BAUD = valor	Especifica a velocidade de comunicação serial (em bits por segundo).
XMIT = pino	Especifica o pino de transmissão de dados (TX).
RCV = pino	Especifica o pino de recepção de dados (RX).
RESTART_WDT	Determina que a função GETC() re-configure o *watchdog* enquanto aguarda a chegada de um caractere.
INVERT	Inverte a polaridade dos pinos de TX/RX. Não pode ser utilizada com hardware interno.
PARITY = x	Seleciona a paridade. Sendo que "x" pode assumir os valores: N = Sem paridade E = Paridade par O = Paridade ímpar
BITS = x	Seleciona o número de bits de dados (5 a 9 para o modo por software e 8 a 9 para o modo por hardware).
FLOAT_HIGH	A saída não vai em nível lógico 1. Utilizado com saídas coletor aberto.
ERRORS	Solicita ao compilador que armazene os erros de recepção na variável RS232_ERRORS, re-configurando os *flags* de erro quando eles ocorrerem.
BRGH10K	Permite a utilização das velocidades disponíveis com o bit BRGH em 1 *chips* que tenham *bugs* nesta configuração do hardware.
ENABLE = pino	Especifica um pino para atuar como saída de habilitação durante uma transmissão. Utilizado no protocolo RS485.
STREAM = identificador	Associa a interface RS232 a um identificador de *stream* de dados. Os *streams* de dados são utilizados em algumas funções internas do compilador.

2.4 Comunicação Serial

Nesta seção, algumas informações complementares são apresentadas.

2.4.1 RS232

O RS232 foi gerado da necessidade de se padronizar a comunicação entre equipamentos produzidos por diferentes fabricantes. Este padrão especifica sinais de tensões, sinais de tempo, sinais de função, protocolo para troca das informações e os conectores físicos a serem utilizados para realização da comunicação. A *Eletronic Industries Association* – EIA, responsável por este padrão, publicou apenas três modificações nos últimos 40 anos, sendo o mais recente a renomeação na nomenclatura para EIA232F. Na comunicação entre equipamentos utilizando o EIA232F é importante denominar os equipamentos participantes da comunicação, sendo aquele no final da comunicação o DTE – *Data Terminal Equipament* e é dotado de um conector DB9[9]. O equipamento responsável pela comutação dos dados (modem) é o DCE – *Data Circuit-terminating Equipament*, dotado também de um DB9[10].

Na figura 4 e 5 é possível verificar a relação de pinos e suas funções de um conector DB9 DTE e DCE respectivamente:

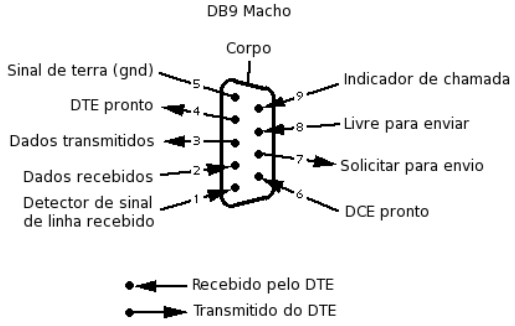

Figura 4 – Conector DB9 DTE.

---

[9] A especificação DB9 é padronizada pela EIA, da seguinte forma: A primeira letra, "D", representa o formato do conector. A segunda letra representa o números de pinos do conector. Inicialmente os conectores de 9 pinos eram representados pela letra "E" (DE-9), porém com a grande expansão dos computadores IBM, que erroneamente chamavam os conectores de 9 pinos de DB-9 (que pelo padrão representa o conector de 25 pinos), o uso da expressão DB-9 caiu no gosto popular (RS-232, WIKIPEDIA, THE ONLINE ENCICLOPEDIA. Disponível em <http://en.wikipedia.org/wiki/RS232>. Acesso em 01/06/2007).

[10] RS232 STANDARD. Disponível em <http://www.camiresearch.com/Data_Com_Basics/RS232_standard.html>. Acesso em 23/05/2007.

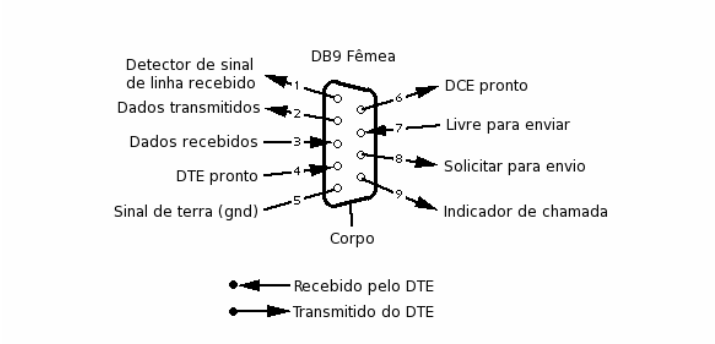

Figura 5 – Conector DB9 DCE.

Na comunicação serial com modens e celulares GSM[11], os comandos de envio das mensagens são chamados comandos AT e o SMS é codificado em modo PDU, conforme padrão utilizado pelo aparelho celular.

2.5 Comandos AT

Originalmente conhecidos como *Hayes Command Set*, os comandos AT (abreviação de *attention*), é uma linguagem de programação desenvolvida para operar alguns modelos de telefones celulares remotamente[12].

Atualmente cada fabricante tem uma especificação própria para seus aparelhos, o que geralmente são variações dos comandos originais (*Hayes Command Set*), sendo os padrões mais utilizados, comumente, os GSM 07.07 e GSM 07.05[13].

Geralmente, a comunicação com um aparelho celular, utilizando comandos AT ocorre da seguinte forma:

```
AT+COMANDO=?
```

---

[11] GSM – *Global System for Mobile Communications*, originalmente *Groupe Spécial Mobile*, é um padrão de comunicação utilizado mundialmente e que permite *roaming* entre diferentes operadoras de celular. (GSM, WIKIPEDIA THE ONLINE ENCICLOPEDIA. Disponível em <http://en.wikipedia.org/wiki/GSM>. Acesso em 02/06/2007). Faz-se importante enfatizar que a tecnologia SMS também esta presente em outras plataformas de comunicação celular como o CDMA – Acesso Múltiplo por Divisão de Código, porém este projeto não estuda tais tecnologias.

[12] HEYES COMMAND SET. Disponível em <http://en.wikipedia.org/wiki/Hayes_command_set>. Acesso em 31/05/2007.

[13] Padrões especificados pela ETSI – European Telecommunications Standards Institute.

Onde AT+ é a inicialização da comunicação, "COMANDO=" é literalmente o comando a ser executado pelo aparelho celular e "?" representa os parâmetros utilizados pelo comando.

Para enviar um SMS, via comando AT, o comando completo seria:

```
AT+CMGS=160<CR>PDU<ctrl-Z>
```

Em que "160" é o tamanho da mensagem enviada, <CR> significa *Carriage Return*, PDU é a informação em si e <ctrl-Z> o fim da comunicação.

2.5.1 PDU

Antes de definir PDU ou *Protocol Data Unit* – Unidade de Dados do Protocolo, é interessante compreender os padrões de comunicação. O objetivo da existência de padrões de comunicação é fornecer uma base comum que permita o desenvolvimento coordenado de padrões para interconexão de sistemas. A ISO (*International Organization for Standardization*) é uma organização que objetiva elaborar padrões internacionais como o das comunicações, os chamados RM-OSI (*Reference Model – Open Systems Interconnection*), ou somente modelo OSI (SOARES, LEMOS e COLCHER, 1995, p. 126). No modelo OSI, um PDU é entendido como o DADO (SDU – *Service Data Unit* – Unidade de Dados do Serviço) juntamente com um cabeçalho chamado de Informação de Controle do Protocolo (*Protocol Control Information* – PCI). Portando PDU pode ser definido como "a unidade de informação trocada pelas entidades pares (fim-a-fim), ao executar o protocolo de uma camada, para fornecer o serviço que cabe à camada em questão" (SOARES, LEMOS e COLCHER, 1995, p.137).

2.6 Programação

O tratamento das informações nos vários módulos do projeto é alcançado via programação. Programação é o trabalho de desenvolver programas com objetivos específicos, o que no caso deste projeto pode ser dividida de acordo com a sua interface. Porém existem apenas duas linguagens de programação utilizadas nas diferentes interfaces do projeto, o PHP e o C, explicados a seguir.

2.6.1 Linguagem de programação C

A linguagem de programação C é uma das linguagens mais utilizadas pela sua facilidade de entendimento, pela vantagem da compilação em várias plataformas e a possibilidade de programar em "alto" ou "baixo" nível, ou seja, aplicações do tipo *front-end*, que fazem interface com usuário, no caso do "alto" nível ou quando realizam instruções diretamente com o hardware, no caso do "baixo nível". O ANSI – *American National Stardard Institut* padroniza a linguagem chamada ANSI C.

2.6.2 Linguagem de programação PHP

Inicialmente chamado de *Personal Home Page*, o PHP foi desenvolvido em linguagem *Perl*[14] para documentar o número de acesso (visitas) que os *sites*[15] recebiam. Ele foi reescrito em C para resolver problemas de *forking*[16] de processos do servidor *Web*[17], e com o passar do tempo foi divulgado para os muitos usuários da internet que começaram a participar no desenvolvimento de novas características do PHP, agora conhecido como *Hypertext Preprocessor*, uma linguagem de *scripts*, que permite manipular códigos HTML (hipertexto)[18].

A vantagem em utilizar o PHP é que, assim como o MySQL e o Apache, esse é um código aberto multi-plataforma, o que dá liberdade de migrar o projeto para o ambiente que for mais usual dentro da concessionária de energia. Além disso, o PHP permite executar programas nativos do Sistema Operacional – S.O., tendo os resultados armazenados em variáveis tornando possível, por exemplo, comunicação serial para leitura de SMS com programa já desenvolvido pela sociedade de software livre (o *getsms*[19] – no caso do S.O. *Linux*[20]*)*.

---

[14] Perl – *Pratical Extraction and Report Language* é uma linguagem otimizada para "varrer" arquivos de texto arbitrariamente, procurando por informações e imprimindo relatórios baseados nestas informações. Informação obtida no manual *Perl* versão 5.8.7 de 16/12/2005, disponível nas distribuições GNU/LINUX.

[15] Forma popular como as páginas de internet são disponibilizadas.

[16] A tradução de *forking* seria bifurcação.

[17] Foma como as operações, disponibilizações e troca de informações na internet é chamada.

[18] PHP. Disponível em < http://br2.php.net/manual/phpfi2.php>. Acesso em 01/06/2007.

[19] O getsms é um programa que serve para ler o primeiro SMS disponível nos modems GSM (padrão 07.05) ou compatíveis. (Disponível no manual do GNU/LINUX)

[20] O kernel Linux junto com diversos aplicativos formam o GNU/LINUX, um sistema operacional de código aberto, ou seja, que qualquer usuário com conhecimentos de sistemas operacionais pode alterar/customizar. (MOTA, 2006, p. 60)

2.7 Banco de Dados

Banco de dados ou, como é realmente chamado, Sistema de Gerenciamento de Banco de Dados – SGBD, são servidores que realizam a organização, disponibilização e manutenção de informações. Eles existem em diferentes tipos, dentre eles os relacionais, que permitem organizar os dados de forma que obedeçam a relacionamentos obrigatórios entre os diferentes dados.

2.7.1 MySQL

O servidor de banco de dados MySQL é senão o mais utilizado, o mais popular banco de dados dentre os desenvolvedores atualmente para uso em pequenos sistemas e *Websites*. Isto acontece porque além de ser gratuito, o MySQL é multi-plataforma e de código aberto, ou seja, pode ser instalado em qualquer sistema operacional, permitindo que cada usuário faça modificações em seu código conforme a necessidade de sua aplicação. Além destas facilidades, o MySQL oferece suporte gratuito em seu *website*, além de diferentes comunidades de suporte espalhados pelo mundo (HOAG, 2002, p. 376).

2.8 Servidor *Web Apache*

Apache é um programa servidor de HTTP – *HyperText Transfer Protocol* ou protocolo criado para apresentar informações estáticas do servidor *Web* no navegador do usuário. Este programa é executado em *background* pelo sistema operacional e atende às requisições feitas pelo navegador do usuário.

O Apache era uma servidor *Web* com poucos recursos, projetado para ser executado apenas como um serviço em ambiente *Unix*. Em meados de 1995, um grupo ou comunidade reorganizou o projeto testando novas funcionalidades e homologando-as para lançar um produto que hoje é o servidor *Web* mais popular e um dos mais seguros da Internet (HOAG, 2002, p.XI)

Atualmente, o *Apache* é executado na maioria dos sistemas operacionais, dentre eles o *MS-Windows*[21] e o *Linux*[22], sendo a versão mais atual lançada a 2.2.4, que

[21] Sistema operacional desenvolvido pela empresa Microsoft Inc.
[22] Sistema operacional mantido pela comunidade de software livre.

também possibilita o uso de módulos como o PHP (*Personal Home Page*), usado para a disponibilização da leitura do consumo por este projeto (HOAG, 2002, p.375).

2.9 Navegador de *Internet e Intranet*

As redes de computadores fechadas, ou seja, que apenas um grupo de usuários pode acessar formam as Intranets enquanto que a rede mundial de computadores, onde qualquer pessoa conectada a ela consegue acessar informações é chamada de Internet. Dentre os vários meios de disponibilização de informações existentes para as Intranets e a Internet, um que se destaca é o HTTP (disponibilizado pelo servidor Apache[23]), descrito anteriormente. Para que um usuário possar acessar tais dados ele precisará de uma ferramenta conhecida como Navegador de Internet e Intranet, ou ainda, *Web Browser*. Existem muitos modelos destas ferramentas, porém duas são amplamente usadas: o Firefox e o Internet Explorer.

[23] NAVEGADOR, WIKIPEDIA A ENCICLOPÉDIA ONLINE. Disponível em <http://pt.wikipedia.org/wiki/Navegador>. Acesso em 01/06/2007.

3 A MONTAGEM DO PROJETO

Neste capítulo, são explicados os passos executados para implementação do projeto, mostrando os equipamentos e dispositivos utilizados e ainda, os *softwares* desenvolvidos.

3.1 O Medidor de Energia Utilizado

O medidor utilizado para implementação deste projeto foi emprestado pela Companhia Energética de Brasília – CEB; e é um modelo fabricado pela *Actaris Metering Systems*, uma empresa líder no mercado de tecnologias em medições elétricas, de gás, água e calor[24].

3.1.1 SL 1621

Este modelo de medidor (SL 1621) é utilizado em redes elétricas com tensão Fase-Neutro de 240 *Volts*, freqüência nominal de 60 Hz e o que é mais importante ter conhecimento, uma constante kd = 3,6 Wh/r, ou seja, a cada rotação do disco principal equivale um consumo de 3,6 Watts x Horas.

3.1.1.1 Características e dimensões

A Tabela III (ver fonte em anexo) mostra as principais características do medidor utilizado no projeto, das quais se pode destacar a freqüência de operação de 60 Hz e o mostrador ciclométrico com cinco registradores.

Este medidor trabalha com tensão de 240 *Volts* (uma fase de 380 *Volts*), sendo um elemento, dois fios, 15/100 *Àmperes*.

---

[24] ACTARIS METERING SYSTEMS. Disponível em <http://www.actaris.com/html/eng/COMPANY/1.html>. Acesso em 01/06/2007.

Tabela III – Características do medidor SL1621.

Discriminação	Unid.	1621-240	Discriminação	Unid.	1621-240
Freqüência nominal	Hz	60	Alimentação	-	Linha/Carga
Corrente máxima em % da corrente nominal	%	666,66	Corrente de partida em % da corrente nominal	%	0,5
Velocidade do elemento móvel para carga nominal	rpm	16 2/3	Conjugado motor para carga máxima	gf.cm	3,9
Margens de calibração Carga nominal Carga pequena Carga indutiva	 % % %	 +/-3 +/-4 +/-2	Perdas em cada circuito Ativa Aparente	W VA	1,1 5
Perdas em cada circuito de corrente com:					
Corrente nominal Ativa Aparente	 W VA	 0,22 0,28	Corrente máxima Ativa Aparente	 W VA	 9,0 12,0
Massa do elemento móvel	g	20	Relação de acoplamento	-	50/1
Tipo do registrador	-	Ciclométrico	Número de indicadores - registrador	-	5
Tipo de mancal	-	Magnético	Número de discos	-	1
Massa do medidor C/tampa de vidro C/tampa de policarbonato	g g	1470 1245	Dimensões máximas (largura x altura x profundidade)	mm	139x173x115
Capacidade da bitola dos terminais - corrente	mm2	35	Terminais de prova		s/terminais ou internos
Tampa bloco terminal	-	Curta alumínio ou plástica	Tampa do medidor	-	Vidro ou policarbonato

Na Figura 6 (extraída do anexo) pode-se verificar as dimensões do SL 1621:

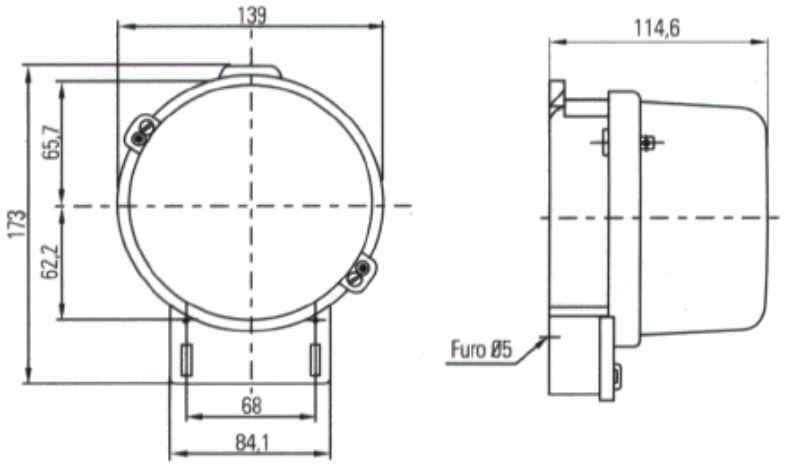

Figura 6 – Dimensões do SL1621.

3.2 Ligação Elétrica do Modelo de Consumo

O modelo de consumo, conforme pode ser visualizado na Figura 7, consiste de dois bocais de lâmpadas comuns, uma tomada universal fêmea onde está ligada uma resistência de aquecimento de água numa tábua de madeira, com fios de 3 mm, o próprio medidor Actaris SL 1621 e três interruptores. Na Figura 7 pode ser visualizada a ligação elétrica do modelo.

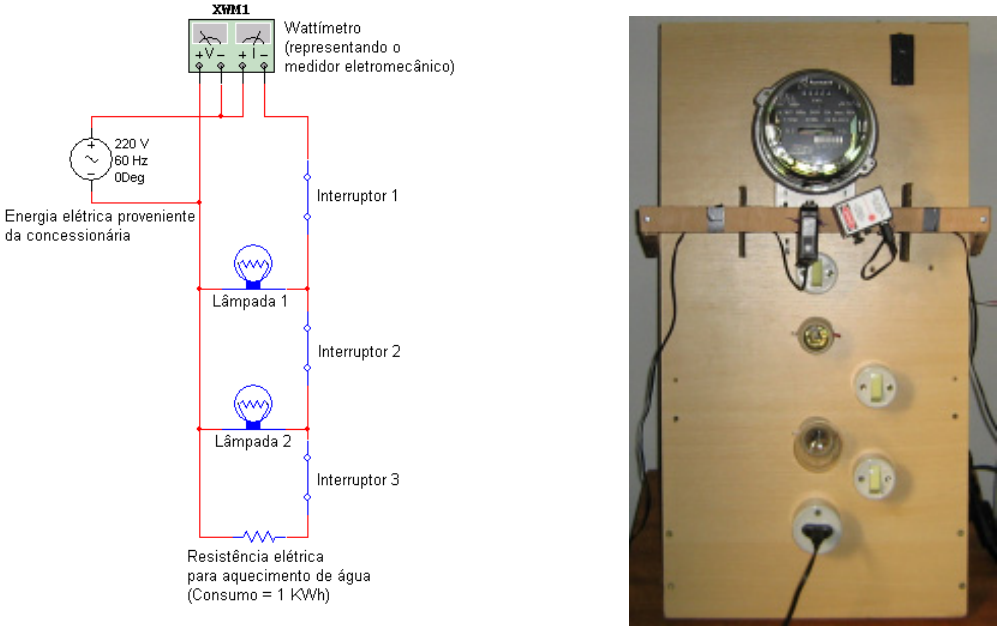

Figura 7 – Ligação elétrica

Na Figura 7 também é possível visualizar o sensor óptico e o emissor *LASER* descritos no próximo capítulo desta dissertação.

Porém, durante os primeiros testes, foi verificado que a luz ambiente juntamente com as lâmpadas do protótipo de consumo causavam interferência na leitura do sensor, conforme explicam os detalhes destes testes explorados no próximo capítulo. Portanto, foi necessário projetar uma forma de cobrir o protótipo de forma a bloquear esta interferência. Na Figura 9 pode-se verificar como ficou o protótipo coberto.

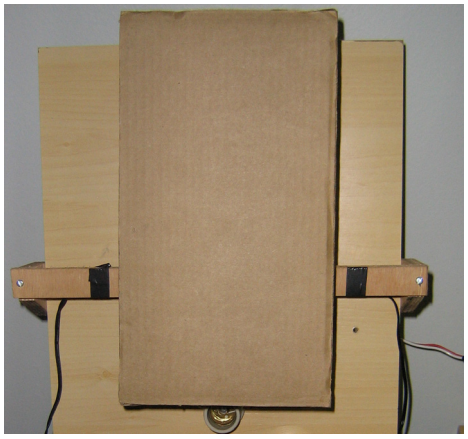

Figura 9 – Modelo de consumo coberto.

3.3 Construção do Acoplador Óptico

Uma das fases mais complexas na implementação do projeto com certeza foi a construção do acoplador óptico (ver capítulo 4, dificuldades com o acoplador óptico). O acoplador óptico, como foi explicado no capítulo 2.2 consiste de um emissor *LASER* e um foto-sensor.

3.3.1 LASER

O *LASER* utilizado neste projeto é o módulo OS-8528A, da Pasco (que o autor conseguiu emprestado do laboratório de física). Este componente emite um *LASER* classe II, de cor vermelha, comprimento de onda entre 660 a 680nm, potência de saída menor que 1mW, sendo alimentado por uma fonte de 9V DC, fornecida pelo fabricante (mais detalhes são fornecidos no *datasheet* em anexo). A Figura 10 mostra uma imagem deste emissor, onde pode-se ver a existência de uma base de regulagem. Esta base não foi utilizada no projeto.

Figura 10 – Emissor *LASER* da Pasco.

3.3.2 O sensor

Após muitos testes, o foto-sensor utilizado no projeto foi o componente encontrado no mercado de São Paulo, o L14G1 da FAIRCHILD SEMICONDUCTOR. Este componente atende aos requisitos do sistema, sendo um fototransistor de sensibilidade de 0 a 940nm, que é superior ao comprimento de onda do *LASER* (640nm). Outros detalhes podem ser verificados no *datasheet* do componente, em anexo.

Mesmo com o protótipo coberto, ainda havia ruído na leitura do sensor porque a caixa não isolou completamente o protótipo, portanto para filtrar os comprimentos de onda que nele incidem, foi construído um compartimento onde a entrada de luz tem uma fina camada de papel celofane vermelha de forma que outros comprimentos de onda não causem ruído no sinal do sensor, Figura 11:

Figura 11 – Foto-sensor com filtro de luz.

Na Figura 12 pode-se visualizar a *pinagem* do fototransistor:

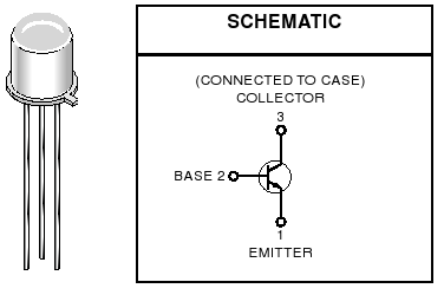

Figura 12 – Imagem do fototransistor.

A seção 2.2.2 explicou o funcionamento de um fototransistor: “o sinal de saída do fototransistor é proporcional à intensidade de luz entrante na sua junção base-emissor”, em outras palavras, para um fototransistor como o da Figura 13, alimentado em 5V DC, a tensão (mostrada pelo multímetro) será proporcional à luz que nele incide, variando de 0 a 5 *Volts* para uma completa ausência de luz à uma luz máxima (informada pelo fabricante), respectivamente.

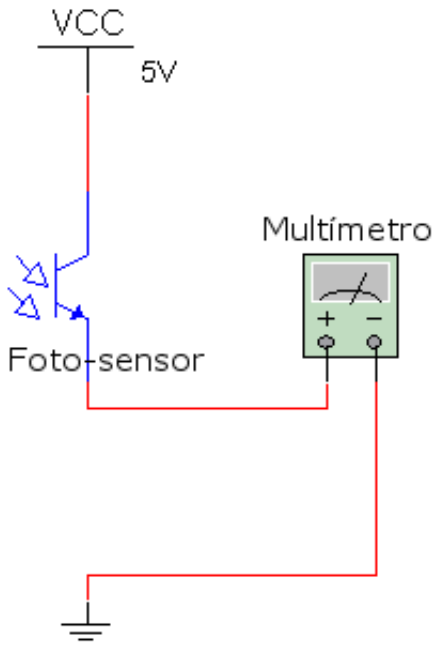

Figura 13 – Ligação elétrica padrão do fototransistor.

O sinal de saída do foto-sensor (resultante da incidência de luz) é conectado ao conversor A/D do microcontrolador, explicado na próxima seção.

3.4 Microcontrolador

O microcontrolador utilizado pelo projeto foi o PIC16F877A, cujas características foram explicadas antes e cujos detalhes podem ser verificados no *datasheet* em anexo. Na Figura 14, pode-se visualizar a descrição dos pinos deste microcontrolador:

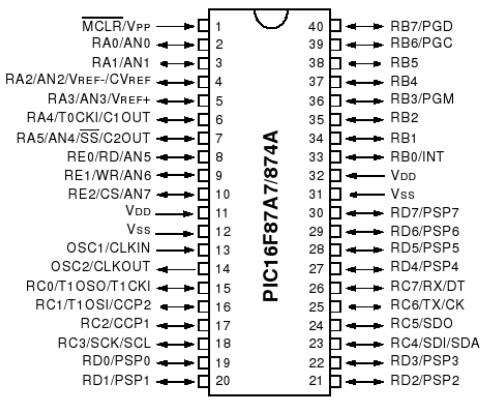

Figura 14 – Microcontrolador PIC16F8771A.

Este modelo de microcontrolador foi escolhido pela quantidade de pinos de entrada e saída (que permite o uso de periféricos adicionais ao projeto), pelo conversor A/D de 10 bits, pela possibilidade de utilizar oscilador de cristal externo de 4MHz e a disponibilização no mercado local.

3.4.1 Circuito eletrônico com o PIC16F877A

No circuito, exposto na Figura 15, é exibido o circuito eletrônico em sua versão final, completa, onde se tem uma vista da ligação do foto-sensor no microcontrolador, o *driver* de comunicação serial explicado mais adiante neste trabalho e também a ligação na fonte de 5 *volts* de todo circuito.

3.4.2 Comunicação serial do microcontrolador

Como puderam ser visto no circuito da Figura 15, os pinos de comunicação serial do microcontrolador são ligados a um *driver* (MAX232). Isto ocorre porque os bits (1's e 0's) que o microcontrolador transmite/recebe são em 5 e 0 *volts*. O *driver* converte as tensões do microcontrolador para as tensões padrão da comunicação serial (12 e -12 *volts*). Pode-se conhecer mais sobre o MAX232 no *datasheet* em anexo.

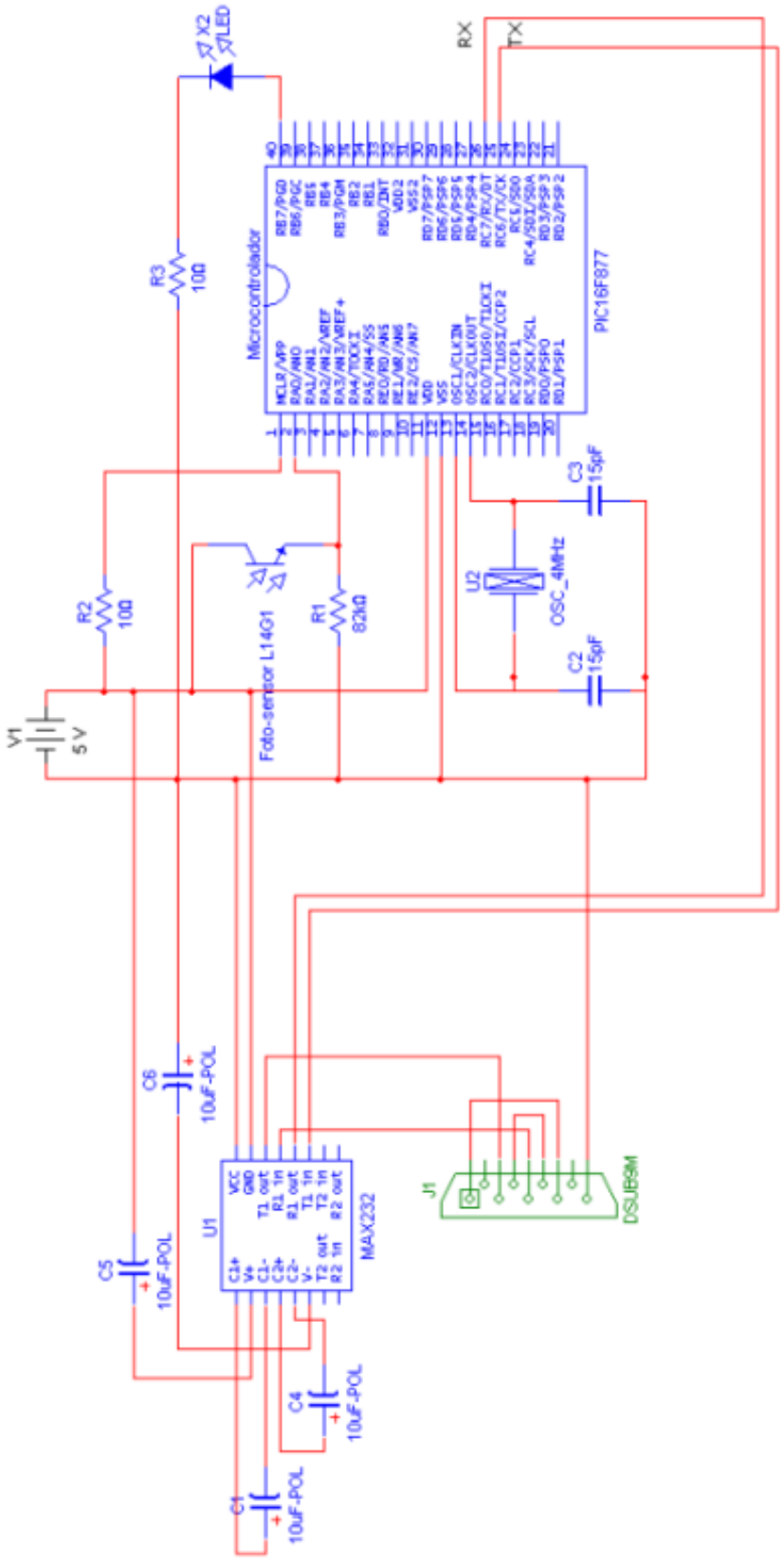

Figura 15 – Circuito eletrônico completo.

3.5 Comunicação Serial

Após realizar a leitura do medidor via *LASER* e tratamento desta leitura com auxílio do microcontrolador, ainda era necessário enviar os dados lidos para o servidor que faz a disponibilização destes dados na Internet. Este envio de dados é realizado via conexão serial com um aparelho celular Siemens C35i.

3.5.1 Siemens C35i

O celular Siemens C35i é um aparelho GSM que permite ser operado remotamente através de comunicação serial via cabo serial.

Figura 16– Celular Siemens C35i.

Figura 17 – Cabo Serial.

Um dos cuidados que foi preciso verificar para a correta ligação do celular com o circuito foi a *pinagem* do cabo serial. Esta *pinagem* representou uma dificuldade, detalhada no capítulo seguinte. Na Figura 18 tem-se demonstrada esta *pinagem*:

Figura 18 – *Pinagem* do cabo serial.

Os sinais de cada pino são detalhados na tabela abaixo:

Tabela IV – *Pinagem* do cabo serial.

Pino	Nome	Direção	Descrição
1	GND	---	GND
2	SELF-SERVICE	IN/OUT	Controle de carga da bateria
3	LOAD	IN	Tensão de carga
4	BATTERY	OUT	Bateria (apenas para S25, não utilizado no projeto)
5	DATA OUT (TX)	OUT	Envio de dados
6	DATA IN (RX)	IN	Recepção de dados
7	Z_CLK	---	Linha de clock para comunicação com acessórios.
8	Z_DATA	---	Linha de dados para comunicação com acessórios.
9	MICG	---	GND do microfone
10	MIC	IN	Entrada do microfone
11	AUD	OUT	Saída de áudio
12	AUDG	--	GND do áudio

Outra informação necessária para a conclusão do projeto foi a forma como enviar os dados ao celular. Na seção 2.5 foram introduzidos os conceitos de comandos AT, que neste ponto faz-se necessário o uso da mensagem (PDU), de acordo com as especificações da própria *Siemens*, fabricante do celular em uso. Todos os detalhes dos comandos AT estão em anexo.

3.6 Programação do Projeto

Neste projeto, como já foi explicado anteriormente, a programação teve que ser dividida de acordo com a interface programada. Inicialmente foram realizados testes com

os comandos AT via *hyperteminal*, um programa nativo do *MS-Windows*. Após esta fase foram gerados os programas do microcontrolador, do *middleware*[25] e do *front-end* (explicado no item 3.6.3, adiante).

3.6.1 Testando os comandos AT

Para controlar o celular Siemens C35i remotamente fez-se necessário o uso de Comandos AT. Este aparelho possui um modem interno padrão V.25 e de acordo com os padrões, os comandos devem iniciar com a *string* "AT+" e terminar com "<CR>" (0x0D) (Carriage Return). O resultado dos comandos serão "OK" em caso de sucesso ou "ERROR" em casso de fracasso.

Na Figura 19 tem-se o uso dos comandos AT+ pelo *HyperTerminal* do *MS-Windows*:

Figura 19 – Tela do *Hyperterminal* do *MS-Windows*.

Na primeira linha foi digitado o comando "AT". Este comando testa a conexão com o celular. O resultado foi "OK" (segunda linha), sinalizando que a conexão foi estabelecida.

Na terceira linha foi digitado o comando "AT+CGMI". Este comando solicita ao celular o nome do fabricante. O resultado foi "SIEMENS" e em seguida "OK".

---

[25] *Middle* = central; meio; mediano. *Middleware* = Software que media transações.

Na sétima linha foi digitado o comando "AT+CGMM". Este comando solicita ao celular o modelo. O resultado foi "C35i" e "OK".

Na décima-primeira linha foi digitado o comando "AT+CGMR", que solicita a versão GSM do celular resultado foi "24" e "OK".

O comando para realizar o envio de um SMS é:

(1) AT+CMGS=160

(2) > 0011000c91551618771363000000005D4E2945A04

(3) +CMGS:12

(4) OK

Onde os comandos significam:

Enviar um SMS de tamanho 160

PDU

Resposta do envio, com o número do envio

Resposta de sucesso.

Estes comandos tratam de enviar o PDU ao aparelho. Para analisar a mensagem (PDU) foi necessário dividir em grupo de caracteres e verificar o significado de cada um destes grupos de acordo com a fabricante. Utilizando o exemplo dado (0011000c91551618771363000000005D4E2945A04) pode-se dividir da forma mostrada na Tabela 5:

Tabela V – Descrição do PDU

PDU		
Parte do PDU	**Grupo de Caracteres**	**Descrição**
PCI	00	Informação sobre o comprimento do SMSC (*Short Message Service Center*). Neste caso, não está passando informações sobre o provedor de SMS.

PDU		
	11	Primeiro Octeto da mensagem.
	00	Permite que o telefone fixe ele mesmo o número de referência da mensagem.
	0c	Comprimento do número de telefone de destino (em decimal = 12)
	91	Tipo de endereço. (91 Hex = 10010001 Bin => Formato internacional)
	551618771363	Número de destino (Codificado segundo GSM 04.11).
	00	Protocolo Identificador.
	00	Esquema de codificação de dados.
	00	Período de validade.
	05	Comprimento da mensagem.
SDU	D4E2945A04	Mensagem (“TESTE”) em octeto de 8 bits representando dados de 7 bits.

A forma de entendimento dos caracteres descritos nesta tabela pode ser obtida no anexo deste documento.

3.6.2 Programação do middleware

O *middleware* do projeto é responsável por ler as mensagens de texto (SMS) do celular, decodificá-las e inserir os dados corretamente no banco de dados para que possam tirar relatórios baseados nestes dados.

O *middleware* foi desenvolvido em PHP (arquivo RecebeSMS.php anexado neste trabalho), sendo seu código de simples entendimento:

a) Uma chamada (*exec*) ao programa getsms é realizada e seus dados (resposta) são armazenados em duas variáveis: `$resultado` e `$status`;

b) A variável `$resultado` é uma matriz com as informações do SMS;

c) A variável `$status` retorna o sucesso ou insucesso na leitura do SMS;

d) Então os dados são devidamente tratados e armazenados no banco de dados;

Como existe a possibilidade de visualizar os dados (quando o script PHP for chamado do navegador de internet – ver no item 3.6.2), Existe um tratamento para imprimir na tela as informações.

Por padrão o script é agendado no *cron*[26] do S.O. para ser executado periodicamente. Os comandos deste agendamento podem ser vistos na Figura 20 (MOTA, 2006,p. 310):

```
root@notebruno:/etc# crontab -e
# m h  dom mon dow   command
* * * * * php /dados/UniCEUB/ProjetoFinal/Monografia/Anexos/php/RecebeSMS.php
~
```

Figura 20 – Editando o crontab.

O modo como foi realizado o agendamento desta figura faz com que o *script* RecebeSMS.php seja executado a cada minuto, todos os dias.

3.6.3 Programação do front-end

O *front-end* é a aplicação desenvolvida em PHP que permite que os usuários visualizem os dados das leituras, alterem dados básicos como nome, endereço, etc. mostrados a seguir:

3.6.3.1 Chamada principal (index.php)

O *script index.php* (em apêndice), é o responsável por dividir a tela em três *frames* (quadros), fazendo chamada para o menu de opções, título do sistema *front-end* e ao corpo do sistema.

---

[26] CRON é um sistema, implementado pelo daemon crond, que permite o agendamento de tarefas no GNU/LINUX (MOTA, 2006, p. 310).

3.6.3.2 Título do sistema (cabeçalho.php)

O *script* apenas mostra o nome do sistema, nomeado de "Sistema de Automação de Leitura de Medidores de Energia".

3.6.3.3 Menu (menu.php)

Disponibiliza as opções de acesso do sistema: clientes, classificação de clientes, tipos de instalações elétricas, relatório de consumo e recebe SMS (que faz chamada ao *middleware*). Este *script* encontra-se em apêndice.

3.6.3.4 Corpo (corpo.php)

O corpo serve como destino de todas chamadas do menu. Inicialmente o corpo apenas faz apresentação do sistema. O *script* do corpo está em apêndice.

3.6.3.5 Clientes (clientes.php)

Para poder relatar as leituras com um nível de inteligibilidade melhor, foram criados módulos para preenchimento de dados de clientes. Estes dados dependem de outros três tipos de dados importantes: dados complementares (preenchidos no próprio *script*), classificação de clientes e tipo de instalação elétrica (clientes.php está em apêndice).

3.6.3.6 Classificações de clientes (classificacoes.php)

Este módulo (em apêndice) permite que diferentes classificações sejam armazenadas no banco de dados para que possam ser referenciadas no módulo de clientes.

3.6.3.7 Tipos de instalações elétricas (instalacoes.php)

Este módulo (em apêndice) permite que os tipos de instalações elétricas sejam armazenados no banco de dados para que possam ser referenciadas no módulo de clientes.

3.6.3.8 Relatório de consumo (relatorio.php)

Este *script*, que se encontra em apêndice, permite a extração de relatórios de

leituras por clientes e por período.

3.6.3.9 Recebe sms (recebesms.php?apresentadetalhes=sim)

Finalmente, o *script* em apêndice RecebeSMS.php, que na realidade é o *middleware* do sistema, mas que no caso passa o parâmetro de apresentação dos detalhes na tela (`ApresentaDetalhes=Sim`), está disponibilizado no menu do sistema.

3.6.4 Programação do microcontrolador

O microcontrolador é responsável por realizar a conversão A/D do foto-sensor, contar as voltas que o disco efetua e, por fim, enviar as informações codificadas ao celular que as enviará ao servidor. O cófigo-fonte da programação do microcontrolador (realizada em linguagem de programação C) está no apêndice deste trabalho.

3.7 Modelagem do Banco de Dados

A modelagem do banco de dados foi baseada numa cobrança de energia da concessionária de energia local. É importante salientar que o modelo criado não pretende ser fiel ao da concessionária, sendo apenas para demonstração dos resultados da leitura remota. O mapa de entidades e relacionamentos pode ser observado na Figura 21:

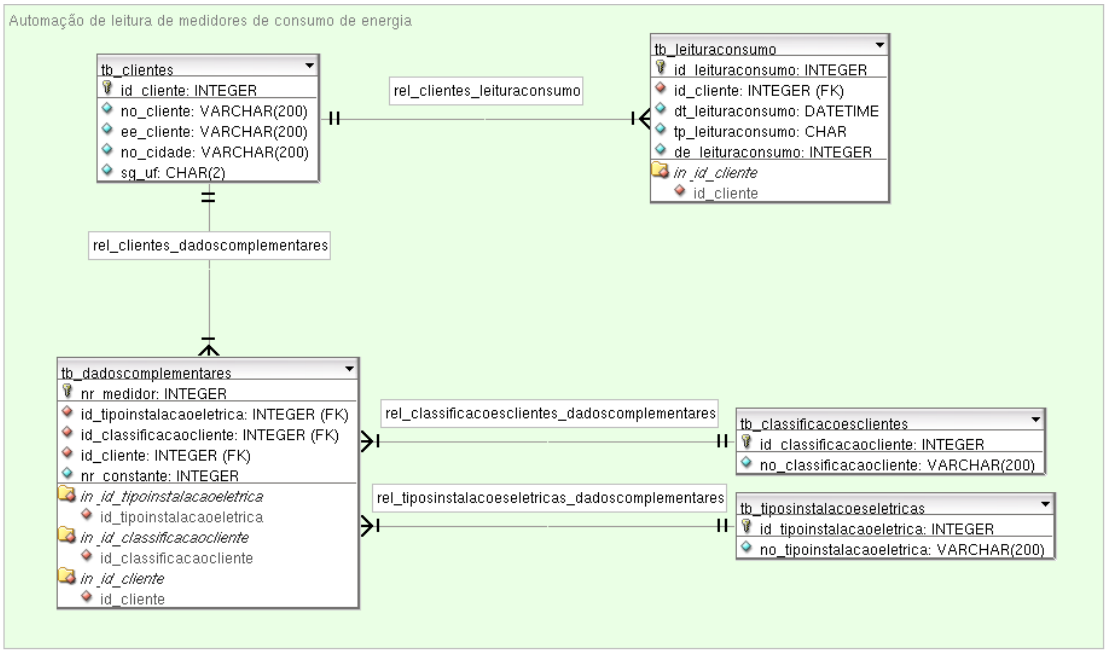

Figura 21 – Mapa de entidades e relacionamentos.

3.7.1 Descrição e codificação das tabelas

As tabelas do banco de dados são descritas nos sub-itens abaixo:

3.7.1.1 Tabela de clientes

Esta tabela contém as informações principais dos clientes:

a) id_cliente: identificador único de cada cliente;

b) no_cliente: nome do cliente;

c) ee_cliente: endereço do cliente;

d) no_cidade: nome da cidade do cliente;

e) sg_uf: sigla da unidade federativa do cliente.

Abaixo segue o código de criação desta tabela:

```
CREATE TABLE tb_clientes (
  id_cliente INTEGER UNSIGNED NOT NULL AUTO_INCREMENT,
  no_cliente VARCHAR(200) NULL,
  ee_cliente VARCHAR(200) NULL,
  no_cidade VARCHAR(200) NULL,
  sg_uf CHAR(2) NULL,
  PRIMARY KEY(id_cliente)
)
TYPE=InnoDB;
```

3.7.1.2 Tabela de classificação de clientes

Esta tabela classifica os clientes (residencial, comercial, industrial etc):

a) id_classificacaocliente: identificador único de cada classificação de cliente;

b) no_classificacaocliente: nome da classificação do cliente;

Abaixo segue o código de criação desta tabela:

```
CREATE TABLE tb_classificacoesclientes (
  id_classificacaocliente INTEGER UNSIGNED NOT NULL AUTO_INCREMENT,
  no_classificacaocliente VARCHAR(200) NULL,
  PRIMARY KEY(id_classificacaocliente)
```

```
)
TYPE=InnoDB;
```

3.7.1.3 Tabela de tipos de instalações elétricas

Esta tabela tipifica a instalação elétrica do clientes (monofásica, bifásica, trifásica etc):

a) id_tipoinstalacaoeletrica: identificador único de cada tipo de instalação elétrica;

b) no_tipoinstalacaoeletrica: nome do tipo de instalação elétrica;

Abaixo segue o código de criação desta tabela:

```
CREATE TABLE tb_tiposinstalacoeseletricas (
  id_tipoinstalacaoeletrica INTEGER UNSIGNED NOT NULL AUTO_INCREMENT,
  no_tipoinstalacaoeletrica VARCHAR(200) NULL,
  PRIMARY KEY(id_tipoinstalacaoeletrica)
)
TYPE=InnoDB;
```

3.7.1.4 Tabela de dados complementares

Esta tabela serve para armazenar dados complementares dos clientes:

a) nr_medidor: identificador único do medidor instalado no cliente;

b) id_tipoinstalacaoeletrica: chave estrangeira do tipo de instalação elétrica;

c) id_classificacaocliente: chave estrangeira da classificação do cliente;

d) id_cliente: chave estrangeira do cliente;

e) nr_constante: constante.

Abaixo segue o código de criação desta tabela:

```
CREATE TABLE tb_dadoscomplementares (
  nr_medidor BIGINT UNSIGNED NOT NULL AUTO_INCREMENT,
  id_tipoinstalacaoeletrica INTEGER UNSIGNED NOT NULL,
  id_classificacaocliente INTEGER UNSIGNED NOT NULL,
  id_cliente INTEGER UNSIGNED NOT NULL,
```

```
  nr_constante INTEGER UNSIGNED NULL,
  PRIMARY KEY(nr_medidor),
  INDEX in_id_tipoinstalacaoeletrica(id_tipoinstalacaoeletrica),
  INDEX in_id_classificacaocliente(id_classificacaocliente),
  INDEX in_id_cliente(id_cliente),
  FOREIGN KEY(id_tipoinstalacaoeletrica)
    REFERENCES tb_tiposinstalacoeseletricas(id_tipoinstalacaoeletrica)
      ON DELETE NO ACTION
      ON UPDATE NO ACTION,
  FOREIGN KEY(id_classificacaocliente)
    REFERENCES tb_classificacoesclientes(id_classificacaocliente)
      ON DELETE NO ACTION
      ON UPDATE NO ACTION,
  FOREIGN KEY(id_cliente)
    REFERENCES tb_clientes(id_cliente)
      ON DELETE NO ACTION
      ON UPDATE NO ACTION
)
TYPE=InnoDB;
```

3.7.1.5 Tabela de leitura de consumo

Esta tabela, por fim, armazena as leituras realizadas para os medidores dos clientes:

a) id_leituraconsumo: identificador único de cada leitura;

b) id_cliente: chave estrangeira que identifica o cliente;

c) dt_leituraconsumo: data da realização da leitura;

d) tp_leiruraconsumo: tipo de leitura realizada (local ou remotamente);

e) de_leituraconsumo: leitura realizada.

Abaixo segue o código de criação desta tabela:

```
CREATE TABLE tb_leituraconsumo (
  id_leituraconsumo INTEGER UNSIGNED NOT NULL AUTO_INCREMENT,
  id_cliente INTEGER UNSIGNED NOT NULL,
  dt_leituraconsumo DATETIME NULL,
  tp_leituraconsumo CHAR NULL,
  de_leituraconsumo FLOAT UNSIGNED NULL,
  PRIMARY KEY(id_leituraconsumo),
  INDEX in_id_cliente(id_cliente),
  FOREIGN KEY(id_cliente)
    REFERENCES tb_clientes(id_cliente)
      ON DELETE NO ACTION
      ON UPDATE NO ACTION
)
TYPE=InnoDB;
```

3.8 O Apache

O servidor *Web* Apache com suporte a PHP já é nativo do *Linux*, podendo ser verificado seu correto funcionamento e instalação através dos comandos mostrados na figura abaixo:

```
root@notebruno:~# dpkg -l apache2
Name                  Version               Description
======================-=====================-==============================================================
apache2               2.0.55-4ubuntu2.1     next generation, scalable, extendable web server
root@notebruno:~# dpkg -l php5
Name                  Version               Description
======================-=====================-==============================================================
php5                  5.1.2-1ubuntu3.3      server-side, HTML-embedded scripting language (meta-package)
```

Figura 22 – Comandos para verificar instalação do Apache e PHP.

O único cuidado que se deve tomar nas configurações do PHP é configurar o arquivo "`php.ini`", alterando a variável "`register_global = on`", o que permite que as variáveis sejam passadas (pelos métodos GET e POST do HTTP) entre os *scripts* sem a necessidade realizar nenhum método de captura de valor das variáveis dentro do *script*.

É importante citar, já que o *middleware* é executado pelo Apache, que o usuário do servidor Apache, o "`www-data`", não tem permissão para realizar alterações nos *devices* do *Linux*, que neste projeto é o "`/dev/ttyUSB0`". USB, porque para conectar o celular ao computador é utilizado um adaptador USB/DB-9. Portanto é necessário executar o comando abaixo, com o usuário "`root`", permitindo então que o "`www-data`" acesse a porta serial:

```
#chmod 777 /dev/ttyUSB0
```

3.9 Acessando o Front-End

A visualização das informações de consumo pode ser realizada utilizando um navegador *Web*. Por padrão do *MS-Windows*, o navegador instalado é o *Internet Explorer* – IE, porém o mostrador neste projeto é o *Firefox*, um navegador de código aberto, gratuito e também multi-plataforma.

Como todo sistema está armazenado no servidor, que no caso deste projeto é também a máquina onde o *front-end* será visualizado, basta digitar na barra de endereços do *Firefox* o endereço: "`http://localhost/index.php`" que o *front-end* será acessado,

conforme demonstrado na figura seguinte:

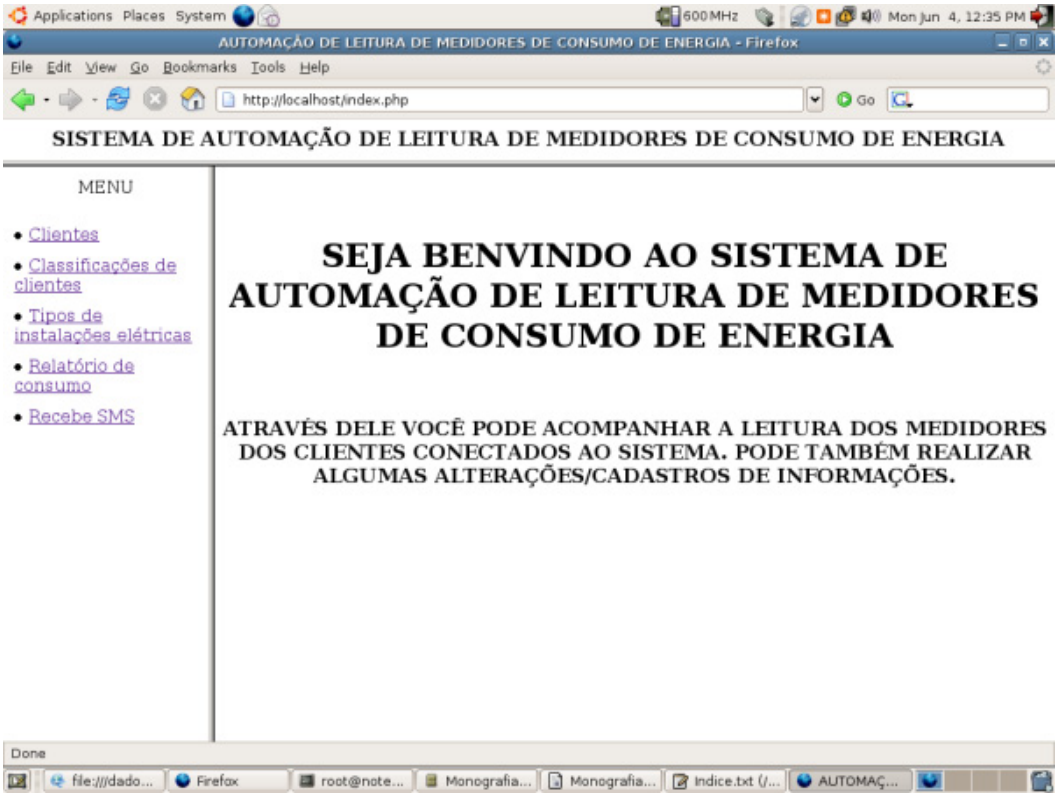

Figura 23 – Acessando o Sistema de Automação de Leitura de Medidores de Consumo de Energia.

A navegação entre os módulos do sistema é feita pelo menu (à esquerda da Figura 23) e é detalhada nos itens a seguir:

3.9.1 Módulo de clientes

Está disposta na Figura 24 a tela do módulo de clientes. O preenchimento é bem simples, sendo que cada campo tem sua descrição e o tipo de dados que se deve colocar.

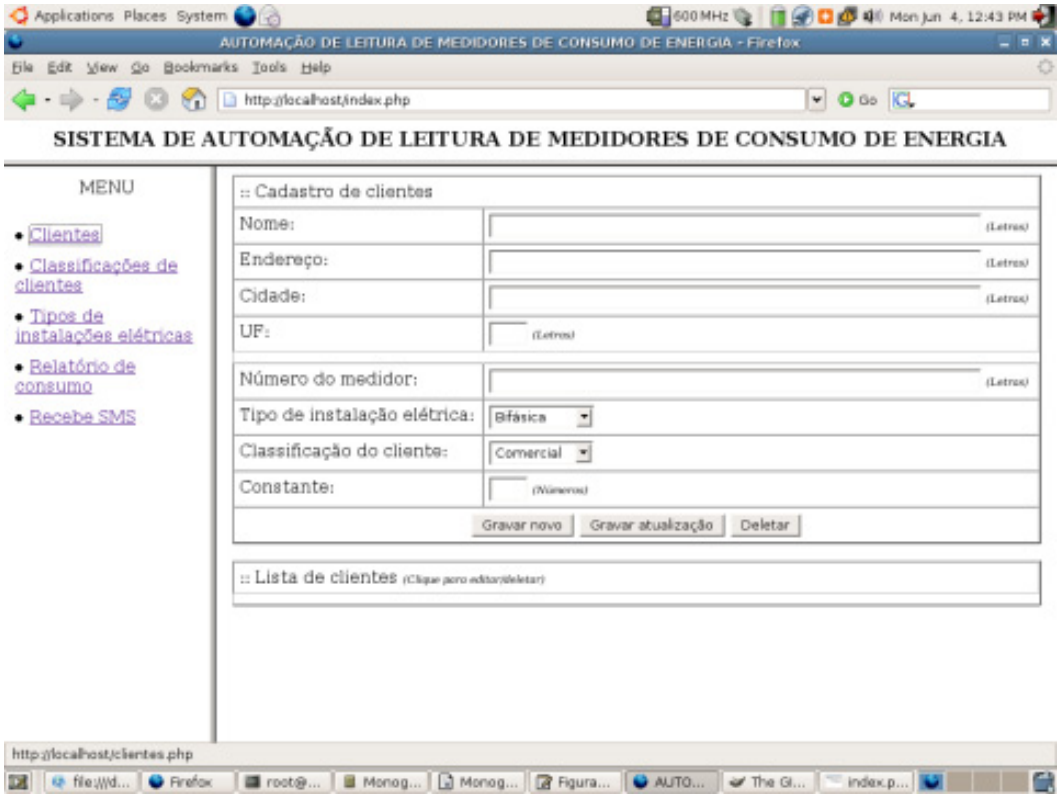

Figura 24 – Módulo de clientes.

Abaixo do formulário de cadastro existe uma lista com os clientes já cadastrados.

3.9.2 Módulo de classificação de clientes

Na Figura 25 a tela do módulo de classificação de clientes é mostrada, onde assim como no módulo de clientes é de preenchimento bem simples, sendo que cada campo tem sua descrição e o tipo de dado que se deve colocar.

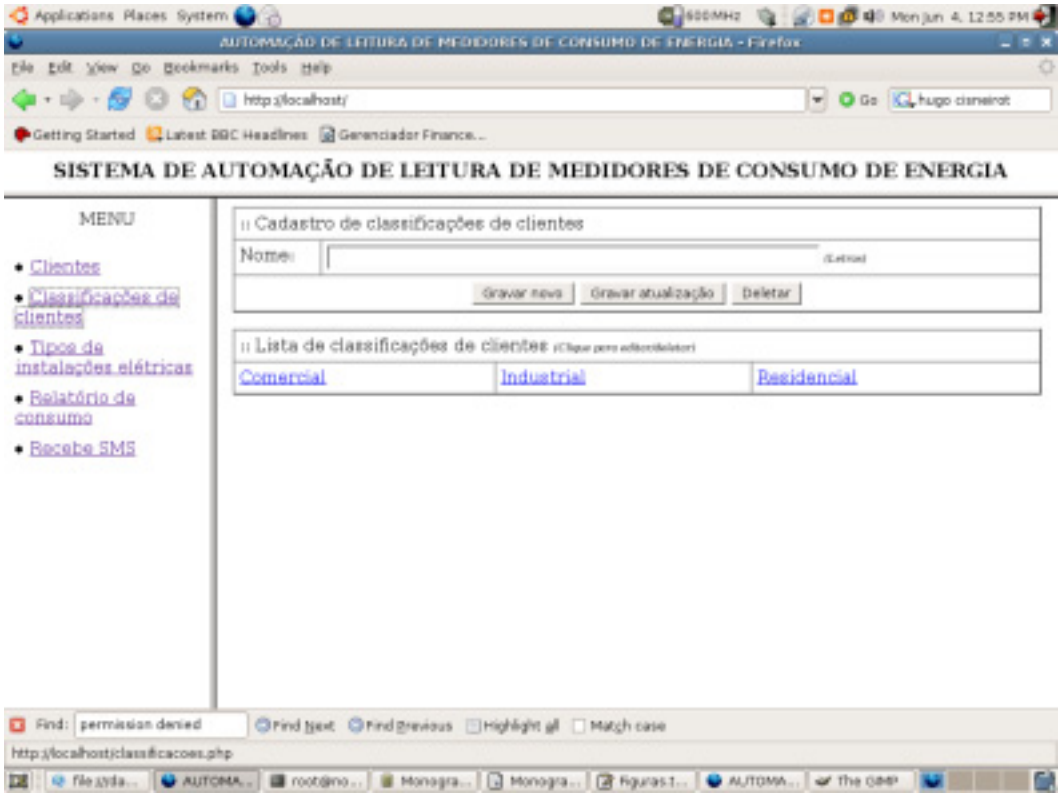

Figura 25 – Módulo de classificação de clientes.

3.9.3 Módulo de tipos de instalações elétricas

Está disposta na Figura 26 a tela do módulo de tipos de instalações elétricas. Assim como no módulo de clientes, o preenchimento é bem simples, sendo que cada campo tem sua descrição e o tipo de dados que se deve colocar.

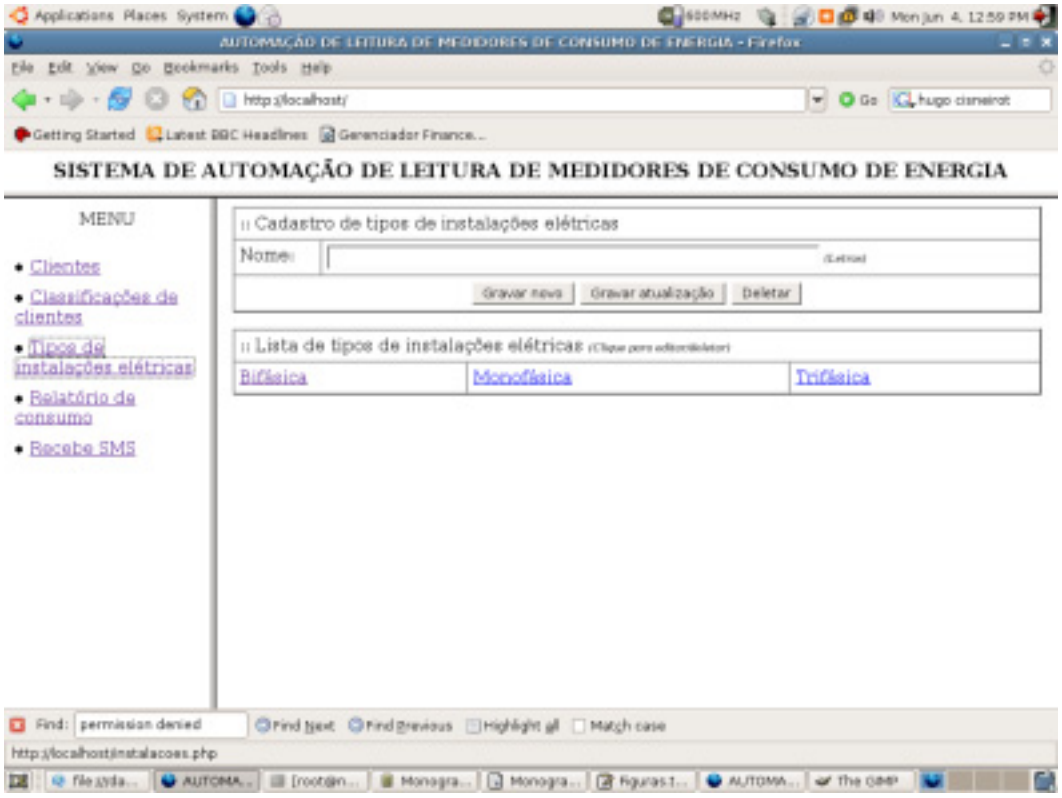

Figura 26 – Módulo de tipos de instalações elétricas.

3.9.4 Módulo de relatórios

É mostrada na Figura 27 a tela do módulo de relatórios. Neste módulo basta selecionar o cliente e o período que se deseja relatar as leituras.

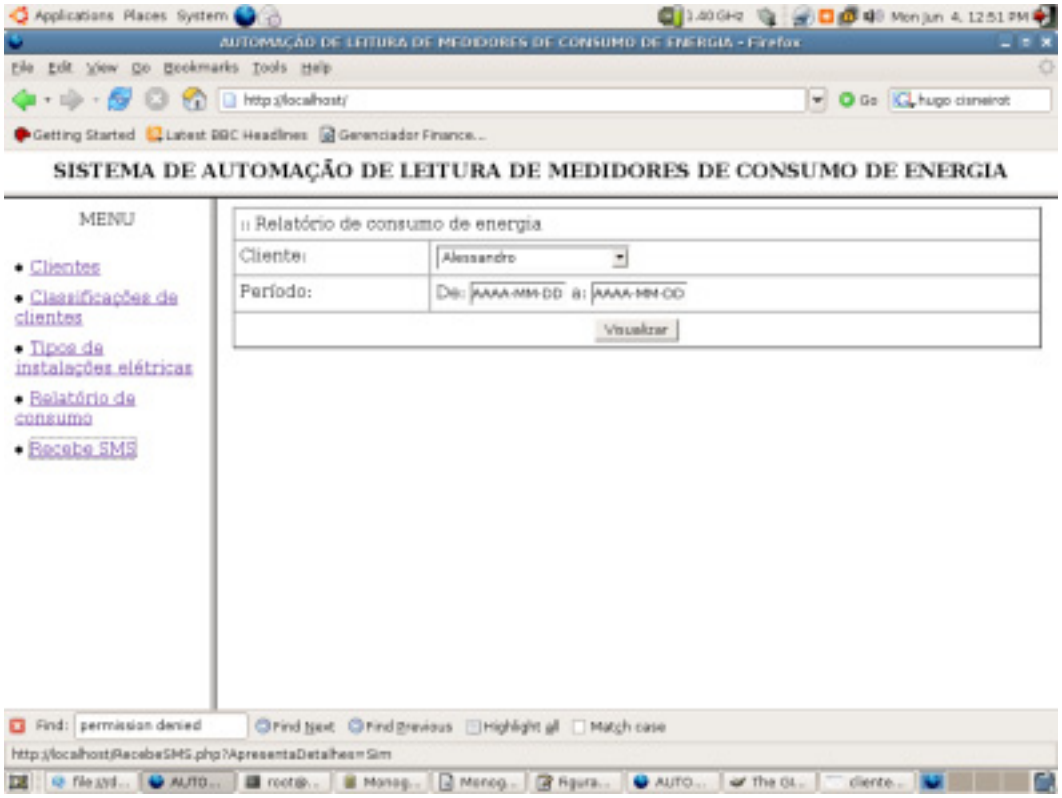

Figura 27 – Módulo de relatórios.

3.9.5 Módulo de recepção de sms

Na Figura 28 a tela do módulo de recepção de SMS pode ser visualizada. Este módulo é o *middleware* do sistema.

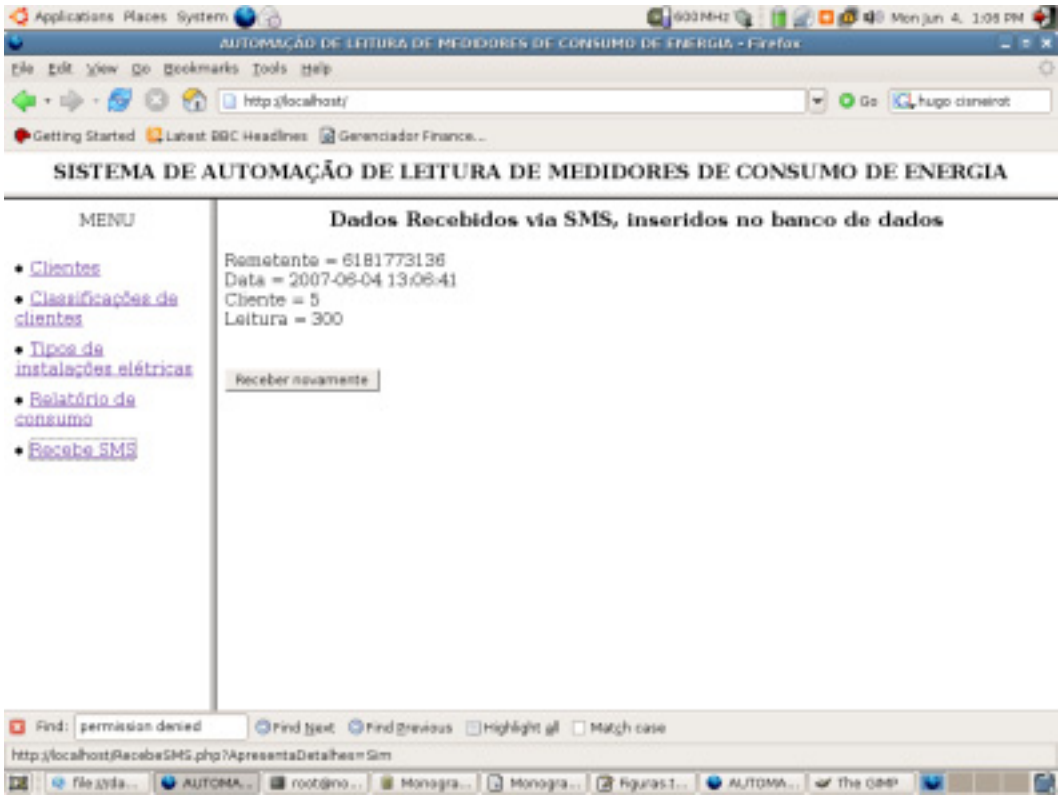

Figura 28 – Módulo de recepção de SMS.

4 DIFICULDADES ENCONTRADAS E TESTES REALIZADOS

Para proceder com o projeto, um grande número de dificuldades foi encontrado, testes foram realizados e soluções alcançadas. Este capítulo dispõe sobre este tema.

4.1 Construção do Acoplador Óptico

Durante a construção do acoplador óptico deparou-se com várias dificuldades:

4.2.1 Posicionamento do foto-sensor e do emissor LASER

Inicialmente, o emissor *LASER* foi posicionado na lateral do medidor, o que trazia dois problemas:

a) A espessura do disco é muito pequena, de forma que a reflexão do *LASER* era mínima;

b) Tendo uma reflexão insuficiente, era mais difícil regular a posição do foto-sensor.

Com ajuda de seu orientador, o autor encontrou a posição ideal para instalação do acoplador óptico, conforme mostrado na figura abaixo.

Figura 29 – Posição final do acoplador óptico.

4.2.2 Foto-sensor

Outra grande dificuldade superada foi a escolha do foto-sensor. No início dos trabalhos, estava sendo utilizado o foto-sensor CI-6504A da PASCO (Figura 30), emprestado pelo UniCEUB. Este dispositivo, na verdade, é um transdutor com todo um circuito interno implementado, que faz com que o sinal de saída varie em amplitude de acordo com a intensidade de luz capturada e ainda, varie a freqüência de acordo com o comprimento de luz.

Figura 30 – Foto-sensor da PASCO.

Era preciso então procurar por outro sensor, mas no mercado local não existe nenhuma empresa que comercialize produtos do tipo, tendo sido necessário pesquisar em outros centros[27] por outros dispositivos a fim de encontrar um que atendesse aos requisitos do projeto.

Após intensa procura, o fototransistor L14G1 foi comprado e finalmente esta fase do projeto superada. Restava então verificar o sinal de saída deste novo componente. Um outro problema foi encontrado.

4.2.3 Sinal do foto-sensor

O novo sensor era tão sensível que a mínima variação de luz fazia com que

---

[27] Com auxílio de Bianca Pinheiro, amiga pessoal do autor foi realizada a pesquisa em São Paulo, onde ela reside.

seu sinal de saída chegasse ao topo (5 *volts*). Para superar este problema o autor construiu uma "caixa" de papelão, pintou o interior de preto com tinta aquarela e cobriu o protótipo.

Mesmo assim, variações de luz refletida pelo disco causavam erro no sinal. Outros dois filtros foram modelados:

a) na abertura do foto-sensor, foi colocado uma pequena película vermelha (de papel celofane) para que apenas luz da cor vermelha (que é a cor do *LASER*) passasse ao sensor.

b) A parte inferior do vidro do medidor, onde está posicionado o foto-sensor, foi coberta com fita isolante, deixando apenas um orifício suficiente para passar o feixe do *LASER* refletido. Na Figura 11 é mostrado o modelo.

4.3 Microcontrolador

O maior problema encontrado na ligação do microcontrolador foi a comunicação serial com o celular. Isto se deu porque as referências encontradas nos livros de microcontrolador instruíam a *curto-circuitar* os pinos 7 com 8, 1 com 4 e com 6, do conector DB-9.

Após verificar o não funcionamento, procedendo desta forma, o autor procurou o laboratório de arquitetura avançada de computadores e, com auxílio do professor, realizou testes pino-a-pino utilizando um osciloscópio.

Pôde-se constatar que o pino 1 do conector deveria ficar isolado (e não em curto com 4 e 6, como sugerem os livros).

Outra dificuldade encontrada foi o tratamento do sinal provindo do foto-sensor, que mesmo coberto e provido de filtro, ainda assim, enviava um sinal alto de tensão. Para resolver este problema foi ligado um resistor de 82 kΩ em paralelo com o microcontrolador e o "terra".

Na Figura 31 é mostrado como ficou o circuito fisicamente ligado. Pode-se verificar a conexão com o celular, o *driver* de comunicação serial, a conexão com o foto-sensor, o microcontrolador e ainda, a fonte de alimentação do circuito.

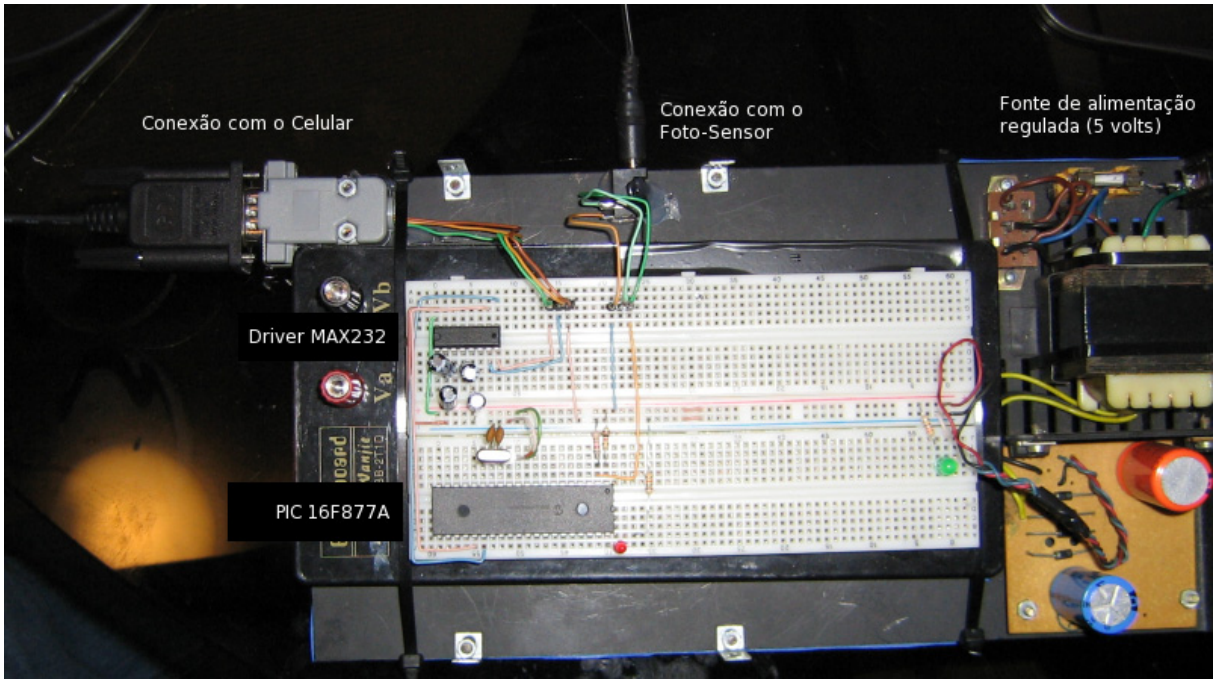

Figura 31 – Circuito fisicamente ligado.

4.4 Comunicação Serial

Outro grande problema enfrentado no projeto foi conseguir realizar comunicação serial do celular com o servidor. Nos primeiros testes, realizados com o *hyperterminal* do *MS-Windows* foi alcançado sucesso, porém fazer um programa que lesse o SMS do celular e armazenasse os dados no banco de dados foi um passo difícil de ser superado.

Inicialmente a idéia era desenvolver um software em linguagem de programação C para executar esta tarefa no próprio *MS-Windows*. Mas após muitos dias de pesquisa em vários sites da internet e em vários livros em diferentes bibliotecas, a solução de instalar o programa *getsms*, do GNU/LINUX e fazer com que o próprio PHP (que já era a ferramenta escolhida para apresentar os relatórios aos clientes) executasse este programa foi a solução. Além de funcionar, permitiu a criação de um *script* passível de ser visto pelo navegador de internet.

4.5 O Apache

A fase de leitura do SMS pelo *middleware* foi também penosa, isto porque o apache não conseguia ter acesso ao *device*, sempre aparecendo a seguinte mensagem:

```
“Jun  4 10:22:59 localhost getsms (/dev/ttyUSB0): Cannot open serial port,
cause: No such file or directory”
```

Conforme recomendado na seção 3.8, a solução seria realizar o comando de permissão para o *device*, sempre antes de executar o *script* “RecebeSMS.php”, a saber:

```
#chmod 777 /dev/ttyUSB0
```

5 CONCLUSÕES

A proposta do projeto foi construir um dispositivo capaz de realizar a leitura dos medidores eletromecânicos (amplamente utilizados pelas concessionárias de energia) de forma remota. Para isto foram contruídos um acoplador óptico (responsável por converter com auxílio de *LASER* e foto-sensor, as voltas realizadas pelo disco do medidor em sinal elétrico), um circuito eletrônico dotado de microcontrolador (que converte o sinal elétrico, proveniente do foto-sensor, em código de máquina) que periodicamente envia a leitura para um servidor. Este envio se dá via telecomunicação celular. E, por fim, foi desenvolvido um sistema baseado em ambiente *Web* que disponibiliza os dados lidos para o usuário. Além de concluir o projeto satisfatoriamente, é importante ressaltar o impacto que tal plano pode causar ao cotidiano das pessoas, consumidoras de energia elétrica em todo mundo: a possibilidade de acompanhar com maior freqüência e facilidade o consumo medido em seu local.

Deve-se citar ainda a vantagem para as concessionárias de energia, com a possível redução nos custos operacionais para realização da medição. O projeto de Automação de Leitura de Medidores de Consumo de Energia, agora finalizado, contempla também muitos conhecimentos adquiridos ao longo do curso de Engenharia de Computação do Centro Universitário de Brasília – UniCEUB, pelo autor que pode, por fim, mostrar que é possível, com uso de acoplador óptico, sistema inteligente de leitura e comunicação *wireless*, realizar a leitura remota dos, publicamente conhecidos, relógios de luz.

5.1 Sugestões de Continuidade

Sugere-se para trabalhos futuros, baseados neste, os seguintes fatores:

a) Pesquisa por *LASER* mais adequado;

b) Pesquisa por foto-sensor mais adequado;

c) Pesquisa por formas de implementação de acoplador óptico com tamanho reduzido;

d) Redução dos custos com dispositivos eletrônicos (microcontrolador, foto-sensor, *LASER*);

e) Outros meios de envio da leitura;

f) Diferentes formas de apresentação da leitura.

REFERÊNCIAS BIBLIOGRÁFICAS

CREDER, Hélio. **Instalações elétricas**. 14ª. ed. Rio de Janeiro: LTC, 2000.

HOAG, Melanie. **Servidor web usando Apache**. Tradução de: Toni Ricardo Cavalheiro. São Paulo: Berkeley Brasil , 2002.

MALVINO, Albert Paul. **Eletrônica volume 1**. 4ª. ed. Tradução de: Romeu Abdo. Revisão técnica de: Antonio Pertence Junior. São paulo: Makron Books, 1995.

GNU/LINUX. **GETSMS - Manual do debian**. Maio, 2006.

MARTINS, Nardênio Almeida. **Sistemas microcontrolados**. São Paulo: Novatec, 2005.

MEDEIROS FILHO, Sólon de. **Medição de energia elétrica**. 4ª. ed. Rio de Janeiro: LTC, 1997.

MOTA FILHO, João Eriberto. **Descobrinto o Linux - entenda o sistema operacional GNU/LINUX**. Sao Paulo: Novatec, 2006.

PEREIRA, Fábio. **Microcontroladores PIC: programação em C**. São Paulo: Érica, 2003.

REZENDE, Sérgio M. **Materiais e dispositivos eletrônicos**. 2ª. ed. São Paulo: Livraria da Física, 2004.

SILVA JUNIOR, Vidal Pereira da. **Microcontroladores**. São Paulo: Érica, 1998.

SOARES, Luiz Fernando G.; LEMOS, Guido; COLCHER, Sérgio. **Redes de computadores: das LAN's, MAN's e WAN's às redes ATM**. 12ª. ed. Rio de Janeiro: Campus, 1995.

SOUZA, David José de; LAVÍNIA, Nicolás César. **Conectando o PIC 16F877A: recursos avançados**. São Paulo: Érica, 2003.

MONOGRAFIA.NET. Disponível em <http://www.monografia.net/abnt/index.htm>. Acesso em 13/03/2007.

GEODESIA ONLINE. Disponível em <http://geodesia.ufsc.br/Geodesia-online/arquivo/cobrac_2002/010/010.htm>. Acesso em 29/05/07.

SENSOR, WIKIPEDIA A ENCICLOPÉDIA ONLINE. Disponível em <http://pt.wikipedia.org/wiki/Sensor>. Acesso em 01/06/2007.

RS232 STANDARD. Disponível em <http://www.camiresearch.com/Data_Com_Basics/RS232_standard.html>. Acesso em 23/05/2007.

RS232, WIKIPEDIA THE ONLINE ENCICLOPEDIA. Disponível em <http://en.wikipedia.org/wiki/RS-232>. Acesso em 01/06/2007.

HAYES COMMAND SET. Disponível em <http://en.wikipedia.org/wiki/Hayes_command_set>. Acesso em 31/05/2007.

DENNIS RITCHIE, WIKIPEDIA THE ONLINE ENCICLOPEDIA. Disponível em <http://en.wikipedia.org/wiki/Dennis_Ritchie>. Acesso em 01/06/2007.

PHP. Disponível em <http://br2.php.net/manual/phpfi2.php>. Acesso em 01/06/2007.

NAVEGADOR, WIKIPEDIA A ENCICLOPÉDIA ONLINE. Disponível em <http://pt.wikipedia.org/wiki/Navegador>. Acesso em 01/06/2007.

INTEGRATED CIRCUIT, WIKIPEDIA THE ONLINE ENCICLOPEDIA. Disponível em <http://en.wikipedia.org/wiki/Integrated_circuit>. Acesso em 01/06/2007.

ACTARIS METERING SYSTEMS. Disponível em <http://www.actaris.com/html/eng/COMPANY/1.html>. Acesso em 01/06/2007.

CASA FERREIRA. Disponível em <http://www.casaferreira.com.br/medicao/actaris/mono_02.htm>. Acesso em 29/05/07.

GSM, WIKIPEDIA THE ONLINE ENCICLOPEDIA. Disponível em <http://en.wikipedia.org/wiki/GSM>. Acesso em 02/06/2007.

APÊNDICE 1 – CÓDIGO FONTE DO SISTEMA FRONT-END

1.1 INDEX.PHP

```
<?
/*********************************************************************/
/* AUTOR: BRUNO MESQUITA SANTANA                                     */
/* DATA: 03/06/2007                                                  */
/* NOME DO ARQUIVO: INDEX.PHP                                        */
/* FUNÇÃO: ESTE É ARQUIVO PRINCIPAL DO SISTEMA FRONT-END DO PROJETO */
/*       DE CONCLUSÃO DO CURSO DE ENGENHARIA DE COMPUTAÇÃO DO     */
/*       AUTOR: AUTOMAÇÃO DE LEITURA DE MEDIDORES DE CONSUMO DE   */
/*       ENERGIA. ESTE ARQUIVO FAZ CHAMADA AO CABEÇALHO, AO MENU  */
/*       E AO CORPO PRINCIPAL.                                     */
/*********************************************************************/
?>
<html>
<head>
      <TITLE>AUTOMAÇÃO DE LEITURA DE MEDIDORES DE CONSUMO DE
ENERGIA</TITLE>
      <frameset rows="40,*">
            <frame           name="cabecalho"           src="cabecalho.php"
scrolling=no></frame>
            <frameset cols="200,*">
                  <frame name="menu" src="menu.php"></frame>
                  <frame name="corpo" src="corpo.php"></frame>
            </frameset>
      </frameset>
</head>
</html>
```

1.2 CABECALHO.PHP

```
<?
/*********************************************************************/
/* AUTOR: BRUNO MESQUITA SANTANA                                     */
/* DATA: 03/06/2007                                                  */
/* NOME DO ARQUIVO: CABECALHO.PHP                                    */
/* FUNÇÃO: ESTE É O CABEÇALHO DO SISTEMA FRONT-END DO PROJETO DE     */
/*       CONCLUSÃO DO CURSO DE ENGENHARIA DE COMPUTAÇÃO DO AUTOR: */
/*       AUTOMAÇÃO DE LEITURA DE MEDIDORES DE CONSUMO DE ENERGIA  */
/*       ESTE ARQUIVO APRESENTA O SISTEMA AO USUÁRIO.             */
/*********************************************************************/
?>
<html>
<head>
      <TITLE>AUTOMAÇÃO DE LEITURA DE MEDIDORES DE CONSUMO DE
ENERGIA</TITLE>
</head>
<body>
<h3><center>SISTEMA DE AUTOMAÇÃO DE LEITURA DE MEDIDORES DE CONSUMO DE
ENERGIA</center></h3>
</body>
</html>
```

1.3 MENU.PHP

```
<?
/*********************************************************************/
/* AUTOR: BRUNO MESQUITA SANTANA                                     */
/* DATA: 03/06/2007                                                  */
/* NOME DO ARQUIVO: MENU.PHP                                         */
/* FUNÇÃO: ESTE É O MENU DO SISTEMA FRONT-END DO PROJETO DE          */
/*       CONCLUSÃO DO CURSO DE ENGENHARIA DE COMPUTAÇÃO DO AUTOR: */
/*       AUTOMAÇÃO DE LEITURA DE MEDIDORES DE CONSUMO DE ENERGIA  */
/*       ESTE ARQUIVO LISTAS AS OPÇÕES DE MENU DO O SISTEMA AO    */
/*       USUÁRIO.                                                  */
/*********************************************************************/
?>
<html>
<head>
      <TITLE>AUTOMAÇÃO DE LEITURA DE MEDIDORES DE CONSUMO DE
ENERGIA</TITLE>
</head>
<body>
<table>
<TR>
      <TD align="center">MENU</TD>
</TR>
<TR>
      <TD height="20"> </TD>
</TR>
<TR>
      <TD><li><a href="clientes.php" target="corpo">Clientes</a></li></TD>
</TR>
<TR>
      <TD height="5"></TD>
</TR>
<TR>
      <TD><li><a href="classificacoes.php" target="corpo">Classificações de
clientes</a></li></TD>
</TR>
<TR>
      <TD height="5"></TD>
</TR>
<TR>
      <TD><li><a href="instalacoes.php" target="corpo">Tipos de instalações
elétricas</a></li></TD>
</TR>
<TR>
      <TD height="5"></TD>
</TR>
<TR>
      <TD><li><a     href="relatorio.php"    target="corpo">Relatório    de
consumo</a></li></TD>
</TR>
<TR>
      <TD height="5"></TD>
</TR>
<TR>
      <TD><li><a                  href="RecebeSMS.php?ApresentaDetalhes=Sim"
target="corpo">Recebe SMS</a></li></TD>
</TR>
</table>
</body>
</html>
```

1.4 CORPO.PHP

```
<?
/*********************************************************************/
/* AUTOR: BRUNO MESQUITA SANTANA                                     */
/* DATA: 03/06/2007                                                  */
/* NOME DO ARQUIVO: CORPO.PHP                                        */
/* FUNÇÃO: ESTE É O CORPO DO SISTEMA FRONT-END DO PROJETO DE         */
/*       CONCLUSÃO DO CURSO DE ENGENHARIA DE COMPUTAÇÃO DO AUTOR: */
/*       AUTOMAÇÃO DE LEITURA DE MEDIDORES DE CONSUMO DE ENERGIA  */
/*       ESTE ARQUIVO FAZ A MENSAGEM DE BOAS VINDAS DO SISTEMA AO */
/*       USUÁRIO.                                                  */
/*********************************************************************/
?>
<html>
<head>
      <TITLE>AUTOMAÇÃO DE LEITURA DE MEDIDORES DE CONSUMO DE
ENERGIA</TITLE>
</head>
<body>
<br><BR>
<H1><center>SEJA BENVINDO AO SISTEMA DE AUTOMAÇÃO DE LEITURA DE MEDIDORES
DE CONSUMO DE ENERGIA</center></H1><BR>
<H3><center>ATRAVÉS DELE VOCÊ PODE ACOMPANHAR A LEITURA DOS MEDIDORES DOS
CLIENTES CONECTADOS AO SISTEMA. PODE TAMBÉM REALIZAR ALGUMAS
ALTERAÇÕES/CADASTROS DE INFORMAÇÕES.</center></H3>
</body>
</html>
```

1.5 CLASSIFICACOES.PHP

```
<?
/*********************************************************************/
/* AUTOR: BRUNO MESQUITA SANTANA                                     */
/* DATA: 03/06/2007                                                  */
/* NOME DO ARQUIVO: CLASSIFICACOES.PHP                               */
/* FUNÇÃO: ESTE É O ARQUIVO DE CONFIGURAÇÃO DAS CLASSIFICAÇÕES       */
/*       EXISTENTES DO SISTEMA FRONT-END DO PROJETO DE            */
/*       CONCLUSÃO DO CURSO DE ENGENHARIA DE COMPUTAÇÃO DO AUTOR: */
/*       AUTOMAÇÃO DE LEITURA DE MEDIDORES DE CONSUMO DE ENERGIA  */
/*********************************************************************/
include_once ("inc/global.php"); // contém informações sobre a conexão com
o banco de dados.
/* se o formulario for preenchido o script entra no “if” abaixo e, conforme
a ação (inserir, editar ou deletar) acessa os “if’s” subseqüentes. */
if($form_status == "ok"){
      if($acao == "inserir"){
            $SQL =      "SELECT no_classificacaocliente ".
                  "FROM tb_classificacoesclientes ".
                  "WHERE              no_classificacaocliente             =
'$no_classificacaocliente'";
            $dbq=$db->execute($SQL);
            $dbq->close();
            if($dbq->getNumOfRows() > 0){
                  $MSG =      "Erro: ". $db->errorMsg() . "<br>".
                        "Mensagem do sistema: Classificação de cliente ja
cadastrado! <br>".
                        "Data: ". date("d/m/Y", time())."<br>";
            }else{
                  $SQL_HISTORICO =  "INSERT  INTO  tb_classificacoesclientes
".
                                    "VALUES                                 ('',
'$no_classificacaocliente')";
                  $dbq=$db->execute($SQL_HISTORICO);
                  if ($db->errorMsg()){
                        $MSG .= "Erro: ". $db->errorMsg() . "<br>";
                  }
                  $MSG .= "Mensagem do sistema: classificação de cliente
cadastrado com sucesso! <br>".
                        "Data: ". date("d/m/Y", time())."<br>";
                  $dbq->close();
            }
      }elseif($acao == "editar"){
            $SQL_HISTORICO =  "UPDATE tb_classificacoesclientes ".
                              "SET
no_classificacaocliente='$no_classificacaocliente'".
                              "WHERE        id_classificacaocliente       =
'$id_classificacaocliente'";
            $dbq=$db->execute($SQL_HISTORICO);
            if ($db->errorMsg()){
                  $MSG .= "Erro: ". $db->errorMsg() . "<br>";
            }
            $MSG  .=  "Mensagem  do  sistema:  classificação  de  cliente
atualizado com sucesso! <br>".
                  "Data: ". date("d/m/Y", time())."<br>";
            $dbq->close();
      }elseif($acao == "deletar"){
            $SQL_HISTORICO = "DELETE FROM tb_classificacoesclientes WHERE
id_classificacaocliente = '$id_classificacaocliente'";
            $dbq=$db->execute($SQL_HISTORICO);
            if ($db->errorMsg()){
                  $MSG .= "Erro: ". $db->errorMsg() . "<br>";
            }
            $MSG .= "Mensagem do sistema: classificação de cliente deletado
com sucesso! <br>".
            "Data: ". date("d/m/Y", time())."<br>";
            $dbq->close();
```

```
	}
}
if(isset($id_classificacaocliente)){
	$STATUS_BUTAO_CADASTRO = " disabled";
	$SQL =	"SELECT * ".
		"FROM tb_classificacoesclientes ".
		"WHERE id_classificacaocliente = '$id_classificacaocliente'";
	$dbq=$db->execute($SQL);
		$ID_CLASSIFICACAOCLIENTE	=	$dbq->fields['id_classificacaocliente'];
		$NO_CLASSIFICACAOCLIENTE	=	$dbq->fields['no_classificacaocliente'];
	$dbq->close();
}
?>
<html>
<head>
<title>AUTOMAÇÃO DE LEITURA DE MEDIDORES DE CONSUMO DE ENERGIA</title>
<script language="JavaScript">
<!-
	// o script seleciona acao, configure a variável que diz ao php que ação deve executar.
	function seleciona_acao(ACAO){
		document.formulario_cad_classificacaocliente.acao.value = ACAO;
		document.formulario_cad_classificacaocliente.submit();
	}
//-->
</script>
</head>
<body>

<?
	// imprime resultados da ações executadas pelo php
	if (isset($MSG)){
?>
<table border=1 cellpadding="5" cellspacing="0" align="center" width="98%">
	<tr >
		<td colspan=2>Resultados </td>
	</tr>
	<tr >
		<td><?=$MSG?></td>
	</tr>
</table><br>
<?
	}
?>


<form	action='<?=$PHP_SELF?>'	method='GET' name='formulario_cad_classificacaocliente'>
<input	type='hidden'	name='id_classificacaocliente' value='<?=$ID_CLASSIFICACAOCLIENTE?>'>
<input type='hidden' name='form_status' value='ok'>
<input type='hidden' name='acao' value=>
<table	border=1	cellpadding="5"	cellspacing="0"	bordercolor="#000000" align="center" width="98%">
<tr>
	<td	height="15"	colspan=2>::	Cadastro	de	classificações	de clientes</td>
</tr>
<tr>
	<td valign='top'>Nome: </td>
	<td><input	type='text'	name='no_classificacaocliente' value='<?=$NO_CLASSIFICACAOCLIENTE?>'	size=65>	<i><font size=1>(Letras)</font></i></td>
</tr>
<tr >
	<td colspan=2 align="center">
```

```
            <input  type='button'<?=$STATUS_BUTAO_CADASTRO?>  value='Gravar
novo' onClick="this.disabled=true;seleciona_acao('inserir')">
            <input      type='button'      value='Gravar      atualização'
onClick="this.disabled=true;seleciona_acao('editar')">
            <input               type='button'               value='Deletar'
onClick="this.disabled=true;seleciona_acao('deletar')">
      </td>
</tr>
</table>
</form>

<table  border=1  cellpadding="5"  cellspacing="0"  bordercolor="#000000"
align="center" width="98%">
<tr>
      <td  height="15"  colspan=3>:: Lista  de  classificações  de  clientes
<font size=1"><i>(Clique para editar/deletar)</i></font></td>
</tr>
<?
      $SQL   =   "SELECT   *   FROM   tb_classificacoesclientes   ORDER   BY
no_classificacaocliente";
      $dbq=$db->execute($SQL);
      while (!$dbq->EOF){
            echo "      <tr>\n";
            for($i=0;$i<3;$i++){
                  echo "            <td><a
href='classificacoes.php?id_classificacaocliente=".$dbq-
>fields['id_classificacaocliente']."'>".$dbq-
>fields['no_classificacaocliente']."</a></td>\n";
                  $dbq->nextRow();
            }
            echo "      </tr>\n";
      }
      $dbq->close();
?>
</table>
</body>
</html>
```

1.6 INSTALACOES.PHP

```
<?
/*********************************************************************/
/* AUTOR: BRUNO MESQUITA SANTANA                                     */
/* DATA: 03/06/2007                                                  */
/* NOME DO ARQUIVO: INSTALACOES.PHP                                  */
/* FUNÇÃO: ESTE É O ARQUIVO DE CONFIGURAÇÃO DOS TIPOS DE INSTALAÇÕES*/
/*       ELÉTRICAS EXISTENTES DO SISTEMA FRONT-END DO PROJETO DE  */
/*       CONCLUSÃO DO CURSO DE ENGENHARIA DE COMPUTAÇÃO DO AUTOR: */
/*       AUTOMAÇÃO DE LEITURA DE MEDIDORES DE CONSUMO DE ENERGIA  */
/*********************************************************************/
// os scripts seguem o mesmo padrão adotado pelo classificações.php
include_once ("inc/global.php");

if($form_status == "ok"){
      if($acao == "inserir"){
            $SQL =      "SELECT no_tipoinstalacaoeletrica ".
                  "FROM tb_tiposinstalacoeseletricas ".
                  "WHERE            no_tipoinstalacaoeletrica            =
'$no_tipoinstalacaoeletrica'";
            $dbq=$db->execute($SQL);
            $dbq->close();
            if($dbq->getNumOfRows() > 0){
                  $MSG =      "Erro: ". $db->errorMsg() . "<br>".
                        "Mensagem do sistema: Tipo de instalacao elétrica
ja cadastrado! <br>".
                        "Data: ". date("d/m/Y", time())."<br>";
            }else{
                  $SQL_HISTORICO =  "INSERT                                 INTO
tb_tiposinstalacoeseletricas ".
                                    "VALUES                                 ('',
'$no_tipoinstalacaoeletrica')";
                  $dbq=$db->execute($SQL_HISTORICO);
                  if ($db->errorMsg()){
                        $MSG .= "Erro: ". $db->errorMsg() . "<br>";
                  }
                  $MSG .= "Mensagem do sistema: tipo de instalação elétrica
cadastrado com sucesso! <br>".
                        "Data: ". date("d/m/Y", time())."<br>";
                  $dbq->close();
            }
      }elseif($acao == "editar"){
            $SQL_HISTORICO =  "UPDATE tb_tiposinstalacoeseletricas ".
                              "SET
no_tipoinstalacaoeletrica='$no_tipoinstalacaoeletrica'".
                              "WHERE       id_tipoinstalacaoeletrica      =
'$id_tipoinstalacaoeletrica'";
            $dbq=$db->execute($SQL_HISTORICO);
            if ($db->errorMsg()){
                  $MSG .= "Erro: ". $db->errorMsg() . "<br>";
            }
            $MSG  .= "Mensagem do sistema: tipo de instalação elétrica
atualizado com sucesso! <br>".
                  "Data: ". date("d/m/Y", time())."<br>";
            $dbq->close();
      }elseif($acao == "deletar"){
            $SQL_HISTORICO  =  "DELETE  FROM  tb_tiposinstalacoeseletricas
WHERE id_tipoinstalacaoeletrica = '$id_tipoinstalacaoeletrica'";
            $dbq=$db->execute($SQL_HISTORICO);
            if ($db->errorMsg()){
                  $MSG .= "Erro: ". $db->errorMsg() . "<br>";
            }
            $MSG  .= "Mensagem do sistema: Tipo de instalação elétrica
deletado com sucesso! <br>".
            "Data: ". date("d/m/Y", time())."<br>";
            $dbq->close();
      }
```

```
}
if(isset($id_tipoinstalacaoeletrica)){
	$STATUS_BUTAO_CADASTRO = " disabled";
	$SQL =	"SELECT * ".
		"FROM tb_tiposinstalacoeseletricas ".
		"WHERE		id_tipoinstalacaoeletrica		=
'$id_tipoinstalacaoeletrica'";
	$dbq=$db->execute($SQL);
		$ID_TIPOINSTALACAOELETRICA	=			$dbq-
>fields['id_tipoinstalacaoeletrica'];
		$NO_TIPOINSTALACAOELETRICA	=			$dbq-
>fields['no_tipoinstalacaoeletrica'];
	$dbq->close();
}
?>
<html>
<head>
<title>AUTOMAÇÃO DE LEITURA DE MEDIDORES DE CONSUMO DE ENERGIA</title>
<script language="JavaScript">
<!--
	function seleciona_acao(ACAO){
		document.formulario_cad_instalacaoeletrica.acao.value = ACAO;
		document.formulario_cad_instalacaoeletrica.submit();
	}
//-->
</script>
</head>
<body>

<?
	if (isset($MSG)){
?>
<table border=1 cellpadding="5" cellspacing="0" align="center" width="98%">
	<tr >
		<td colspan=2>Resultados </td>
	</tr>
	<tr >
		<td><?=$MSG?></td>
	</tr>
</table><br>
<?
	}
?>


<form		action='<?=$PHP_SELF?>'		method='GET'
name='formulario_cad_instalacaoeletrica'>
<input		type='hidden'		name='id_tipoinstalacaoeletrica'
value='<?=$ID_TIPOINSTALACAOELETRICA?>'>
<input type='hidden' name='form_status' value='ok'>
<input type='hidden' name='acao' value=>
<table	border=1	cellpadding="5"	cellspacing="0"	bordercolor="#000000"
align="center" width="98%">
<tr>
	<td	height="15"	colspan=2>::	Cadastro	de	tipos	de	instalações
elétricas</td>
</tr>
<tr>
	<td valign='top'>Nome: </td>
	<td><input		type='text'		name='no_tipoinstalacaoeletrica'
value='<?=$NO_TIPOINSTALACAOELETRICA?>'		size=65>		<i><font
size=1>(Letras)</font></i></td>
</tr>
<tr >
	<td colspan=2 align="center">
		<input	type='button'<?=$STATUS_BUTAO_CADASTRO?>	value='Gravar
novo' onClick="this.disabled=true;seleciona_acao('inserir')">
		<input	type='button'	value='Gravar	atualização'
```

```
onClick="this.disabled=true;seleciona_acao('editar')">
            <input                type='button'                value='Deletar'
onClick="this.disabled=true;seleciona_acao('deletar')">
      </td>
</tr>
</table>
</form>

<table   border=1   cellpadding="5"   cellspacing="0"   bordercolor="#000000"
align="center" width="98%">
<tr>
      <td height="15" colspan=3>:: Lista de tipos de instalações elétricas
<font size=1"><i>(Clique para editar/deletar)</i></font></td>
</tr>
<?
      $SQL   =   "SELECT   *   FROM   tb_tiposinstalacoeseletricas   ORDER   BY
no_tipoinstalacaoeletrica";
      $dbq=$db->execute($SQL);
      while (!$dbq->EOF){
            echo "      <tr>\n";
            for($i=0;$i<3;$i++){
                  echo "            <td><a
href='instalacoes.php?id_tipoinstalacaoeletrica=".$dbq-
>fields['id_tipoinstalacaoeletrica']."'>".$dbq-
>fields['no_tipoinstalacaoeletrica']."</a></td>\n";
                  $dbq->nextRow();
            }
            echo "      </tr>\n";
      }
      $dbq->close();
?>
</table>
</body>
</html>
```

1.7 RELATORIO.PHP

```
<?
/*********************************************************************/
/* AUTOR: BRUNO MESQUITA SANTANA                                     */
/* DATA: 03/06/2007                                                  */
/* NOME DO ARQUIVO: RELATORIO.PHP                                    */
/* FUNÇÃO: ESTE É O RESPONSAVEL POR DISPONIBILIZA A LEITURA DO       */
/*       SISTEMA FRONT-END DO PROJETO DE                           */
/*       CONCLUSÃO DO CURSO DE ENGENHARIA DE COMPUTAÇÃO DO AUTOR: */
/*       AUTOMAÇÃO DE LEITURA DE MEDIDORES DE CONSUMO DE ENERGIA  */
/*********************************************************************/
// os scripts seguem o mesmo padrão adotado pelo classificações.php
include ("inc/global.php");
?>
<html>
<head>
      <TITLE>AUTOMAÇÃO DE LEITURA DE MEDIDORES DE CONSUMO DE
ENERGIA</TITLE>
</head>
<body>
<?
if(!$id_cliente){
?>
      <form               action='<?=$PHP_SELF?>'               method='GET'
name='formulario_relatorio'>
      <table border=1 cellpadding="5" cellspacing="0" bordercolor="#000000"
align="center" width="98%">
      <tr>
            <td   height="15"   colspan=2>::   Relatório   de   consumo   de
energia</td>
      </tr>
      <tr>
            <td valign='top'>Cliente: </td>
            <td>
                  <select name="id_cliente">
                  <?
                        $SQL   =   "SELECT   *   FROM   tb_clientes   ORDER   BY
no_cliente";
                        $dbq = $db->execute($SQL);
                        while(!$dbq->EOF){
                              if($dbq->fields['id_cliente']==$ID_CLIENTE){
                                    $cliente = "selected";
                              }else{
                                    $cliente = "";
                              }
                              echo        "\t\t<option        value='".$dbq-
>fields['id_cliente']."'                              ".$classificacao.">".$dbq-
>fields['no_cliente']."</option>\n";
                              $dbq->nextRow();
                        }
                        $dbq->close();
                  ?>
                  </select>
            </td>
      </tr>
      <tr>
            <td valign='top'>Período: </td>
            <td>De:  <input  type="text"  name="periodo_ini"  value="AAAA-MM-
DD" size="10"> a: <input type="text" name="periodo_fim" value="AAAA-MM-DD"
size="10"></td>
      </tr>
      <tr >
            <td colspan=2 align="center">
                  <input type='submit' value='Visualizar'>
            </td>
      </tr>
      </table>
```

```
      </form>
<?
}else{
      $SQL =      "SELECT c.*, d.*, t.*, l.* ".
      "FROM tb_clientes c ".
      "LEFT JOIN tb_dadoscomplementares d ON (d.id_cliente = c.id_cliente)
".
      "LEFT       JOIN       tb_tiposinstalacoeseletricas       t       ON
(t.id_tipoinstalacaoeletrica = d.id_tipoinstalacaoeletrica) ".
      "LEFT JOIN tb_classificacoesclientes l ON (l.id_classificacaocliente
= d.id_classificacaocliente) ".
      "WHERE c.id_cliente = '$id_cliente'";
      $dbq=$db->execute($SQL);
            $ID_CLIENTE             = $dbq->fields['id_cliente'];
            $NO_CLIENTE             = $dbq->fields['no_cliente'];
            $EE_CLIENTE             = $dbq->fields['ee_cliente'];
            $NO_CIDADE              = $dbq->fields['no_cidade'];
            $SG_UF                        = $dbq->fields['sg_uf'];
            $NR_MEDIDOR             = $dbq->fields['nr_medidor'];
            $NO_TIPOINSTALACAOELETRICA    =                                $dbq-
>fields['no_tipoinstalacaoeletrica'];
            $NO_CLASSIFICACAOCLIENTE      =                                $dbq-
>fields['no_classificacaocliente'];
            $NR_CONSTANTE                 = $dbq->fields['nr_constante'];
      $dbq->close();
?>

      <table border=1 cellpadding="5" cellspacing="0" bordercolor="#000000"
align="center" width="98%">
      <tr>
            <td   height="15"   colspan=2>::   Relatório   de   consumo   de
energia</td>
      </tr>
      <tr>
            <TD                                    colspan="2"><?=$ID_CLIENTE?>
<?=$NO_CLIENTE?><br><?=$EE_CLIENTE?><br><?=$NO_CIDADE?> - <?=$SG_UF?></TD>
      </tr>
      <tr>
            <TD  colspan="2">Medidor:  <?=$NR_MEDIDOR?><br>Classificação  do
cliente:  <?=$NO_CLASSIFICACAOCLIENTE?><br>Tipo  de  instalação  elétrica:
<?=$NO_TIPOINSTALACAOELETRICA?><br>Constante: <?=$NR_CONSTANTE?></TD>
      </tr>
      <tr>
            <td     height="15"     colspan=2>::     Período     solicitado:
<?=$periodo_ini?> a <?=$periodo_fim?></td>
      </tr>
      <tr>
            <td      align="center">Data</td><td      align="center">Leitura
(Wh)</td>
      </tr>
<?
      $SQL =      "SELECT * FROM tb_leituraconsumo ".
            "WHERE id_cliente='$id_cliente' ".
            "AND     dt_leituraconsumo     BETWEEN     '$periodo_ini'     AND
'$periodo_fim'";
      $dbq=$db->execute($SQL);
      while (!$dbq->EOF){
            echo                "\t<tr><td                width='50%'>".$dbq-
>fields['dt_leituraconsumo']."</td><td                width='50%'>".$dbq-
>fields['de_leituraconsumo']."</td></tr>\n";
            $dbq->nextRow();
      }
      $dbq->close();
?>
      </table>
<?
}
?>
```

```
</body>
</html>
```

1.8 RECEBESMS.PHP

```
<?php
/*********************************************************************/
/* AUTOR: BRUNO MESQUITA SANTANA                                     */
/* DATA: 04/06/2007                                                  */
/* NOME DO ARQUIVO: RECEBESMS.PHP                                    */
/* FUNÇÃO: ESTE É O ARQUIVO DO SISTEMA FRONT-END DO PROJETO DE       */
/*       CONCLUSÃO DO CURSO DE ENGENHARIA DE COMPUTAÇÃO DO AUTOR: */
/*       AUTOMAÇÃO DE LEITURA DE MEDIDORES DE CONSUMO DE ENERGIA  */
/*       ESTE ARQUIVO FAZ A LEITURA DOS SMS DO CELULAR E INSERE   */
/*       OS DADOS DA LEITURA NO BANCO DE DADOSUSUÁRIO.            */
/*********************************************************************/
// os scripts seguem o mesmo padrão adotado pelo classificações.php
include("inc/global.php");
// a função “exec”, como explicada na monografia, permite executar um
comando do sistema operacional (no caso o LINUX).
exec ('getsms -d/dev/ttyUSB0',$resultado,$status) or die ("Não foi possível
executar o comando getsms");
if($status == 0){
      $nr_remetente           = $resultado[0];
      $nr_centralremetente    = $resultado[1];
      $dt_smsenviado          = $resultado[2];
      $dt_smsrecebido         = $resultado[3];
      $assunto          = $resultado[4];
      $alfabeto         = $resultado[5];
      //$seis                 = $resultado[6]; //--> Não utilizado
      $texto                  = $resultado[7];
      // Tratamento dos dados para serem armazenados no banco de dados
      $nr_remetente           = substr($nr_remetente,7);
      $dt_smsrecebido         = "20".substr($dt_smsrecebido,10);
      $id_cliente       = substr($texto,0,strpos($texto,";"));
      $leitura          = (substr($texto,2))*3.6;
      if ($ApresentaDetalhes == "Sim"){
            echo "<h3><center>Dados Recebidos via SMS, inseridos no banco
de dados</center></h3>\n";
            echo "Remetente = $nr_remetente\n";
            echo "<br>\n";
            echo "Data  = $dt_smsrecebido\n";
            echo "<br>\n";
            echo "Cliente     = $id_cliente\n";
            echo "<br>\n";
            echo "Leitura     = $leitura\n";
            echo "<br><br><br>\n";
            echo    "<input    type='button'    value='Receber    novamente'
onClick=\"JavaScript:document.location.href='RecebeSMS.php?ApresentaDetalhe
s=Sim'\">\n";
      }
      $SQL      =      "INSERT      INTO      tb_leituraconsumo      values
('',$id_cliente,'$dt_smsrecebido','1',$leitura)";
      $dbq = $db->execute($SQL);
      $dbq->close();
}else{
      $fp                                                               =
fopen("/dados/UniCEUB/ProjetoFinal/Monografia/Anexos/php/logErros.log",
"a+");
      $data = "Data: ".date("Y-m-d H:i:s",time())."\t - \tErro: ";
      if ($ApresentaDetalhes == "Sim"){
            echo  "<h3><center>Ocorreu  erro  na  tentativa  de  receber  o
SMS:</center></h3>";
      }
      switch ($status){
            case 1:
                  if ($ApresentaDetalhes == "Sim"){
                        echo "$data Não foi possível conectar o celular\n";
                        fwrite($fp,"$data  Não  foi  possível  conectar  o
celular\n");
                  }else{
```

```
                        fwrite($fp,"$data Não foi possível conectar o
celular\n");
                }
                break;
            case 2:
                if ($ApresentaDetalhes == "Sim"){
                    echo "$data Erro no chip (PIN)\n";
                    fwrite($fp,"$data Erro no chip (PIN)\n");
                }else{
                    fwrite($fp,"$data Erro no chip (PIN)\n");
                }
                break;
            case 3:
                if ($ApresentaDetalhes == "Sim"){
                    echo "$data Problema na inicialização do
celular\n";
                    fwrite($fp,"$data Problema na inicialização do
celular\n");
                }else{
                    fwrite($fp,"$data Problema na inicialização do
celular\n");
                }
                break;
            case 4:
                if ($ApresentaDetalhes == "Sim"){
                    echo "$data Não existe SMS para receber\n";
                    fwrite($fp,"$data Não existe SMS para receber\n");
                }else{
                    fwrite($fp,"$data Não existe SMS para receber\n");
                }
                break;
            case 5:
                if ($ApresentaDetalhes == "Sim"){
                    echo "$data SMS retornou erro de status\n";
                    fwrite($fp,"$data SMS retornou erro de status\n");
                }else{
                    fwrite($fp,"$data SMS retornou erro de status\n");
                }
                break;
        }
        if ($ApresentaDetalhes == "Sim"){
            echo "<br><br><br>\n";
            echo "<input type='button' value='Receber novamente'
onClick=\"JavaScript:document.location.href='RecebeSMS.php?ApresentaDetalhe
s=Sim'\">\n";
        }
}
?>
```

1.9 INC/GLOBAL.PHP

```
<?
/*********************************************************************/
/* AUTOR: BRUNO MESQUITA SANTANA                                     */
/* DATA: 03/06/2007                                                  */
/* NOME DO ARQUIVO: GLOBAL.PHP                                       */
/* FUNÇÃO: ESTE É O ARQUIVO CONTEM AS CONFIGURACOES PARA ACESSAR O   */
/*       BANCO DE DADOS. ESTE ACESSO OCORRE COM AUXILIO DE UMA       */
/*       BIBLIOTECA GPL.                                             */
/*********************************************************************/
// Informações de acesso ao banco de dados
$database_type          = "mysql";
$database_host          = "localhost";
$database_user          = "root";
$database_user_pass     = "";
$database_name          = "salmce";
// Informações de acesso aos arquivos do sistema.
$dir_inc          =
"/dados/UniCEUB/ProjetoFinal/Monografia/Anexos/php/inc/";
$dir_raiz         = "/dados/UniCEUB/ProjetoFinal/Monografia/Anexos/php";
$dir_home         = "";

// Mysql:
  include ($dir_inc . "phpDB-mysql.inc");
  $db = new phpDB();
  $db->pconnect("$database_host", "$database_user", "$database_user_pass",
"$database_name") or die ("<center><font color='red'><p><b>Ocorreu um erro
no     banco     de     dados.     Entre     em     contato     com     <a
href='mailto:bruno@bruno.eng.br'>Bruno     Mesquita     Santana,</a>     o
administrador desse site.</b></p></font></conter>");

?>
```

1.10 PHPDB-MYSQL.INC

```
<?
/*
   File name         : phpDB-mysql.inc
   Version           : 1.1.0
   Author            : Joe Thong
   Purpose           : phpDB MySQL module
   Last modified     : 03 Sep 2000

   Copyright (c) Joe Thong Chean Fonk.
   All rights reserved.
   Este arquivo é de licensa livre, tendo sido utilizado para realizar as
operações de acesso à banco de dados pelo sistema desenvolvido por Bruno M
Santana em seu trabalho de conclusão de curso.
*/

if (!defined("_PHPDB_ABSTRACT_LAYER")) {
  define("_PHPDB_ABSTRACT_LAYER", 1 );
}


class phpDB {

      /*    public variables  */
      var $version = '1.02bR6';           /*    Version number of phpDB */
      var $databaseType = '';             /*    This  variable  keeps  what
database type is going to be used. Current supported database server are
MySQL, MSQL, SQL Server, and Sybase*/
      var $databaseName = '';             /*    Specifies which database is
going to be used. */
      var $hostname = '';                 /*    The  hostname  of  the
database server, port number is optional. e.g: "db.devNation.com"     */
      var $username = '';                 /*    The username which is
used to connect to the database server. */
      var $password = '';                 /*    Password    for    the
username.   */

      /*    private variables */
      var $_queryIDList = array();        /*    An  array  of  executed
querys.  For results cleanup purposes.    */
      var $_connectionID     = -1;        /*    The     returned
link identifier whenever a successful database connection is made.     */
      var $_errorMsg = '';                /*    A variable which
was used to keep the returned last error message.   The value will then
returned by the errorMsg() function*/
      var $_queryID = -1;                 /*    This
variable keeps the last created result link identifier.          */
      var   $_isPersistentConnection = false;  /*    A boolean variable to
state whether its a persistent connection or normal connection.  */
      var $_tempResultObj = '';           /*    Holds  the  newly
created result object, returned via the execute() method.  */

      /*    A   constructor   function   for   the   phpDB   object.     When
initializing, specify the dbType
            i.e: "mysql", "msql", "postgresql", "mssql", and "sybase"  */

      function phpDB($dbType = "mysql") {
            switch ($dbType) {
                  case "mysql":
                  case "msql":
                  case "postgresql":
                  case "mssql":
                  case "sybase":
                  case "informix":
                        $this->databaseType = $dbType;
                        break;
                  default:
```

```
                        return false;
                }
        }

        /*    Returns: A positive link identifier on success, or false on
error.
                Connect to the server with the provided arguments. The
connection to the server will be closed when the script
                terminates, unless close() function is called beforehand.  */

        function connect($argHostname = "", $argUsername = "", $argPassword =
"", $argDatabaseName = "") {
                $boolDBSelected;
                if ($argHostname != "") {
                        $this->hostname = $argHostname;
                }
                if ($argUsername != "") {
                        $this->username = $argUsername;
                }
                if ($argPassword != "") {
                        $this->password = $argPassword;
                }
                if ($argDatabaseName != "") {
                        $this->databaseName = $argDatabaseName;
                }

                $this->_connectionID  =  @mysql_connect($this->hostname,  $this-
>username, $this->password);

                if ($this->databaseName && $this->_connectionID) {
                        $boolDBSelected = @mysql_select_db($this->databaseName);
                        if(!$boolDBSelected) {  /*    If DB selection fails   */
                                @mysql_close($this->_connectionID);/*    Close   the
current connection      */
                                return false;
                        }
                }
                return $this->_connectionID;
        }

        /*    Returns: A positive link identifier on success, or false on
error.
                Connect  to  the  server  with  the  provided  arguments.  The
connection to the server will not be closed when the
                script  terminates.  Instead  it  will  be  kept  for  later  future
use.  */

        function pconnect($argHostname = "", $argUsername = "", $argPassword
= "", $argDatabaseName = "") {
                global $boolDBSelected;
                if ($argHostname != "") {
                        $this->hostname = $argHostname;
                }
                if ($argUsername != "") {
                        $this->username = $argUsername;
                }
                if ($argPassword != "") {
                        $this->password = $argPassword;
                }
                if ($argDatabaseName != "") {
                        $this->databaseName = $argDatabaseName;
                }

                $this->_connectionID  =  @mysql_pconnect($this->hostname,  $this-
>username, $this->password);
                if ($this->_connectionID) {
                        $this->_isPersistentConnection = true;
                }
```

```
            if ($this->databaseName && $this->_connectionID) {
                  $boolDBSelected = @mysql_select_db($this->databaseName);
                  if(!$boolDBSelected) {  /*    if DB selection fails   */
                        return false;     /*    Persistent  connection  can't
be closed   */
                  }
            }
            return $this->_connectionID;
      }

      /*    Returns: true on success, false on error
            Select the database name to be used*/

      function selectDB($dbName) {
            $this->databaseName = $dbName;
            if ($this->_connectionID) {
                  return @mysql_select_db($dbName);
            }
            else {
                  /*    No database selected    */
                  return false;
            }
      }

      /*    Returns: the Recordset object disregard success or failure
            Send the sql statement to the database server. */

      function execute($sql = "") {
            $this->_queryID = @mysql_query($sql, $this->_connectionID);
            /*    Instantiate  an  object  without  considering  whether  the
query return any results or not.    */
            $this->_tempResultObj = new Recordset($this->_queryID);
            $this->_insertQuery($this->_queryID);
            return $this->_tempResultObj;
      }

      /*    Returns:  the  last  error  message  from  previous  database
operation   */

      function errorMsg() {
            $this->_errorMsg = @mysql_error($this->_connectionID);
          return $this->_errorMsg;
      }

      /*    Returns: true on success, false on failure
            Close the database connection.      */

      function close() {
            if ($this->_queryIDList && sizeof($this->_queryIDList > 0)) {
                  while(list($_key,      $_resultID)      =      each($this-
>_queryIDList)) {
                        @mysql_free_result($_resultID);
                  }
            }
            if ($this->_isPersistentConnection != true) {  /*    If its not
a persistent connection,

                  then only the connection needs to be closed    */
                  return @mysql_close($this->_connectionID);
            }
            else {
                  return true;
            }
      }

      /*    A  PRIVATE  function  used  by  the  constructor  function  of  the
query object.  insert
```

```
            the successful returned query id to the query id list.  Used
for later
            results cleanup.  A private function that's never meant to be
used directly.    */

      function _insertQuery($query_id) {
            $this->_queryIDList[] = $query_id;
      }
}

/*--------------------------------------------------------------------------
-------------
       Class Name: Recordset
--------------------------------------------------------------------------
-----------*/

class Recordset {
      /*    public variables  */

      var $fields;

      var $BOF = null;  /*    indicates that the current record position is
before the first record in a Recordset object. */

      var $EOF = null;  /*    indicates that the current record position is
after the last record in a Recordset object.    */

      /*    private variables */

      var $_numOfRows = -1;   /*    NEVER change the value!  READ-ONLY! */

      var $_numOfFields = -1; /*    NEVER change the value!  READ-ONLY! */

      var $_tempResult = '';  /*    Holds anything that was returned from
the database specific functions.    */

      var $_queryID = -1;     /*    This variable keeps the result link
identifier. */

      var $_currentRow = -1;  /*    This variable keeps the current row in
the Recordset.    */

      /*    Returns: query id on success and false if failed
            Constructor function    */

      function Recordset($queryID) {
            $this->_queryID = $queryID;
            if ($queryID) {
                  $this->_numOfRows = @mysql_num_rows($this->_queryID);
                  $this->_numOfFields = @mysql_num_fields($this->_queryID);
            }
            else {
                  $this->_numOfRows = 0;
                  $this->_numOfFields = 0;
            }
            if ($this->_numOfRows > 0 && $this->_currentRow == -1) {   /*
      If result set contains rows   */
                  $this->_currentRow = 0;
                  $this->fields = @mysql_fetch_array($this->_queryID);
                  $this->EOF = false;
                  $this->BOF = false;
            }
            return $this->_queryID;
      }

      /*    Returns: true if successful, false if fail
            Set the Recordset pointer to a specified field offset. If the
next call to fetchField() won't include a field offset, this
```

```
            field would be returned.      */

      function fieldSeek($fieldOffset = -1) {
            $this->_tempResult     =     @mysql_field_seek($this->_queryID,
$fieldOffset);
            return $this->_tempResult;
      }

      /*    Returns: an object containing field information.
            Get column information in the Recordset object. fetchField()
can be used in order to obtain information about
            fields in a certain query result. If the field offset isn't
specified, the next field that wasn't yet retrieved by
            fetchField() is retrieved.    */

      function fetchField($fieldOffset = -1) {
            if ($fieldOffset != -1) {
                  $this->_tempResult  =  @mysql_fetch_field($this->_queryID,
$fieldOffset);
            }
            else if ($fieldOffset == -1) {      /*    The       $fieldOffset
argument is not provided thus its -1      */
                  $this->_tempResult = @mysql_fetch_field($this->_queryID);
            }
            return $this->_tempResult;
      }

      /*    Returns: true if there still rows available, or false if there
are no more rows.
            Moves to the next row in a specified Recordset object and makes
that record the current row
            and the data corresponding to the row will be retrieved into
the fields collection.
            Note:  Unlike  the  moveRow()  method,  when  _currentRow  is
getNumOfRows() - 1, EOF will immediately be true.
            If row number is not provided, the function will point to the
first row automatically.      */

      function nextRow() {
            if ($this->getNumOfRows() > 0) {
                  $this->fields = array();
                  $this->_currentRow++;
                  $this->fields = @mysql_fetch_array($this->_queryID);
                  if ($this->fields) {    /*    This is not working.   True
all the time      */
                        $this->_checkAndChangeEOF($this->_currentRow - 1);
                        return true;
                  }
            }
            $this->EOF = true;
            return false;
      }

      /*    Returns: true on success, false on failure
            moveRow()  moves  the  internal  row  pointer  of  the  Recordset
object to point to the specified row number
            and the data corresponding to the row will be retrieved into
the fields collection.
            If row number is not provided, the function will point to the
first row automatically.      */

      function moveRow($rowNumber = 0) {
            if ($rowNumber == 0) {
                  return $this->firstRow();
            }
            else if ($rowNumber == ($this->getNumOfRows() - 1)) {
                  return $this->lastRow();
            }
```

```
            if ($this->getNumOfRows() > 0 && $rowNumber < $this->getNumOfRows()) {
                  $this->fields = null;
                  $this->_currentRow = $rowNumber;
                  if(@mysql_data_seek($this->_queryID, $this->_currentRow)) {
                        $this->fields      =      @mysql_fetch_array($this->_queryID);
                        if ($this->fields) {    /*    This is not working. True all the time */
                              $this->EOF = false; /* No need to call _checkAndChangeEOF() because the possibility of moving to the last row has been handled by the above code.     */
                              return true;
                        }
                  }
            }
            $this->EOF = true;
            return false;
      }

      /*    Returns: true on success, false on failure
            firstRow() moves the internal row pointer of the Recordset object to the first row
            and the data corresponding to the row will be retrieved into the fields collection. */

      function firstRow() {
            if ($this->getNumOfRows() > 0) {
                  $this->fields = array();
                  $this->_currentRow = 0;
                  if        (@mysql_data_seek($this->_queryID,        $this->_currentRow)) {
                        $this->fields      =      @mysql_fetch_array($this->_queryID);
                        $this->EOF = false;
                        if ($this->fields) {    /*    This is not working. True all the time */
                              return true;
                        }
                  }
            }
            $this->EOF = true;
            return false;
      }

      /*    Returns: true on success, false on failure
            lastRow() moves the internal row pointer of the Recordset object to the last row
            and the data corresponding to the row will be retrieved into the fields collection. */

      function lastRow() {
            if ($this->getNumOfRows() > 0) {
                  $this->fields = array();
                  $num_of_rows = $this->getNumOfRows();
                  $this->_tempResult = @mysql_data_seek($this->_queryID, --$num_of_rows);
                  if ($this->_tempResult) {
                        $this->_currentRow = $num_of_rows; /*
      $num_of_rows decemented at above    */
                        $this->fields      =      @mysql_fetch_array($this->_queryID);
                        if ($this->fields) {    /*    This is not working. True all the time */
                              $this->EOF = false;     /*    Special case for making EOF false. */
                              return true;
```

```
                        }
                    }
                }
                $this->EOF = true;
                return false;
        }

        /*    close() only needs to be called if you are worried about using
too much memory while your script
                is running. All associated result memory for the specified
result identifier will automatically be freed. */

        function close() {
                $this->_tempResult = @mysql_free_result($this->_queryID);
                return $this->_tempResult;
        }

        /*    Returns: the number of rows in a result set.
        Get number of rows in result.*/

        function getNumOfRows() {
                return $this->_numOfRows;
        }

        /*    Returns: the number of fields in a result set.
        Get number of fields in result.    */

        function getNumOfFields() {
                return $this->_numOfFields;
        }

        /*    Check and change the status of EOF.*/
        function _checkAndChangeEOF($currentRow) {
                if ($currentRow >= ($this->_numOfRows - 1)) {
                        $this->EOF = true;
                }
                else {
                        $this->EOF = false;
                }
        }
}
?>
```

APÊNDICE 2 – CÓDIGO FONTE DO MICROCONTROLADOR

2.1 CONVERSOR.C

```
#include <16f877A.h>
#use delay(clock=4000000)
#fuses XT,NOWDT,PUT
#use rs232(baud=19200, xmit=PIN_C6,rcv=PIN_C7, parity=n, bits=8)
#include <regs_16f87x.h>

int32 amostragem[6];
int32 media_ref = 180;
int32 media_adq;
int i,j,k=0,l=0,m=0;

void enviaSMS(){
      /*
      ---------------------------------------------
      Pré-configurações para envio de mensagem
      ---------------------------------------------
      id_cliente;de_leitura   PDU
      ---------------------------------------------
      1;5         0011000c91551618771363000000003B15D0D
      1;10  0011000c9155161877136300000004B15D0C06
      1;15  0011000c9155161877136300000004B15DAC06
      2;5         0011000c91551618771363000000003B25D0D
      2;10  0011000c9155161877136300000004B25D0C06
      2;15  0011000c9155161877136300000004B25DAC06
      ---------------------------------------------
      */
      printf("AT+CMGS=160");
      putc(0x0d);
      delay_ms(1000);
      printf("0011000c91551618771363000000003B15D0D");
      putc(0x1A);
      delay_ms(1000);
}

main(){
      long int valor;
      int32 val32;
      setup_ADC_ports (RA0_analog);
      setup_adc(ADC_CLOCK_INTERNAL );
      set_adc_channel(0);
      i = 0;
      while (true){
            output_low(PIN_B7);
            valor = read_adc(); // efetua a conversão A/D
            if (valor) valor += 1;
            val32 = valor * 4 + ((int32)valor * 113)/128;

            if(i>5){
                  i = 0;
                  amostragem[i] = val32;
            }else{
                  amostragem[i] = val32;
            }

            media_adq = 0;
            for(j=0;j<6;j++){
                  media_adq += amostragem[i]/6;
            }
            /*
            printf("media_adq= %lu\r\n",media_adq);
            printf("media_ref= %lu\r\n\n",media_ref);
            */
            if(media_adq <= media_ref){
                  output_high(PIN_B7);
                  if(k >= 5){
```

```
                        /* printf("GOL\n\n\n"); */
                        enviaSMS();
                        k = 0;
                        delay_ms(2000);
                  }
                  k++;
                  delay_ms(4000);
            }
            /*printf ("Tensao = %lu\r\n",val32);*/
            delay_ms (400); // aguarda 400 ms
            i++;
      }
}
```

ANEXOS