UNIVERSIDADE REGIONAL DE BLUMENAU

CENTRO DE CIÊNCIAS EXATAS E NATURAIS

CURSO DE CIÊNCIAS DA COMPUTAÇÃO – BACHARELADO

PROTÓTIPO DE SISTEMA DE CAPTURA DE DADOS MULTIPONTO WIRELESS PARA CONTROLE DE CONSUMO DE ÁGUA

BENNO MARTIM SCHUBERT

BLUMENAU
2008

2008/2-04

BENNO MARTIM SCHUBERT

PROTÓTIPO DE SISTEMA DE CAPTURA DE DADOS MULTIPONTO WIRELESS PARA CONTROLE DE CONSUMO DE ÁGUA

Trabalho de Conclusão de Curso submetido à Universidade Regional de Blumenau para a obtenção dos créditos na disciplina Trabalho de Conclusão de Curso II do curso de Ciências da Computação — Bacharelado.

Prof. Miguel Alexandre Wisintainer - Orientador

BLUMENAU
2008

2008/2-04

PROTÓTIPO DE SISTEMA DE CAPTURA DE DADOS MULTIPONTO WIRELESS PARA CONTROLE DE CONSUMO DE ÁGUA

Por

BENNO MARTIM SCHUBERT

Trabalho aprovado para obtenção dos créditos na disciplina de Trabalho de Conclusão de Curso II, pela banca examinadora formada por:

	_____
Presidente:	Prof. Miguel Alexandre Wisintainer – Orientador, FURB
	_____
Membro:	Prof. Mauro M. Matos – FURB
	_____
Membro:	Prof. Antonio Carlos Tavares – FURB

Blumenau, 11 de fevereiro de 2009

Dedico este trabalho aos meus familiares e amigos, especialmente aqueles que foram pacientes durante a realização deste.

AGRADECIMENTOS

À Deus, por tudo.

À minha família, pelo apoio contínuo.

Ao meu orientador, Miguel Alexandre Wisintainer, pelo apoio e por ter acreditado na conclusão deste trabalho.

Aos meus amigos, pelos empurrões e cobranças.

"Quando todos os homens abrirem as portas dos seus corações, desaparecerão as trevas que envolvem este mundo."

Meishu-Sama

RESUMO

Este trabalho tem por objetivo apresentar o desenvolvimento de um protótipo de sistema de captura de dados de consumo e de análise de vazamentos um uma rede de distribuição de água. A captura dos dados dar-se-á através de microcontroladores PIC instalados nos hidrômetros e transmitidas para o PC através de transceptores *ZigBee* (*wireless*). No PC um software desenvolvido em Java armazena estes dados em um banco de dados MySQL. Conforme a solicitação do usuário o software do PC executa rotinas para verificar vazamentos, gerar relatórios periódicos de consumo, enviar aos microcontroladores PIC sinais de controle para atuar válvulas e outras funcionalidades.

Palavras-chave: Telemetria. *Wireless.* Microcontrolador. Desperdício de água.

ABSTRACT

This work´s objective is to present the development of a prototype system to capture data for consumption and leak analysis of a water distribution network. The capture of data will be given through PIC microcontrollers installed in water meters and transmitted to the PC via tranceptores *ZigBee* (wireless). In the PC a software developed in Java stores these data in a MySQL database. According to users requests, the PC software executes routines to check leaks, generate periodic consumption reports, send signals to PIC microcontrollers to control valves and other functionalities.

Key-words: Telemetry. Wireless. Microcontroler. Waste of water.

LISTA DE ILUSTRAÇÕES

SUMÁRIO

1 INTRODUÇÃO

Analisando a atmosfera de acontecimentos sociais que envolvem o ser humano, é possível perceber que algumas atitudes devem ser tomadas hoje, para reduzir os efeitos no futuro. Pode-se citar como exemplo, o aquecimento global e a falta de água potável no mundo.

> Relatórios recentes da Organização da [sic] Nações Unidas (ONU), mostram que cerca de 1,1 bilhão de pessoas em todo o mundo não têm acesso a água potável. Nos países em desenvolvimento, esse problema aparece relacionado a 80% das mortes e enfermidades. [..] Um total de 26 países sofrem escassez crônica de água e a previsão é de que em 2025 serão 3,5 bilhões de pessoas em 52 países nessa situação. (NÓRCIO, 2007, p. 1).

É possível perceber também a quantidade de informações que a evolução tecnológica permite ao ser humano trafegar pelos diversos meios de comunicação, seja por ondas de rádio, internet, fibra óptica, rede elétrica e outros. Permite inclusive, comandar máquinas remotamente, sem o uso de fios entre operador e máquina, ou o acesso a dados entre computadores a milhares de quilômetros de distância, cruzando diversos países e até continentes inteiros.

Seria interessante se os profissionais da área de tecnologia aproveitassem tais avanços, em prol de soluções para as situações críticas citadas anteriormente, em relação ao consumo de água potável. Alguns governos apóiam a criação e a manutenção de projetos que tenham este objetivo, como é o caso do Programa Nacional de Combate ao Desperdício de Água (PNCDA), onde em suas fases I e II apóia os planos de combate de desperdício de água.

Seguindo o raciocínio de redução de desperdício de água, a *Millennium Development Goals* (MDGs), criada pela Organização das Nações Unidas (ONU), tem como um de seus objetivos "[...] reduzir em 50% a proporção de pessoas sem acesso a água potável e saneamento básico." (UN, 2006, p. 223, tradução nossa). Um dos meios para contribuir com o cumprimento deste objetivo é através da redução de desperdício de água tratada.

Uma das soluções seria o desenvolvimento de um sistema informatizado que permitisse auxiliar na redução de desperdício de água tratada, através da análise constante na rede de distribuição, para identificar vazamentos e acessos clandestinos. É possível observar que a maioria das empresas de distribuição de água tratada contam apenas com a análise visual, onde em alguns casos, para se detectar um vazamento, podem levar alguns dias ou até meses e no caso de consumo clandestino o tempo pode ser muito maior. Com relação ao consumo clandestino, ressalta-se que, não havendo cobrança financeira, por não existir o

devido controle, a probabilidade do consumo desta rede possuir um alto nível de desperdício é grande, seja por falta de informação ou por falta de capital para manter a rede em plenas condições de uso. Conforme as informações apresentadas o trabalho proposto visa desenvolver um sistema de telemetria *wireless* para analisar o consumo e vazamentos em redes de distribuição de água para combater o desperdício de água.

1.1 OBJETIVOS DO TRABALHO

Com base nestas informações, o trabalho tem como objetivo o desenvolvimento de um protótipo de sistema de telemetria, para analisar o consumo e o desperdício de água tratada, através de sensores instalados nos consumidores finais e nas redes de distribuição, interligados remotamente a distribuidora através de uma rede sem fio.

Os objetivos específicos do trabalho são:

a) permitir visualizar remotamente o consumo de água;
b) detectar possíveis vazamentos na rede de distribuição de água tratada e nos consumidores finais;
c) gerar relatórios de consumo;
d) gerar mensagens de alerta;
e) permitir controlar remotamente válvulas para controle de vazamentos instaladas nas redes de distribuição e consumidores finais.

1.2 ESTRUTURA DO TRABALHO

O capítulo 2 apresenta uma introdução sobre microcontroladores, protocolo de comunicação, transceptores, transdutores, hidrômetros, telemetria, comunicação wireless, protocolo *ZigBee*, Circuito Integrado (CI) MAX 232, CI 555 e trabalhos correlatos.

No capítulo 3 são apresentadas a especificação e a implementação do protótipo. E, por fim, no capítulo 4, são apresentadas as considerações finais e as sugestões para extensão deste trabalho.

2 FUNDAMENTAÇÃO TEÓRICA

São vistos a seguir microcontroladores, protocolo de comunicação, transceptores, transdutores, hidrômetros, telemetria, comunicação *wireless*, protocolo *ZigBee*, CI MAX-232, CI 555, Proteus e trabalhos correlatos.

2.1 MICROCONTROLADORES

Microcontroladores são "[…] processadores com uma grande quantidade de recursos integrados" (ZELENOVSKY; MENDONÇA, 2006, p. 1).

Mikroeletronica (2003, p. 1) afirma que a diferença entre os microcontroladores e os microprocessadores é que para se usar um microprocessador é necessário o uso de outros componentes associados a ele, como memória e componentes para enviar e receber dados, ao contrário do microcontrolador que já possui estes componentes dentro dele.

Conforme Brain (2007, p. 1), microcontroladores são computadores, porém, de baixa potência. Enquanto um computador normal, ligado à rede elétrica consome aproximadamente 50 W (watts), um microcontrolador ligado a uma bateria pode consumir aproximadamente 50 mW (miliwatts). Os microcontroladores estão presentes nos mais diversos equipamentos, utilizados pelo homem, por exemplo, em microondas, televisores, rádios com *display* gráfico, carros, câmeras digitais, telefones celulares e outros.

2.2 PROTOCOLO DE COMUNICAÇÃO

Conforme Tafner, Loesch e Stringari (1996, p. 32), "O protocolo é um conjunto de regras pré-estabelecidas, cuja função é fazer com que a comunicação de dados entre equipamentos seja realizada com segurança e de forma ordenada.".

Os protocolos de comunicação podem ser classificados quanto aos modos de transmissão e quanto aos controles de transmissão.

Em relação aos modos de transmissão, Tafner, Loesch e Stringari (1996, p. 32)

afirmam que existem dois tipos, o assíncrono e o síncrono. O tipo de protocolo assíncrono utiliza bits de *start* e *stop* para delimitar um caracter, o que torna a comunicação menos eficiente em relação ao protocolo síncrono. Utiliza o código *American Standard Code for Information Interchange* (ASCII). Nesse modo de transmissão as estações secundárias podem iniciar uma transmissão a qualquer momento, sem a necessidade de autorização por parte da estação primária. Já no protocolo síncrono, os dados trabalham sincronicamente, ou seja, as estações de transmissão e recepção trabalham no mesmo *clock*, não sendo necessário a utilização de bits de *start* e *stop* e o código utilizado pode ser ASCII ou *Extended Binary Coded Decimal Interchange Code* (EBCDIC).

Quanto ao controle de transmissão, Tafner, Loesch e Stringari (1996, p. 32) afirmam que existem o orientado a caracter e o orientado a bit. O orientado a caracter utiliza símbolos especiais para efetuar o controle de operação e tráfego de suas mensagens. São os mais antigos, porém utilizados com freqüência em linhas de longa distância. Cita-se como exemplo o *Binary Sysnchronous Communication* (BSC). O formato típico de um bloco de protocolo orientado a caracter BSC pode ser visualizado no Quadro 1.

Quadro 1 – Formato de protocolo orientado a caracter BSC

Quanto ao controle de transmissão do tipo orientado a bit, Tafner, Loesch e Stringari (1996, p. 36) afirmam que são protocolos que não utilizam caracteres especiais, todo o controle é tratado em nível de bit, ou seja, os campos são formados por combinações binárias bem definidas. Um exemplo é o protocolo X-25 do Comitê Consultivo Internacional de Telegrafia e Telefonia (CCITT) e outro exemplo é o protocolo *Syncronous Data Link Control* (SDLC), presente na arquitetura de rede *Systems Network Architeture* (SNA) da International Business Machines (IBM) orientado de forma síncrona. Exemplo de um formato típico do protocolo orientado a bit pode ser visualizado no Quadro 2.

Quadro 2 - Formato de protocolo orientado a bit

Conforme Inteligência Computacional Aplicada (ICA) (ICA, 2007, p. 5), os campos apresentados na figura 2 podem ser definidos como:

flag: definem o início e o fim do quadro;

endereço: utilizado para identificar a estação secundária que transmitiu, ou que deve receber o quadro, necessário apenas em linhas multiponto;

controle: descreve o tipo do quadro; se ele é do tipo informação, supervisão ou não numerado;

texto: contém os dados a serem transmitidos. O tamanho é sempre múltiplo de 8 bits;

Frame Check Sequence (FSC): contém o código de detecção de erro, geralmente CRC[1].

2.3 TRANSCEPTORES E TRANSDUTORES.

Transceptor é um dispositivo eletrônico capaz de atuar como transmissor e receptor, utilizando componentes de circuito comuns para ambas as funções num só componente (TIOSAM, 2007, p. 1).

"Transdutores são componentes eletrônicos que efetuam conversão de energia de uma modalidade para outra onde, uma delas, é necessariamente energia elétrica." (NETTO, 2007, p. 1).

Transdutores de entrada são componentes onde uma energia qualquer será convertida em energia elétrica. Como exemplo citam-se: microfone, interruptor, gerador, hidrômetros, etc. Transdutores de saída são componentes onde a energia elétrica será convertida em uma energia qualquer. Como exemplos têm-se: alto-falantes, lâmpadas de filamento, motores elétricos, relés, etc (NETTO, 2007, p. 1).

2.4 HIDRÔMETROS

A Superintendência de Água e Esgotos de Ituituba (SAE) (SAE, 2002, p. 1) define que hidrômetro é um relógio que mede o consumo de água, ou seja, ele mede a quantidade de água tratada fornecida a cada cliente da distribuidora. Geralmente sua unidade de medida é o volume, expressos em m³ ou litros.

O hidrômetro digital permite que os dados sejam acessados através de uma saída serial (ZENNER, 2007, p. 8). Desta forma os dados podem ser transmitidos para estações remotas, onde podem ser analisados, armazenados em bancos de dados e/ou mostrados em um monitor de computador.

---

[1] Algoritmo utilizado para identificar erros em uma transmissão de dados. Efetua cálculo sobre os bits de uma mensagem (SILVA, 2007, p. 10).

2.5 TELEMETRIA

"TELEMETRIA é a transferência (via rede fixa ou sem fio) e utilização de dados provindos de múltiplas máquinas remotas, distribuídas em uma área geográfica de forma pré-determinada, para o seu monitoramento, medição e controle" (TELECO, 2007, p. 1).

De acordo com Cabral e Morelli Neto (2007, p. 1) telemetria é "[...] a técnica que mede quantidades, transmitindo os resultados para um centro de controle que é responsável pela interpretação, apresentação e/ou armazenamento dos valores medidos".

2.6 COMUNICAÇÃO *WIRELESS*

Conforme Kondo e Martineli (2007, p. 1), comunicação *wireless* é a comunicação sem fio entre dois pontos ou dispositivos, geralmente empregado na indústria de telecomunicações para definir sistemas de comunicação à distância, como controles remoto, redes de computadores, transmissores e receptores de rádio, telefones celulares, entre outros. Os dados podem trafegar por ondas de rádio, luz infravermelha, laser, ondas sonoras e outros.

> Dentro deste modelo de comunicação, enquadram-se várias tecnologias, como Wi-Fi, InfraRed (infravermelho), bluetooth e Wi-Max.
>
> Seu controle remoto de televisão ou aparelho de som, seu telefone celular e uma infinidade de aparelhos trabalham com conexões wireless. Podemos dizer, como exemplo lúdico, que durante uma conversa entre duas pessoas, temos uma conexão wireless, partindo do principio [sic] de que sua voz não utiliza cabos para chegar até o receptor da mensagem. (I9SOLUÇÔES, 2008).

"A tecnologia de comunicação wireless é composta de padrões estabelecidos pelo IEEE – *Institute of Electrical and Electronics Engineers* […]" (PEIXOTO, 2002, p. 1, grifo nosso). Conforme Peixoto (2002, p. 1), os padrões que recebem mais atenção ultimamente correspondem à família de especificações batizada de 802.11. Estes padrões especificam a interconexão de computadores, impressoras, dispositivos de vídeo e demais aplicações proporcionando o estabelecimento de redes e comunicações entre um aparelho cliente e uma estação ou ponto de acesso, com o uso de microondas de frequência de rádio. Estas redes são conhecidas como *Wireless Local Area Network* (WLAN), e são atualmente estabelecidas quatro especificações na família 802.11 onde cita 802.11, 802.11a, 802.11b, 802.11g. Estes padrões utilizam-se do protocolo Ethernet, comum em computadores pessoais e portáteis.

2.7 PROTOCOLO *ZIGBEE*

De acordo com André Teixiera da Silva (2007, p. 19) o protocolo *ZigBee* teve a sua primeira versão apresentada ao público em 27 de Julho de 2005, desenvolvida pela *ZigBee Alliance*, uma aliança constituída por mais de 200 empresas, oriundas de mais de 20 países distintos, na qual se integram também especialistas da área de telecomunicações e semicondutores, incluindo membros do IEEE. O objetivo do desenvolvimento do protocolo *ZigBee* pretende associar a transmissão de dados sem fios a um reduzido consumo energético e com elevada fiabilidade.

Apesar de permitir seu uso em diversas topologias de rede, foi através da topologia em malha que lhe foi associado o nome. Uma malha *ZigBee* apresenta múltiplos caminhos possíveis entre cada dispositivo, o que permite eliminar um possível ponto de falha, através do "zig" e "zag" da informação pela rede. O nome *ZigBee* surgiu da analogia da estrutura e modo de funcionamento da rede de comunicações para com o modo de vida das abelhas (SILVA, André, 2007, p. 21).

As características do protocolo *ZigBee* são:

a) baixo custo (EMBEDDEDWORLD, 2008, p. 1);
b) baixo consumo (EMBEDDEDWORLD, 2008, p. 1);
c) alta confiabilidade (EMBEDDEDWORLD, 2008, p. 1);
d) segurança (AES 128 bits) (EMBEDDEDWORLD, 2008, p. 1);
e) redes tipo estrela, árvore e malha (EMBEDDEDWORLD, 2008, p. 1);
f) uso do protocolo IEEE 802.15.4 (EMBEDDEDWORLD, 2008, p. 1);
g) velocidade de transmissão de até 250Kbps (SILVA, André, 2007, p. 21);
h) permite a utilização de redes com mais de 65.535 nós por cada coordenador *ZigBee* (SILVA, André, 2007, p. 20);
i) dois modos de operação da rede: *beaconing* e *non-beaconing* (SILVA, André, 2007, p. 20);
j) suporte para duas classes de dispositivos físicos definidos na norma IEEE 802.15.4, podendo ambos coexistir numa mesma rede, sendo o *Full Function Device* (FFD) e o *Reduced Function Device* (RFD), podem corresponder aos tipos lógicos coordenador, roteador e ponto final (SILVA, André, 2007, p. 20).

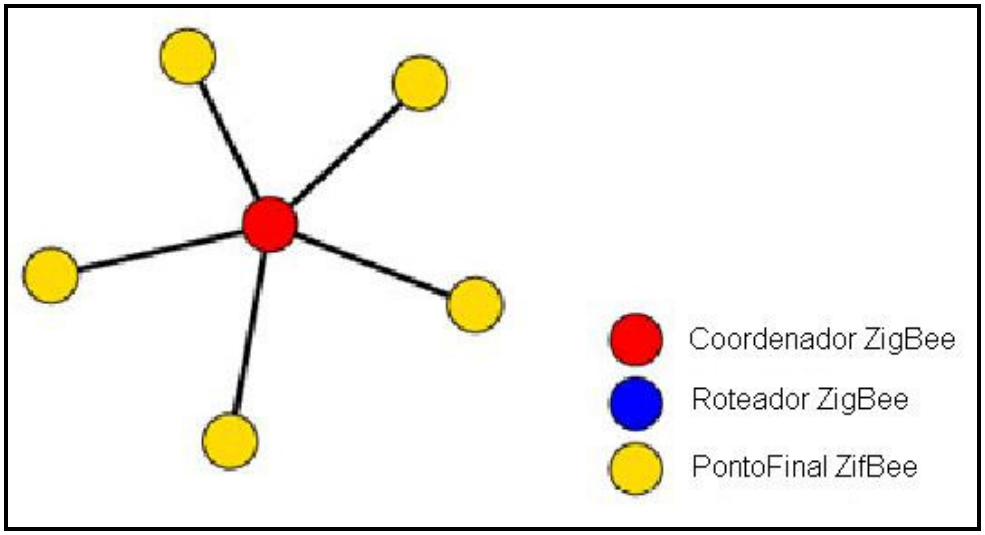
Os dispositivos lógicos e suas funções são apresentados no Quadro 3.

Dispositivo	Tipo de dispositivo físico associado (IEEE 802.15.4)	Função
Coordenador	FFD	Forma a rede, atribui endereços, suporta binding table. Existe apenas um por rede.
Roteador	FFD	Permite que mais nós se juntem à rede, ao aumentar o seu alcance físico. Pode também efetuar funções de controle ou monitoração. A sua existência é opcional.
Ponto final	RFD ou FFD	Efetua ação de controle ou monitoração através do dispositivo que lhe esteja associado (sensor, controlador, atuador...).

Fonte: adaptado e traduzido de André Teixiera da Silva (2007, p. 24)

Quadro 3 - Dispositivos lógicos e suas funções

De acordo com André Teixiera da Silva o protocolo *ZigBee* admite diferentes topologias da rede: estrela (*star*), malha (*mesh*) ou árvore (*cluster tree*). Na rede *ZigBee* tipo Estrela cabe ao coordenador efetuar todo o controlo da rede, assumindo um papel central e de comunicação direta com todos os dispositivos de ponto final, portanto é o coordenador que inicia e mantém os dispositivos na rede. Toda a informação em circulação na rede passa pelo nó coordenador. A topologia de rede *ZigBee* do tipo estrela é apresentada no Quadro 4.

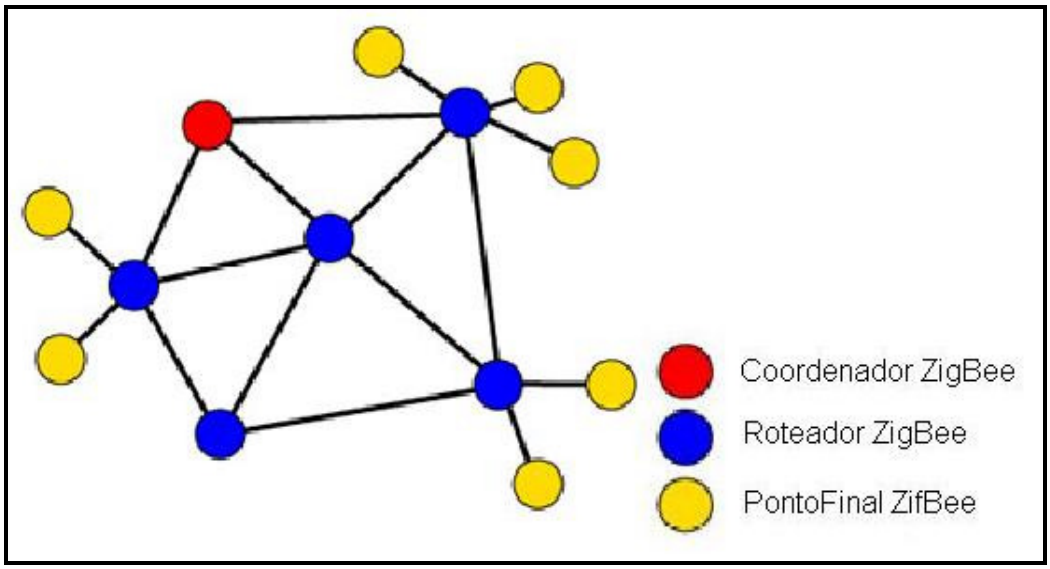


Fonte: traduzido de Silva, André (2007, p. 22).

Quadro 4 - Rede *ZigBee* tipo estrela

Na rede *ZigBee* tipo malha os dispositivos do tipo FFD (coordenadores/roteadores) são livres para comunicar com outro dispositivo FFD. Isto permite, quando necessária, a expansão física da rede (maior alcance). O coordenador registra toda a entrada e saída de dispositivos,

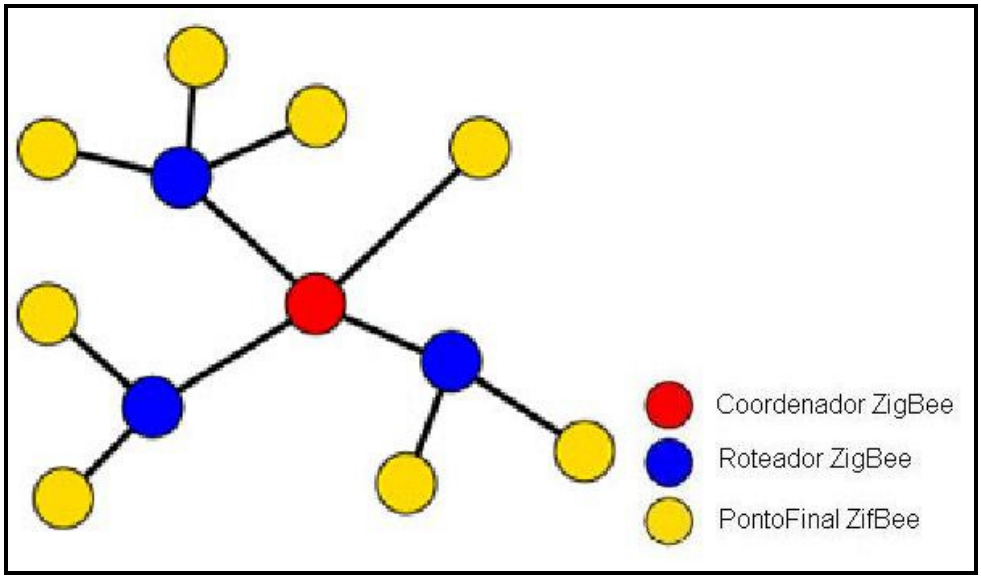
mas não assume como papel principal o fluxo de informação como na configuração anterior. A topologia de rede *ZigBee* do tipo malha é apresentada no Quadro 5.



Fonte: traduzido de Silva, André (2007, p. 23).

Quadro 5 - Rede *ZigBee* tipo malha

Na rede *ZigBee* tipo árvore apresenta semelhanças à rede em malha, também são usados dispositivos roteadores. No entanto, nesta topologia efetua-se a distribuição de dados e mensagens de controle numa estrutura hierárquica, onde o coordenador assume o papel de nó "raiz" da rede. A topologia de rede *ZigBee* do tipo árvore é apresentada no Quadro 6.



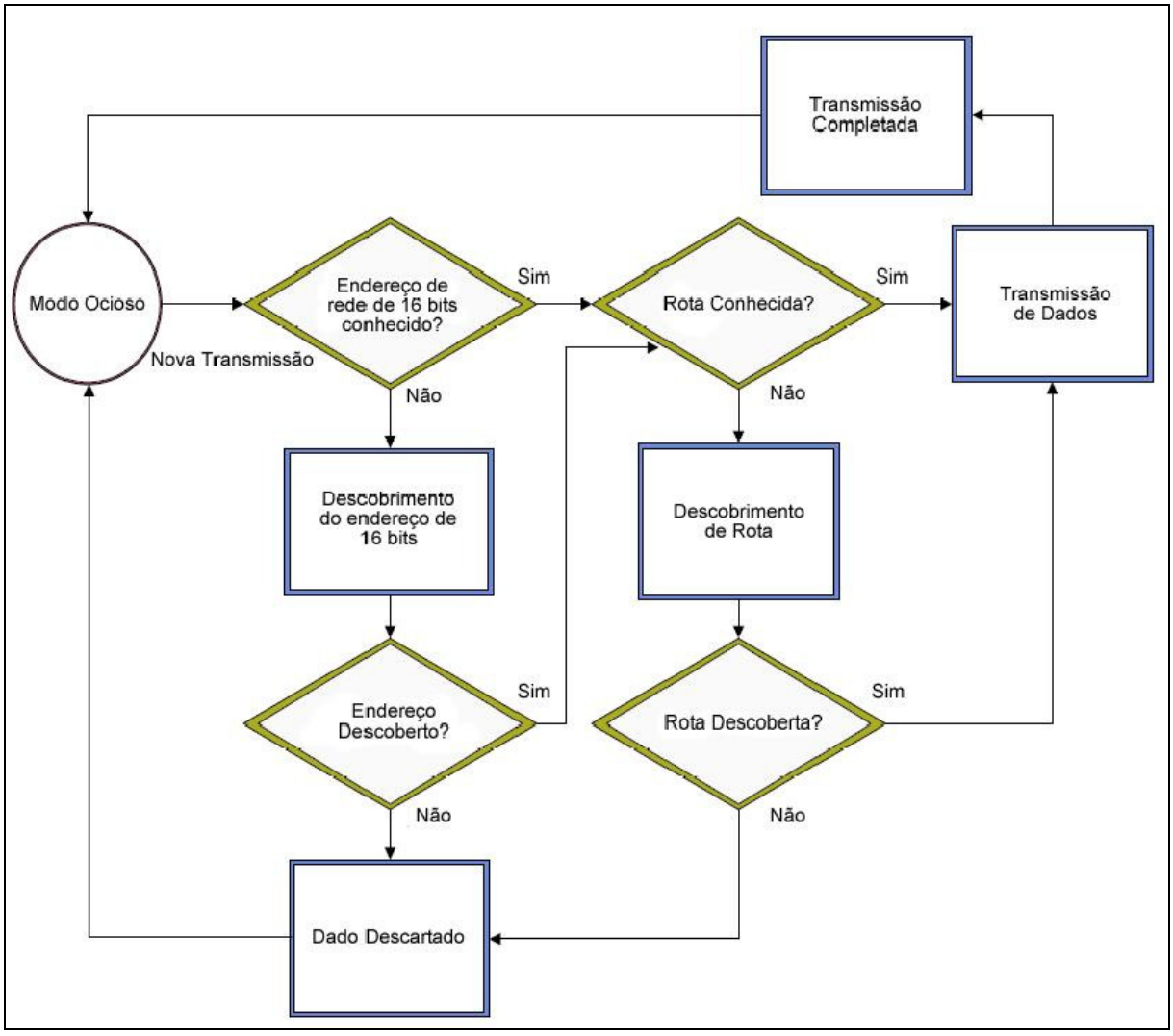
Fonte: traduzido de Silva, André (2007, p. 23).

Quadro 6 - Rede *ZigBee* tipo árvore

Quando um dado é enviado a um módulo ZigBee, este módulo sai do modo ocioso

(Idle Mode) para iniciar a transmissão de dados. O endereço de destino determina qual nó receberá a informação. Antes que inicie a transmissão, o módulo certifica-se que o endereço de 16 bits e a rota para nó de destino tenha se estabelecido. Caso o endereço de destino ou a rota não sejam conhecidos, são acionados procedimentos de descoberta de endereço ou de rota (VIKACONTROLS, 2008?, p. 6).

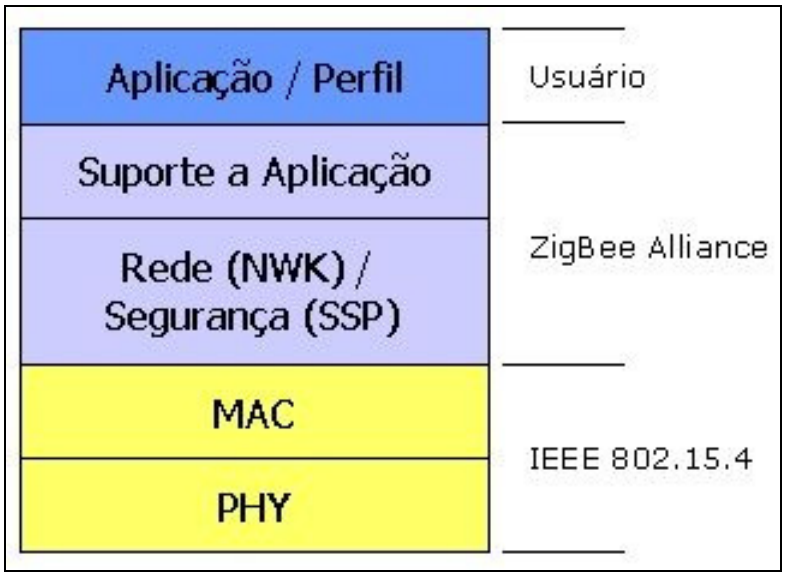
O modo de transmissão de uma rede ZigBee é apresentado no Quadro 7.



Quadro 7 - Modo de transmissão de uma rede ZigBee

"O protocolo Zigbee é estruturado em cinco camadas: PHY (física), MAC (enlace), NWK (rede), Suporte a Aplicação e Aplicação Perfil" (VIVASEMFIO, 2007, p. 1).

As camadas do protocolo ZigBee são apresentadas no Quadro 8.



Quadro 8 - Camadas do protocolo ZigBee

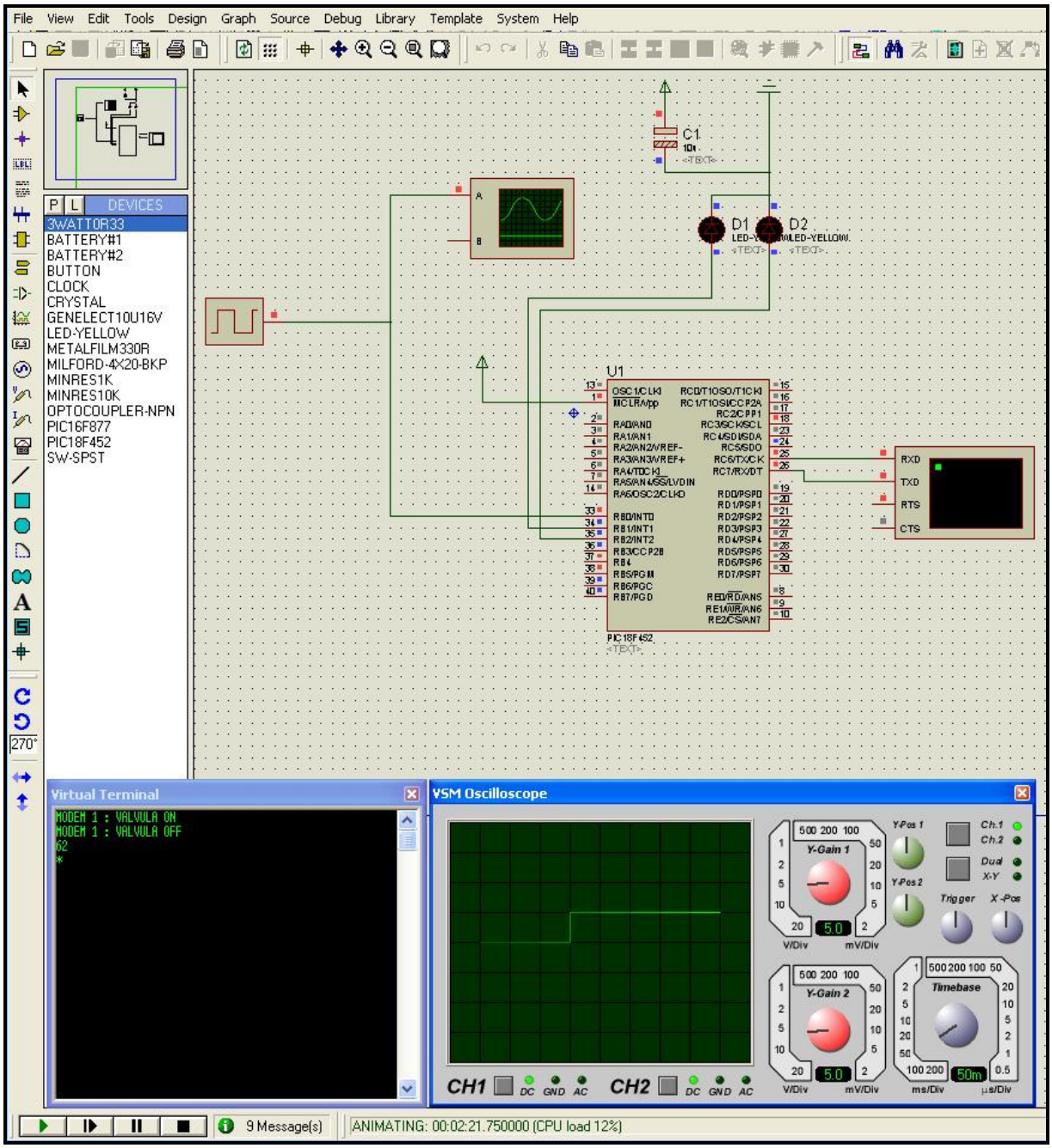
2.8 CI MAX 232 E CI 555

"O MAX 232 é um circuito integrado conversor de nível, que converte sinais TTL em RS232 e vise-versa." (ELETRÔNICA.ORG, 2008, p. 1).

O 555 é um circuito integrado dedicado, projetado para aplicações de temporizador e oscilador. Geralmente utilizado no modo monoestável (temporizador) ou astável (oscilador) (GTA, 2003, p. 1).

2.9 PROTEUS

O Proteus é uma ferramenta utilizada para desenvolver e simular diagramas e circuitos eletrônicos.Uma imagem do Proteus em execução é apresentada no Quadro 9.



Quadro 9 - Ferramenta Proteus 7 simulando circuito

2.10 TRABALHOS CORRELATOS

É possível encontrar no mercado o sistema Hydronet da empresa alemã Hydrometer. O Hydronet é um sistema informatizado que captura dados de hidrômetros digitais

interconectados através de uma rede com protocolo *Meter Bus* (M-Bus[2]). O sistema armazena os dados em um banco de dados, gera relatórios, gráficos, tarifas e outras funcionalidades. Está sendo utilizado pela Universidade de São Paulo (USP), integrado ao Programa de Uso Racional de Água (PURA), coordenado pela USP.

De acordo com Tamaki at al. (2005), o sistema Hydronet obtém as informações dos hidrômetros (leitura de volume, entre outros) a cada cinco minutos, o que permite a geração de perfis de vazão detalhados, através dos quais podem ser identificadas anomalias no consumo. Faz-se o acompanhamento diário dos perfis dos hidrômetros com o intuito de identificar possíveis anomalias no consumo, sendo a mais comum a ocorrência de vazamentos, que pode ser identificada pelo aumento súbito no patamar de vazão ou pela ocorrência de vazão durante a madrugada (quando teoricamente não haveria consumo). A implementação da leitura remota de hidrômetros, no contexto do PURA da Universidade de São Paulo, teve como objetivo principal o de constituir um instrumento de gestão da demanda de água para promover a economia de água de abastecimento no campus Cidade Universitária Armando de Salles Oliveira (CUASO), o qual resultou em uma redução de 36% no consumo de água.

A companhia de SAneamento Básico do Estado de São Paulo (SABESP) possui um sistema chamado Sistema de Controle da Operação do Abastecimento (SCOA), "[...] o qual monitora e controla à distância o nível de 258 reservatórios, 127 bombas telecomandadas, 389 bombas automáticas, 207 pontos de pressão e 214 de vazão. Os dados são transmitidos por meio de 180 linhas telefônicas." (SABESP, 2007, p. 1). Estes dados são registrados e armazenados em banco de dados e possibilitam análise operacional, ensaios matemáticos para melhorar o sistema e o gerenciamento das condições de abastecimento. Desta forma aumentaram a velocidade na detecção de vazamentos e a redução dos custos com operação e manutenção (SABESP, 2007, p. 1).

[2] Conforme M_BUS (2000, tradução nossa), "[...] é um novo padrão europeu para leitura remota, [...] para qualquer tipo de medidor de consumo, assim como vários sensores e atuadores.".

3 DESENVOLVIMENTO

Neste capítulo são abordadas os requisitos, a especificação e a implementação do hardware e do software do protótipo através de diagramas específicos de cada tópico como também trechos do código fonte utilizado em seu desenvolvimento.

3.1 REQUISITOS PRINCIPAIS DO PROBLEMA A SER TRABALHADO

Os Requisitos Funcionais (RF) e Requisitos Não-Funcionais (RNF) são:

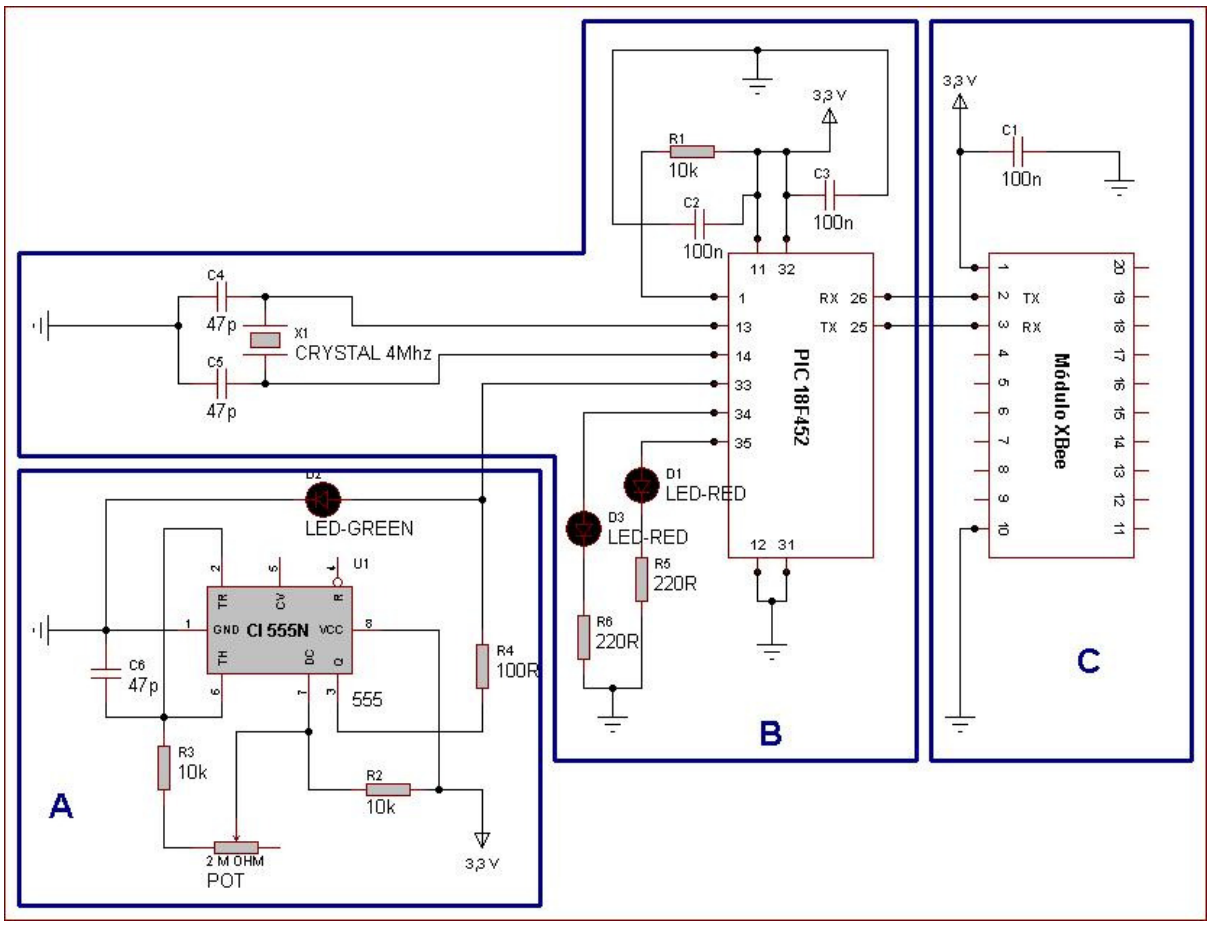
a) o sistema deverá coletar dados de consumo de água na rede (RF);
b) o sistema deverá gerar relatórios periódicos de consumo (RF);
c) o sistema deverá detectar possíveis vazamentos na rede (RF);
d) o sistema deverá gerar mensagens de alerta quando detectar possíveis vazamentos (RF);
e) o sistema deverá permitir o cadastro de consumidores (RF);
f) o sistema deverá permitir enviar sinais de comando às válvulas (RF);
g) a interface gráfica deverá ser multi-plataforma (RNF);
h) a interface gráfica deverá ser desenvolvida na linguagem Java (RNF);
i) o sistema deverá armazenar os dados em um banco de dados MySQL (RNF);
j) o sistema deverá utilizar comunicação *wireless* entre o hardware e o software (RNF);
k) o sistema deverá utilizar o microcontroladores PIC 18F452 para efetuar a comunicação entre transceptor e hidrômetro (RNF).

3.2 ESPECIFICAÇÃO

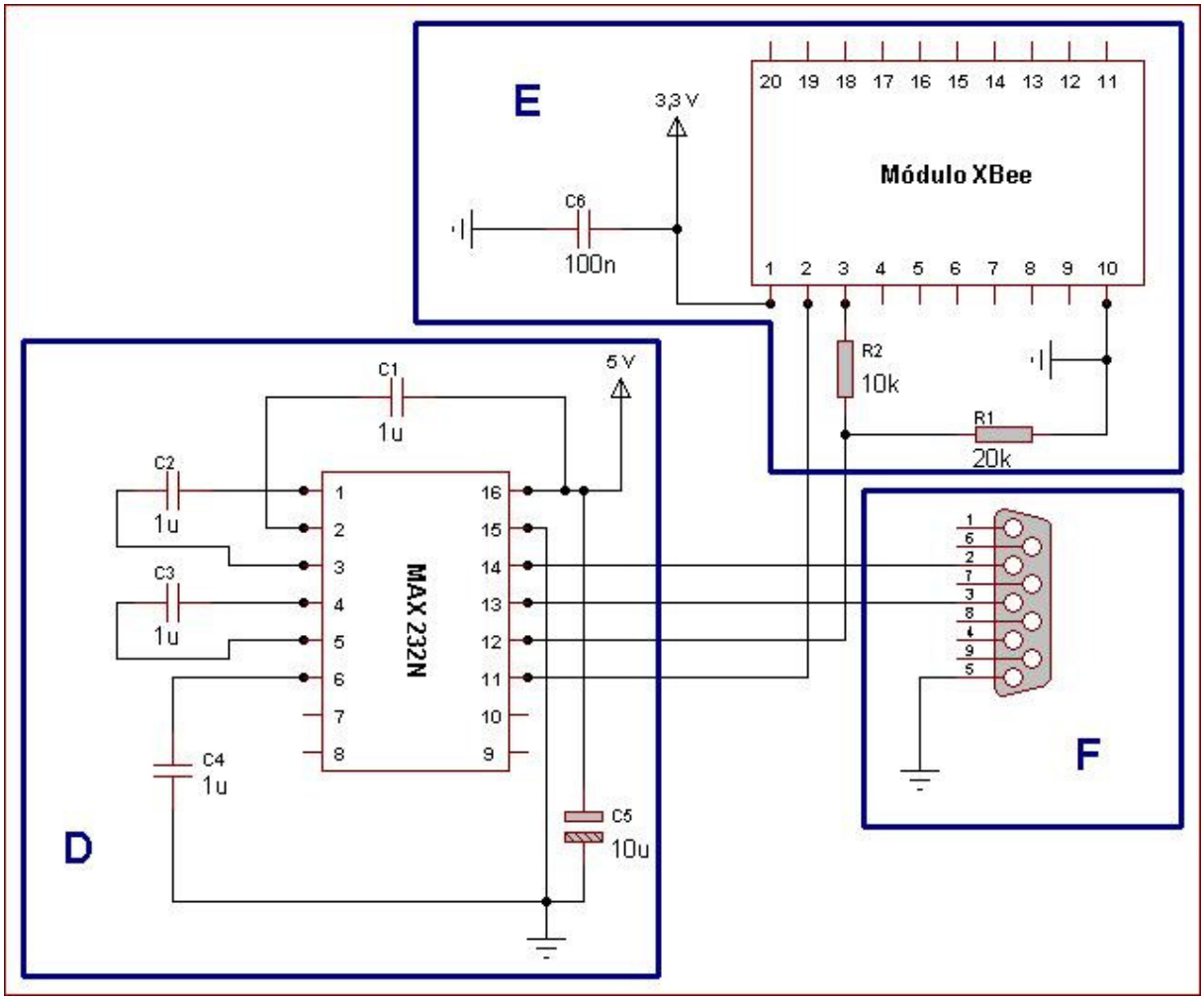
A especificação está dividida em quatro partes, sendo primeiro a especificação do circuito eletrônico do protótipo, em segundo a especificação do software embarcado, em terceiro a especificação do software para o PC e por último a especificação da base de dados.

3.2.1 Especificação do circuito eletrônico

Como o circuito eletrônico é composto por dois sub-circuitos, a especificação destes é apresentada no Quadro 10 e no Quadro 11 e foram desenvolvidos com a ferramenta Proteus 7, sendo o Quadro 10 referente ao circuito dos terminais de captura de dados e o Quadro 11 referente ao circuito de comunicação com o PC através do CI MAX-232N.



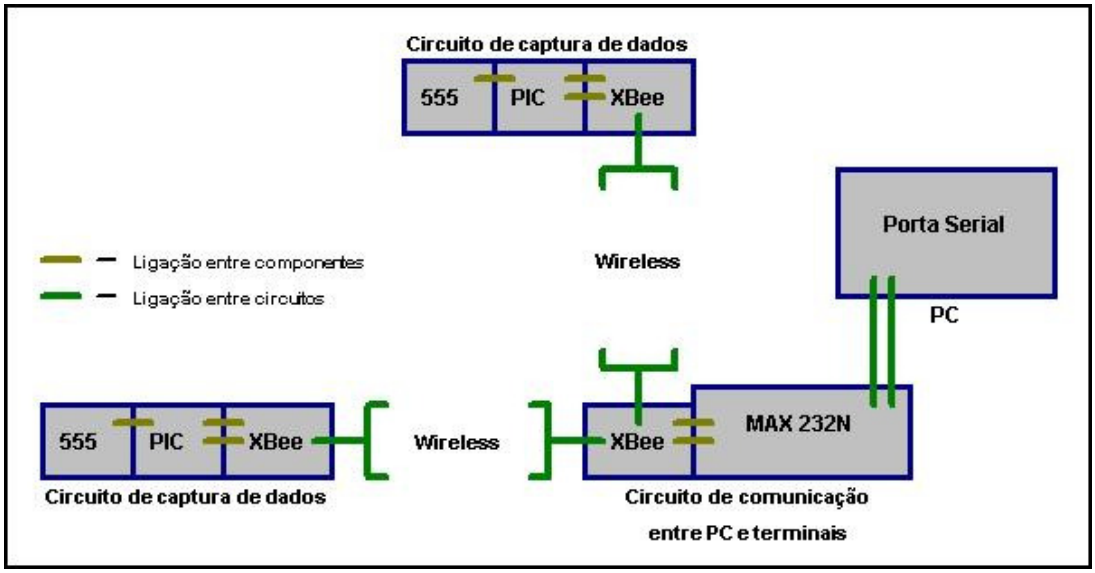
Quadro 10 - Circuito dos terminais de captura de dados



Quadro 11 - Circuito de comunicação com o PC através do CI MAX-232N

No Quadro 10 o detalhe “A” representa o circuito do CI 555 operando em modo astável(oscilador) simulando um hidrômetro digital com saída pulsada, o detalhe “B” representa o circuito do PIC 18F452 e o detalhe “C” representa o circuito do transceptor Xbee. A função do CI 555 conforme está apresentado no detalhe “A” é gerar uma onda quadrada no pino 3, essa onda terá sua frequência alterada conforme a posição do pino central do potenciômetro “POT” (detalhe “A”), desta forma será possível simular a saída pulsante de um hidrômetro digital e o volume de água que flui pelo mesmo. No Quadro 11 o detalhe “D” representa o circuito do CI MAX 232N que converte a comunicação RS232 para o nível TTL e vice-versa, o detalhe “E” representa o circuito do Módulo Xbee e o detalhe “F” representa a ligação do conector DB9 fêmea que será conectado à porta serial do PC.

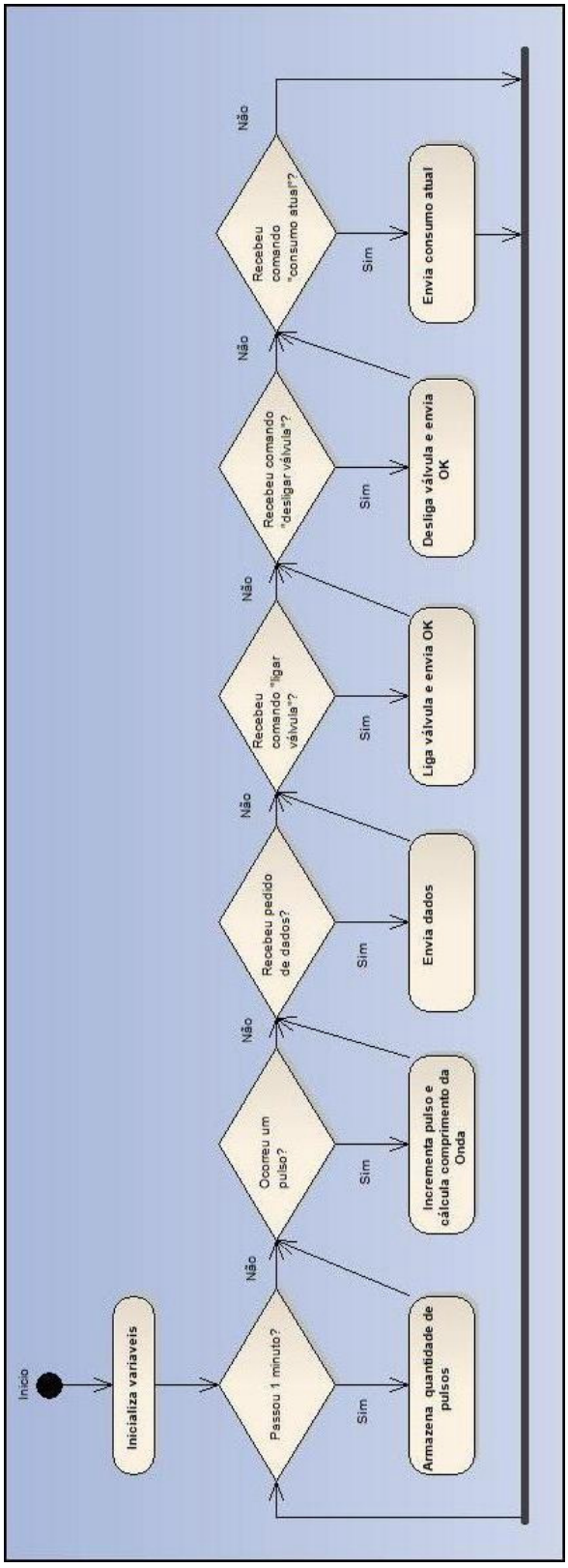
O Quadro 12 apresenta a distribuição dos circuitos formando uma rede do tipo “estrela” onde o circuito de comunicação entre PC e terminais, conectado ao PC tem o papel de coordenador e os circuitos de captura de dados tem o papel de terminais finais.



Quadro 12- Distribuição dos circuitos

3.2.2 Especificação do software embarcado

A especificação do software embarcado é apresentada no Quadro 13, em seu desenvolvimento foi utilizado a técnica UML através do diagrama de atividades. Produzidos com a ferramenta Enterprise Architect 7.1.

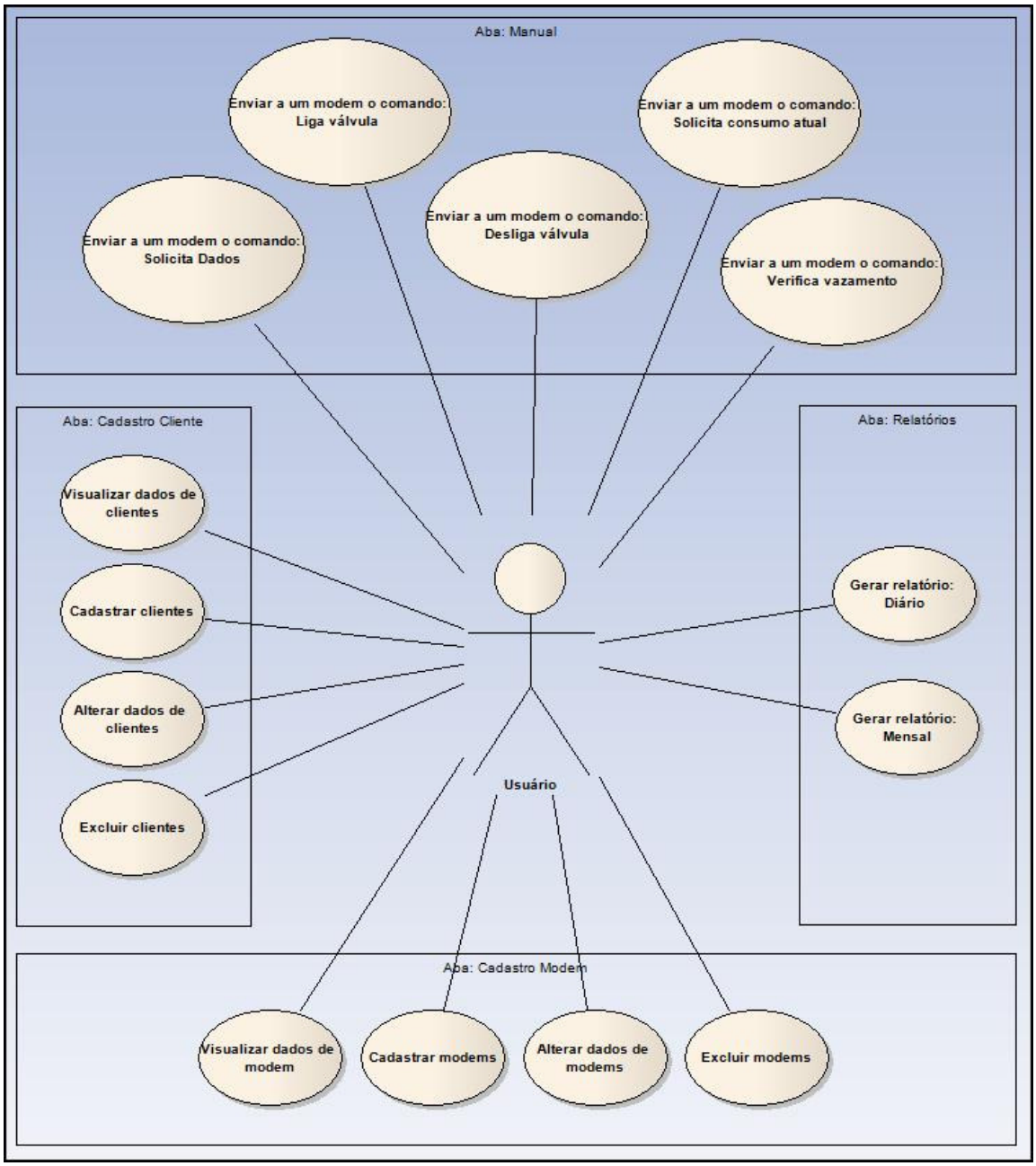


Quadro 13 - Diagrama de atividades para o software embarcado

Primeiramente o software inicializa as variáveis, após isto entra em um loop infinito onde o primeiro losango de decisão é executado no software embarcado pelo método main() e os demais são executados através das interrupções.

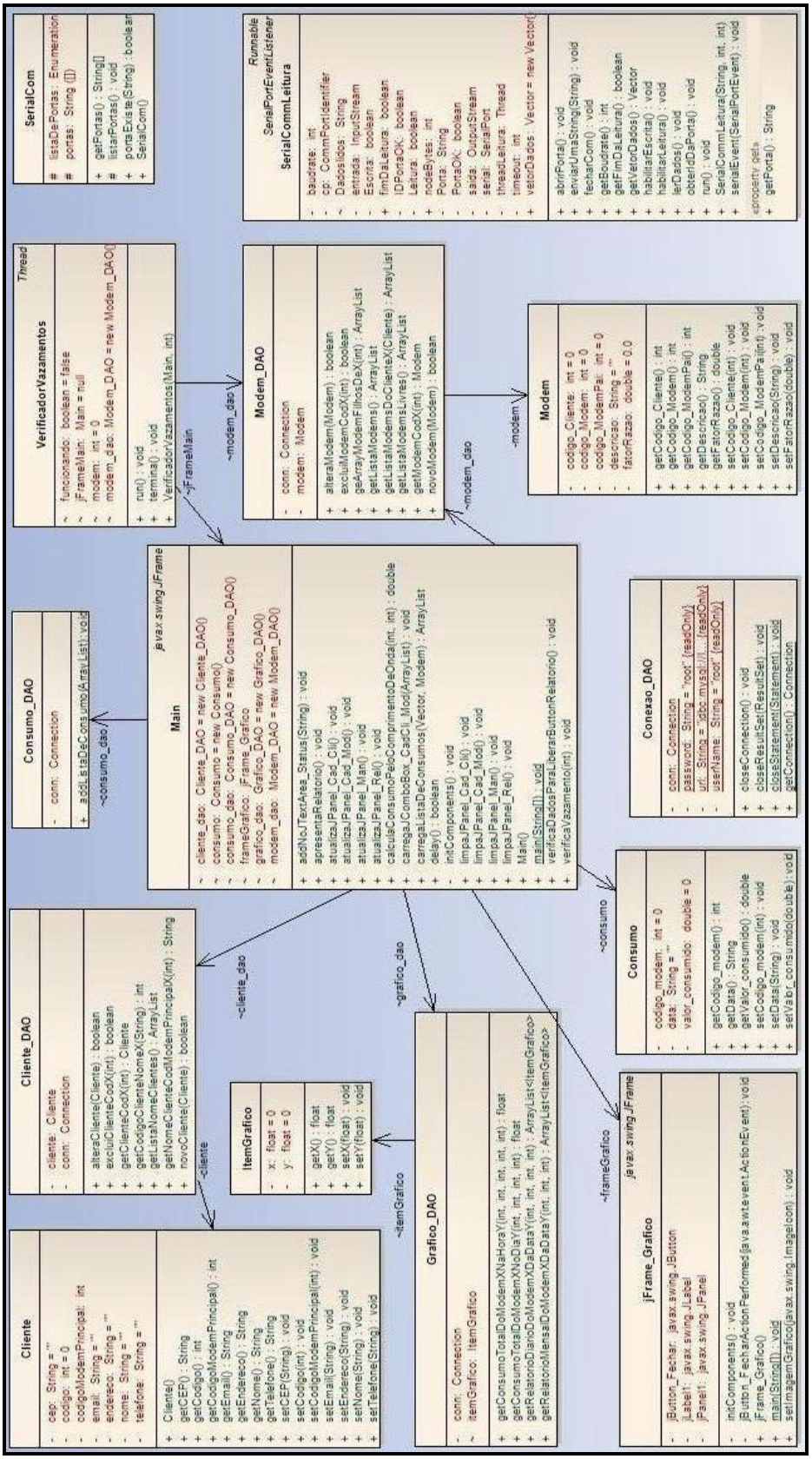
3.2.3 Especificação do software do PC

A especificação do software do PC é apresentada pelo diagrama de casos de uso no Quadro 14 e pelo diagrama de classes no Quadro 15, foram desenvolvidos com a ferramenta Enterprise Architect 7.1. No diagrama da classes os atributos e os métodos da classe Main, referente a interface gráfica da aplicação, como jButtons, jText*Label*s e outros foram removidos do diagrama para viabilizar (tamanho) a apresentação do mesmo.



Quadro 14 - Diagrama de casos de uso para o software do PC

O diagrama de casos de uso demonstra quais as ações que o usuário pode solicitar à aplicação em cada uma das quatro abas disponíveis na interface gráfica.

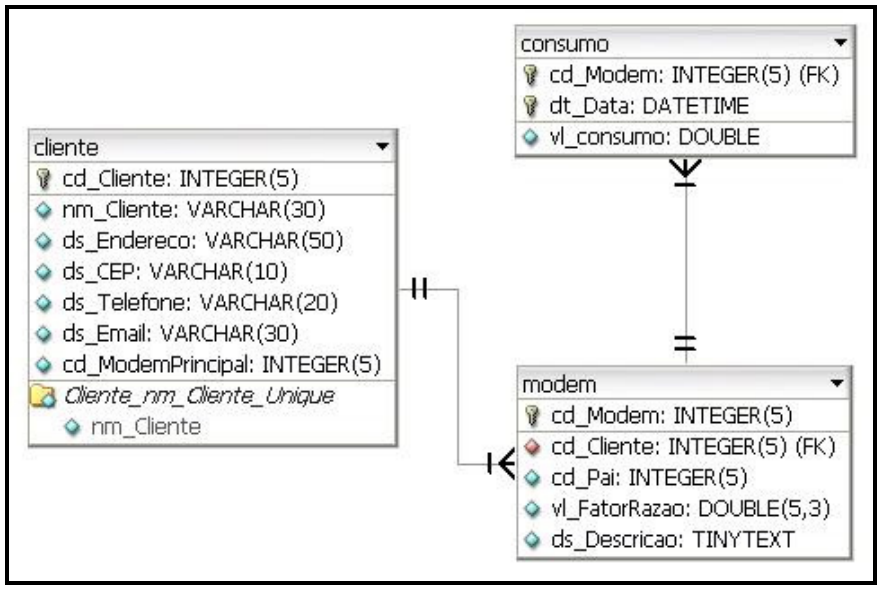


Quadro 15 - Diagrama de classes para o software do PC

O diagrama de classes demonstra as classes da aplicação, os métodos e atributos de cada classe e através das setas representando a instância de outras classes. As classes Cliente_DAO, Consumo_DAO e Modem_DAO são conhecidas como classes *Data Acess Object* (DAO), são responsáveis por efetuar os comandos SQL para inserir, alterar, remover e coletar dados da base de dados respectivamente às classes Cliente, Consumo e Modem que por sua vez são classes com os mesmo atributos das tabelas cliente, consumo e modem respectivamente e com os métodos *get()* e *set()* de cada atributo. A classe Conexao_DAO é responsável por controlar a conexão com o banco de dados, todas as classes DAO antes de executar qualquer comando no banco de dados, solicitam a conexão a esta classe, para evitar que cada classe DAO crie uma nova conexão com o banco de dados. As classes SerialCom e SerialCommLeitura são responsáveis por efetuar a comunicação com a porta serial. A classe Grafico_DAO é responsável por efetuar os comando SQL referente aos relatórios periódicos, armazena estes dados em objetos da classe ItemGrafico que por sua vez possuem os atributos do tipo inteiro “x” e “y” que irão formar cada ponto dos gráficos de relatório. A classe JframeGrafico é responsável pela criação e configuração da janela dos gráficos. A classe Main é responsável pela criação e configuração da interface gráfica com o usuário.

3.2.4 Especificação da base de dados

A especificação da base de dados é apresentada no diagrama entidade relacionamento no Quadro 16, foi desenvolvido com a ferramenta DBDesigner 4.0.



Quadro 16 - Diagrama entidade relacionamento da base de dados

3.3 IMPLEMENTAÇÃO

A seguir são apresentadas as técnicas, as ferramentas utilizadas e a operacionalidade da implementação.

3.3.1 Técnicas e ferramentas utilizadas

A seguir serão apresentadas informações sobre o módulo Xbee e o microcontrolador PIC 18F452.

3.3.1.1 Módulo XBee XB24-AWI-001

O módulo Xbee é utilizado para efetuar a comunicação entre os circuitos de hardware e a aplicação no PC. O circuito para a ligação do módulo Xbee com o PIC é apresentado no Quadro 10.O circuito para a ligação do módulo Xbee com o PC através do MAX-232N é apresentado no Quadro 11.

Para utilizar os módulos Xbee, foi necessário configurá-los antes, esta configuração foi feita através do aplicativo X-CTU, foram configuradas a variável “MY” responsável pelo endereço do módulo e a variável *Destination adress Low* (DL) responsável pelo endereço de destino dos dados. Como o protótipo desenvolvido constitui uma rede do tipo “estrela”, então o módulo instalado no circuito de comunicação entre o PC e os terminais finais foi definido como coordenador e teve sua variável “MY” configurada com o valor “0” e a variável “DL” com o valor “FFFF” (FFFF é o endereço de *broadcast*), os demais módulos tiveram a variável “MY" configurada com os valores 1, 2, 3 e assim por diante, e a variável DL configurada com o valor “0” ou seja, todos os terminais finais enviando para o endereço “0” que neste caso é o endereço do módulo coordenador. As características do módulo Xbee são apresentadas no Quadro 17.

Performance
- Rendimento da Potência de saída: 1 mW (0 dBm);
- Alcance em ambientes internos/zonas urbanas: 30m;
- Alcance de RF em linha visível para ambientes externos: 100m;
- Sensibilidade do receptor: -92 dBm;
- Freqüência de operação: ISM 2.4 GHz;
- Taxa de dados de RF: 250.000 bps;
- Taxa de dados da Interface (Data Rate): 115.200 bps;

Alimentação
- Tensão de alimentação: 2.8 à 3.4v;
- Corrente de transmissão (típico): 45 mA @ 3.3 V;
- Corrente de Recepção (típico): 50 mA @ 3.3 V;
- Corrente de Power-down Sleep: <10 µA;

Propriedades físicas
- Dimensões: (2.438cm x **2.761cm**);
- Peso: 0.10 oz (3g);
- Temperatura de operação: -40 to 85º C (industrial);
- Opções de antena: Conector U.FL RF, Chip ou Chicote (whip);

Rede
- Tipo de espalhamento espectral: DSSS (Direct Sequence Spread Spectrum);
- Manipulação de erro: Retransmite novamente (Retries) & reconhecimento (acknowledgements);
- Topologia de Rede: Peer-to-peer(Par-a-par), ponto-a-ponto, ponto-a-multiponto e malha;
- Endereçamento: 65.000 endereços de rede disponíveis para cada canal;
- Opções de filtros: PAN ID, canais e endereços;
- Criptografia: 128-bit AES;
- Número de canais selecionáveis via software: 16 canais de seqüência direta;

Geral
- Faixa de freqüência: 2.4000 - 2.4835 GHz;

Fonte: MESSIAS (2007, p. 1).

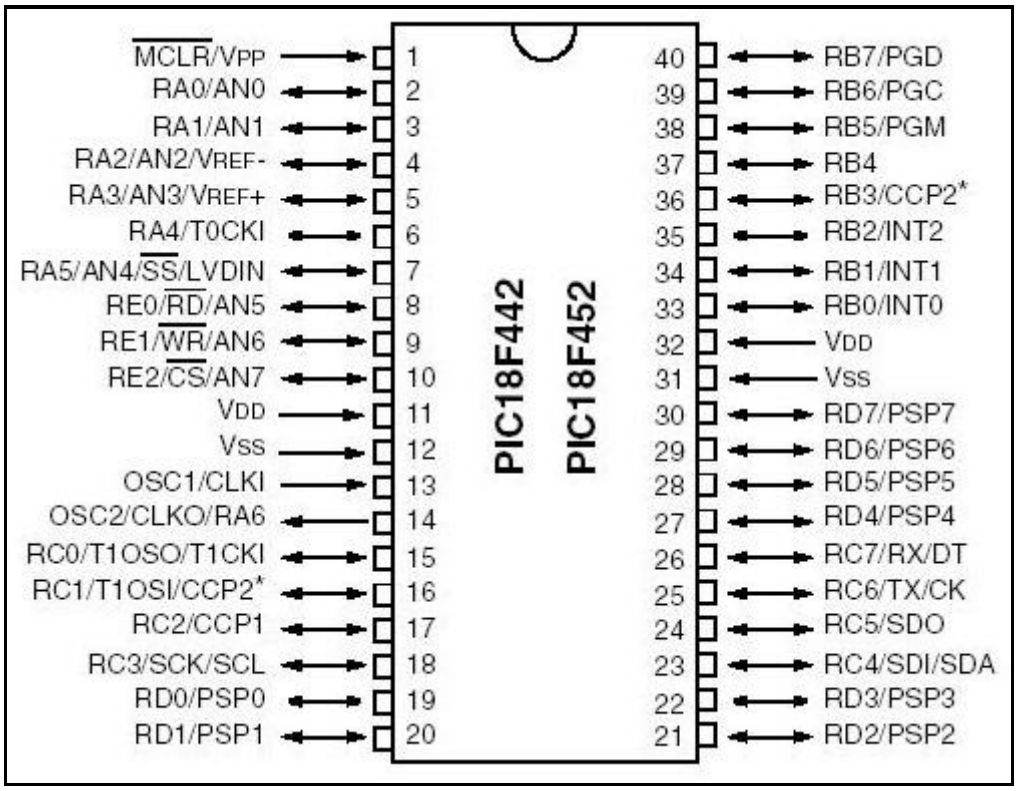
Quadro 17 - Características do módulo Xbee XB24-AWI-001

3.3.1.2 Microcontrolador PIC 18F452

As funcionalidades do microcontrolador PIC 18F452 utilizadas para o desenvolvimento deste trabalho são:

a) Memória RAM de 1536 bytes, utilizada para armazenar os dados de consumo;
b) Interrupção do Timer0;
c) Interrupção do Timer1;
d) Interrupção Externa pela porta RB0;
e) Interrupção pela entrada de dados na porta serial;
f) Comunicação USART via RS-232.

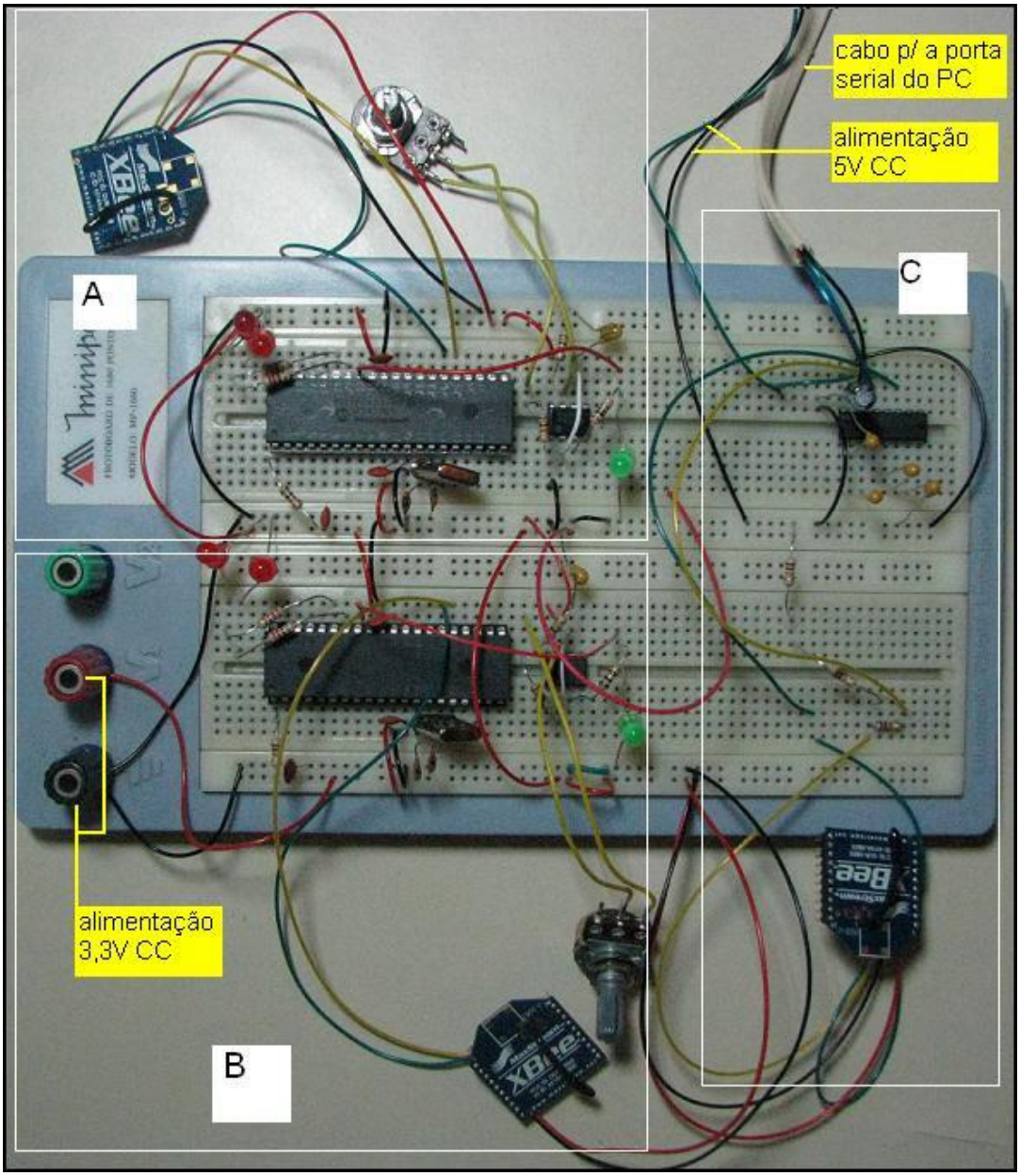
A pinagem do microcontrolador PIC 18F452 é apresentado no Quadro 18.



Quadro 18 – Pinagem do PIC 18F452

3.3.2 Implementação do circuito eletrônico do protótipo

O circuito eletrônico do protótipo foi montado em um protoboard, sendo composto por dois sub-circuitos montados conforme o quadro 3 representando dois hidrômetros e um sub-circuito conforme o quadro 4 para efetuar a comunicação com o PC. O circuito eletrônico do protótipo é apresentado no Quadro 19.



Quadro 19 – Circuito do protótipo montado em um *protoboard*

Os detalhes “A” e “B” representam dois sub-circuitos de captura de dados compostos por um microcontrolador PIC, um transceptor Xbee e um CI 555 cada um. O detalhe “C” representa o sub-circuito de comunicação entre o PC, o CI MAX232 e um transceptor XBee.

Os detalhes em amarelo mostram os pontos de alimentação e o cabo de comunicação com a porta serial do PC, o CI MAX232 é alimentado com 5V CC enquanto o resto do circuito do protótipo é alimentado com 3,3V CC.

3.3.3 Implementação do software embarcado

O software embarcado foi desenvolvido na linguagem C utilizando a ferramenta PCH 3.34 da empresa CCS. As funcionalidades do software embarcado são, contar os pulsos de consumo, calcular o comprimento do pulso de consumo para informar ao software do PC quando solicitado o valor de consumo em tempo real, armazenar os valores de consumo, receber comandos para ativar ou desativar uma saída (simulando uma válvula), enviar ao software do PC os dados de consumo armazenados quando solicitado pelo mesmo.

Para contar os pulsos de consumo e calcular o comprimento do pulso foi utilizada a interrupção externa da porta RB0, quando chega um pulso elétrico à porta RB0, esta gera uma interrupção desviando o código para a rotina de tratamento de interrupção externa chamada "int_ext" nesta rotina foi inserido o método "`interrupçãoExterna(void)`", esse método é apresentado no Quadro 20.

```
#int_ext
void interrupcaoExterna(void)
{
      if (aguardandoFimDaOnda == 1){      // significa: está calculando comprimento da onda ???
         comprimentoDaOnda = countTimer0; // atribui à variável comprimentoDaOnda a qtd. de estouros do Timer0
         ext_int_edge(1);                 // define interupção de RB0 pela subida de borda
         aguardandoFimDaOnda = 0;         // significa: terminou de calcular o comprimento da onda
         pulsos++;                        // incrementa a variável pulsos
         disable_interrupts(int_timer0);  // desativa interrupção do Timer0
      }else{
         set_timer0(valorInicialTimer0);  // seta Timer0 com seu valor inicial (variável valorInicialTimer0)
         enable_interrupts(int_timer0);   // ativa interrupção do Timer0
         countTimer0 = 0;                 // zera contador de estouros do Timer0
         ext_int_edge(0);                 // define interupção de RB0 pela descida de borda
         aguardandoFimDaOnda = 1;         // significa: está calculando comprimento da onda
      }
}
```

Quadro 20 - Método `interrupçãoExterna(void)`

O método `interrupçãoExterna(void)` ocorre duas vezes por pulso, uma quando o valor lógico da porta RB0 vai de 0 para 1, e outra quando vai de 1 para 0, no intervalo de tempo entre estas duas execuções é calculado o comprimento do pulso pelo registrador Timer0 e na última é incrementada a variável pulsos. Segue abaixo um explicação mais detalhada da execução deste método.

A princípio a interrupção ocorre pela subida de borda, ou seja, quando o valor lógico na entrada RB0 vai de 0 para 1. Na primeira vez que este método é executado a variável "aguardandoFimDaOnda" tem seu valor lógico = 0, portanto o código executado pela primeira vez será a cláusula "else", neste ponto o registrador Timer0 tem sou valor setado com o valor da variável valorInicialTimer0 (valor = 248), após isto a interrupção do registrador Timer0 é ativada e o valor da variável countTimer0 é zerada. O próximo passo é o comando "`ext_int_edge(0)`" que define que a interrupção externa seja ativada pela descida

de borda, e após isto a variável aguardandoFimDaOnda tem seu valor lógico = 1 para que, quando ocorrer uma interrupção, agora pela descida de borda, a execução entre na cláusula "if" e não mais na "else".

Quando ocorrer outra interrupção externa, agora pela descida de borda, o método irá executar a cláusula "if", o primeiro passo é atribuir o valor da variável comprimentoDaOnda com o valor da variável countTimer0, ou seja a quantidade de vezes que o registrador Timer0 atingiu seu valor limite no tempo entre a interrupção da subida de borda (cláusula "else") e agora da decida de borda (cláusula "if") resultando no valor do comprimento do pulso, esse valor multiplicado por 0,002048 será igual ao valor do comprimento do pulso em segundos[3]. O próximo passo executar o método "`ext_int_edge(1)`" que define que a interrupção externa seja ativada pela subida de borda e após isto a variável aguardandoFimDaOnda tem seu valor lógico = 0, para que na próxima vez que ocorrer uma interrupção a execução entre na cláusula "else", por final a variável pulsos é incrementada em 1 e a interrupção do registrador Timer0 desativada.

O cálculo do consumo é efetuado através da quantidade de pulsos ocorridos em um determinado tempo, no caso do protótipo desenvolvido é a quantidade de pulsos em 1 (um) minuto. O cálculo do comprimento da onda é utilizado para verificar vazamentos em tempo real e para evitar a necessidade de aguardar um tempo de até 1 (um) minuto em cada medidor de consumo filho, para verificar o valor de consumo dos mesmos e a partir destas informações verificar se há vazamentos entre um medidor pai e seus medidores filhos.

Para armazenar os valores de consumo foi utilizada um vetor de inteiros de 16 bits chamada "array" e uma variável inteira de 16 bits como ponteiro para os campos deste vetor chamada endereço. O método responsável para armazenar os dados de consumo é o método "`armazenaPulsos(void)`", o mesmo é apresentado no Quadro 21.

```
void armazenaPulsos(void){
    array[endereco] = pulsos; // armazena quantidade de pulsos
    endereco++;               // incrementa endereco
    pulsos = 0;               // zera contador de pulsos
    countTimer1 = 0;          // zera a variável countTimer0 (contador de 1 minuto)
    output_high(piscapisca);  // ativa saída RB1
    delay_ms(500);            // aguarda 500 milisegundos
    output_low(piscapisca);   // desativa saída RB1
}
```

Quadro 21 - Método `armazenaPulsos(void)`

Este método armazena o valor atual da variável "pulsos" na próxima posição livre do

[3] o valor 0,002048 foi calculado conforme as configurações de registrador Timer0, o *clock* do processador e da variável valorInicialTimer0. Este valor foi calculado utilizando a ferramenta PIC *Timer Calculator and Source Code Generator* da empresa *Dring Engineering Services*.

vetor “array”, após isto zera a variável “countTimer1”, utilizada em conjunto com o registrador Timer1 para chamar este método a cada 1 minuto, e por final ativa a porta RB1 (led que fica piscando) por 500 milisegundos para informar que um dado de consumo foi armazenado.

Para receber comandos para ativar ou desativar uma saída (simulando uma válvula) e enviar ao software do PC os dados de consumo armazenados e o consumo atual quando solicitado pelo mesmo foi implementado o método “`recebeDados(void)`” o qual é apresentado no Quadro 22.

```
#int_rda
void recebeDados(void){
    dado1 = getc();         // recebe primeiro dado (ID do modem) e atribui valor à variável dado1
    dado2 = getc();         // recebe segundo dado (comando) e atribui valor à variável dado2
    if (dado1 == ID){       // ID é identificador do modem, está perguntando se o dado é para este modem
      switch (dado2){       // direciona o comando recebido para a rotina correspondente
      case ('1') : {                            // comando == 1, Enviar dados ao PC
         if(endereco > 0){
            for (i = 0; i < endereco ; i++){    // de 0 até a última posição utilizada
               printf("%li\r",array[i]);        // envia ao PC o valor do dado do vetor na posição "i"
               array[i] = 0;                    // limpa o valor do dado do vetor na posição "i"
            }
            printf("*");                        // envia ao PC o símbolo "*", informando fim de envio
            endereco = 0;                       // zera variável ponteiro para os dados do vetor
         }else{
            printf("@");                        // envia ao PC o símbolo "@", informando vetor VAZIO
         }
         break;
      }
      case ('2') : {                            // comando == 2, Liga válvula
         output_high(valvula);                  // ativa saída RB1
         printf("MODEM %c : VALVULA ON\r",ID);  // envia ao PC confirmação do comando
         break;
      }
      case ('3') : {                            // comando == 3, Desliga válvula
         output_low(valvula);                   // desativa saída RB1
         printf("MODEM %c : VALVULA OFF\r",ID); // envia ao PC confirmação do comando
         break;
      }
      case ('4') : {                            // comando == 4, Enviar consumo atual ao PC
         printf("%li\r",comprimentoDaOnda);     // envia ao PC o valor do comprimento da onda
         break;
      }
      }
    }
}
```

Quadro 22 - Método `recebeDados (void)`

O comando “`printf()`” é utilizado para enviar dados pela porta RC6 via RS-232. Este método é executado quando ocorrer uma interrupção causada pela recepção de dados na porta RC5.

A configuração inicial do software embarcado é apresentada no Quadro 23.

```
#include <18F452.h>                             // biblioteca do PIC 18F452
#use delay(clock=4000000)                       // definição do cristal conectado nos pinos 13 e 14
#use rs232(baud=9600, xmit=PIN_C6, rcv=PIN_C7)  // configuração do protocólo RS-232 sendo{
                                                // taxa de transferência = 9600 bits/s
                                                // xmit -> porta RC6 responsável por enviar os dados (TX)
                                                // rcv -> porta RC7 responsável por receber os dados (RX)
                                                // }
#fuses nowdt, XT   // desativa WatchDogTimer e
                   // define modo de oscilação como XT-> 455 kHz - 4 MHz


#define  piscapisca         PIN_B1   // "renomeando" PIN_B1 (porta RB1) com piscapisca
#define  valvula            PIN_B2   // "renomeando" PIN_B2 (porta RB2) com valvula
```

Quadro 23 - Configuração inicial do software embarcado

O método `main()` é apresentado no Quadro 24. No detalhe A são apresentadas as inicializações das variáveis e as configurações inicias de execução bem como as configurações do Timer0 e do Timer1. O valor da variável "ID" é definida nesta parte do código (em ID = '1') sendo definida antes da compilação, o ideal seria a definição através de uma *DIP switch*[4] de oito posições conectada ao PIC podendo gerar 256 combinações diferentes, porém por motivos de tempo optou-se pela definição por código.

No detalhe B é apresentado o *loop* de execução principal, o qual fica ativando e desativando a porta RB1 (pisca-pisca) e a cada 1 minuto executa o método armazenaPulsos().

```
void main()
{
   ID = '1';                  // define o ID do modem
   output_high(valvula);      // Ativa saída RB1 (válvula)
   endereco = 0;              // zera variável endereco                    A
   pulsos = 0;                // zera variável pulsos
   countTimer1 = 0;           // zera variável countTimer1
   aguardandoFimDaOnda = 0;   // seta variável aguardandoFimDaOnda com o valor lógico 0
   SET_TRIS_B(0b00000001);    // seta TRISB sendo apenas a porta RB0 como entrada e o resto com saída
   setup_adc_ports(NO_ANALOGS);  // define SEM nenhuma porta analógica
   setup_adc(ADC_OFF);           // desativa função Conversor Analógico-Digital
   setup_psp(PSP_DISABLED);
   setup_spi(SPI_SS_DISABLED);

   enable_interrupts(global);    // ativa globalmente as interrupções
   setup_timer_0(RTCC_INTERNAL | RTCC_8_BIT |RTCC_DIV_256); // configura Timer0 sendo {
                                                            // Referência de clock INTERNA |
                                                            // Utilizando registrador de 8 bits (até 256) |
                                                            // pré-scaler = 256       }
   setup_timer_1(T1_INTERNAL | T1_DIV_BY_8);                // configura Timer1 sendo {
                                                            // Referência de clock INTERNA |
                                                            // Utilizando registrador de 8 bits (até 256) }
   enable_interrupts(int_ext);      // habilita interrupção externa (RB0)
   enable_interrupts(int_rda);      // habilita interrupção da porta serial (RC6)
   enable_interrupts(int_timer1);   // habilita interrupção externa (RB0)
   set_timer1(valorInicialTimer1);  // seta o Timer1 com seu valor inicial (variável valorInicialTimer1)

   while (true) {                // ao infinito
      if (countTimer1 >= 120){   // verifica se de acordo com o Timer1 já se passou 1 minuto
         armazenaPulsos();
      }
      output_high(piscapisca);   // ativa porta RB1 (led piscando)
      delay_ms(100);             // aguarda 100 milisegundos
      output_low(piscapisca);    // desativa porta RB1 (led piscando)          B
      delay_ms(100);             // aguarda 100 milisegundos
   }
}
```

Quadro 24 - Método `main()`

3.3.4 Implementação do software do PC

O software do PC foi implementado utilizando-se a ferramenta NetBeans 6.0, junto a ela foram integrados três componentes, um deles é a API RXTX para efetuar a comunicação através da porta serial do PC, outro foi a biblioteca JfreeChart para efetuar a apresentação dos relatórios de consumo através de gráficos de linha e por último o driver MySQL

[4] é um conjunto de minúsculos interruptores montados em um único suporte plástico.

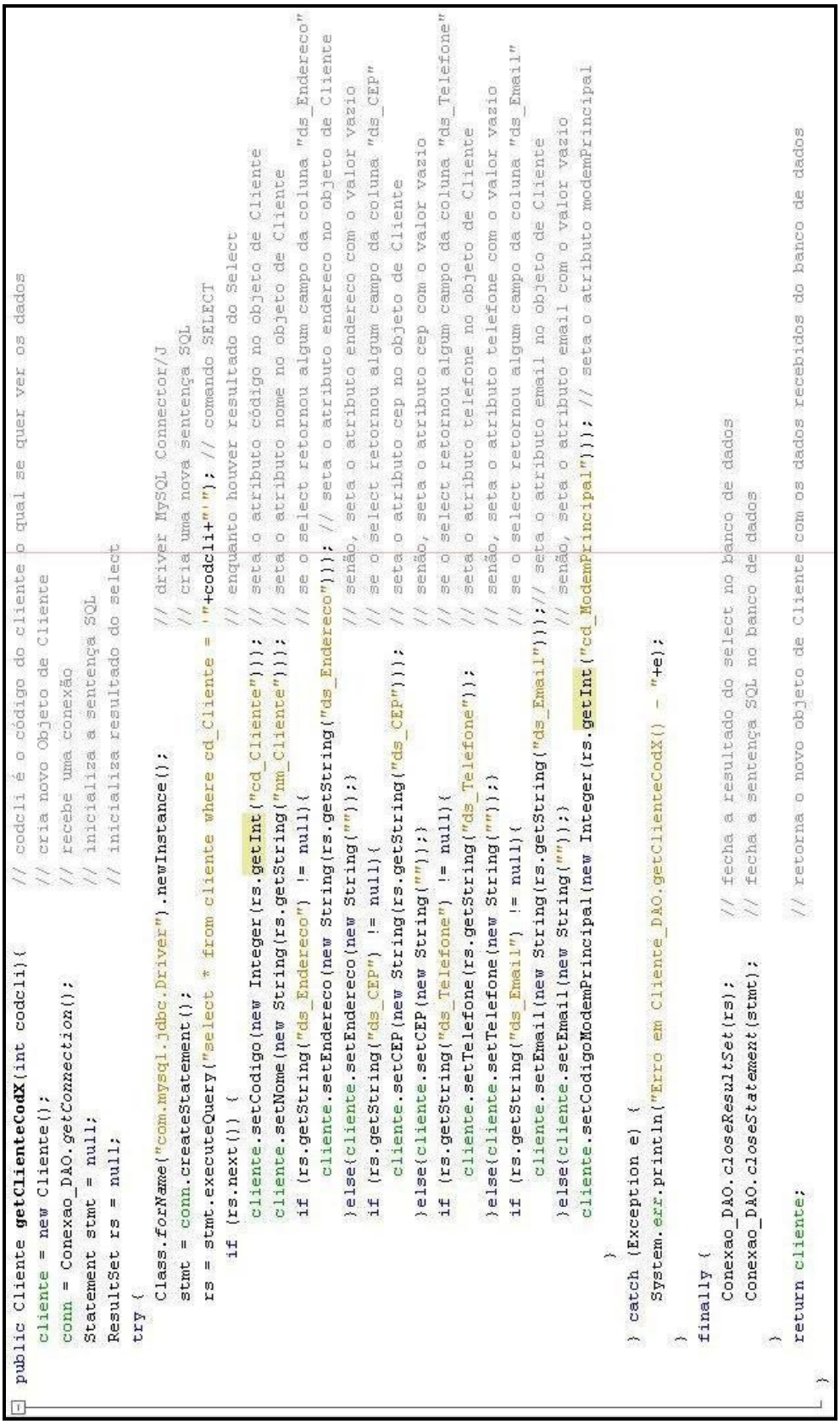
Connector/J 3.0.17 utilizado para efetuar a comunicação com o banco de dados MySQL.

As principais funcionalidades do software do PC são:

a) manipular[5] dados de clientes e modems;
b) apresentar relatórios de consumo;
c) enviar comandos aos modems;
d) armazenar os dados de consumo no banco de dados;
e) verificar vazamentos e alertar sobre vazamentos;

Para manipular os dados de clientes e modems foram criadas as classes DAO para facilitar o desenvolvimento e o entendimento do código, as classes DAO executam os métodos de manipulação de dados com o banco ou seja, executam *script*s SQL de inserção, remoção e seleção, cada classe DAO está associada a uma classe objeto, por exemplo: a classe Cliente_DAO está associada a classe Cliente, onde a classe Cliente possui exatamente os mesmos atributos da tabela "cliente"(no banco de dados), quando a classe Cliente_DAO executa um comando de select no banco para ver os dados de um determinado cliente, a classe Cliente_DAO pega estes dados e armazena em um novo objeto da classe Cliente através dos métodos do tipo "`set()`" da classe Cliente, e retorna este objeto de Cliente pelo campo "return" do método, para facilitar o entendimento, o método `getClienteCodX` é apresentado no Quadro 25. As classes `Consumo_DAO` e `Modem_DAO` seguem o mesmo princípio.

[5] significa apresentar, inserir, alterar e remover dado do banco de dados.



```
public Cliente getClienteCodX(int codcli){      // codcli é o código do cliente o qual se quer ver os dados
    cliente = new Cliente();                    // cria novo Objeto de Cliente
    conn = Conexao_DAO.getConnection();         // recebe uma conexão
    Statement stmt = null;                      // inicializa a sentença SQL
    ResultSet rs = null;                        // inicializa resultado do select
    try {
        Class.forName("com.mysql.jdbc.Driver").newInstance();   // driver MySQL Connector/J
        stmt = conn.createStatement();                          // cria uma nova sentença SQL
        rs = stmt.executeQuery("select * from cliente where cd_Cliente = '"+codcli+"'"); // comando SELECT
            if (rs.next()) {                                    // enquanto houver resultado do Select
                cliente.setCodigo(new Integer(rs.getInt("cd_Cliente")));    // seta o atributo código no objeto de Cliente
                cliente.setNome(new String(rs.getString("nm_Cliente")));    // seta o atributo nome no objeto de Cliente
                if (rs.getString("ds_Endereco") != null){                   // se o select retornou algum campo da coluna "ds_Endereco"
                    cliente.setEndereco(new String(rs.getString("ds_Endereco")));} // seta o atributo endereco no objeto de Cliente
                }else{cliente.setEndereco(new String(""));}                 // senão, seta o atributo endereco com o valor vazio
                if (rs.getString("ds_CEP") != null){                        // se o select retornou algum campo da coluna "ds_CEP"
                    cliente.setCEP(new String(rs.getString("ds_CEP")));     // seta o atributo cep no objeto de Cliente
                }else{cliente.setCEP(new String(""));}                      // senão, seta o atributo cep com o valor vazio
                if (rs.getString("ds_Telefone") != null){                   // se o select retornou algum campo da coluna "ds_Telefone"
                    cliente.setTelefone(rs.getString("ds_Telefone"));       // seta o atributo telefone no objeto de Cliente
                }else{cliente.setTelefone(new String(""));}                 // senão, seta o atributo telefone com o valor vazio
                if (rs.getString("ds_Email") != null){                      // se o select retornou algum campo da coluna "ds_Email"
                    cliente.setEmail(new String(rs.getString("ds_Email")));// seta o atributo email no objeto de Cliente
                }else{cliente.setEmail(new String(""));}                    // senão, seta o atributo email com o valor vazio
                cliente.setCodigoModemPrincipal(new Integer(rs.getInt("cd_ModemPrincipal"))); // seta o atributo modemPrincipal
            }
    } catch (Exception e) {
        System.err.println("Erro em Cliente_DAO.getClienteCodX() - "+e);
    }
    finally {
        Conexao_DAO.closeResultSet(rs);     // fecha a resultado do select no banco de dados
        Conexao_DAO.closeStatement(stmt);   // fecha a sentença SQL no banco de dados
    }
    return cliente;                         // retorna o novo objeto de Cliente com os dados recebidos do banco de dados
}
```

Quadro 25 - Método getClienteCodX

Para apresentar relatórios de consumo foi utilizada a biblioteca JfreeChart para gerar os gráficos do tipo XYLineChart. Os dados que devem aparecer no gráfico são armazenados

em um objeto de XYSeriesCollection através de um *ArrayList* contendo os valores de ‘x’ e ‘y’ para cada posição do gráfico, sendo os valores x para representar o tempo (dia ou hora) e y para representar o consumo (total por dia ou total por hora). Este *ArrayList* é montado pela classe Grafico_DAO através de seus métodos “getRelatorioDiarioDoModemXDaDataY” ou “getRelatorioMensalDoModemXDaDataY” ambos os métodos retornam um objeto de “*ArrayList*” com objetos do tipo “ItemGrafico” que por sua vez possuem apenas os atributo do tipo float “x” e “y”. Como exemplo o método getRelatorioMensalDoModemXDaDataY e os demais métodos necessários para o entendimento são apresentados no Quadro 26. E o trecho de código que recebe este *ArrayList* e gera o gráfico de consumo mensal é apresentado no Quadro 27.

```
public ArrayList<ItemGrafico> getRelatorioMensalDoModemXDaDataY(int modem, int mes, int ano){
    ItemGrafico itemGrafico;
    ArrayList<ItemGrafico> arrayItemGrafico = new ArrayList();
    for (int dia = 1; dia < 31; dia++){  // cria ponteiro com o nome "dia"
         itemGrafico = new ItemGrafico();
         itemGrafico.setX(dia); // seta a variável "x" em itemGrafico com o dia (1 - 31)
         itemGrafico.setY(getConsumoTotalDoModemXNoDiaY(modem, dia,  mes,  ano));
         // comando acima -> seta a variável "y" em itemGrafico com o valor do consumo total do dia,
         //                  recebido do método getConsumoTotalDoModemXNoDiaY(modem, dia,  mes,  ano)
         arrayItemGrafico.add(itemGrafico); // adiciona itemGrafico ao ArrayList de rotorno
    }
    return arrayItemGrafico;
}
public float getConsumoTotalDoModemXNoDiaY(int modem,int dia, int mes, int ano){
    float totalDoDia = 0;
    for (int hora = 0; hora < 24; hora++){  // cria ponteiro com o nome "hora"
         totalDoDia += getConsumoTotalDoModemXNaHoraY(modem, dia,  mes,  ano, hora);
    }
    return totalDoDia;
}
public float getConsumoTotalDoModemXNaHoraY(int modem,int dia, int mes, int ano, int hora){
    float totalDaHora = 0;
    conn = Conexao_DAO.getConnection();
    Statement stmt = null; ResultSet rs = null;
    try {
        stmt = conn.createStatement();
                    rs = stmt.executeQuery("SELECT vl_Consumo "+
                            " FROM consumo" +
                            " WHERE cd_Modem = "+modem+" AND "+
                            " Year (dt_Data) = "+ano+" AND "+
                            " Month (dt_Data) = "+mes+" AND "+
                            " Day (dt_Data) = "+dia+" AND "+
                            " Hour (dt_Data) = "+hora);
                    while (rs.next()){
                        totalDaHora += rs.getFloat("vl_Consumo");
                    }
    }catch(Exception e){
         System.err.println("Erro em Grafico_DAO.getConsumoTotalDoModemXNaHoraY() - "+e);
    }
    finally { // independente de gerar exceção ou não, este código será executado
        Conexao_DAO.closeResultSet(rs);
        Conexao_DAO.closeStatement(stmt);
    }
    return  totalDaHora;
}
```

Quadro 26 - Método `getRelatorioMensalDoModemXDaDataY`

```
if (this.jComboBox_Rel_Tip.getSelectedItem().toString().compareToIgnoreCase("Mensal") == 0){ // Se Relatório for Mensal
    ArrayList<ItemGrafico> arrayItens = grafico_dao.getRelatorioMensalDoModemXDaDataY(modem, mes, ano);
    float x,y;      // o comando acima, retorna um ArrayList com 31 campos, cada um contendo o consumo total daquele dia
    for (int i = 0; i < arrayItens.size(); i++){ // para cada item do array
        x = arrayItens.get(i).getX(); // x = dia do mês no ano
        y = arrayItens.get(i).getY(); // y = total consumido no dia
        series.add(x,y);
    }
   XYSeriesCollection dataset = new XYSeriesCollection();
   dataset.addSeries(series);   // adiciona ao dataset o objeto de XYSeries contendo posições(x,y)
   JFreeChart chart = ChartFactory.createXYLineChart(  // Cria um Grafico do tipo XYLineChart
            "Consumo mensal de "+nomeCliente, // título do gráfico,
            "Tempo (DIAS) - "+this.jComboBox_Rel_Mes.getItemAt(mes)+"/"+ano,            // Título do eixo 'X'
            "Valor consumido de agua em m3",                                            // Título do eixo 'Y'
            dataset,                          // dataset - array de pontos (x,y)
            PlotOrientation.VERTICAL,         // sentido do gráfico
            true,                             // legenda é necessária?
            false,                            // configurar o gráfico para gerar tool tips?
            false);                           // configurar gráfico para gerar urls?
   imagem_chart = chart.createBufferedImage(800,600);    // coloca o resultado do gráfico em um BufferedImage
   this.frameGrafico.setImagemGrafico(new javax.swing.ImageIcon(imagem_chart)); // pinta o JFrame com o grafico
   this.frameGrafico.setVisible(true);  // torna a janela com o gráfico visivel
}
```

Quadro 27 - Trecho de código que gera o gráfico de consumo mensal

Para enviar comandos aos modems, utilizou-se a API RXTX, forma criados os métodos SerialCom e SerialCommLeitura utilizando-se como base o tutorial desenvolvido por Petter Rafael Villa Real Silva (2007, p. 1-4). No Quadro 28 é apresentado um pseudo código para enviar e receber dados utilizando métodos da classe SerialCommLeitura.

```
public void exemploEnviarReceberDados(char dado1, char dado2){
    final int VALOR_TIMEOUT = 5000;
    SerialCommLeitura comunicacao = new SerialCommLeitura("COM1",9600,0);
    comunicacao.habilitarEscrita();
    comunicacao.obterIdDaPorta();
    comunicacao.abrirPorta();
    comunicacao.enviarUmaString(dado1); // envia o ID do modem que irá receber o comando
    comunicacao.enviarUmaString(dado2); // envia o numero do comando
    // enviados os dados agora ele espera uma resposta
    comunicacao.habilitarLeitura();
    comunicacao.lerDados();
    tempoInicial = System.currentTimeMillis();

    while (true){ // loop infinito
        tempoPercorrido = System.currentTimeMillis() - tempoInicial; // atualiza timeout
        if (tempoPercorrido > VALOR_TIMEOUT){                        // se ocorreu timeout
            System.out.println("TIMEOUT");
            comunicacao.fecharCom();
            break;
        }
        if (comunicacao.getFimDaLeitura()){             // se finalizou recepção dos dados
            Vetor vetor = comunicacao.getVetorDados().get(0).toString();
            comunicacao.fecharCom();
            break;
        }
    }
}
```

Quadro 28 - Pseudo código para enviar e receber dados à serial

Para armazenar os dados de consumo no banco de dados foi adotada a seguinte estratégia, o microcontrolador não armazena a data em que cada dado foi armazenado, porém como ele armazena um dado a cada 1 minuto, após solicitar os valores ao microcontrolador, o software executa o método carregaListaDeConsumos passando como parâmetros o vetor com

os dados e o objeto do modem que enviou os dados. O método carregaListaDeConsumos então instanciada um objeto do tipo Calendar com o horário atual, deste subtrai em minutos o valor igual a quantidade de itens no vetor, e então através de um for de 0 até a quantidade de itens no vetor, o método calcula o valor consumido e gera uma data(YYYY-MM-DD hh:mm:ss) no forma de String incrementando em 1 minuto (para cada iteração do for), após isto o método cria um objeto de Consumo e seta os valores com o valor consumido, o código do modem e a data gerada, e armazena este objeto em um *ArrayList*, ao fim do for o método retorna este *ArrayList*. Agora o software do PC executa o método addListaDeConsumo da classe Consumo_DAO passando este *ArrayList* como parâmetro, então o método addListaDeConsumo armazena no banco de dados 1 registro para cada objeto de Consumo armazenado no *ArrayList*, agora já com o campo "data" preenchido. Para facilitar o entendimento o método `carregaListaDeConsumos` é apresentado no Quadro 29 e o método addListaDeConsumo no Quadro 30. O valor consumido foi calculado multiplicando o valor recebido pelo microcontrolador pelo valor da atributo fatorRazao do objeto Modem, por que o microcontrolador armazena a quantidade de pulsos recebidos e a variável fatorRazao no objetoModem significa quantos m³ representam 1 pulso. O valor fatorRazao pode variar dependendo da construção do hidrômetro.

```
public ArrayList carregaListaDeConsumos(Vector listaDadosSerial, Modem modem){
ArrayList arrayDeConsumos = new ArrayList();
Calendar c = Calendar.getInstance();  // cria um objeto de Calendar com a data/horario atual
String data;
double valorConsumido = 0.0;
    try {
        c.add(Calendar.MINUTE, -listaDadosSerial.size()); // subtrai em minutos o tamanho do vetor
        for (int i =0; i < listaDadosSerial.size(); i++){
            // calcula valor consumido
            valorConsumido = Integer.parseInt(String.valueOf(listaDadosSerial.get(i)))*
                             modem.getFatorRazao(); // multiplica o dado recebido pelo fatorRazao
            // cria data em que foi efetuada a leitura
            c.add(Calendar.MINUTE, 1); // adianta 1 minuto no alendario c
            data = c.get(Calendar.YEAR)+"-"+    // gera uma String de data(YYYY-MM-DD hh:mm:ss)
                   c.get(Calendar.MONTH)+"-"+
                   c.get(Calendar.DAY_OF_MONTH)+" "+
                   c.get(Calendar.HOUR_OF_DAY)+":"+
                   c.get(Calendar.MINUTE)+":"+
                   c.get(Calendar.SECOND);
            Consumo consumo = new Consumo();                    // cria novo objeto de Consumo
            consumo.setValor_consumido(valorConsumido);
            consumo.setCodigo_modem(modem.getCodigo_Modem());
            consumo.setData(data);

            arrayDeConsumos.add(consumo);
        }
    } catch (Exception e) {
        System.err.println("Erro no Main.carregaListaDeConsumos"+e);
    }
 return arrayDeConsumos;
}
```

Quadro 29 - Método `carregaListaDeConsumos`

```
public void addListaDeConsumo(ArrayList array){
    ArrayList<Consumo> arrayDeConsumo = array;
    conn = Conexao_DAO.getConnection();
    Statement stmt = null;
    try {
        for (int i =0; i < arrayDeConsumo.size(); i++){
        stmt = conn.createStatement();
        stmt.executeUpdate("insert into consumo " +
                           "(cd_modem," +
                           "dt_data," +
                           "vl_consumo)" +
                           "values ("+
                           arrayDeConsumo.get(i).getCodigo_modem()+",'"+
                           arrayDeConsumo.get(i).getData()+"',"+
                           arrayDeConsumo.get(i).getValor_consumido()+
                           ")");
        }
    } catch (Exception e) {
        System.err.println("Erro em Consumo_DAO.addListaDeConsumo - "+e);
    }
    finally {
        Conexao_DAO.closeStatement(stmt);
    }
}
```

Quadro 30 - Método `addListaDeConsumo`

Para calcular o consumo a partir do comprimento da onda é necessário primeiramente solicitar ao modem em questão o valor do comprimento da onda que por sua vez, é um valor inteiro correspondendo à quantidade de estouros do timer0, este valor é multiplicado por 0,002048 segundos (valor definido pelas configurações do timer0) definindo em segundos o comprimento da onda, após isto divide-se 30 (segundos) pelo comprimento da onda obtido ou seja, quantos pulsos com o comprimento obtido "cabem" em 1 minuto, com este resultado têm-se um previa do consumo em 1 minuto. Divide-se 30 pelo comprimento da onda e não 60 pois o valor do comprimento da onda na verdade é o valor do comprimento de meia-onda (apenas o período alto da onda) portanto ou multiplica-se o comprimento da onda por 2(dois) ou divide-se o minuto por 2(dois), no projeto foi optado por dividir 1 minuto por 2(dois) resultando no valor 30 (trinta segundos). Esta previa é multiplicada pelo fatorRazão do Modem em questão (informação armazenada na base de dados mySQL) e assim a previsão de consumo em 1 minuto. O código do método `calculaConsumoPeloComprimentoDaOnda` é apresentado no Quadro 31.

```
public double calculaConsumoPeloComprimentoDeOnda(int compOnda, int codModem){
    double valor = 0.0;
    Modem modem = modem_dao.getModemCodX(codModem);
    valor = compOnda * 0.00204800;          // quantidade de estouros do timer0 * tempo (em seg) de um estouro do timer0
    valor = 30 / valor;                     // 1 minuto / pelo resultado anterior
    valor = valor * modem.getFatorRazao();  // resultado anterior * a razao do modem em questao
    return valor;
}
```

Quadro 31 - Método `calculaConsumoPeloComprimentoDaOnda`

Para verificar vazamentos foi criada uma classe chamada `VerificadorVazamentos` que extende da classe Thread. Quando o usuário executa o comando "`Verificar Vazamentos`" o software do PC cria um objeto desta classe e executa o método run(), que por sua vez ativa o método, este método calcula os vazamentos da seguinte forma, ele solicita ao modem que está selecionado na variável jComboBox_Man_Mod (JComboBox de seleção de modem) o seu valor de consumo atual e coloca esta valor na variável consumoPai, então ele verifica no banco de dados quem são os modems que tem este como Pai, ou seja quem está ligado após este modem, e também solicitando o consumo dos mesmos adiciona a soma dos valores de consumo dos filhos na variável consumoFilhos, após isto é verificado se a variável consumoPai é maior que a variável consumoFilhos com uma margem de erro de 2%, se for maior então significa que existe um vazamento, neste instante o *label* j*Label*_Alarme é setado como visível mostrando um sinal de aviso e no área de Status também é apresentada uma mensagem de que existe vazamento na linha do modem selecionado. Caso contrário o *label* j*Label*_Alarme é setado com não-visível. Esta rotina de verificação ocorre repetidamente até que o checkBox na aba "Manual" seja deselecionado. Um pseudocódigo representando como são verificados os vazamentos é apresentado no Quadro 32.

```
ArrayList filhos;
double consumoPai = 0;
double consumoFilhos = 0;

while (funcionando){     // enquanto o checkBox na aba "Manual" estiver selecionado
    consumoPai = 0;
    consumoFilhos = 0;
    Solicita ao modem selecionado o consumo atual;
    armazena o valor em consumoPai;
    filhos = lista de modems que tem como a coluna cd_ModemPai o modem selecionado;
            for (int i = 0 ; i < filhos.size(); i++){
               Solicita ao modem filho[i] o consumo atual;
               adiciona o valor a variavel consumoFilhos;
            }
            double margemDeErro = 0.02; // 2%
            if (consumoFilhos*(1+margemDeErro) < consumoPai){
                Escreve aviso de vazamento na área de status
                seta o label jLabel_Alarme como visivel
            }else{
                Escreve que não há vazamento na área de status
                seta o label jLabel_Alarme como não-visivel
            }
        }
    }
```

Quadro 32 - Pseudocódigo representando como são verificados os vazamentos

3.3.5 Implementação da base de dados

A base de dados foi implementada conforme o *script Structured Query Language* (SQL) apresentado no Quadro 33, com nome do usuário e senha igual a “root”.

```
CREATE DATABASE base_consumo;

CREATE TABLE `cliente` (
  `cd_Cliente` int(5) NOT NULL auto_increment,
  `nm_Cliente` varchar(30) NOT NULL,
  `ds_Endereco` varchar(50) default NULL,
  `ds_CEP` varchar(10) default NULL,
  `ds_Telefone` varchar(20) default NULL,
  `ds_Email` varchar(30) default NULL,
  `cd_ModemPrincipal` int(5) default NULL,
  PRIMARY KEY  (`cd_Cliente`),
  UNIQUE KEY `Cliente_nm_Cliente_Unique` (`nm_Cliente`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

CREATE TABLE `consumo` (
  `cd_Modem` int(5) NOT NULL,
  `dt_Data` datetime NOT NULL,
  `vl_consumo` double NOT NULL,
  PRIMARY KEY  (`dt_Data`,`cd_Modem`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

CREATE TABLE `modem` (
  `cd_Modem` int(5) NOT NULL auto_increment,
  `cd_Cliente` int(5) unsigned zerofill default NULL,
  `cd_Pai` int(5) NOT NULL,
  `vl_FatorRazao` double(5,3) NOT NULL,
  `ds_Descricao` tinytext,
  PRIMARY KEY  (`cd_Modem`),
  KEY `Modem_Cliente_FK` (`cd_Cliente`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
```

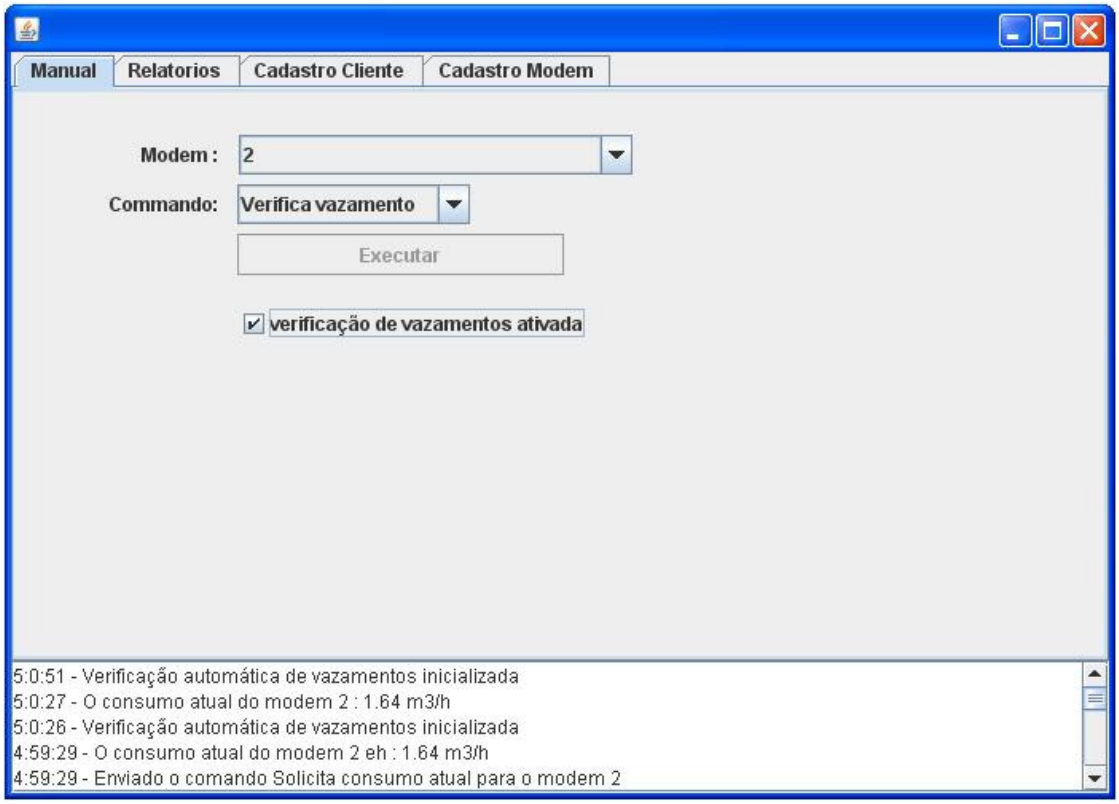
Quadro 33 – *Script* SQL para a implementação da base de dados

3.3.6 Operacionalidade da implementação

O usuário pode acessar qualquer uma das 4 abas disponíveis, sendo a aba “Manual”, “Relatórios”, “Cadastro Cliente” e “Cadastro Modem”.

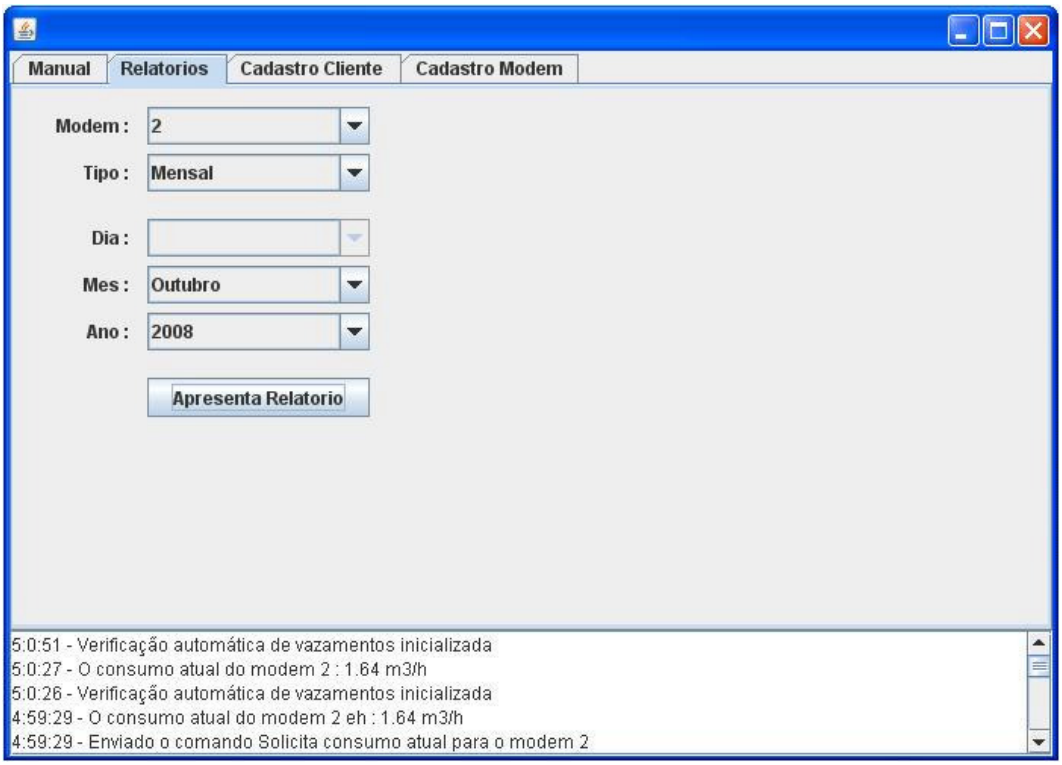
Na aba “Manual” o usuário pode inicialmente selecionar um modem, opós isto é ativada a opção para selecionar um comando e em seguida é ativado o botão “Executar”. Se o usuário selecionar o comando “Verifica Vazamento” então será ativado o checkBox apresentando a descrição “verificação de vazamento ativada” e serão desativados os demais itens desta aba, até que o checkBox seja deselecionado, após isto a aplicação irá esperar até

que a thread que fica verificando vazamentos tenha finalizado, para daí então desativar o checkBox e novamente ativar os demais itens. Enquanto o comando “Verifica Vazamento” estiver ativo, as demais abas podem ser acessadas e trabalhadas sem afetar o mesmo, pois é executado por uma thread. Os comandos possíveis são: “Solicita dados”,”Liga Valvula”, “Desliga Valvula”, “Solicita consumo atual” e “Verifica vazamentos”. A aba “Manual” em execução poder ser visualizada no quadro Quadro 34.

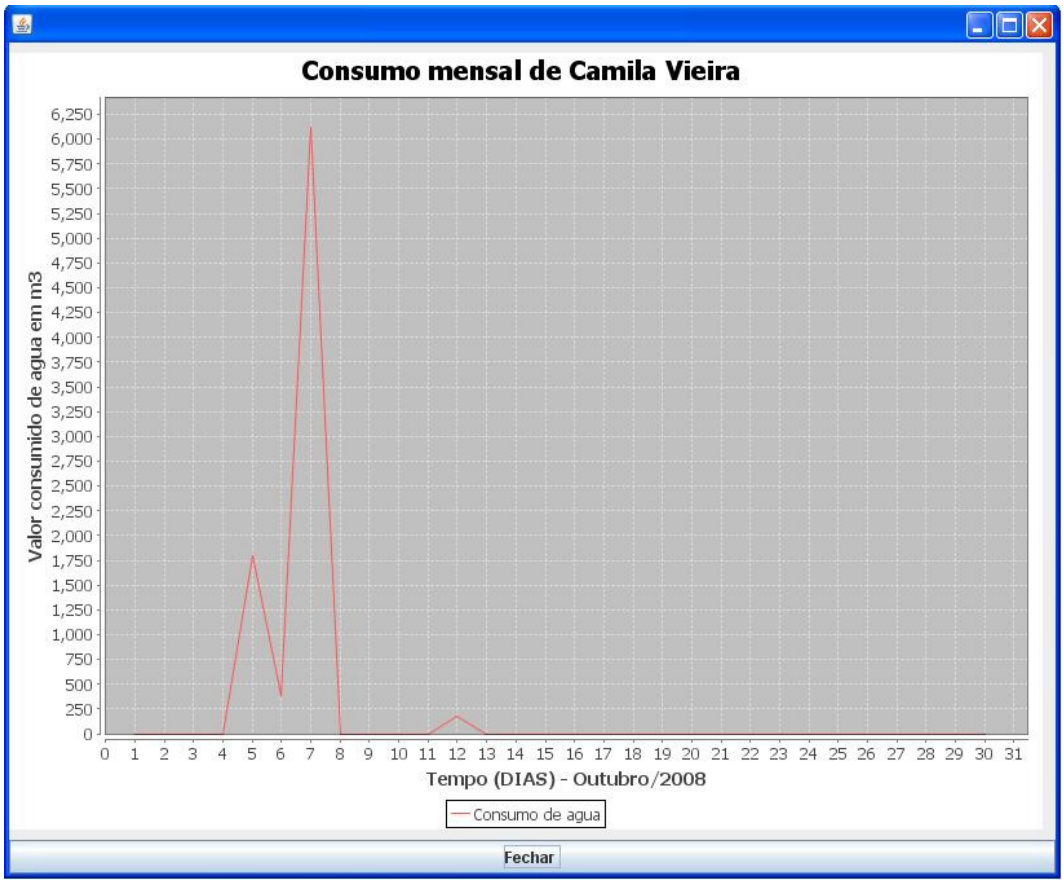


Quadro 34 - Aba "Manual" em execução

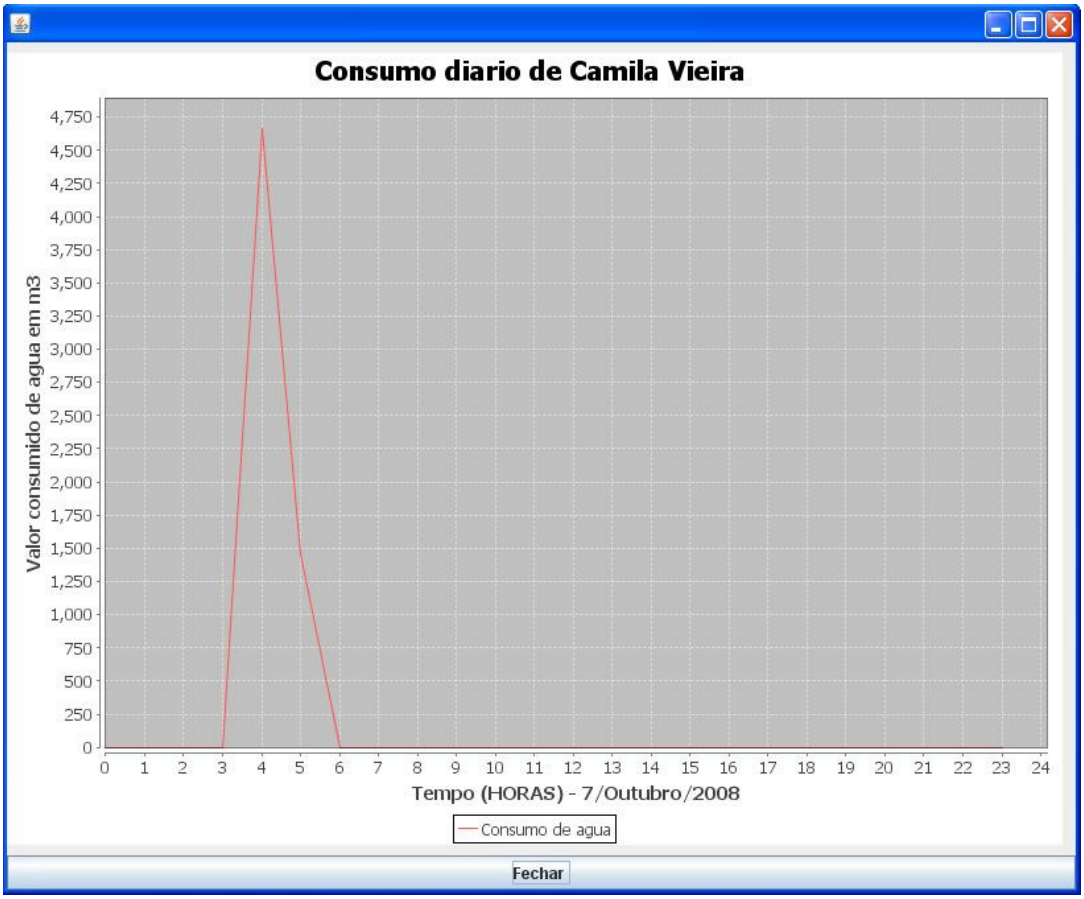
Na aba “Relatorios” o usuário pode inicialmente selecionar um modem, após isto é ativada a opção para selecionar um tipo de gráfico e conforme a opção selecionada serão ativados os demais itens com exceção do botão “Apresenta Relatório” que será ativado somente quando todos os campos que estiverem ativos estiverem selecionados com um valor válido. Quando for clicado sobre o botão “Apresenta Relatório” ativo a aplicação irá gerar o gráfico correspondente a apresentá-lo em uma janela externa, que por sua vez pode ser fechada no botão “Fechar”, o gráfico irá apresentar os dados de consumo solicitados e também o nome do cliente vinculado ao modem selecionado. Os tipos de gráfico implementados são: diário, retornando o consumo total de cada hora do dia selecionado e mensal, retornando o consumo total de cada dia do mês selecionado. A aba “Relatorios” em execução poder ser visualizada no quadro Quadro 35. Nos Quadro 36 e Quadro 37 são apresentados um exemplo de gráfico mensal e um exemplo de gráfico diário respectivamente.



Quadro 35 - Aba "Relatorios" em execução

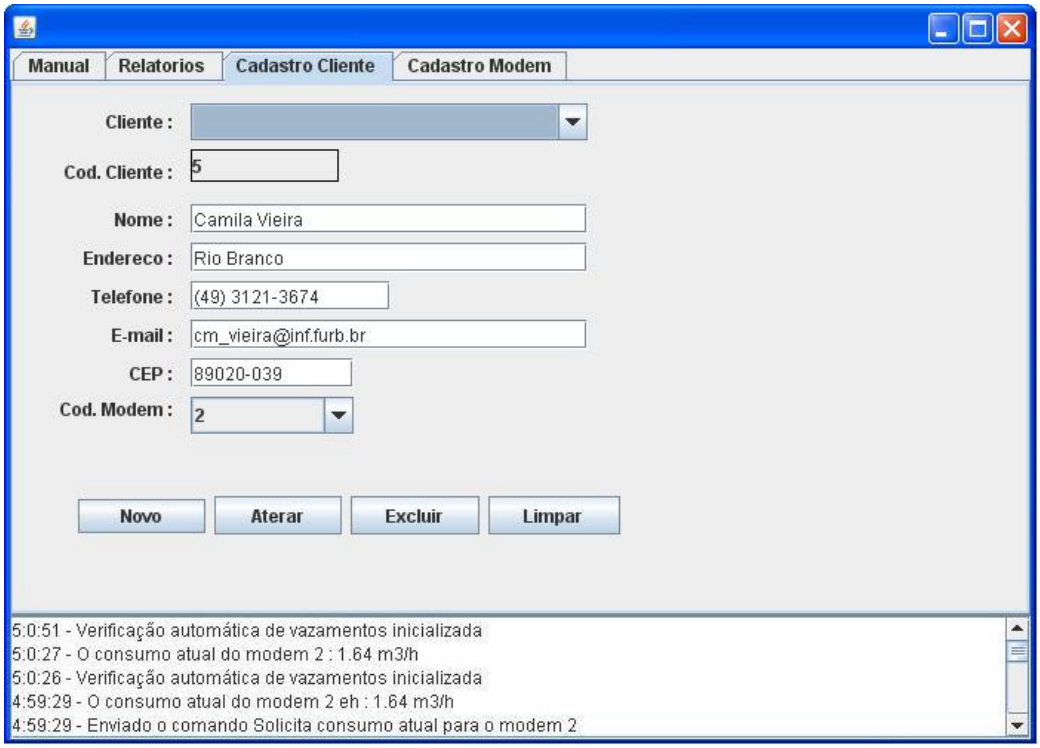


Quadro 36 - Exemplo de gráfico mensal



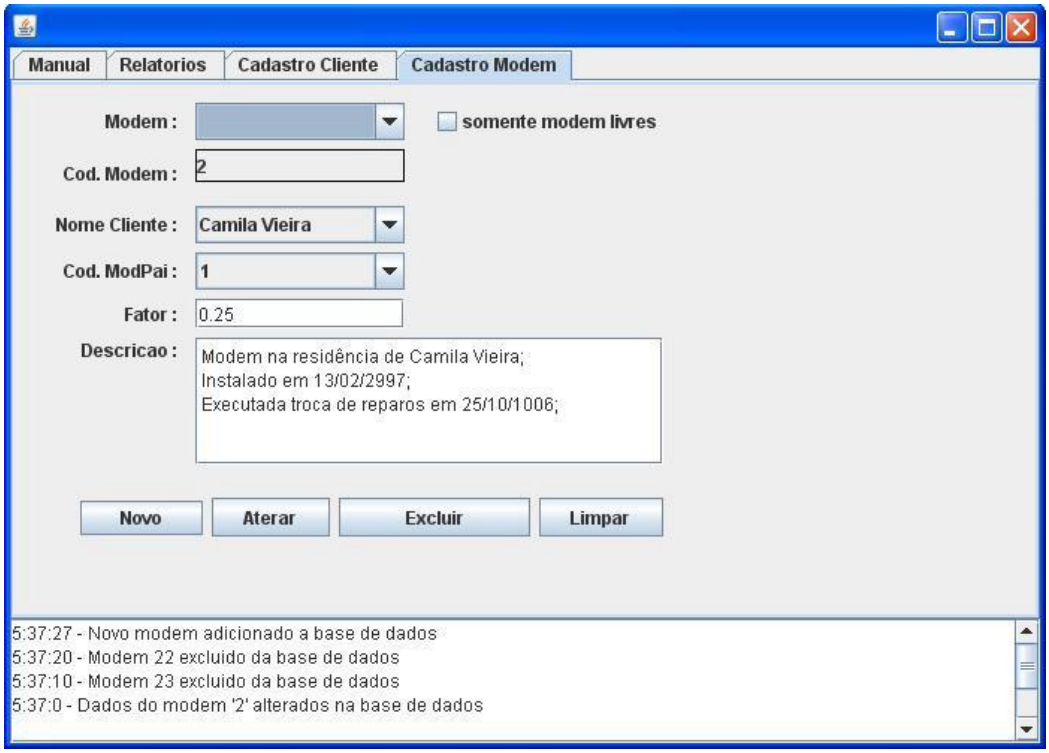
Quadro 37 - Exemplo de gráfico diário

Na aba “Cadastro Cliente” o usuário pode cadastrar novos clientes, alterar os dados cadastrais de clientes, remover clientes e visualizar os dados de clientes. A aba “Cadastro Cliente” em execução poder ser visualizada no quadro Quadro 38.



Quadro 38 - Aba "Cadastro Cliente" em execução

Na aba “Cadastro Modem” o usuário pode cadastrar novos modems, alterar os dados de modems, remover modems e visualizar os dados de modems. O checkBox com a descrição “somente modems livres” ao ser selecionado filtra na seleção de modems para que sejam apresentados apenas os modems que ainda não tem um cliente vinculado a eles. A aba “Cadastro Modem” em execução poder ser visualizada no quadro Quadro 39.



Quadro 39 - Aba "Cadastro Modem" em execução

3.4 RESULTADOS E DISCUSSÃO

Os objetivos propostos no desenvolvimento deste trabalho foram alcançados. Porém, vale salientar alguns desafios e dificuldades que foram encontrados durante o desenvolvimento, tais como:

a) a idéia inicial do projeto seria com o uso de hidrômetros digitais, porém devido ao custo dos mesmos optou-se por simulá-los através de uma saída pulsada de um CI 555. Converter um hidrômetro convencional em hidrômetro digital com um custo mais baixo é possível, como mostra o artigo de Fontes et al. (2004), porém o custo de cada hidrômetro convencional seria de no mínimo R$ 95,00 e ainda faltariam os demais componentes para a conversão;
b) o uso do microcontrolador PIC 18F452 gerou inicialmente alguns contratempos, pois acreditava-se que para fazê-lo funcionar necessitavam os mesmos componentes associados que em outros microcontroladores PIC já estudados como o PIC 16F84, foi necessário um estudo mais profundo para chegar a conclusão que seriam necessários capacitores associados ao cristal para que o microcontrolador operasse corretamente;
c) o uso de protoboard facilitou o processo de testes no quisito montagem do circuito porém por ser um equipamento para fins didáticos e devido a sua construção o mesmo também gera um grande nível de ruído no circuito, tornando o circuito instável e inoperável em diversos momentos;
d) o uso da ferramenta PCH para compilação do software embarcado gerou alguns contratempos pois além da necessidade de estudar a linguagem, a ferramenta possui suas próprias rotinas de configuração de registros, como por exemplo o Timer0, tornando diversos exemplos genéricos encontrados na internet inúteis pois necessitava configurações específicas;
e) a ferramenta NetBeans 6.0 também teve seu momento de complicação, apesar de possuir um gerador de interfaces simples de trabalhar (motivo pelo uso desta ferramenta), em alguns momentos ao colocar um novo JButton por exemplo, o mesmo ao ser colocado no JPanel desconfigurava a posição e largura de alguns componentes associados ao mesmo Jpanel por este motivo alguns componente ficaram ligeiramente deslocados ou com tamanho diferente em relação aos demais;
f) para calcular o consumo atual, seria inviável aguardar 1 minuto até que fossem

totalizados os pulsos para calcular o consumo, por isso optou-se por calcular pela comprimento da onda, ou seja calcula-se o tempo em que o pulso fica ativo, e assim é assim calcular a frequência em que os pulsos ocorrem sem ter que aguardar a totalização em 1 minuto, essa rotina para calcular o comprimento da onda, também consumiu um bom tempo;

g) os orçamentos dos transceptores geraram muito contratempo pois dos 10 orçamentos solicitados sendo 5 referente a transceptores XBee e 5 referente a transceptores RF 2,4Ghz comuns, retornaram apenas 3 orçamentos dos 10, o que atrasou o início do desenvolvimento, pois ainda não havia sido definida qual tecnologia de transceptor utilizar.

O custo dos componentes para o desenvolvimento do protótipo foi em torno de R$ 550,00. Um quadro comparativo entre o trabalho desenvolvido e os trabalhos correlatos é apresentado no Quadro 40.

	Meio comunicação	Limite pontos de leitura	Foco de atuação	Tipo hidrômetro	Custo
Trabalho desenvolvido	Wireless	10	Geral	Digital/Convêncional convertido	Baixo
Sistema Hidronet na USP	Cabo	n	Geral	Digital	Alto
Sistema SCOA na SABESP	Cabo	n	Rede distribuição	Digital	Alto

Quadro 40 - Quadro Comparativo

4 CONCLUSÕES

O trabalho atingiu todos os objetivos propostos e para isto foram realizados estudos sobre o microcontrolador PIC 18F452, a ferramenta PCH da empresa CCS, a API RXTX, a biblioteca JfreeChart, CI MAX-232, CI 555 e acessos a banco de dados MySQL via driver MySQL Connector/J, todos os itens descritos acima foram de suma importância para que o objetivo fosse atingido.

A limitação do trabalho está na conexão máxima 10 pontos de consumo ao contrário do ideal que seria no mínimo 1 para cada residência conectada à rede de distribuição de água da central de distribuição.

O trabalho realizado traz noções de usabilidade da tecnologia *ZigBee* e do módulo Xbee que por sua vez, são considerados tecnologias recentes, também apresenta o uso da API RXTX utilizada para efetuar comunicação entre aplicações no PC e sistemas embarcados, e a biblioteca JfreeChart para a criação diversos tipos de gráficos, neste caso foi um gráfico de linhas.

O trabalho realizado também mostrou que é possível reduzir os índices de desperdício de água tratada através de sistemas de telemetria, apesar de nenhum teste em campo ter sido realizado, a aplicação prova que é possível detectar vazamentos com eficiência, tanto em uma rede de distribuição quanto em uma rede residencial ou industrial.

O software embarcado foi desenvolvido para totalizar e armazenar os pulsos a cada 1 minuto e assim sendo necessário que o software no PC solicite os dados a cada 12 horas, porém se o software embarcado totalizar e armazenar os pulsos a cada 1 hora, o software no PC poderá solicitar os dados a cada 30 dias.

O trabalho desenvolvido em relação ao trabalhos correlatos, tem a vantagem de ser através de comunicação wireless

4.1 EXTENSÕES

Fica como sugestão para o aprimoramento deste trabalho os seguintes itens:

a) implementar o uso de uma DIP switch para setar o ID no microcontrolador;
b) ajustar o software embarcado e o software do PC para que possa atender mais do

que apenas 10 terminais finais;

c) utilizar hidrômetros reais (digitais ou convertidos para digitais);
d) desenvolver um circuito para efetuar comunicação pela porta USB do PC;
e) efetuar testes em uma rede de distribuição;
f) desenvolver um software para PDA ou celular permitindo solicitar os dados através dos mesmo.
g) desenvolver um circuito gerador através do movimento da água para gerar a energia para alimentar cada terminal final.

REFERÊNCIAS BIBLIOGRÁFICAS

BBC. **2,7 bilhões podem ficar sem água em 2025, diz ONU.** [S.l.], 2002. Disponível em: <http://www.bbc.co.uk/portuguese/noticias/2002/020322_secaml.shtml>. Acesso em: 20 set. 2008.

BRAIN, Marshall. **Como funcionam os microcontroladores?:** o que é um microcontrolador? São Paulo,[2007?]. Disponível em: <http://eletronicos.hsw.uol.com.br/microcontroladores1.htm>. Acesso em: 18 nov. 2008.

ELETRÔNICA.ORG. **Alternativas de baixo custo ao MAX 232.** [S.l.], 2008. Disponível em: <http://www2.eletronica.org/hack-s-dicas/alternativas-de-baixo-custo-ao-max-232/>. Aceso em: 12 nov. 2008.

EMBEDDEDWORLD. **F.A.Q - Módulos *ZigBee*:** quais as características principais do padrão *ZigBee*?. São Paulo, [2008?]. Disponível em: <http://www.embeddedworld.com.br/ex07.asp>. Acesso em: 12 nov. 2008.

FONTES, Ivo R. et al. **Desenvolvimento de um hidrômetro eletrônico de baixo custo.** Bauru, [2004?]. Disponível em: <http://www.lti.pcs.usp.br/robotics/grva/publicacoes/outras/cba2004-cd-rom/cba2004/pdf/1112.pdf >. Acesso em: 10 set. 2008.

GTA. **Apostila básica sobre o 555.** Rio de Janeiro, 2003. Disponível em: <http://www.gta.ufrj.br/grad/01_1/contador555/555 >. Acesso em: 12 nov. 2008.

ICA. **Protocolo HDLC:** formato do quadro. Rio de Janeiro, [2007?]. Disponível em: <http://www.ica.ele.puc-rio.br/cursos/download/CC-enlace_HDLC.pdf>. Acesso em: 19 nov. 2007.

KONDO, Rogerio T.; MARTINELI, Edmar. **Serviço de wireless.** São Carlos, 2007. Disponível em: <http://www.cisc.usp.br/wireless/>. Acesso em: 20 set. 2008.

M_BUS. **The M_Bus:** an overview**.** [Padernborn], 2000. Disponível em: <http://www.m-bus.com/info/mbuse.html>. Acesso em: 20 set. 2008.

MIKROELETRÔNICA. **Introdução aos microcontroladores:** microcontroladores versus microprocessadores. [S.l.], 2003. Disponível em: <http://www.mikroe.com/pt/product/books/picbook/capitulo1.htm>. Acesso em: 20 set. 2008.

NETTO, Luiz F. **Resistores:** para que servem os resistores? São Paulo, [2007?]. Disponível em: <http://www.feiradeciencias.com.br/sala12/12_T02.asp>. Acesso em: 13 nov. 2008.

NÓRCIO Lúcia. **Falta de água potável no mundo aparece relacionada a 80% das mortes e doenças.** Foz do Iguaçu, 2007. Disponível em: <http://www.agenciabrasil.gov.br/noticias/2007/03/22/materia.2007-03-22.6414269867/view>. Acesso em: 18 set. 2008.

MESSIAS, Antônio R. **Módulos *ZigBee*/XBee da MaxStream.** São Paulo, 2007. Disponível em: <http://www.rogercom.com/ZigBee/ZigBee.htm>. Acesso em: 12 nov. 2008.

SABESP. **Controle operacional.** [São Paulo]. Disponível em: <http://www.sabesp.com.br/CalandraWeb/CalandraRedirect/?temp=4&proj=sabesp&pub=T&db=&docid=3C4261631C5429C0832571B100558EFC>. Acesso em: 20 set. 2008.

SAE. **Perguntas e respostas – FAQ:** o que é hidrômetro e qual a sua importância? Ituituba, 2002. Disponível em : <http://www.saeituiutaba.com.br/?contexto=040309>. Acesso em: 13 nov. 2007.

SILVA, Ardemírio de Barros. **Dicionário ilustrado:** teste de redundância cíclico. Feira de Sanatana, [2007?]. Disponível em: <http://www.uefs.br/disciplinas/exa519/DICIONARIO_14.pdf>. Acesso em: 19 nov. 2007.

SILVA, André T. **Módulos de comunicação *wireless* para sensores.** Porto, 2007. Disponível em: <http://paginas.fe.up.pt/~ee02055/RelatorioTEC15.pdf>. Acesso em: 12 nov. 2008.

SILVA, Petter R. V. R. **Utilizando a API RXTX para manipulação da serial .** Maringá, 2007. Disponível em: <http://www.devmedia.com.br/articles/viewcomp.asp?comp=6722>. Acesso em: 12 nov. 2008.

TAFNER, Malcon A.; LOESCH, Claudio; STRINGARI, Sérgio. **Comunicação de dados usando linguagem C:** aplicação em DOS e Windows. Blumenau. Editora da FURB, 1996.

TELECO. **Telemetria:** descrição do Setor. [S.l.], [2007?]. Disponível em: <http://www.teleco.com.br/tutoriais/tutorialmtelemetria/pagina_2.asp>. Acesso em: 21 set. 2008.

TIOSAM. **Transceptor.** [S.l.], [2007?]. Disponível em: <http://www.tiosam.com/enciclopedia/?q=Transceptor>. Acesso em: 22 set. 2008.

UN. **Water a shared responsibility:** the United Nations world water development report 2. Paris, 2006. Disponível em: <http://unesdoc.unesco.org/images/0014/001454/145405E.pdf>. Acesso em: 20 set. 2007.

VIKACONTROLS. **Teoria sobre o Xbee.** Modo de Transmissão. [S.l], [2008?]. Disponível em: <http://www.vikacontrols.com.br/catalogos_pdf/Teoria_XBee.pdf>. Acesso em 15 fev. 2009.

VIVASEMFIO. **ZigBee**. [S.l], 2007. Disponível em: <http://www.vivasemfio.com/blog/zigbee/>. Acesso em: 15 fev. 2009.

ZELENOVSKY, Ricardo; MENDONÇA, Alexandre. **Arquitetura de microcontroladores modernos:** o que é um microcontrolador? [S.l.], [2006?]. Disponível em: <http://www.mzeditora.com.br/artigos/mic_modernos.htm>. Acesso em: 20 set. 2008.

ZENER. **Hidrômetros Monojato.** Novo Hamburgo, 2007. Disponível em: <http://www.zenner.com.br/pdf/monojatos.pdf>. Acesso em : 20 set. 2008.