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

Acadêmico: Benno Martim Schubert

Orientador: Miguel Alexandre Wisintainer

ROTEIRO

- Introdução
- Objetivos do trabalho
- Fundamentação teórica
- Desenvolvimento do trabalho
- Resultados e Discussão
- Conclusão
- Extensões

Introdução

- Importância da água tratada no mundo
- Necessidade de reduzir o desperdício de água tratada
- Possível solução: uso de sistemas informatizados para identificar vazamentos e controlar o consumo de água tratada
- Sistema de captura de dados para controle de consumo de água

Objetivos do trabalho

- Desenvolver um protótipo de sistema de telemetria, para analisar o consumo e o desperdício de água tratada
- Efetuar a leitura dos dados através de uma comunicação sem fio
- Detectar vazamentos
- Gerar relatórios de consumo
- Gerar mensagens de alerta

Fundamentação teórica

- Conceitos Básicos
  - ✓ Transceptores
  - ✓ Protocolo *ZigBee*
- Trabalhos correlatos
  - ✓ Hydronet
  - ✓ Sistema de Controle da Operação do Abastecimento (SCOA)

Transceptores

- Responsáveis pela comunicação wireless
- Chamados Modems
- Módulo Xbee XB24-AWI-001 da MaxStream

Protocolo ZigBee

- Nível de abstração
- Segurança (AES 128 bits)
- Redes tipo estrela, árvore e malha
- Uso do protocolo IEEE 802.15.4

Hydronet

- Hydrometer
- Protocolo M-Bus
- Programa de Uso Racional de Água (PURA)
- Redução de 36% no consumo de água no campus Cidade Universitária Armando de Salles Oliveira (CUASO)

SCOA

- Linha Telefônica
- Aumento na velocidade de detecção de vazamentos
- Redução do custo de operação e manutenção

Desenvolvimento

- Requisitos funcionais e não funcionais
- Especificações
- Implementação do hardware
- Implementação do software

Requisitos funcionais

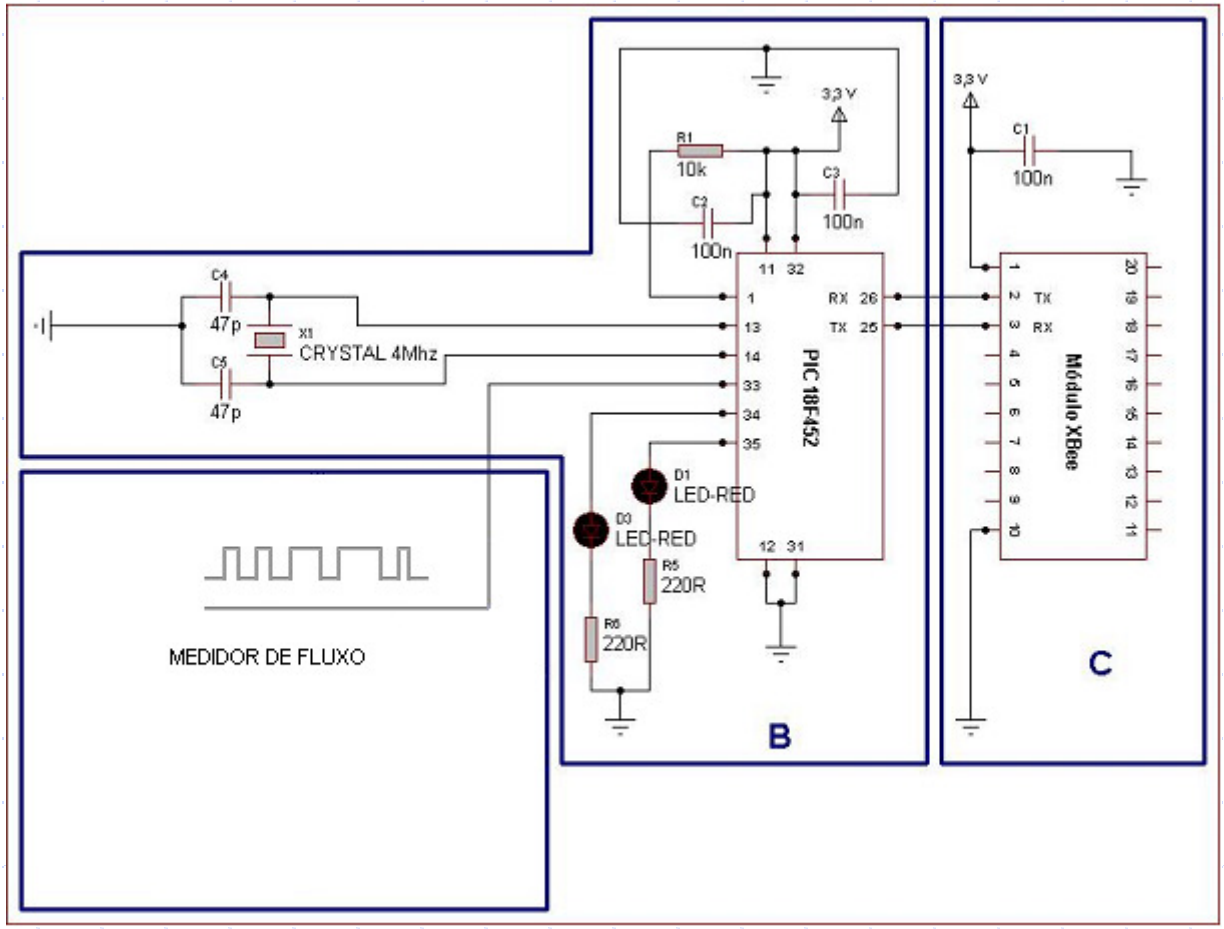
- coletar dados de consumo de água na rede
- gerar relatórios periódicos de consumo
- detectar possíveis vazamentos na rede
- gerar mensagens de alerta quando detectar possíveis vazamentos
- permitir o cadastro de consumidores
- permitir enviar sinais de comando às válvulas

Requisitos não funcionais

- a interface gráfica deverá ser multi-plataforma
- a interface gráfica deverá ser desenvolvida na linguagem Java
- armazenar os dados em um banco de dados MySQL
- utilizar comunicação *wireless* entre o hardware e o software
- deverá utilizar o microcontroladores PIC 18F452 para efetuar a comunicação entre transceptor e hidrômetro

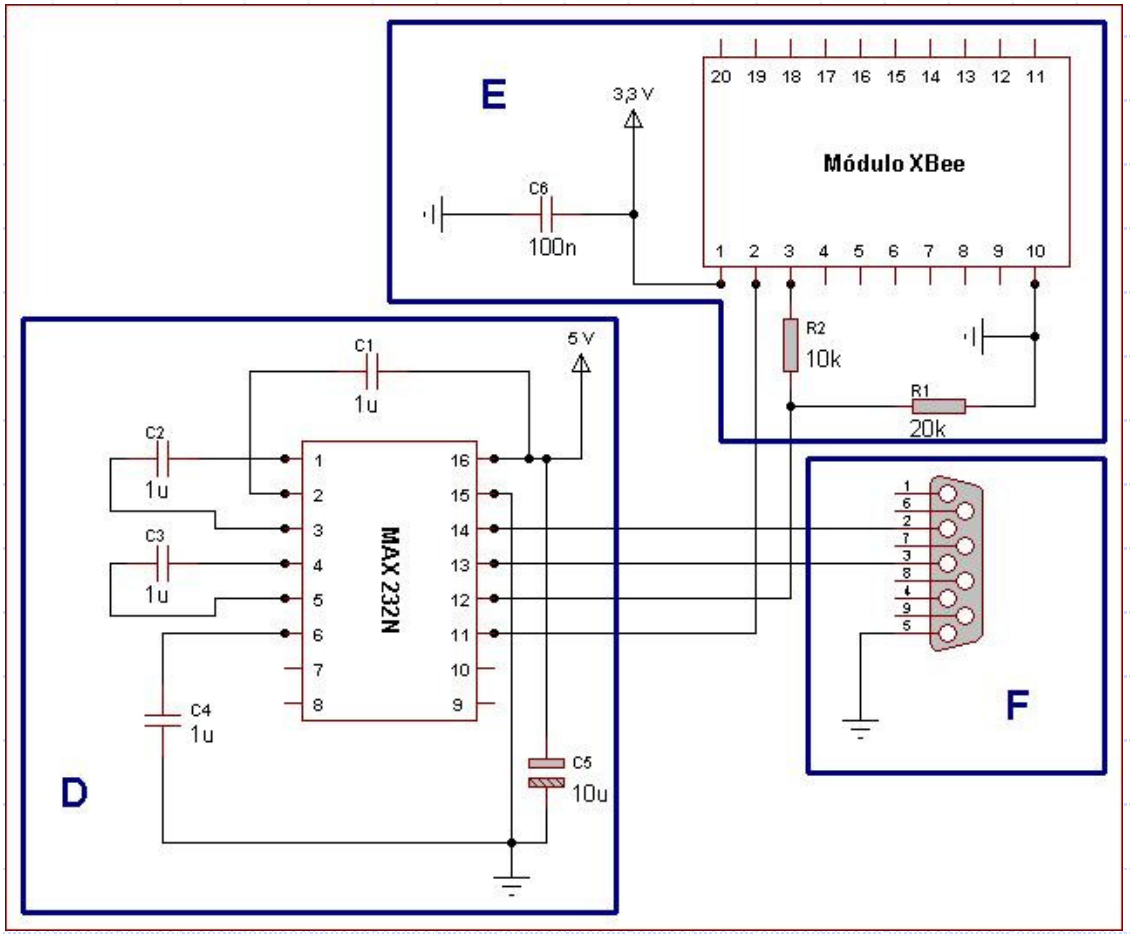
Especificação

- Circuito dos terminais de captura de dados:



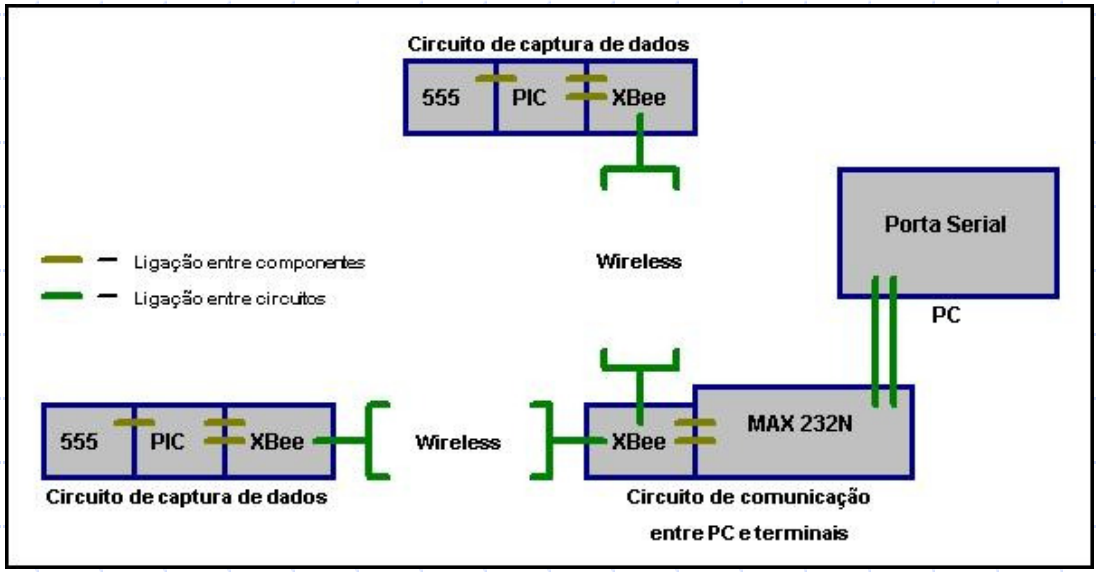
Especificação

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



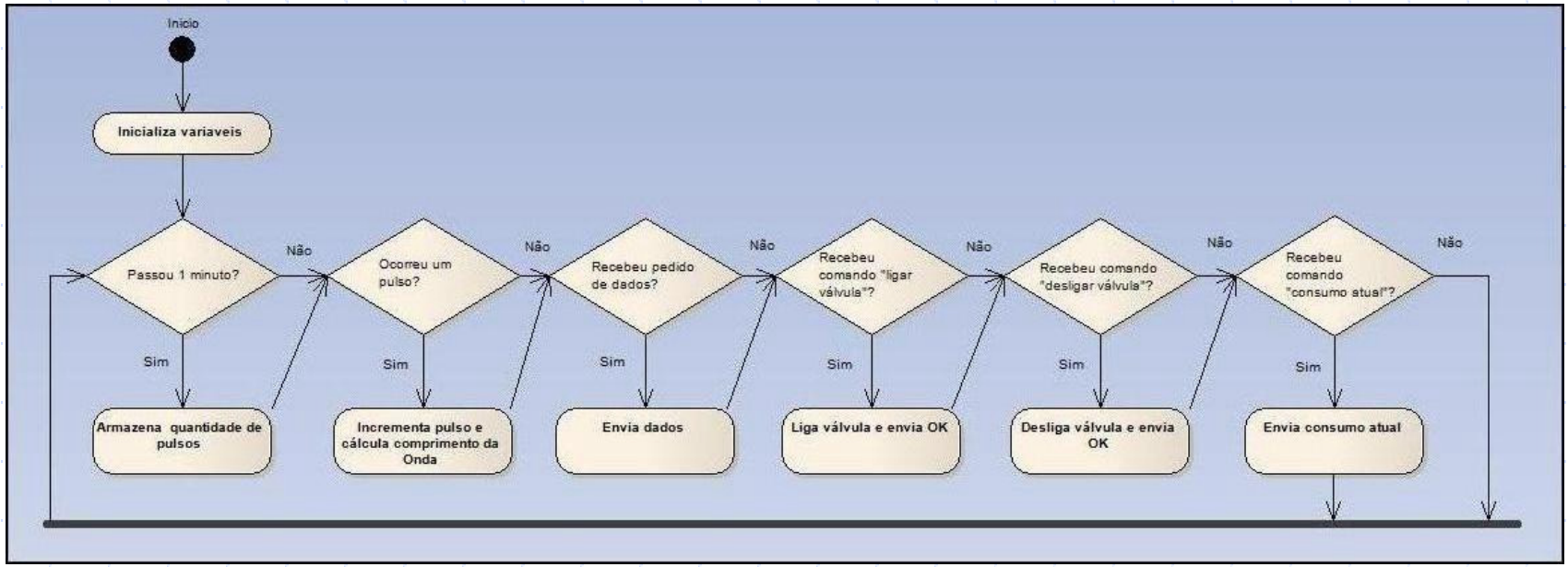
Especificação

- Distribuição dos circuitos:



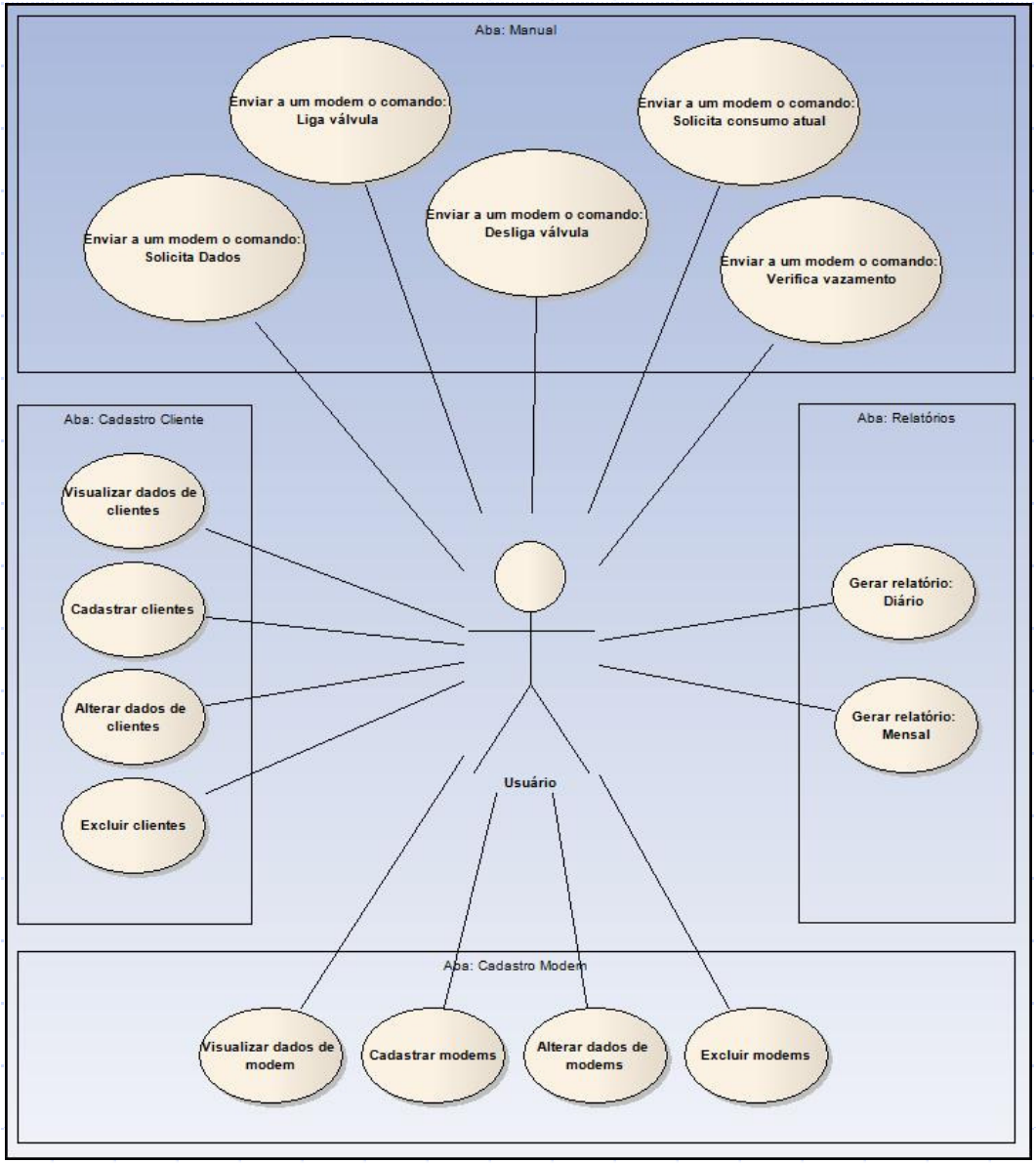
Especificação

- Diagrama de atividades para o software embarcado :



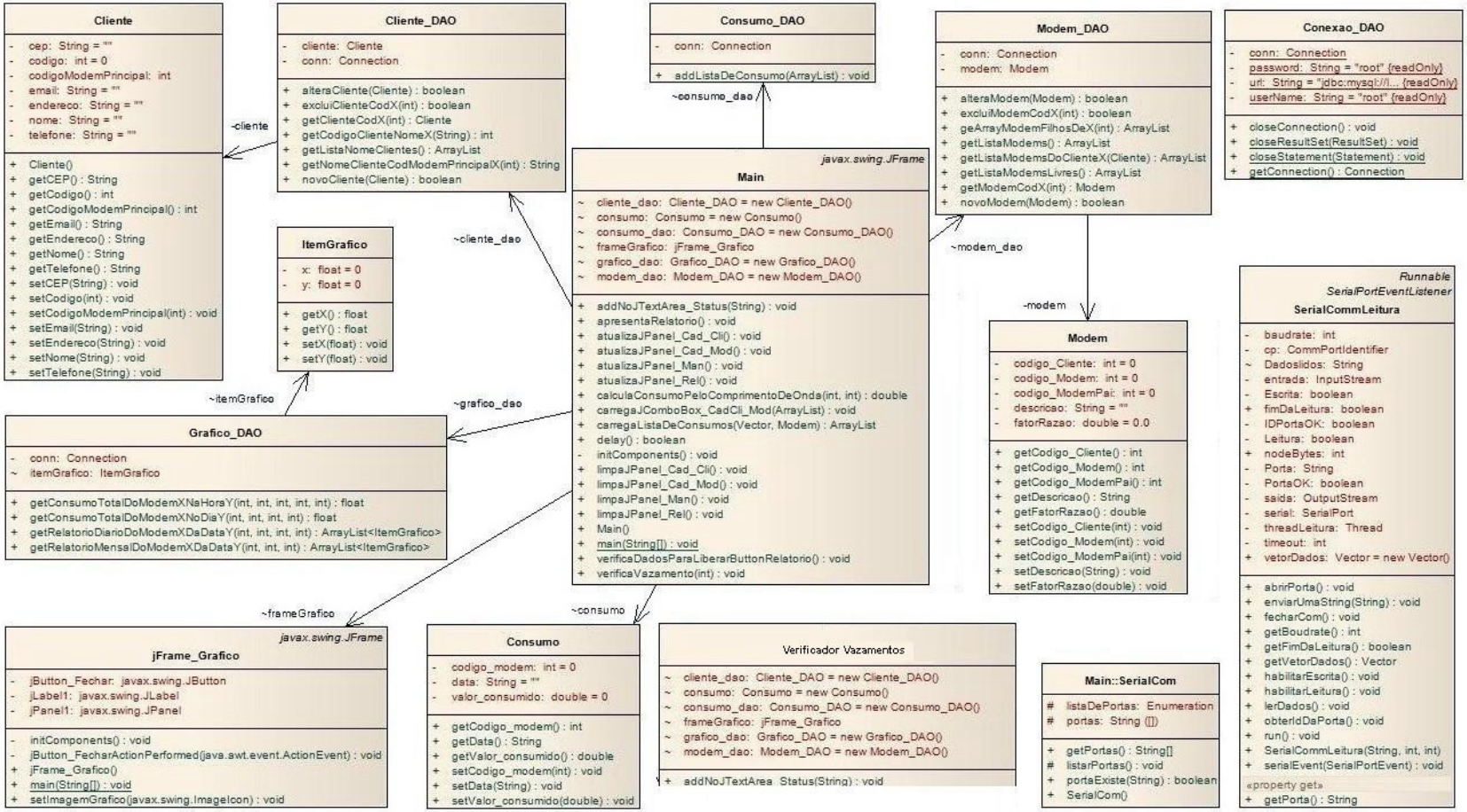
Especificação

- Caso de uso:



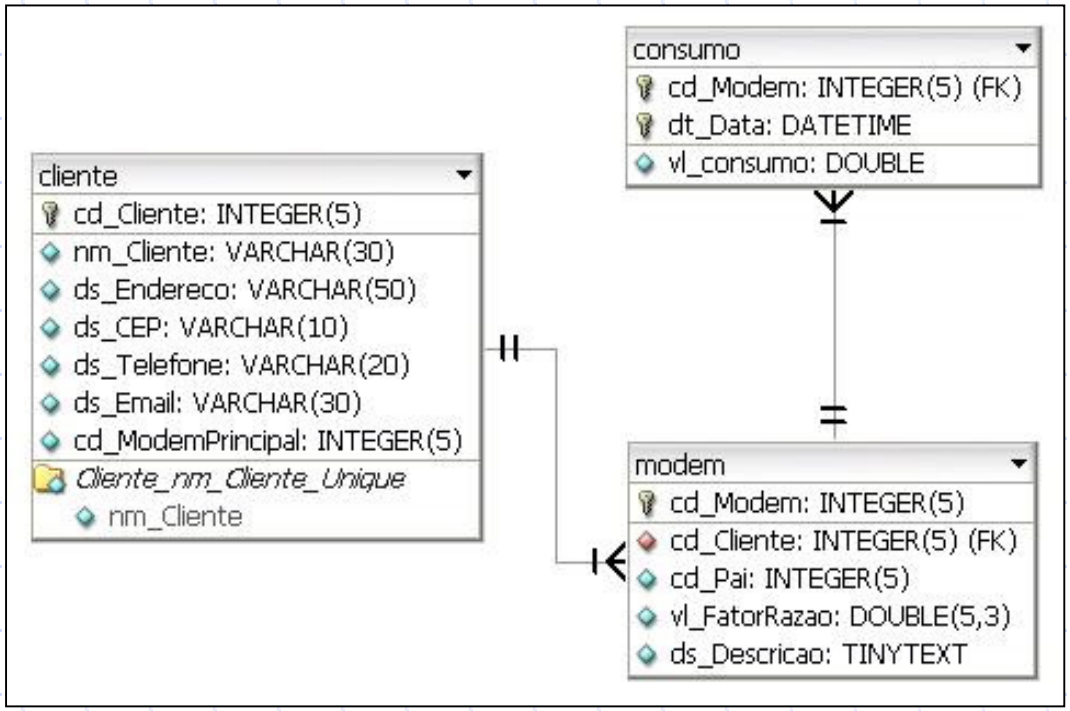
Especificação

Diagrama de classe para o software do PC:



Especificação

- Diagrama entidade relacionamento da base de dados:

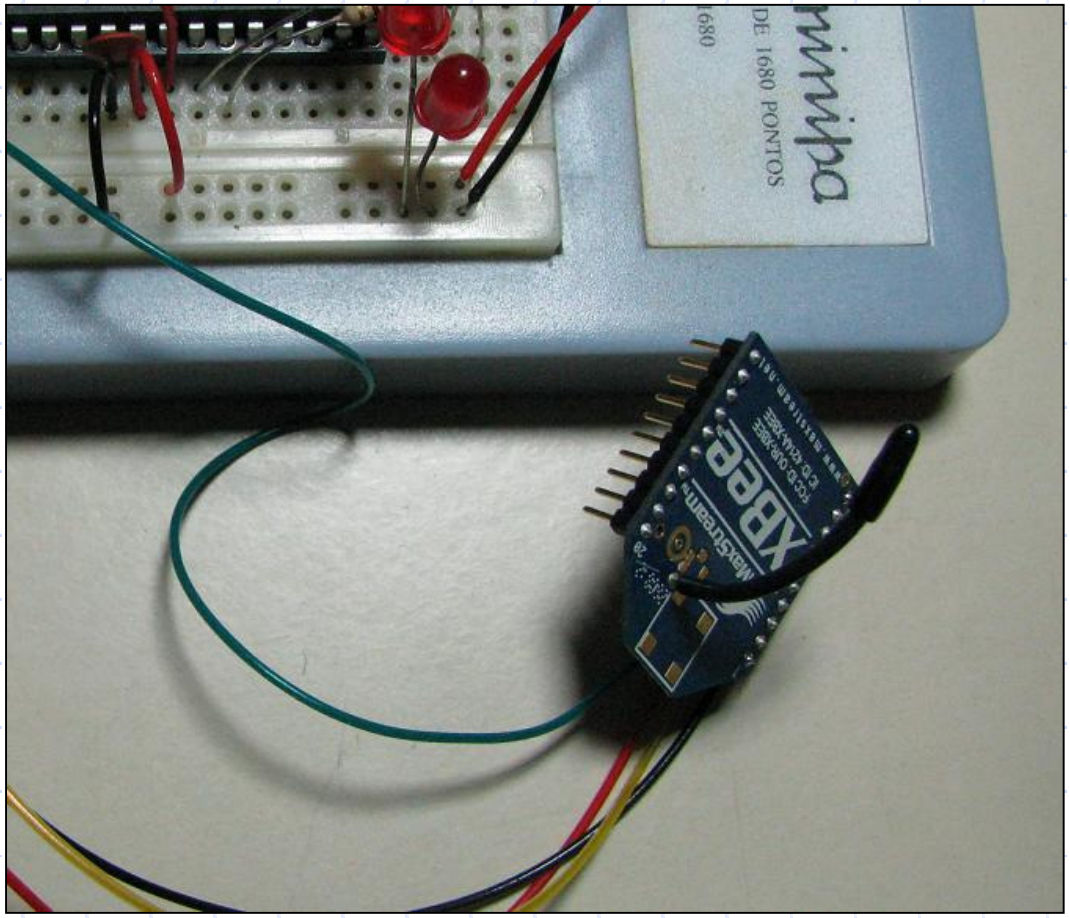


Implementação

- Técnicas e ferramentas utilizadas:
  - ✓ Módulo Xbee
  - ✓ Microcontrolador PIC 18F452
  - ✓ Ferramenta Proteus 7

Implementação

- Módulo XBee



Você está em: Desenvolvimento do trabalho → Implementação → Técnicas e ferramentas Utilizadas -> Módulo XBee

Implementação

Módulo XBee

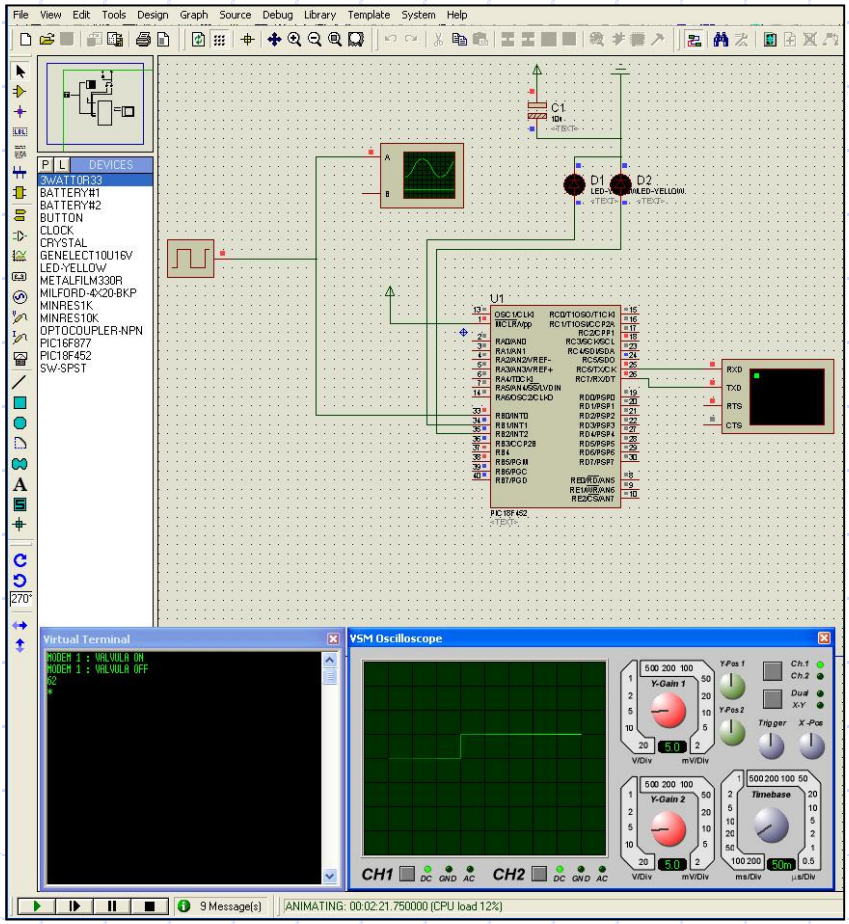
- ✓ Alcance em ambientes internos : 30 m
- ✓ Alcance em ambientes externos/aberto : 100 m
- ✓ Frequência de operação: 2,4 Ghz
- ✓ Taxa de transferência: até 250 Kbps
- ✓ Endereçamento: mais de 65.000 endereços disponíveis por canal
- ✓ Quantidade de canais: 16
- ✓ Tensão alimentação: 2,8 V à 3,4 V
- ✓ Criptografia: 128 bit AES
- ✓ Corrente TX/RX: 45mA / 50mA – 3,3 V

Implementação

- Funcionalidades do PIC 18F452 utilizadas no projeto:
  - ✓ Memória RAM de 1536 bytes, utilizada para armazenar os dados de consumo
  - ✓ Interrupção do Timer0
  - ✓ Interrupção do Timer1
  - ✓ Interrupção Externa pela porta RB0
  - ✓ Interrupção pela entrada de dados na porta serial
  - ✓ Comunicação USART via RS-232

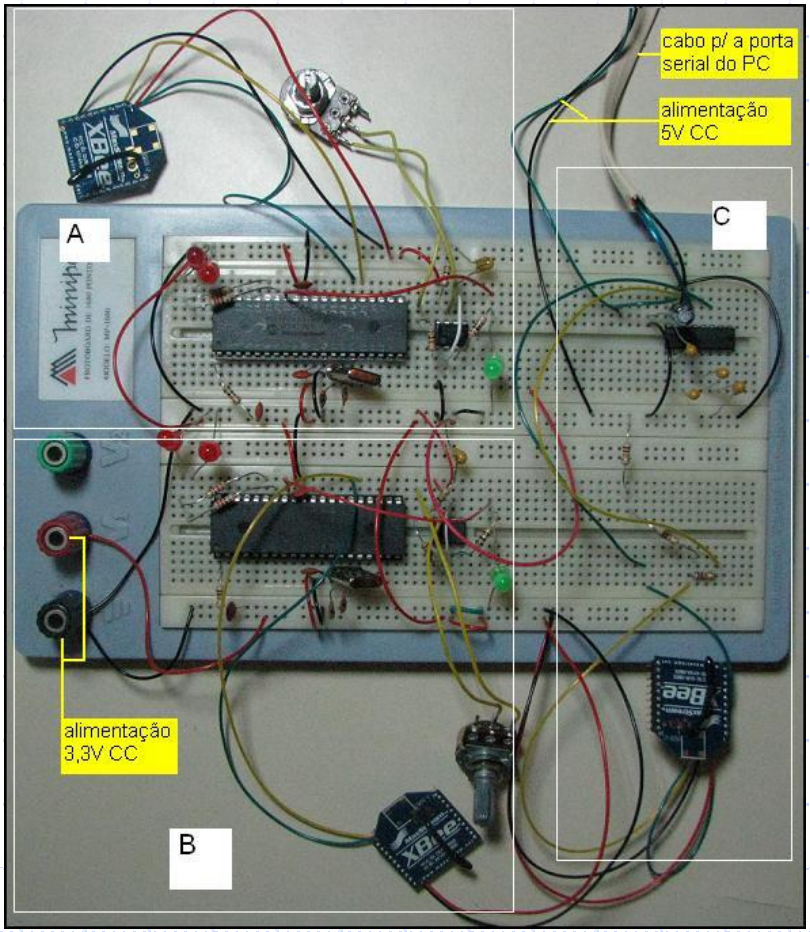
Implementação

- Ferramenta Proteus 7 simulando circuito:



Implementação

- Circuito do protótipo montado em um protoboard:



Implementação

- Códigos Fonte:
  - ✓ Códigos fonte da aplicação embarcada
  - ✓ Códigos fonte da aplicação Java
  - ✓ Script SQL para criação da base de dados

Implementação

Método interrupçãoExterna(void):

```
#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
      }
}
```

Implementação

- Método armazenaPulsos (void):

```
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
}
```

Implementação

Método recebeDados (void):

```
#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;
      }
      }
    }
}
```

Implementação

Método carregaListaDeConsumos:

```
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;
}
```

Implementação

- Pseudocódigo representando como são verificados os vazamentos:

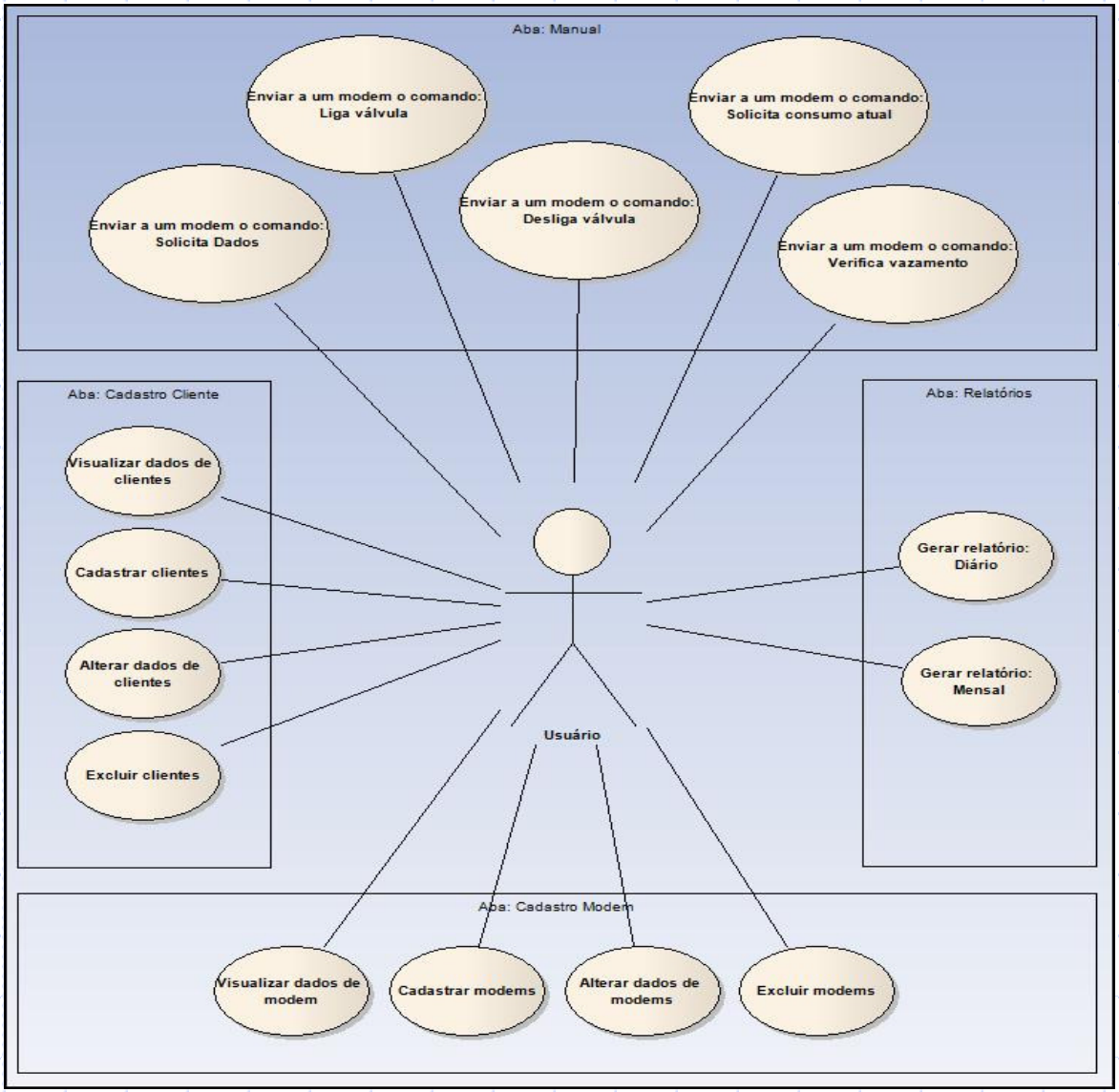
```
ArrayList filhos;
double consumoPai = 0;
double consumoFilhos = 0;

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
            }
        }
    }
```

Implementação

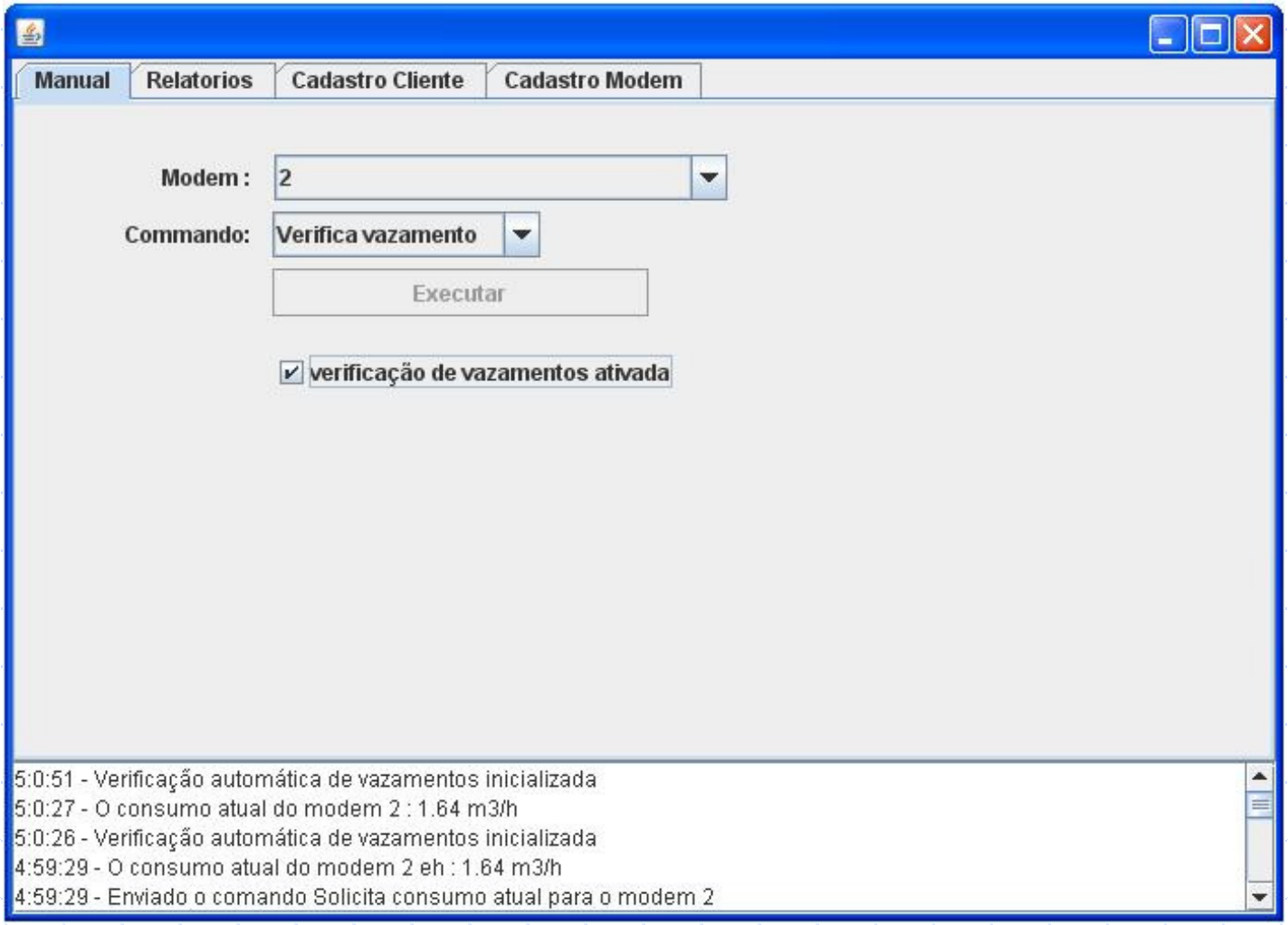
- *Script* SQL para a implementação da base de dados:

```
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;
```

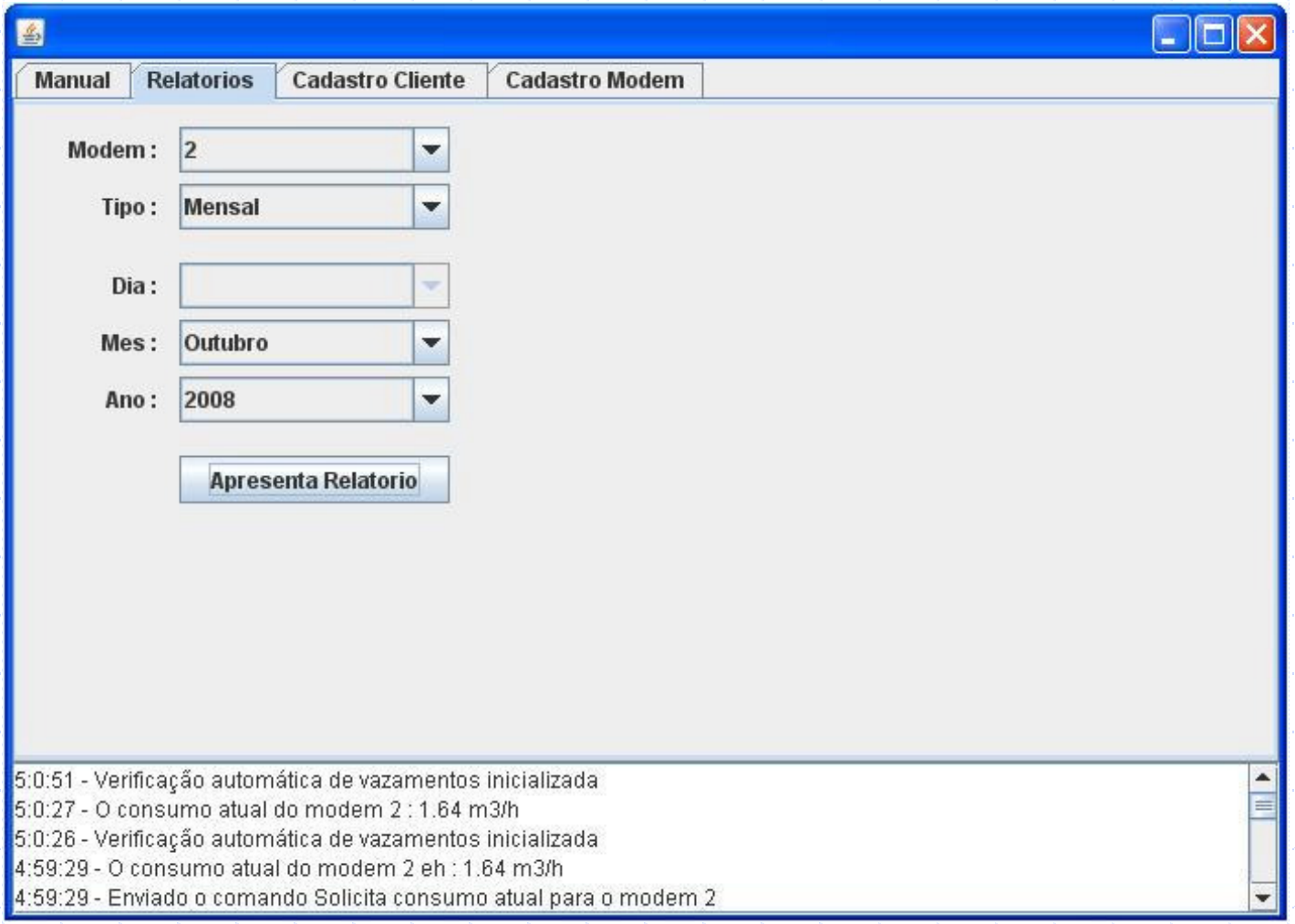
Operacionalidade



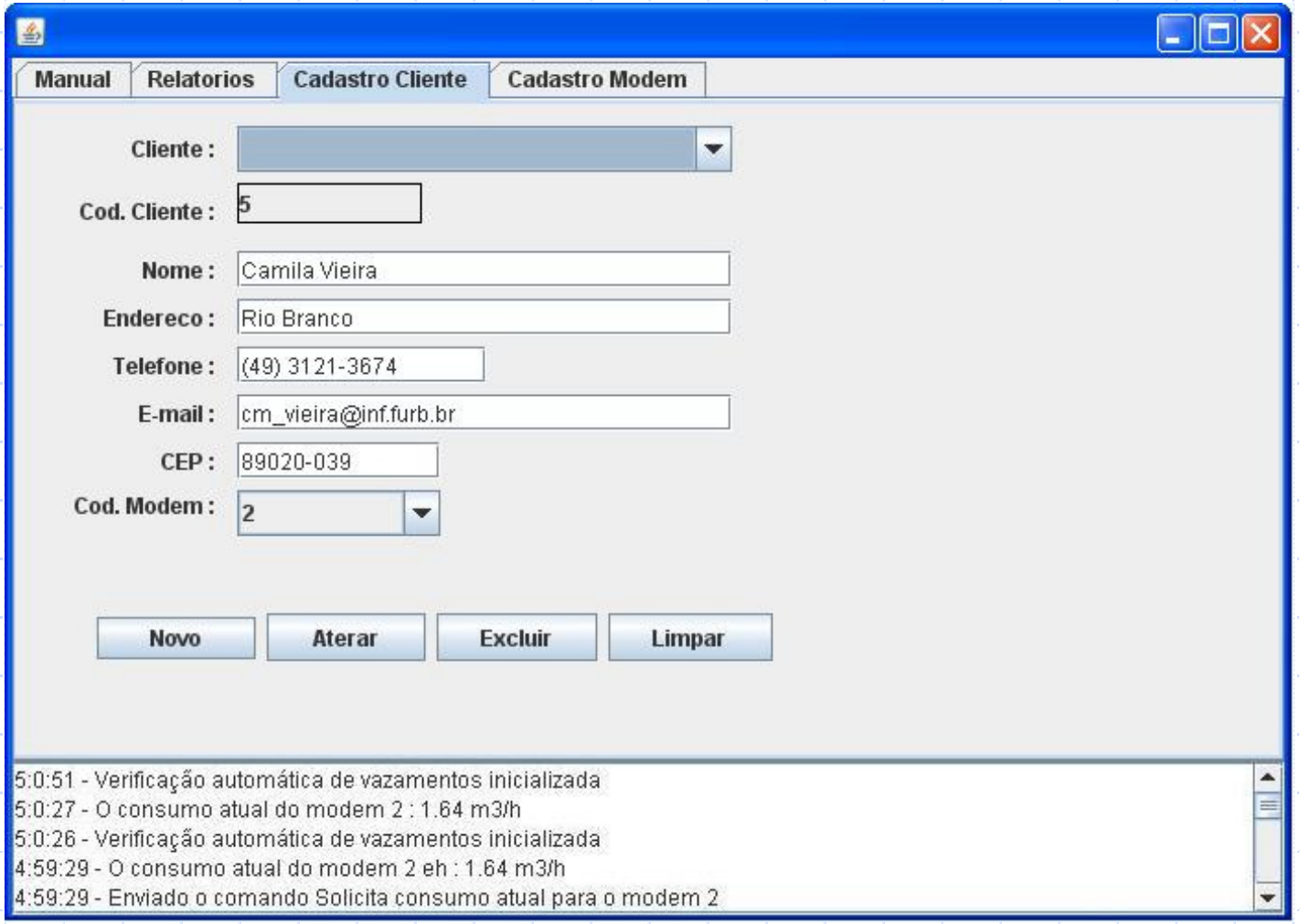
Operacionalidade



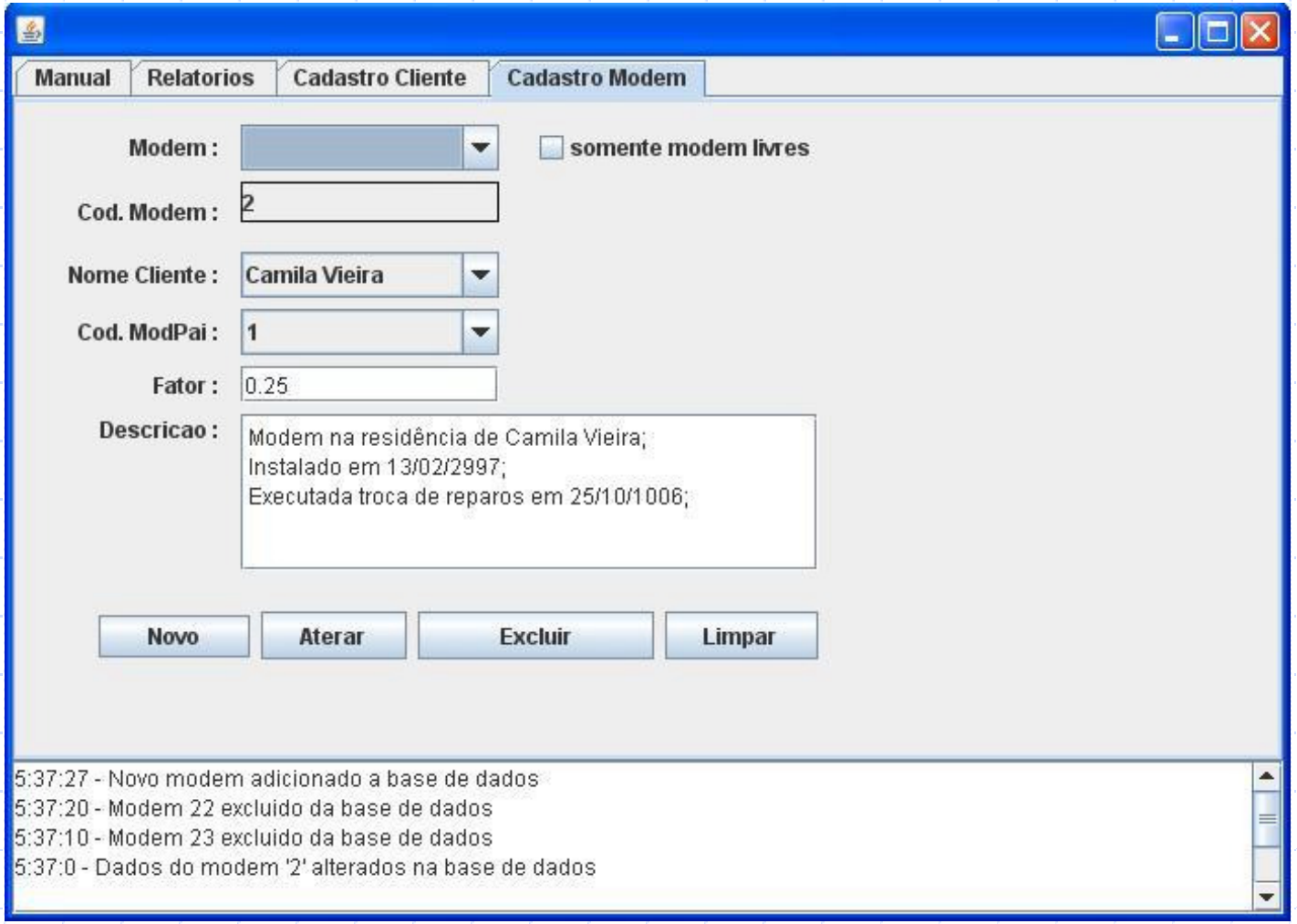
Operacionalidade



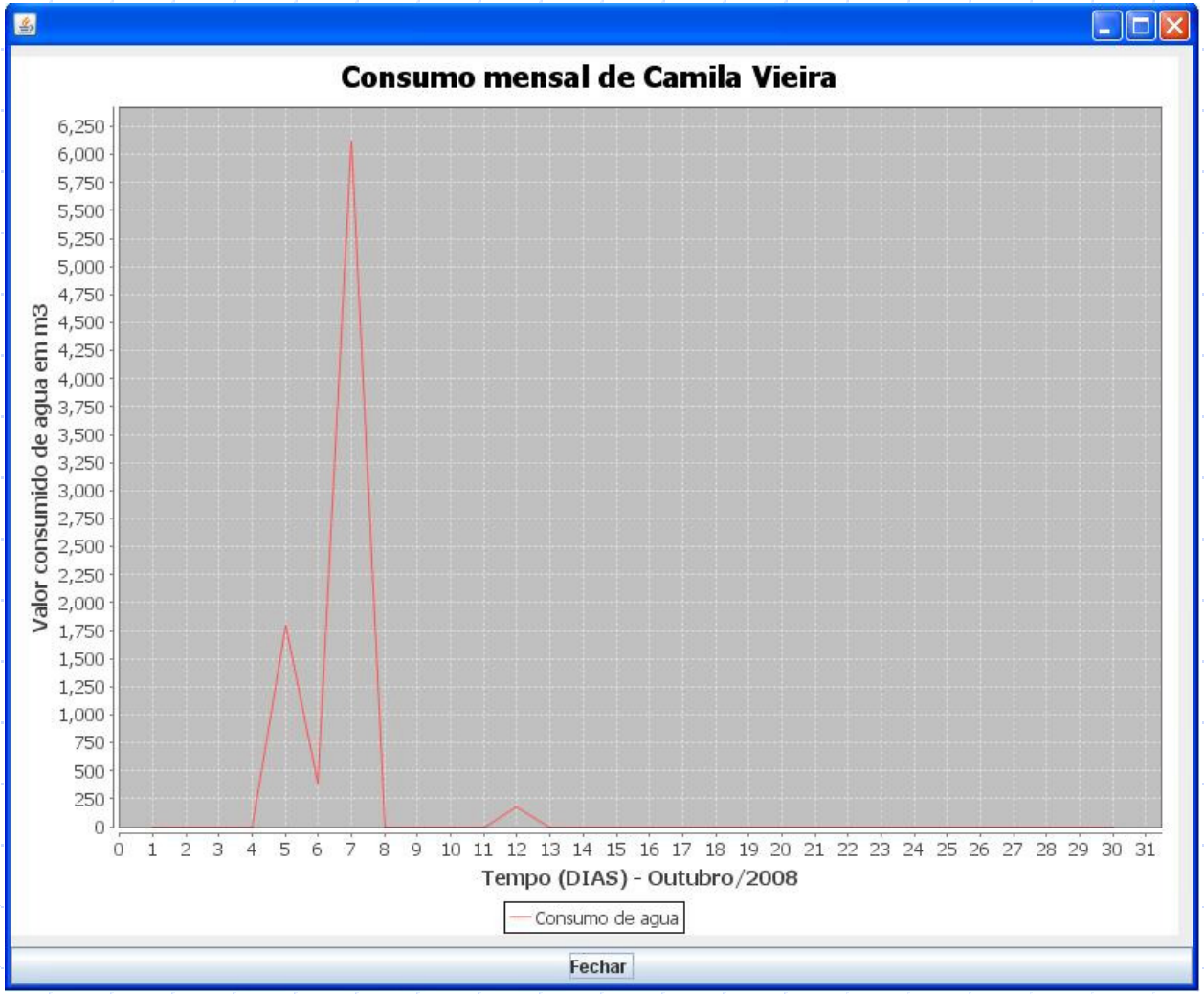
Operacionalidade



Operacionalidade



Operacionalidade



Você está em: Desenvolvimento do trabalho → Operacionalidade

Resultados e Discussão

	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

Resultados e Discussão

- Prós em relação aos trabalhos correlatos
  - ✓ Uso de hidrômetros digitais/convencionais convertidos
  - ✓ Baixo custo
  - ✓ Wireless
- Contras em relação aos trabalhos correlatos
  - ✓ Poucos terminais de consumo

Conclusão

- ✓ Objetivos foram atingidos
- ✓ Conhecimentos na API RXTX, tecnologia ZigBee, banco de dados, API JFreeChart, comunicação entre componentes eletrônicos, linguagem C para microcontroladores
- ✓ Limitação em 10 terminais de consumo
- ✓ Eficiência de sistemas de telemetria no combate ao desperdício de água tratada
- ✓ Se os dados forem totalizados por hora o buffer permite solicitar os dados a cada 30 dias.

Extensões

- ✓ implementar o uso de uma DIP switch para setar o ID no microcontrolador;
- ✓ ajustar o software embarcado e o software do PC para que possa atender mais do que apenas 10 terminais finais;
- ✓utilizar hidrômetros reais (digitais ou convertidos para digitais);
- ✓desenvolver um circuito para efetuar comunicação pela porta USB do PC;
- ✓efetuar testes em uma rede de distribuição;
- ✓desenvolver um software para PDA ou celular permitindo solicitar os dados através dos mesmos;
- ✓desenvolver um circuito gerador de eletricidade através do movimento da água para alimentar individualmente cada terminal final.

FIM

Obrigado

Benno Martim Schubert