Nathalie Sousa Cardone

Tarifador de Chamadas Telefônicas em Tempo Real

São José – SC
Agosto / 2008

Nathalie Sousa Cardone

Tarifador de Chamadas Telefônicas em Tempo Real

Monografia apresentada à Coordenação do Curso Superior de Tecnologia em Sistemas de Telecomunicações do Instituto Federal de Educação, Ciência e Tecnologia de Santa Catarina para a obtenção do diploma de Tecnólogo em Sistemas de Telecomunicações.

Orientador:
Prof. Fábio Alexandre de Souza

Co-orientador:
Prof. Eraldo Silveira da Silva

CURSO SUPERIOR DE TECNOLOGIA EM SISTEMAS DE TELECOMUNICAÇÕES
INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DE SANTA CATARINA

São José – SC
Agosto / 2008

Monografia sob o título “Tarifador de Chamadas Telefônicas em Tempo Real”, defendida por Nathalie Sousa Cardone e aprovada em 04 de setembro de 2008, em São José, Santa Catarina, pela banca examinadora assim constituída:

_____

Prof. Fábio Alexandre de Souza
Orientador

_____

Prof. Eraldo Silveira da Silva
Co orientador

_____

Prof. Marcos Moecke
IF-SC – São José - SC

_____

Eng. Ricardo Cláudio Voigt
Digitro Tecnologia

A felicidade às vezes é uma bênção,
mas geralmente é uma conquista.
Paulo Coelho

Agradecimentos

Ao Deus Trindade: Agradeço ao Espírito Santo por ter me iluminado nos momentos de incertezas, por ser meu companheiro ao longo destes anos de estudo. Ao Deus filho que sempre intercede por mim, e ao Deus pai por atender minhas orações.

Aos meus familiares: Meus pais sempre presentes na minha vida que com amor me educaram, e confiaram em mim, me apoiando e incentivando a estudar. Kassia minha irmã, fonte de inspiração, que está sempre pronta a ajudar em tudo que eu precisar. Bruno meu noivo, que soube ser paciente nos meus momentos de estudo, e principalmente por ter me apresentado ao IFSC, pois sem o seu apoio talvez não estivesse me formando neste momento. Bruno você é meu alicerce.

Aos Mestres: Agradeço aos professores, que me formaram uma profissional, que insistiram contra minhas teimosias. Agradeço principalmente ao professor Fábio Alexandre de Souza, meu orientador do projeto final de curso, que foi um excelente professor e me ajudou sempre que precisei me ensinou a ter mais confiança em mim mesma.

Aos colegas e amigos: Agradeço pelo companheirismo dos meus colegas: Cleimar, Diego, Fabrício e Patrick. Que são mais que colegas, são verdadeiros amigos. A todos os outros colegas que estudaram comigo, pelo apoio. Desejo sucesso a todos.

Ao IF-SC e a Prefeitura: Agradeço ao IF-SC por ter me acolhido e ajudado quando necessário, e a prefeitura de Garopaba por ter dado gratuitamente o transporte escolar por todos estes anos.

Resumo

Este trabalho apresenta uma solução para o controle dos gastos com telefonia fixa dos usuários brasileiros. O tarifador é uma arquitetura composta por circuitos eletrônicos e um programa, que juntos realizam a tarifação da chamada em curso. A tarifação é baseada nas regras de tarifação do plano básico de telefonia, determinadas pela Agência Nacional de Telecomunicações - ANATEL.

Hoje o usuário de telefonia fixa não tem controle sobre a conta telefônica. O registro e custo das chamadas são realizados pela operadora, que reúne os dados e os envia mensalmente ao assinante, na conta telefônica deste.

O uso do tarifador possibilitará aos assinantes um controle maior dos gastos com telefonia, uma vez que o custo é exibido e atualizado durante a chamada.

Palavras chave: Tarifador, Java, normas telefônicas, ANATEL, tempo real.

Abstract

This work presents a solution to control the costs of telephone calls in the brazilian PSTN (Public Switched Telephone Network). The Call Billing System is a hardware and software implementation which shows the cost of ongoing calls based on the ANATEL (Agência Nacional de Telecomunicações - BrazilianNational Telecommunications Agency) rules. Nowadays PSTN subscribers do not have control of their originated call costs. This information is controlled by the local carrier and will be charged on telephone bill. By using the proposed Call Billing System, subscribers can control their telephone operating expense, once the cost is shown and updated during the call.

Keywords: Billing system, telecommunication rules, DTMF, ANATEL, JAVA, Real-Time.

Sumário

Lista de Figuras

Lista de Tabelas

1 Introdução

Este projeto visa desenvolver um circuito eletrônico e um programa associado que permita ao usuário de telefonia saber o custo das ligações em curso. O programa deverá atender as regras de tarifação impostos pela Agência Nacional de Telecomunicações, que estão dispostas no documento (ANATEL, 2004). Para tanto, serão utilizados os planos básicos de tarifas das principais prestadoras. Este projeto possibilitará o aprimoramento dos conhecimentos adquiridos no decorrer do Curso Superior de Sistemas de Telecomunicações, além de proporcionar uma experiência de projeto e desenvolvimento, unindo conhecimentos teóricos e práticos.

O programa deverá apresentar o custo da ligação em curso, segundo as regras da Anatel. Ele utilizará como base de dados do custo das chamadas somente o plano básico das principais operadoras de telefonia. Será capaz também de tarifar qualquer chamada realizada para qualquer destino dentro de Santa Catarina, e para qualquer país do grupo[1] escolhido para estudo, sendo o destino um acesso do sistema telefônico fisico comutado. Deste modo serão implementadas todas as tarifas do plano básico, pois Santa Catarina possui todos os degraus tarifários previstos no regulamento de tarifação da Anatel. Será desafiador desenvolver um programa que atenda todas as regras da Anatel, pois há vários requisitos a serem atendidos, como por exemplo, a troca de tarifa numa ligação em curso.

O circuito realizará a interface do telefone com o computador, que estará executando o programa, capturando os dígitos do telefone de destino, para enviar ao tarifador. Ele também irá identificar o sinal de inversão de polaridade[2].

[1] Para tarifação de chamadas internacionais, os países são separados em sete grupos.
[2] Sinal enviado pela central publica, que indica o atendimento da chamada.

2 Fundamentação Teórica

Este capítulo descreve de forma sucinta a pesquisa realizada no início do desenvolvimento deste trabalho.

Subdividido em sete itens, este capitulo apresenta os conhecimentos básicos para o completo entendimento do desenvolver do projeto. É apresentado um resumo dos principais critérios de tarifação, breve explicação dos componentes utilizados no circuito, conceitos da linguagem de programação utilizada e dados das sinalizações e temporizações de telefonia fixa.

2.1 Regulamento de Tarifação

A regulamentação da tarifação das chamadas telefônicas no Brasil, seguem o Anexo IV da Resolução 424 de 6 de dezembro de 2004.

Para entender este documento é necessário o conhecimento de alguns termos utilizados e seus significados, portanto seguem abaixo as definições:

2.1.1 Definições básicas

1) Um número telefônico completo é composto por 12 dígitos, ex; 55 48 3254 1234, representados da seguinte maneira n12 n11 n10... n1.

n12 e n11 são o código do pais;

n10 e n9 representam a área de numeração;

n8, n7, n6 e n5 formam o prefixo que é o número do telefone relacionado com sua central;

n4, n3, n2 e n1 são os dígitos que identificam o número do telefone na sua central;

2) Área de numeração: área geográfica do território nacional na qual os acessos

telefônicos são identificados pelo código nacional.

3) Área tarifária: área geograficamente contínua constituída por um conjunto de áreas locais agrupadas segundo critérios socio-geo-econômicos e contidas em uma mesma área de numeração.

4) Área local: área geograficamente continua de prestação de serviço, definida pela agência segundo critérios técnicos e econômicos.

5) Centro de área de tarifação: localidade definida pela agência segundo critérios técnicos e econômicos, utilizada como referência na determinação da distância geodésica entre áreas de tarifação;

6) Distância geodésica: distância entre dois pontos no mapa terrestre, em linha reta, levando em consideração a curvatura da terra.

7) Degrau tarifário: intervalo de distâncias geodésicas entre centro de áreas de tarifação, para o qual são atribuídos valores tarifários específicos.

8) Duração da chamada: período compreendido entre a ocorrência do sinal de atendimento e o sinal de desconexão.

9) Modulação horária: segmentação das vinte e quatro horas do dia, considerada a sua natureza de dia útil, sábado, domingo ou feriado nacional, em intervalos de uma ou mais horas, aos quais são atribuídos valores tarifários específicos. As chamadas que se estenderem além de um horário de tarifação devem ser tarifadas em função do tempo utilizado em cada um dos horários, observados as respectivas tarifas e a duração total da chamada.

10) Plano básico: plano de serviço de oferta obrigatória e não discriminatória a todos os usuários.

11) Região fronteiriça: é aquela compreendida por localidades situadas no Brasil e em país que com ele faça fronteira, distântes entre si até 50 Km, em distância geodésica.

2.1.2 Tarifação

A tarifação é o processo de medição da utilização do STFC (sistema telefônico fixo comutado) para atribuição de valor, em moeda nacional a ser pago em contrapartida à prestação do serviço. A tarifação é calculada em função do tempo de utilização.

Dos critérios gerais relativos às chamadas originadas de acesso individual:

•Unidade de tempo de tarifação: 6 segundos.

•Tempo de tarifação mínima: 30 segundos.

•Duração de chamadas faturáveis: maior que 3 segundos.

Segundo a noma uma chamada somente pode ser tarifada se a sua duração for superior a 3 segundos. Caso seja, o tempo mínimo de tarifação é de 30 segundos, até os 30 primeiros segundos a chamada terá um valor fixo, caso dure mais que 30 segundos, a partir daí o valor será acrescido a cada 6 segundos do valor do minuto dividido por 10.

Chamadas sucessivas com duração inferior a 30 segundos, efetuadas entre os mesmos acessos de origem e de destino, e quando o intervalo entre o final de uma ligação e o início da seguinte for inferior a 120 segundos são tarifadas como uma única ligação, cuja duração é igual ao somatório das durações das chamadas sucessivas ou igual ao tempo de tarifação mínima (30 segundos).

É obrigatório o truncamento da fração do centavo na apresentação do valor final de qualquer registro individual constante da fatura.

Para fins de tarifação, a duração da chamada é expressa em horas, minutos e segundos, no formato hh:mm:ss, e em valores múltiplos da unidade de tempo de tarifação, admitindo-se o arredondamento para cima da duração real da chamada.

2.1.3 Definição dos degraus tarifários no Serviço Móvel Pessoal

VC[3] - 1- degrau tarifário entre um acesso do STFC e um do SMP (serviço móvel pessoal) : As chamadas são tarifadas como VC - 1 quando ambos os caracteres da área de numeração [n10 e n9] da origem e destino são iguais.

VC - 2- degrau tarifário entre um acesso do STFC e um do SMP: As chamadas são tarifadas como VC - 2 quando o primeiro algarismo da área de numeração [n10] da origem e destino são iguais.

VC - 3- degrau tarifário entre um acesso do STFC e um do SMP: As chamadas são tarifadas como VC- 3 quando ambos caracteres da área de numeração [n10 e n9] da origem e destino são diferentes.

[3] Valor da comunicação.

2.1.4 Definição dos degraus tarifários no Serviço Telefônico Físico Comutado

Em função da distância geodésica entre os centros das áreas tarifárias onde estão situadas as localidades de origem e destino, a chamada LDN (longa distância nacional) é classificada em degraus tarifários, a saber:

•I – degrau 1 (D1) compreendendo distâncias até 50 km;

•II – degrau 2 (D2) compreendendo distâncias maiores que 50 km e até 100 km;

•III – degrau 3 (D3) compreendendo distâncias maiores que 100 km e até 300 km; e

•IV – degrau 4 (D4) compreendendo distâncias maiores que 300 km.

O cálculo da distância geodésica entre duas localidades, expressa em quilômetros, obedece à equação:

$$D = 111{,}18 \arccos (X1.X2 + Y1.Y2) \quad (1)$$

onde:

$$X1 = \text{sen } K1; \quad (2)$$

$$X2 = \text{sen } K2; \quad (3)$$

$$Y1 = \cos K1.\cos K2; \quad (4)$$

$$Y2 = \cos (L2 - L1) = \text{sen } L1.\text{sen } L2 + \cos L1.\cos L2; \quad (5)$$

onde: K1 é a latitude da Localidade 1 em graus; L1 é a longitude da Localidade 1 em graus; K2 é a latitude da Localidade 2 em graus; e L2 é a longitude da Localidade 2 em graus.

Para efeito de cálculo da distância geodésica, a latitude de uma localidade situada no hemisfério norte é expressa com sinal negativo.

2.1.5 Modalidade de chamadas do plano básico

Nesta seção são descritas apenas os tipos de chamadas originadas do um acesso do STFC.

a) Estão compreendidas na modalidade local (Local) as chamadas:

I - realizadas entre acessos do STFC situados na mesma área local;

II - realizadas entre acessos do STFC situados em localidades que compõem uma área com continuidade urbana, mesmo que localizadas em áreas locais distintas;

III - originadas em acesso do STFC e destinadas a acesso do Serviço Móvel Pessoal (SMP) ou Serviço Móvel Especializado (SME), cuja área de registro é idêntica à área de numeração do acesso de origem;

b) Estão compreendidas na modalidade longa distância nacional (LDN) as chamadas:

I- realizadas entre acessos do STFC situados em áreas locais distintas, exceto aquelas entre localidades que têm tratamento local, conforme previsto no Regulamento sobre Áreas Locais para o STFC;

II- destinadas a acesso do SMP ou do SME e originadas em acesso do SMP ou do SME localizados em área de registro distinta da área de registro do acesso de destino.

c) Estão compreendidas na modalidade longa distância internacional (LDI) as chamadas:

I - originadas em acessos do STFC, SMP ou SME e destinadas a acessos localizados no exterior;

d) Chamadas da modalidade especial recebem tratamento diferenciado, descrito a seguir:

A tarifação das chamadas originadas de acessos do STFC e destinadas aos serviços Hora Certa (130) e Despertador Automático (134) é definida de acordo com os termos

do Ato n.º 50.660, de 1º de junho de 2005, da Superintendência de Serviços Públicos da Anatel.

A tarifação das chamadas destinadas aos códigos não geográficos 0300, 0500 e 0900, bem como as chamadas destinadas aos serviços de utilidade pública e de apoio ao STFC é definida em regulamentação específica.

A tarifação das chamadas destinadas ao código não geográfico 0800 obedece aos critérios estabelecidos neste regulamento, observando-se a área local e a área de numeração nas quais está localizado o acesso identificado pelo código 0800.

2.1.6 Modulação horária

Nesta seção será descrito a modulação horária do plano básico determinado pela normada Anatel, que deve ser seguida pelas operadoras em seu plano básico.

Somente será descrito as modulações de chamadas originadas de acesso telefônico físico.

a) Chamadas locais

Chamadas locais envolvendo acessos do SMP e SME:

Em função do dia e hora de realização da chamada aplica-se a seguinte modulação horária:

•Horário Normal

Plano Básico: Cobra-se um mínimo de 30 segundos e o tempo de utilização adicional é tarifado a cada seis segundos. Somente são tarifadas as chamadas com duração superior a três segundos.

O horário de tarifa normal é de segunda-feira a sábado, de 7 h às 21 h.

•Horário Reduzido

Plano Básico: Cobra-se um valor por chamada atendida (VCA) , equivalente ao valor de dois minutos, por chamada completada, independentemente do tempo de utilização.

Horário de tarifa reduzida, de segunda-feira a sábado de 0 h às 7 h e das 21 h às 24 h, e aos domingos e feriados nacionais, de 0 h às 24 h.

b) Chamadas LDN

Chamadas LDN entre acessos do STFC

Em função do dia e hora de realização da chamada aplica-se a seguinte modulação horária:

1. Horário de tarifa super-reduzida que corresponde ao horário de 0 h às 6 h;

2. Horário de tarifa reduzida que corresponde ao horário de 6 h às 7 h e de 21 h às 24 h de segunda-feira a sexta-feira; de 6 h às 7 h e de 14 h às 24 h aos sábados; e de 6 h às 24 h aos domingos e feriados nacionais;

3. Horário de tarifa normal que corresponde ao horário de 7 h às 9 h, de 12 h às 14 h e de 18h às 21 h de segunda-feira a sexta-feira; e de 7 h às 14 h aos sábados; e

4. Horário de tarifa diferenciada que corresponde ao horário de 9 h às 12 h e de 14 h às 18 h de segunda-feira a sexta-feira.

c) Chamadas LDI

Chamadas LDI (longa distância internacional)

Em função do dia e hora de realização da chamada aplica-se tarifa reduzida, para cada Grupo de Países de destino, ficando estabelecido que os demais horários sejam horários de tarifa normal:

1. Grupo 1, das 20 horas às 5 horas de segunda-feira a sábado e de 0 h às 24 horas aos domingos e feriados nacionais;

2. Grupo 2, das 20 horas às 5 h de segunda-feira a sábado e de 0 h às 24 h aos domingos e feriados nacionais;

3. Grupo 3, das 20 h às 5 h de segunda-feira a sábado e de 0 h às 24 h aos domingos e feriados nacionais;

4. Grupo 4, das 20 h às 5 h de segunda-feira a sábado e de 0 h às 24 h aos domingos e feriados nacionais;

5. Grupo 5, das 20 h às 5 h de segunda-feira a sábado e de 0 h a 24 h aos domingos e feriados nacionais;

6. Grupo 6, das 20 h às 5 h de segunda-feira a sábado e de 0 h às 24 h aos domingos e feriados nacionais;

7. Grupo 7, de 1 h às 6 h e das 13 h às 17 h de segunda-feira a sábado e de 0 h às 24 h aos domingos e feriados nacionais;

8. Grupo 8, das 20 h às 5 h de segunda-feira a sábado e de 0 h às 24 h aos domingos e feriados nacionais; e

9. Grupo 9, de 1 h às 6 h e das 13 h às 17 h de segunda-feira a sábado e de 0 h às 24 h aos domingos e feriados nacionais.

Na tarifação de chamada originada de acesso do STFC entre localidades situadas em uma região fronteiriça aplicam-se os critérios correspondentes ao degrau 1 do plano básico do STFC na modalidade longa distância nacional.

2.2 Detector de MF

Esta seção é importante para o entendimento do circuito que é descrito em uma seção a seguir, que utiliza um circuito integrado que realiza a decodificação multi-frequencial.

DTMF é a sigla de "Dual Tone MultiFrequential", os tons de duas freqüências utilizados na discagem dos telefones mais modernos. Nos primeiros telefones a discagem era feita através de um "disco" que gerava uma seqüência de pulsos na linha telefônica ("discagem decádica" ou "discagem usando sinalização decádica") . Ao se ocupar a linha, o "laço" ("loop") era fechado e, ao se efetuar a discagem, ocorriam aberturas periódicas deste "laço", tantas vezes quanto o número discado: para a discagem do 1, uma abertura, para a discagem do 2, duas aberturas, e assim sucessivamente até o 0 (zero) que, na verdade, significava 10 aberturas. Com o advento dos telefones com teclado, das centrais telefônicas mais modernas e com a disseminação dos filtros, passou-se a utilizar a sinalização multifrequêncial, uma combinação de tons para discagem.

Hz	1209	1336	1477	1633
697	1	2	3	A
770	4	5	6	B
852	7	8	9	C
941	*	0	#	D

Tabela 1 - Frequencias dos tons e suas combinações

Na Tabela 1 são mostradas as freqüências "altas" na linha superior e as baixas na coluna mais à esquerda. No centro estão os números do teclado. Nos teclados dos telefones são mostrados apenas os números de 1 até 0 e os caracteres "*" e "#". A

freqüência de 1633 Hz (e conseqüentemente os algarismos "A", "B", "C" e "D") é utilizada apenas internamente entre equipamentos de teste e medida.

Na central o sinal elétrico é constantemente analisado para detectar a presença simultânea de uma das freqüências baixas e uma das freqüências altas, quando então é identificado pela central o algarismo correspondente ao cruzamento destas duas freqüências.

A escolha destas freqüências se deve principalmente pela baixa probabilidade de se produzir estas combinações de freqüências com a voz humana.

A Tabela 2 apresenta os dígitos e seus correspondentes valores em binário, o decodificador irá retornar na sua saída a palavra binária correspondente ao dígito inserido.

Dígito	1	2	3	4	5	6	7	8	9	0
Binário	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010

Tabela 2 -Dígitos e seus correspondentes em binário

O detector multi frequencial é formado por tres filtros passa banda e dois circuitos decodificadores digitais para converter o tom MF em um sinal digital, ao realizar as filtragens é possível determinar qual frequecia alta e qual freqüência baixa, foi utilizada para formar o tom recebido, e assim decodificar na palavra binária correspondente.

2.3 Foto-Acoplador

Um acoplador óptico consiste em um circuito integrado de 6 pinos que por dentro contém um diodo emissor de luz que consome em media 30mA e 1.15V de tensão.

Ao conduzir o diodo emissor aciona com sua luminosidade um transistor que pode ser usado para ligar algum outro dispositivo, como um relé por exemplo. Aí é que está o segredo: os pinos da porta paralela apenas ligarão o diodo emissor de luz do acoplador, sem nenhum contato físico com o resto do circuito, eliminando qualquer risco a porta. A Figura 1 apresenta o diagrama do 4N25, exemplo de acoplador óptico. (BURGOS, 2008).

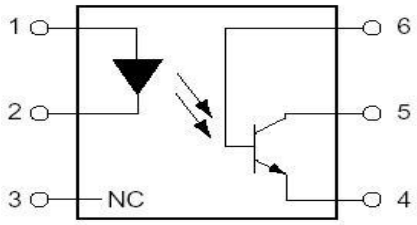

Figura 1 - Foto Acoplador

2.4 Sinalizações

Em uma chamada telefônica existem varias sinalizações entre as centrais telefônicas envolvidas, e entre os acessos, neste projeto especificamente utilizamos o sinal de atendimento para iniciar a tarifação, este sinal é recebido através da inversão de polaridade da linha telefônica do assinante chamador, este sinal é descrito na linha dois da tabela a seguir.

Estas sinalizações são regidas pela norma n° 252 de 20 de Dezembro de 2000, e suas praticas são estão descritas em um documento disponível no sitio da Anatel (ANATEL, 2008).

SINAIS DE LINHA				
N°	SINAL	DURAÇÃO DA EMISSÃO	SENTIDO A B	RECONHECIMENTO
1	OCUPAÇÃO	150 +/- 30	⇒	80 +/- 20 até 375 +/- 75
2	ATENDIMENTO	150 +/- 30	⇐	80 +/- 20 até 375 +/- 75
3	DESLIGAR PRA TRAS	600 +/- 120	⇐	Acima de 375 +/- 75
4	DESLIGAR PRA FRENTE	600 +/- 120	⇒	Acima de 375 +/- 75
5	CONFIRMAÇÃO DE DESCONEXÃO	600 +/- 120	⇐	Acima de 375 +/- 75
6	DESCONEXÃO FORÇADA	600 +/- 120	⇐	Acima de 375 +/- 75
7	BLOQUEIO	PERMANENTE	⇐	Acima de 375 +/- 75
8	TARIFAÇÃO	150 +/- 30	⇐	80 +/- 20 até 375 +/- 75
9	RECHAMADA	150 +/- 30	⇒	80 +/- 20 até 375 +/- 75

Tabela 3 - Sinais de Linha

3 Circuito

A placa AqDT (aquisição de dados para tarifação) é composta por quatro blocos como mostrado na Figura 4 para realizar a conexão entre a linha telefônica e a porta paralela do computador. O bloco DTMF (Detector Multifrequêncial) está conectado em paralelo com a linha telefônica, tendo a função de capturar os dígitos e enviar para o bloco IHC (Interface *hardware* - computador) , que adapta o sinal para a paralela. Os blocos DIP (Detector da Inversão de Polaridade) e DCL (Detector de corrente de loop) trabalham em conjunto e servem para indicar que o telefone foi retirado do gancho e quando o sinal de atendimento, que é representado pelo sinal de inversão de polaridade foi recebido.

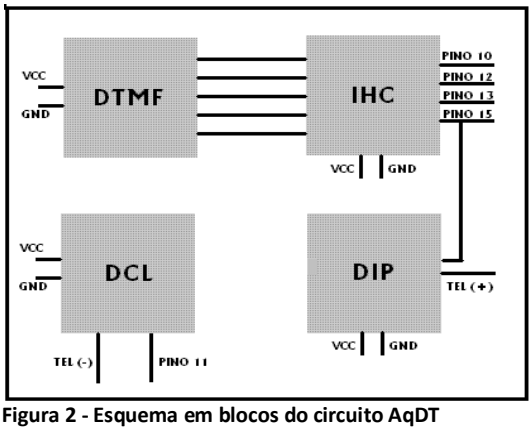

Figura 2 - Esquema em blocos do circuito AqDT

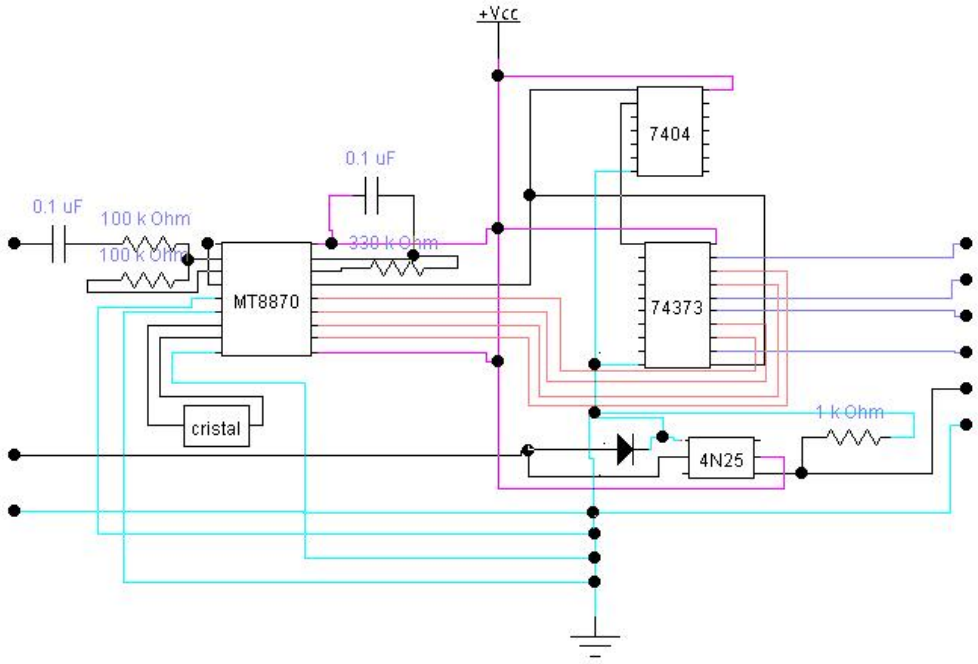

Figura 3 - Esquema eletrônico do circuito AqDT

O bloco DTMF é responsável pela detecção do sinal multifrequêncial, gerado pelo aparelho telefônico, formado por capacitores que fazem a eliminação da tensão contínua da linha telefônica, por resistores que limitam a corrente e por um circuito integrado que faz a conversão dos sinais propriamente dita.

Este CI é usado em paralelo com a linha irá ler os dígitos MF e colocar nos pinos 10, 12, 13 e 15 uma palavra binária que represente este. Os dígitos serão enviados para o bloco IHC para serem tratados. O circuito realiza o isolamento da tensão da linha através dos capacitores conectados ao pin. 2 e 18 do CI 9170 em série com a linha e com o retorno, respectivamente.

Entre cada dígito os pinos da entrada de dados são mantidos em nível baixo "0000".

O bloco IHC faz a interface do bloco DTMF com o computador. Este bloco é formado pelo 74LS04 que é uma porta inversora, pelo CI 74LS373 que adapta os sinais para enviar à porta paralela do computador. O conector da porta paralela foi realizado com base em (MESSIAS, 2008).

Os bits da palavra binária gerados pelo bloco DTMF entram nos pinos 13, 14, 17 e 18 para serem transmitidos para a paralela nos pinos 12, 15, 16 e 19.

A entrada e saída de bits serão acionadas pelo sinal vindo do pino 15 do CI 9170 do bloco DTMF, pelo pino 1 que é ativo alto e pelo 11 que é ativo baixo, para acionar o

pino 11 com o mesmo sinal do pino 1 utilizamos uma porta inversora (74LS04) antes do pino 11.

O bloco DCL (detector de corrente de loop) detecta quando o telefone é retirado do gancho. Este bloco é composto por um diodo, um resistor e pelo foto acoplador 4N25. Este bloco irá gerar um sinal de tensão igual à VCC, que será enviado diretamente à porta paralela do computador. Vale lembrar que não é necessário passar pelo bloco de acoplamento, porque o sinal é enviado por um foto-acoplador.

Este bloco é conectado em série com a linha antes do aparelho telefônico. É responsável por monitorar a linha, para a detecção de corrente de loop, e para detectar a inversão de polaridade.

Ao detectar a corrente de loop um pino da porta paralela será alterado em nível alto para que o programa seja iniciado. Este mesmo pino receberá o sinal de atendimento através da inversão de polaridade, e isso ocorrerá quando o pino for alterado em nível baixo com a mesma duração da inversão de polaridade. O outro foto-acoplador do bloco DIP irá conduzir durante a inversão, alterando os dois pinos mais significativos da entrada dos dados em nível alto, isto para que o sinal não seja confundido com um dígito, pois as palavras binárias que representam os dígitos vão de “0001” até “1010”, e neste caso a saída será “1100”.

O bloco DIP (detector de inversão de polaridade) tem a função de detectar a inversão de polaridade, que é um sinal fornecido pela central telefônica, para identificar o atendimento da chamada

Este bloco é necessario para evitar que sinais gerados pelo usuário sejam confundidos com o sinal de atendimento. Quando o sinal de atendimento é enviado o bloco detector de corrente de loop irá perceber a ausência de sinal, mas como a duração desse pico será igual a 150 ms, o programa irá entender como sinal de atendimento. Aparentemente o bloco detector de inversão de polaridade não se faz necessário, já que o bloco detector de corrente de loop consegue identificar o sinal de atendimento, mas o mesmo não consegue diferencia-lo de um sinal gerado pelo usuário

Este bloco é muito similar ao detector de corrente de loop a diferença é que ao invés de detectar o sinal positivo ele detecta o negativo, para que ao ocorrer a inversão de polaridade o sinal seja notado, e para que caso o telefone seja colocado e retirado do gancho rapidamente o pico na corrente não seja confundido com o sinal de inversão.

O circuito é ligado com alinha telefônica de duas maneiras, o bloco detector de corrente de loop e o bloco detector de inversão de polaridade são conectados em serie

com a linha telefônica antes do aparelho telefônico. Já o bloco DTMF é conectado em paralelo com a linha, é necessário cuidado com a polaridade da linha telefônica, o fio de polaridade positiva é conectado ao pino 2 do NT9170 e o de polaridade negativa é conectado ao terra.

4 Programa

Esta seção descreve em detalhes o programa que foi desenvolvido, o seu funcionamento e sua lógica. Foi desenvolvido em Java uma linguagem orientada a objeto. Este programa tem por objetivo realizar a tariação da chamada, para isso é necessário o monitoramento do circuito e dos sinais que este recebe. (SUN MICROSYSTEMS,2008).

Sem muita interatividade com o usuário ele basicamente identifica a chamada e realiza a tarifação comforme as normas da Anatel em tempo-real.

4.1 Descrição geral

O programa como um todo é composto por três tarefas que ocorrem em paralelo. Uma delas o `Autômato` controla todas as ações, desde o início do programa até a tarifação propriamente dita. É ele também quem inicia as outras duas tarefas que são executadas enquanto ele monitora o circuito.

A tarefa `Autômato` tem oito estados, que são: `Reconhecendo_circuito`, `Aguardando_chamada`, `Aguardando_dígitos`, `Salvando_dígitos`, `Determinando_tarifa`, `Aguardando_atendimento`, `Validando_atendimento e tarifando`. Cada um destes estados é ligado ao anterior, sendo que uma ação altera de um estado para o outro. Como a qualquer momento pode ocorrer o final da execução do ciclo do programa, todos os estados ao serem encerrados retornam ao estado inicial do programa `Aguardando_chamada`. Todas as tarifações iniciam e encerram neste estado.

O `Tarifador` tem cinco métodos, inicialmente o método `número de dígitos`, `operadora` e `Modalidade da chamada` são acessados pelo estado `Salvando_dígitos`, pois é necessário saber quantos dígitos serão salvos. Neste

ponto o Autômato pára, e esse bloco de código é executado sequencialmente. Em um segundo momento ele é executado paralelamente com o Autômato, enquanto o Autômato monitora a condição da linha o Tarifador determina a tarifa da chamada em questão.

A tarefa exibe custo é executada ciclicamente e em paralelo com Autômato. Enquanto é exibido o custo da chamada em tempo-real, o Autômato monitora o circuito, ocorrendo o desligamento, a rotina é encerrada.

Utilizando a figura 5 do programa, descrevemos abaixo todo o processo desde a detecção do circuito até a tarifação de uma chamada. Quando o programa é iniciado o estado Reconhecendo_circuito do Autômato, verifica-se o telefone esta no gancho, para garantir que não tem nenhuma ligação em curso. Isso é realizado através do método lê pino da classe Paralela. (PORTILLO,2008) Sendo assim o estado do programa passa a ser aguardando chamada, neste ponto a execução do Autômato passa a ser seqüencial e cíclica, e o início deste ciclo é sempre Aguardando_chamada. A classe Paralela é novamente acessada e a leitura do pino da corrente de *loop* fica sendo monitorada. Quando o telefone for retirado do gancho o estado é alterado. O usuário tem até sete segundos para inserir um novo dígito, esse tempo é cronometrado pela classe Temporizador com o início de um cronomêtro decrescente de 7 segundos.

O estado Aguardando_dígitos fica monitorando os pinos de entrada dos dígitos na porta paralela. Ao receber o primeiro dígito ele será salvo e o estado será modificado para salvando dígitos. Salvando dígitos acessa alguns métodos da classe tarifador, e após o término da inserção de dígitos o estado será determinando tarifa. Nesta etapa outros métodos do tarifador são executados em paralelo com o Autômato, com a tarifa determinada o próximo passo é aguardar o início da chamada, que é representado pelo atendimento da chamada. O assinante chamado tem até noventa segundos para atender a chamada, o *timer* é reconfigurado.

Como o sinal de atendimento pode ser confundido com picos da corrente de loop, o próximo estado valida-o. Com a certeza do sinal de atendimento o estado do Autômato passa a ser tarifando. Com o início da classe exibe custo em paralelo com o Autômato, a tarifação é iniciada. Ao fim da chamada, que é detectada com a ausência de corrente de *loop* o Autômato volta ao estado Aguardando_chamada.

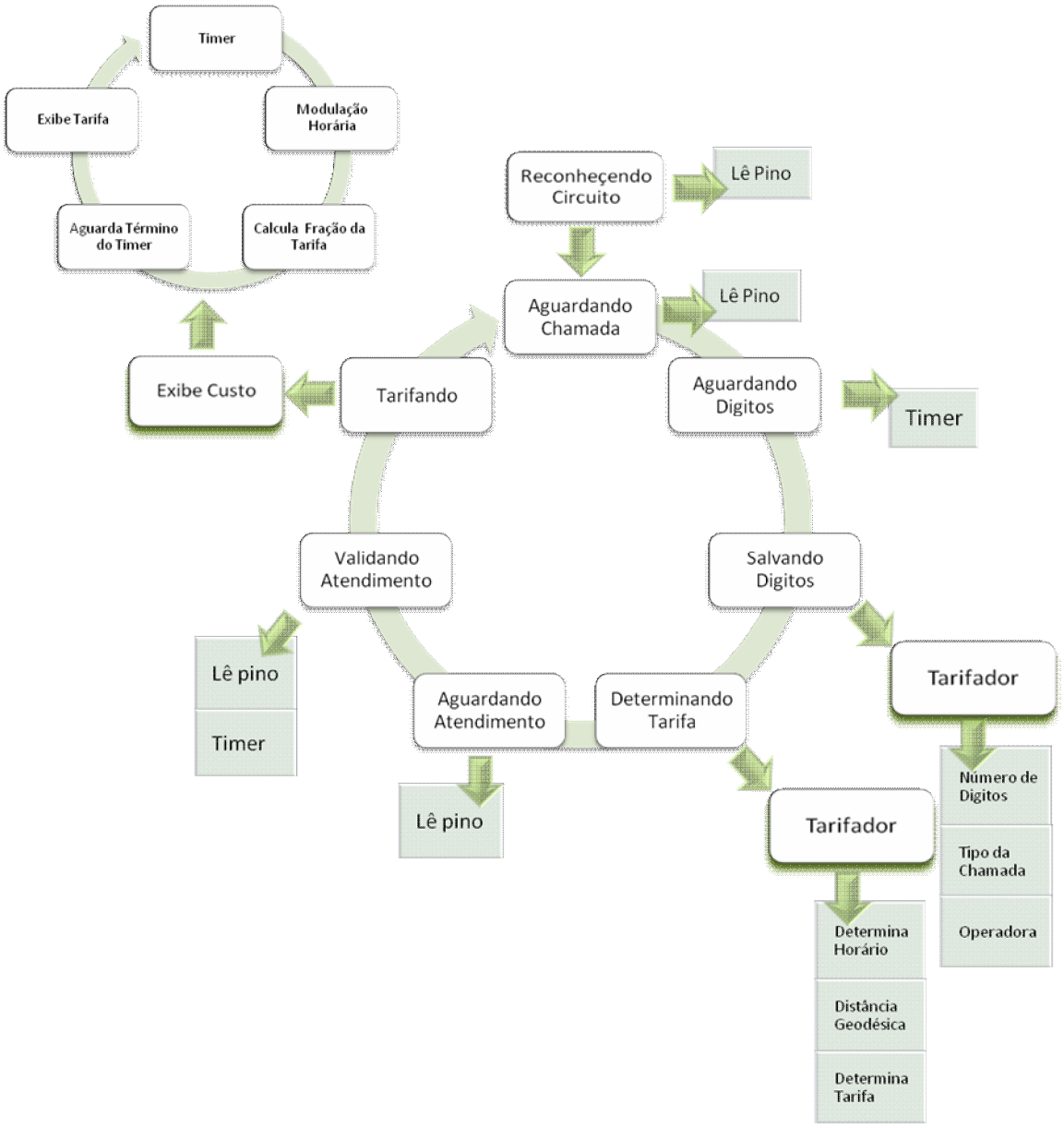

Figura 4 - Esquema do programa

4.2 Autômato

A tarefa `Autômato` controla toda a rotina do programa, desde o reconhecimento do circuito até o momento de exibir o custo ao usuário. O `Autômato` é uma tarefa que ocorre em paralelo com o `Exibe Custo` e o `Tarifador`, que funciona ciclicamente,

tendo oito possíveis estados. Cada estado executa uma pequena tarefa e a mudança de um estado para o outro é gerada por uma ação. A cada ciclo do `Autômato` um estado é executado por um `case` de um switch como mostrado na Figura 5.

A seguir o detalhamento passo a passo da rotina do `Autômato`, onde cada estado é descrito, sendo sempre referenciado ao código em apêndice.

a) Estado Reconhecendo Circuito:

O estado `Reconhecendo_circuito` identifica as condições iniciais do circuito. Garantindo que o programa se inicie somente quando o telefone esteja fora do gancho. Este estado é o primeiro a ser assumido pelo `Autômato` e, ao iniciar o `switch`, o primeiro bloco de código que será executado. Inicialmente é verificado o estado do pino 11. Caso o valor do pino for nível baixo, significa que o telefone está no gancho. E o `Autômato` altera seu estado para `Aguardando_chamada`.

b) Estado Aguardando Chamada:

Este estado tem a função de detectar o início de uma chamada telefônica. Isto ocorre com a alteração do nível do pino 11. O estado do programa é alterado para `Aguardando_digitos` quando o telefone for retirado do gancho. Ainda neste estado é iniciado um *timer* de 7 segundos, tempo este máximo para a entrada do primeiro dígito do número do telefone. Este *timer* é gerado com o método `Tempo` da classe `Temporizador`. Tal método é evocado pelo objeto *timer* previamente criado. Neste estado é iniciado também o vetor de caracteres que irá receber os dígitos.

c) Estado Aguardando Dígitos:

Esta parte do `Autômato` tem a função de aguardar a entrada dos dígitos do telefone, duas condições são testadas: a entrada dos dígitos e o desligamento da chamada.

O desligamento pode ocorrer por dois motivos, o fim do *timer* de 7 segundos, tempo máximo para entrada do primeiro dígito conforme a norma, ou a ausência de corrente de *loop* que ocorre no caso do telefone ser colocado no gancho. Estas

condições são testadas por um `if` e caso verdadeiro o estado do `Autômato` é modificado para `Aguardando_chamada`.

Se o estado dos pinos da porta paralela é diferente de zero a entrada de dígitos é detectada Neste caso zero é representado pelo valor 120 em hexadecimal. Uma vez lido, o primeiro dígito é repassado para a classe `salvar_dígitos`, que irá realizar algumas conversões para determinar o valor do dígito correspondente. Este valor é retornado para a variável `Retorno_Salva`, que por sua vez é salvo no vetor. Neste momento o estado do `Autômato` é alterado para `Salvando_digitos`, e é iniciado um novo *timer* de 3 segundos, tempo máximo entre dígitos.

d) Estado Salvando Dígitos:

Os dígitos posteriores ao primeiro lido no estado anterior serão salvos no vetor durante este estado. Nesta etapa também é determinado o número de leituras que devem ser realizadas, inicialmente são 6 leituras, porque com seis leituras é possível determinar a modalidade de qualquer chamda.

O `Salvando_digitos` tem três ciclos, que são percorridos em duas etapas, sendo estas divididas em antes e depois da determinação do número de leituras excedentes que deverão ser realizadas.

No ciclo interno temos uma rotina que fica presa enquanto não for inserido um novo dígito. Ao sair deste ciclo, ou seja, retornando ao intermediário, é iniciado novamente o timer entre dígitos de 3 segundos.

Ciclo intermediário garante que não serão realizadas mais leituras que o determinado. Dentro deste é capturado o valor dos pinos da porta paralela. Caso o valor recebido não corresponda a 0 e a variável novo_dígito seja 1, significa que foi inserido um novo dígito e este é adicionado ao vetor .

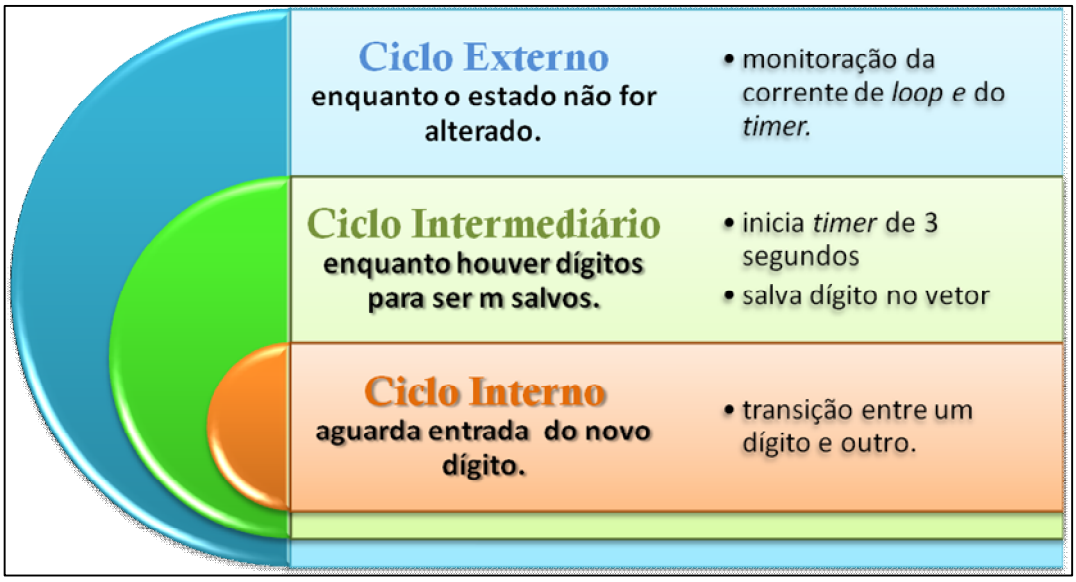

Figura 5- Esquema do Salva Dígitos

O ciclo externo garante que não ocorreu o desligamento da chamada. Isto é realizado através de testes que são realizados em função do *timer* entre dígitos e da corrente de *loop*. Caso ocorra o desligamento o estado é alterado para `Aguardando_chamada`. Dentro do ciclo externo são realizados vários testes que alteram o comportamento do programa. Quando for realizado o número de leituras inicialmente configurado, será verdadeiro e neste momento é calculado o número de leituras excedentes. Pois cada modalidade de chamada tem um número especifico de dígitos. O Modalidade da chamada também é determinado neste momento.

Caso o teste não seja verdadeiro então será testado se todas as leituras já foram realizadas e se já foi realizado o cálculo das leituras excedentes. Em caso positivo todos os dígitos foram salvos e o estado do `Autômato` é modificado para `Determinando_tarifa`.

e) Estado Determinando Tarifa:

Este estado cria um objeto do tipo Tarifador, e inicia a *thread*. Com base nos números salvos no vetor juntamente ao Modalidade da chamada, o custo da tarifa por minuto da chamada em questão será determinado. Enquanto a tarifa é determinada, o programa prossegue, ele fica monitorando a porta paralela para que caso ocorra o desligamento o estado passe para AGUARDANDO_ CHAMADA.

Depois de determinar o custo da chamada, o estado do `Autômato` é configurado como AGUARDANDO_ATENDIMENTO.

Aguardando Atendimento: o atendimento é sinalizado através do sinal de inversão de polaridade.O sinal de atendimento é aguardado por até 90 segundos(TEMPORIZAÇÃO). Esta condição é verificada através do estado do *timer*, se ele acabar o estado do `Autômato` é alterado para AGUARDANDO_CHAMADA. Caso contrário o estado será validando atendimento, também é iniciado o timer de 150ms.

Validando Atendimento: caso o timer de 150ms tenha terminado e o telefone ainda esteja fora do gancho significa que o sinal recebido é de atendimento. Neste caso o estado passa a ser TARIFANDO , caso contrário será AGUARDANDO_CHAMADA.

f) Tarifando:

O último estado do `Autômato` é o que realiza a tarifação propriamente dita. Este procedimento é realizado com pela *thread* `Exibe_custo`, que é instanciada e iniciada neste estado. A execução do programa prossegue, onde é iniciado um ciclo que existe enquanto não ocorrer o desligamento. Quando a chamada for encerrada a *thread* exibe_custo é encerrada, e o estado é alterado para AGUARDANDO_CHAMADA.

4.3 Tarifador

Esta tarefa tem várias funções, que são: determinar o número de dígitos que serão lidos, encontrar a prestadora utilizada, através dos prefixos determinar o Modalidade da chamada, definir a modulação horária, determinar a tarifa e calcular a distância geodésica. Cada uma destas funções é implementada por um método desta classe, sendo descritos a seguir.

a) Método Número de Dígitos:

O método `Número_de_dígitos` tem a função de determinar o número de leituras excedentes que serão realizadas pelo `Autômato` e o Modalidade da chamada. Ele recebe um vetor, que contém os dígitos que já foram recebidos, caso exista, é

necessário determinar a operadora, para isso o método operadora é acessado. Pois nem todas as chamadas necessitam de operadora, por exemplo, chamadas locais.

Se for do tipo LDI obrigatoriamente irá começar com "00". Com essa informação é possível determinar esta modalidade de ligação. Após determinar o Modalidade da chamada igual à longa distância nacional, é determinado o número de leituras excedentes que deverão ser realizadas.

Caso a chamada seja do tipo 0800 ou 0300, testamos se o segundo dígito do vetor é igual a 8 ou 3, e se o primeiro a 0. Se sim, a chamada será determinada como especial, não sendo necessária mais nenhuma leitura, pois com essas informações o custo já pode ser determinado.

Quando a chamada for longa distância nacional, ou seja, de área tarifária diferente do chamador, podemos ter três tipos de chamada, que são: VC-2, VC-3 ou LDN_FIXO.

Determinamos que a chamada é longa distância nacional, se o primeiro dígito é 0, e o código da operadora é valida. Caso positivo, temos que diferenciar a chamada entre os possíveis tipos de chamada longa distância nacional, se começar com 9, 8 ou 7, é celular. A partir deste ponto temos que determinar se é VC-2 ou VC-3. Será VC-2 se o número chamado pertencer a uma área de registro (AR) diferente da área de numeração (AN) do acesso de origem, porém com 1º algarismo do código nacional da AN de origem igual ao 1º algarismo do código nacional da AR de destino. E será VC3 se a área de numeração for diferente. Se não for nenhuma das opções citadas acima, então a chamada é longa distância nacional para fixo.

O final deste método define o tipo de chamada local, que pode ser subdividido como: fixo, celular ou chamada à cobrar. Será local quando o primeiro dígito for diferente de 0. Primeiro determinamos que a chamada é local fixo, caso ela não se encaixe em nenhuma das outras opções abaixo ela é de fato local fixo, caso ela seja outro tipo de chamada, o tipo será modificado. É celular se o número do telefone começar com 9, 8 ou 7 e o segundo dígito for diferente de 0. A chamada será à cobrar se o primeiro número for 9, e o segundo 0.

b) Operadora:

Esta rotina determina a operadora que está sendo utilizada. O programa aceita somente as operadoras Brasil Telecom e GVT. Será Brasil Telecom quando o terceiro e quarto dígito forem 1 e 4, respectivamente. GVT, quando forem 2 e 5.

c) Prefixos:

Este método formata o vetor dos dígitos e determina a latitude e longitude relacionada com o prefixo em questão, Estes dados estão contidos em um documento fornecido pela Anatel(ANATEL, 2008). Cada tipo de chamada envolvendo telefones fixos tem um tratamento especifico. Esta rotina só é executada quando o Modalidade da chamada é para telefone fixo.

Primeiro o prefixo é salvo na variável `pre`. Estas variáveis contém a área tarifária do telefone de origem. Caso seja LDN, irá salvar do 4° ao 9° dígito.

Depois de possuir o prefixo o arquivo que contém a latitude e longitude de cada localidade relacionado ao seu prefixo será lido e salvo em um *buffer*. Outra variável recebe o prefixo e outras duas recebem o valor do início e do fim da posição na linha do arquivo que deverá ser lido.

O arquivo será lido linha a linha, até que o prefixo digitado seja igual ao lido do arquivo. Quando isso ocorrer serão lidas a latitude e longitude da localidade, e também o nome da localidade. Caso o final do arquivo seja alcançado e o prefixo não seja encontrado a rotina é encerrada.

d) Determina horário:

Este método tem a função de determinar a modulação horária, com base na data do ano e do horário

Para determinar que a chamada de fixo para fixo seja da modulação de domingos e feriados, testamos se o dia é igual a domingo ou a data de um feriado, em caso positivo testamos se a hora é entre 00:00 e 5:59:59, neste caso a modulação é super reduzida, no restante é reduzida.

Será de segunda à sexta quando o dia da semana for diferente de sábado e domingo e não sendo feriado, é testado o horário. Quando a hora for entre 7:00 e 08:59:59 ou entre 12:00 e 13:59:59, ou ainda entre 18:00 e 20:59:59, a modulação é normal. Quando for entre 9:00 e 11:59:59, ou entre 14:00 e 17:59:59, será diferenciado. Para hora entre 6:00 e 06:59:59, ou entre 21:00 e 23:59:59, será reduzido. E por fim hora entre 00:00 e 05:59:59, será super reduzido.

Sendo sábado e hora estando entre 7:00 e 13:59:59, será da modulação normal. Sendo hora entre 6:00 e 6:59:59 ou hora entre 14:00 e 23:59:59, será reduzido. E por

fim estando entre 00:00 e 05:59:59, será super reduzido.

Para chamadas locais de fixo para fixo, na Brasil Telecom existem apenas duas modulações, reduzida e normal. Nos dias úteis é normal das 6:00:00 as 23:59:59 e reduzida nos demais horários. Nos sábados das 6:00:00 as 13:59:59, no demais reduzido. E por fim em domingos e feriados é reduzido o dia todo. (BRASIL TEECOM S.A., 2008).

De maneira bem semelhante determinamos a modulação para chamadas de fixo para celular. O detalhe mais importante é que chamadas do tipo VC1, têm modulação horária diferente das demais, mesmo não tarifando chamadas do tipo VC2 e VC3, a determinação da modulação horária é realizada.

e) Distância Geodésica:

Para chamadas do tipo longa distância nacional, é necessário determinar a distância geodésica entre as localidades, para que o degrau tarifário seja determinado, pois a tarifa esta relacionada com a distância entre as localidades. No sistema de tarifação de chamadas longa distância nacional brasileira existem quatro degraus tarifários, onde será degrau 1 se a distância for menor que 50 km, degrau 2 para distâncias entre 50 e 100 km, degrau 3 para distâncias entre 100 e 300 km e degrau 4 para distâncias maiores que 300 km. Com a implementação desta equação a distância é facilmente encontrada, e o degrau já é determinado.(GEOCOMP SYSTEMS's, 2008).

f) Determina tarifa:

este método faz a leituras dos arquivos das tarifas e encontra a tarifa conforme o Modalidade da chamada e a modulação horária determinada. Os tipos da chamada podem ser: BRT_NORMAL_CEL, BRT_REDUZIDO_CEL, BRT_LOCAL_NORMAL, BRT_LOCAL_REDUZIDO, BRT_NORMAL, BRT_REDUZIDO, BRT_DIFERENCIADA e BRT_SUPERREDUZIDO. Os tipos terminados em CEL são relacionados às chamadas de fixo para celular, e estas são diferenciadas pela operadora de destino que podem ser: Brasil Telecom (BRT) , Tim, Vivo, Claro e Nextel.

4.4 Exibe Custo

Esta tarefa realiza a tarifação propriamente dita. Com o valor da tarifa por minuto ela implementa as regras de tarifação descritas na resolução 424. Inicialmente o valor da chamada é igual ao da tarifa dividida por dois, isto até os primeiros 30 segundos da chamada, após a chamada será tarifada a cada seis segundos e o valor acrescido é igual ao do minuto dividido por dez. Caso durante a chamada a modulação horária mude o valor da tarifa também irá mudar e com isso o valor da fração de tarifa. Esta possibilidade é testada antes do acréscimo da tarifa, caso positivo o valor da fração será alterado.

5 Problemas Encontrados

Este capitulo tem o intuito de apresentar as dificuldades encontradas no desenvolvimento do trabalho, com suas respectivas soluções ou justificativas no caso de problemas sem solução viaveis.

5.1 O Projeto do Circuito

O circuito seria projetado e testado com o auxílio de simuladores, mas nenhum dos programas conhecidos tinha o CI (circuito integrado) MT8870 ou similar. Diante deste contratempo partimos direto para os testes na matriz de contatos.

5.2 Alimentação

O circuito é alimentado com uma fonte de celular que fornece uma tensão de 5,7 Volts e uma corrente de 800mA, inicialmente o circuito seria alimentado pela porta paralela, mas a porta paralela se mostrou instável e não fornece a potência necessária, por este motivo é utilizado o carregador de celular como fonte.

5.3 Detector de corrente de loop

Embora o circuito seja simples, não é evidente, pois o circuito é conectado serialmente com a linha telefônica, mas não pode drenar corrente, pois diante dessa situação o telefone estaria sempre ocupado. O problema foi resolvido com ajuda de um foto-acoplador.

5.4 Número de sinais de entrada

A porta paralela tem apenas cinco entradas e com essa limitação surgiu um problema, pois tínhamos 6 sinais para enviar para a porta. A solução foi trocar o sinal que iria indicar a inserção de um novo dígito, por detectar o símbolo “0000” entre dígitos.

5.5 Detector de inversão de polaridade

Como o circuito foi projetado inicialmente, não era possível impedir que a inversão de polaridade fosse confundida com um pulso rápido gerado pelo gancho, no caso do usuário colocar e retirar o telefone rapidamente do gancho. Para solucionar este problema foi necessário utilizar mais um foto-acoplador que irá conduzir somente durante a inversão de polaridade. Este sinal será enviado por dois pinos da porta paralela.

5.6 Variável de caminho

Variável de caminho (*path*) : é uma variável de ambiente do sistema Linux que indica trajetória dos binários, que podem ser executados sem indicar o caminho completo de onde eles estão.

A configuração do caminho foi um problema, pois desconhecíamos a necessidade de realizar tais procedimentos. Abaixo são apresentados os comandos que foram utilizados:

- Esteja como root.
- Descompacte o arquivo coprimido no diretório HOME do root.
- Crie um diretório 'classes'.
- Defina a variável

  CLASSPATH: export CLASSPATH=$HOME/classes.
- Defina a variável

LD_LIBRARY_PATH: export LD_LIBRARY_PATH=$HOME/parport.

5.7 Bibioteca Parport

A captura de sinais na porta paralela foi viabilizada com o uso desta biblioteca. ParallelPort é uma simples classe Java que permite ler e escrever bytes de e para as portas paralelas em seu computador.

5.8 Tarifação de chamadas internacionais

Neste trabalho um dos objetivos era tarifar chamadas internacionais, contudo não foi possível, pois o custo de chamadas internacionais para fixo e móvel é diferenciado. E para isso é necessário identificar cada um dos tipos de chamada. Os dados que identificam se a chamada é para fixo ou móvel não são informadas pela operadora. Uma única opção para solucionar esse problema é realizar uma pesquisa para saber todos os prefixos de todos os países.

6 Resultados

Para testar se a porta paralela detecta os sinais enviados pelo circuito utilizamos o DSPCOM (MESSIAS, 2008), um programa encontrado na Internet, que apresenta o nível da porta paralela.

Para testar o circuito não interfere no áudio da linha telefônica conectamos em uma linha telefônica, e geramos duas ligações, uma interna, ramal – ramal e outra externa, ramal- tronco. O circuito funcionou corretamente e não interferiu no áudio da chamada telefônica.

Na etapa de tarifação de chamadas, dois testes diferentes foram necessários. O teste que garante que a tarifa utilizada é correta e o teste que assegura que a tarifação será realizada conforme as normas da Anatel.

Um conjunto de testes verificou todas as possíveis tarifas de chamadas e mostrou que os custos encontrados, atendem o esperado. Nas tabelas abaixo temos o teste de chamadas da modalidade local fixo. A Tabela 3 tem o teste, e a Tabela 4 o resultado.

	Dados				
n°	Número	Modalidade da chamada	Modulação Horária	Operadora	Custo Previsto
1	32810000	Local	Normal 6 – 24	BRT	0,10486
2	32810000	Local	Reduzido 0 – 6	BRT	0,20972
3	32810000	Local	Reduzido 0 – 24	BRT	0,20972
4	32810000	Local	Reduzido 14 – 24	BRT	0,20972

Tabela 4 - Bateria de testes de chamadas locais

	Resultados			
	Hora Início	Tarifa	Classificação	Correto?
1	14:33:38	R$ 0,10	BRT local normal	sim
2	05:04:50	R$ 0,21	BRT local reduzido	sim
3	Dom	R$ 0,21	BRT local reduzido	sim
4	Sab 14:05	R$ 0,21	BRT local reduzido	sim

Tabela 5 - Resultados das chamadas locais

Nestes testes a origem é um número do município de São José, cada linha contém o teste que verifica um período da modulação horária.

Foram realizados testes na modalidade longa distância nacional, que testa os quatro possíveis degraus tarifários. Estes testes são exibidos na Tabela 5 e o seu resultado na Tabela 6.

	Número	Modalidade da chamada	Modulação horária	Operadora	Custo previsto
1	0144832550	LDL DG 1	7h às 9h (1)	BRT	0,19367
2	0144832550	LDL DG 1	9h às 12h (5)	BRT	0,33760
3	0144832550	LDL DG 1	6h às 7h (7)	BRT	0,09677
4	0144832550	LDL DG 1	0h às 6h (12)	BRT	0,04833
5	0144832810	LDL DG 2	12h às 14h (2)	BRT	0,32285
6	0144832810	LDN DG 2	14h às 18h (6)	BRT	0,46889
7	0144832810	LDN DG 2	21h às 24h (8)	BRT	0,16134
8	0144832810	LDN DG 2	0h às 6h (13)	BRT	0,08061
9	0144733350	LDN DG 3	18h às 21h (3)	BRT	0,32695
1	0144733350	LDN DG 3	14h às 18h (6)	BRT	0,48095
11	0144733350	LDN DG 3	6h às 7h (9)	BRT	0,21967
12	0144733350	LDN DG 3	0h às 6h (14)	BRT	0,12099
13	0144933610	LDN DG 4	7h às 14h (4)	BRT	0,32696
14	0144933610	LDN DG 4	14h às 18h (6)	BRT	0,48130
15	0144933610	LDN DG 4	14h às 24h (10)	BRT	0,21974
16	0144933610	LDN DG 4	0h às 6h (14)	BRT	0,16134
17	0144733350	LDN DG 4	6h às 24h (11)	BRT	0,21974

Tabela 6 - Testes de chamadas LDN

	Hora início	Tarifa	Classificação	Correto?
1	07:01:00	R$ 0,19	normal	sim
2	11:43:00	R$ 0,34	diferenciada	sim
3	06:02:00	R$ 0,10	reduzido	sim
4	00:11:00	R$ 0,05	super reduzido	sim
5	12:01:00	R$ 0,32	normal	sim
6	14:01:00	R$ 0,47	diferenciada	sim
7	21:01:00	R$ 0,16	reduzido	sim
8	00:01:00	R$ 0,08	super reduzido	sim
9	18:03:00	R$ 0,33	normal	sim
10	14:02:00	R$ 0,48	diferenciada	sim
11	06:01:00	R$ 0,22	reduzido	sim
12	00:02:00	R$ 0,12	super reduzido	sim
13	07:01:00	R$ 0,33	normal	sim
14	14:01:00	R$ 0,48	diferenciada	sim
15	19:01:00	R$ 0,33	normal	sim
16	00:01:00	R$ 0,16	super reduzido	sim
17	06:01:00	R$ 0,22	reduzido	sim

Tabela 7 - Resultados de chamadas LDN

Repare que na Tabela 5, que contém os testes, a quarta coluna tem a modulação horária a qual a chamada deve pertencer, ao lado do horário tem um número entre parênteses, estes estão relacionados a um intervalo específico da Tabela 7 de modulação horária de chamadas de longa distância nacional.

	Normal	Diferenciado	Reduzido	Super-reduzido
Dias úteis	7h às 9h (1)	9h às 12h (5)	6h às 7h (7)	0h às 6h (12)
	12h às 14h (2)	14h às 18h (6)	21h às 24h (8)	
	18h às 21h (3)			
Sábados	7h às 14h (4)		6h às 7h (9)	0h às 6h (13)
			14h às 24h (10)	
Domingos e Feriados Nacionais			6h às 24h (11)	0h às 6h (14)

Tabela 8 - Modulação Horária de chamadas LDN

A Tabela 7 apresenta a modulação horária de chamadas da modalidade longa distância nacional.

Os últimos testes realizados, que são apresentados na Tabela 8 e seus resultados na Tabela 9, completam as possibilidades de chamadas comuns tarifáveis.

	Número	Modalidade da chamada	Modulação horária	Operadora
1	91000000	Móvel – vc1	Normal	vivo
2	92000000	Móvel – vc1	Reduzido	vivo
3	93000000	Móvel – vc1	Normal	vivo
4	94000000	Móvel – vc1	Reduzido	vivo
5	88000000	Móvel – vc1	Normal	claro
6	88000000	Móvel – vc1	Reduzido	claro
7	84000000	Móvel – vc1	Normal	brt
8	85000000	Móvel – vc1	Reduzido	brt
9	96000000	Móvel – vc1	Normal	tim
10	97000000	Móvel – vc1	Reduzido	tim
11	98000000	Móvel – vc1	Normal	tim
12	99000000	Móvel – vc1	Reduzido	tim
13	78000000	Móvel – vc1	Normal	nextel
14	78000000	Móvel – vc1	Reduzido	nextel
15	014478400	Móvel – vc2	Normal	brt
16	014478400	Móvel – vc2	Reduzido	brt

Tabela 9 - Testes de chamadas destinadas ao SMP

A tarifação de chamadas de fixo para móvel na modalidade local é relacionada com a operadora de destino, por isso é que na quinta coluna apresenta a operadora, esta é determinada pelo número do telefone.

Ao final dos testes, podemos perceber que todas as chamadas receberam a tarifa correta, conforme a modulação horária proposta pelas operadoras, e também as respectivas tarifas. Os testes também verificaram a classificação das chamadas dentro das possíveis modalidades. Portanto ao final destes 37 testes, asseguramos que as tarifas são encontradas corretamente dentro de suas modulações horárias.

	hora início	tarifa	Classificação	correto?
1	07:51:00	R$ 0,70	vivo normal	sim
2	21:01:00	R$ 0,49	vivo reduzido	sim
3	07:50:00	R$ 0,70	vivo normal	sim
4	21:16:00	R$ 0,49	vivo reduzido	sim
5	07:49:00	R$ 0,72	claro normal	sim
6	21:17:00	R$ 0,50	claro reduzido	sim
7	07:48:00	R$ 0,72	brt normal	sim
8	21:18:00	R$ 0,56	brt reduzido	sim
9	07:51:00	R$ 0,72	tim normal	sim
10	21:26:00	R$ 0,50	tim reduzido	sim
11	07:52:00	R$ 0,72	tim normal	sim
12	21:28:00	R$ 0,50	tim reduzido	sim
13	07:54:00	R$ 0,66	nextel normal	sim
14	21:30:00	R$ 0,46	nextel reduzido	sim
15	07:54:00	Não tarifada	brt normal	sim
16	21:01:00	Não tarifada	brt reduzido	sim

Tabela 10- Resultados de chamadas para celular

O outro teste realizado, verifica se a tarifação é realizada conforme as regras da Anatel. A seguir as figuras são capturas da tela do computador executando a tarifação.

Chamadas com duração inferior a 3 segundos, não são tarifáveis. Confira o resultado na Figura 7. A figura mostra que a tarifação foi iniciada e a hora do inicio, mas como esperado a tarifa não é exibida, caso a chamada não fosse encerrada dentro de 3 segundos o custo inicial seria apresentado.

```
DETERMINANDO_TARIFA
prefixo setado 473335
prefixo 473335
Nome da localidade  BLUMENAU
latitude 2.6550984E7
longitude 4.9035609E7
AGUARDANDO_ATENDIMENTO
iniciou thread tarifador
acessando a função determina_tarifa
VALIDANDO_ATENDIMENTO
modulação: BRT_DIFERENCIADA
Distancia: 175.49176323415745
Degrau: 3
leu arquivo brt_diferenciada.txt
tarifa: 0.48
iniciou a tarifação, hora: 16:17:42
Time's up!
```

Figura 6 - Chamada com duração inferior a 3 segundos

A figura 8 a seguir mostra uma chamada com duração superior a 3 segundos e

inferior a 30 segundos, conforme a norma da Anatel, chamadas com duração entre 3 e 30 segundos têm o custo da chamada igual a metade da tarifa por minuto.

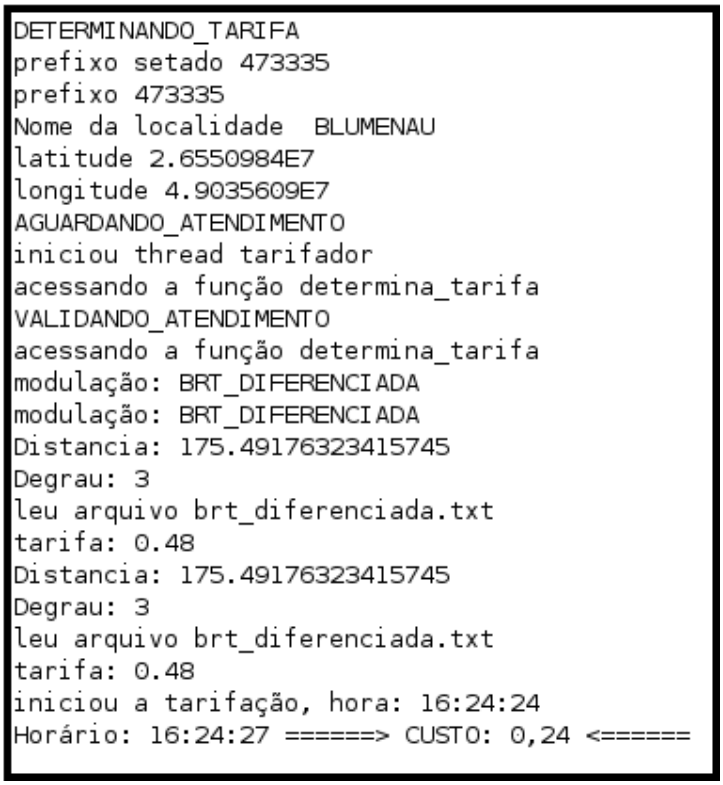

Figura 7 - Chamada com duração entre 3 e 30 segundos.

Por último temos uma chamada realizada por volta das 11:59, onde a modulação horária é diferenciada, a chamada durou até depois das 12:00, onde a modulação é normal, portanto conforme a norma o custo é alterado passando a ser de diferenciado para normal. Nesta figura também podemos ver que a tarifação é atualizada a cada 6 segundos, obedecendo às normas da Anatel.

```
modulação: BRT_DIFERENCIADA
Distancia: 175.49176323415745
Degrau: 3
leu arquivo brt_diferenciada.txt
tarifa: 0.48
Horário: 11:59:52 ======================> CUSTO: 0,34 <======
modulação: BRT_DIFERENCIADA
acessando a função determina_tarifa
modulação: BRT_DIFERENCIADA
Distancia: 175.49176323415745
Degrau: 3
leu arquivo brt_diferenciada.txt
tarifa: 0.48
Horário: 11:59:58 ======================> CUSTO: 0,38 <======
modulação: BRT_NORMAL
acessando a função determina_tarifa
modulação: BRT_NORMAL
Distancia: 175.49176323415745
Degrau: 3
leu arquivo brt_normal.txt
tarifa: 0.33
Horário: 12:00:04 ======================> CUSTO: 0,42 <======
modulação: BRT_NORMAL
acessando a função determina_tarifa
modulação: BRT_NORMAL
Distancia: 175.49176323415745
Degrau: 3
leu arquivo brt_normal.txt
tarifa: 0.33
Horário: 12:00:10 ======================> CUSTO: 0,45 <======
```

Figura 8 - Chamada com troca de modulação horária

7 Conclusões e Comentários Finais

Apesar de algumas implementações inicialmente previstas não terem sido executadas, pode-se dizer que os objetivos gerais do trabalho foram alcançados. O hardware desenvolvido não interfere na linha telefônica e o software consegue determinar o custo da chamada telefônica em tempo-real. Na pesquisa realizada não foi encontrado produto ou serviço similar, o que demonstra a originalidade e relevância do trabalho.

É provável que uma patente do produto/serviço possa vir a ser registrada. Para tal, estudos estão sendo realizados.

Um dos objetivos do trabalho era implementar todas as regras de tarifação da Anatel, mas isso não foi realizado, pois existe uma norma que diz que chamadas sucessivas com duração inferior a 30 segundos, efetuadas entre os mesmos acessos de origem e de destino, e quando o intervalo entre o final de uma ligação e o início da seguinte for inferior a 130 segundos são tarifadas como uma única ligação, cuja duração é igual ao somatório das durações das chamadas sucessivas ou igual ao tempo de tarifação mínima (30s) . E isto não foi desenvolvido.

Outro objetivo era tarifar o plano básico das operadoras mais populares. Este não foi alcançado porque as operadoras não atendem completamente as normas do plano básico, fazendo com que para cada operadora fosse necessário que gerar um bloco de código específico. Esta etapa do trabalho não foi concluída por indisponibilidade de tempo.

O objetivo de atender a todos os telefones de Santa Catarina foi praticamente alcançado por completo, com exceção dos números especiais como 102, 4004 e outros.

A tarifação das chamadas internacionais não foi implementada devido à dificuldade de diferenciar o destino como fixo ou móvel, não sendo é possível encontrar a tarifa correta.

O desenvolvimento deste trabalho foi de grande valia, mais como aprimoramento do conhecimento adquirido ao longo do curso do que como aprendizado de novas

informações.

A etapa de pesquisa foi muito interessante, e trouxe um entendimento profundo do sistema e das normas de tarifação. Os meses de pesquisa que antecederam o início do desenvolvimento do trabalho trouxeram um conhecimento necessário, possibilitando maior segurança no momento do desenvolvimento do mesmo. Embora tenha sido realizada uma pesquisa prévia, surgiram alguns contratempos, como por exemplo, a dificuldade de implementação das chamadas internacionais, que por fim não foi realizada.

No desenvolvimento do trabalho a montagem do circuito se apresentou como um desafio, pois eu não possuía o conhecimento necessário, mas meus orientadores me auxiliaram e antes do prazo previsto a etapa foi concluída. Esta etapa foi a que me trouxe mais informação, adquiri um conhecimento que não possuía. Mesmo o circuito sendo simples, os poucos problemas apresentados valeram como experiência e me somaram conhecimento.

O programa foi a etapa mais longa do desenvolvimento, nela foram aprimorados os conhecimentos de programação. Aprendi alguns conceitos básicos que nunca ouvira falar, como a variável de caminho. O maior conhecimento nesta etapa foi descobrir que quase todas as necessidades de um programador já foram desenvolvidas, e que estas devem ser utilizadas. Conheci classes do Java as quais não imaginava que poderiam existir, mas percebi que minhas necessidades eram comuns às de outras pessoas.

Por fim a última etapa foi a escrita da monografia. Posso dizer que a maior barreira foi vencida neste momento, já que nunca tive grande facilidade de me expressar, tão pouco de escrever. Minha grande dificuldade com a gramática e a ortografia aos poucos foram deixadas para trás. Inicialmente nos relatórios, no artigo submetido à II Jornada da Produção Científica da Educação Profissional e Tecnológica da Região Sul e agora na monografia. O meu orientador tem grande participação nesta conquista, pois sempre me incentivou.

Com as dificuldades aprendi que os professores existem para nos ensinar. E que bons livros são amigos inseparáveis. Percebi que muitas dificuldades que passei foram devido ao fato que alguns conhecimentos básicos eu não possuía, alguns destes porque não me foram ensinados, outros porque não interiorizei.

A construção de um produto do tarifador seria muito interessante aos usuários de telefonia, mas devido a grande dificuldade para possuir todas as tabelas de tarifas de todas as operadoras, sua implementação se torna inviável, pois isso seria necessário no

desenvolvimento de um produto sem vínculos com qualquer operadora. A telefonia móvel também é um problema já que nas ligações locais o custo da tarifa é diferente conforme a operadora do telefone móvel do destino, que sendo assim teria que ser identificado. Mas a construção deste produto poderá ser realizada pelos próprios alunos do curso, onde os interessados podem implementar apenas parte dele e no final teríamos o tarifador de chamadas telefônicas em tempo-real completo.

8 Trabalhos futuros

8.1 Parte gráfica

Consiste no desenvolvimento de uma interface gráfica, para a exibição da tarifação, pois atualmente o programa exibe a tarifação diretamente no *prompt* onde for executado, isso pode ser um problema em ligações de longa duração, pois a quantidade de informação a ser exibida pode ocasionar o travamento do computador.

8.2 Dados usuário

Este trabalho é o desenvolvimento de um programa que receba o número do telefone e configure os parâmetros dos demais programas, que são os dados de origem da chamada. Esta implementação é muito necessária, pois sem ela o programa terá que ser modificado a cada novo prefixo de origem.

8.3 Inserir tarifas

Até então o programa somente realiza a tarifação do plano básico, e esta idéia é uma opção ao usuário, que poderá adicionar uma modulação horária diferente com um custo promocional para determinado número ou tipo de chamada.

8.4 Atualização

Tanto os arquivos que contém os prefixos, quanto os das tarifas necessitam ser

atualizados, o desenvolvimento de um programa de atualização desta base de dados é de alta relevância, pois existem inúmeras aplicações que necessitam de atualização.

A idéia inicial da atualização do tarifador é um servidor remoto, que irá enviar os dados através da linha telefônica, isso será viabilizado com um protocolo independente, que utiliza sinais multifrequênciais. Em cada ponta do canal haverá um circuito, capaz de decodificar os sinais recebidos e enviar ao computador da mesma maneira que os números do telefone são enviados.

8.5 Montagem da placa

O circuito está montado em uma matriz de contatos, e para evitar defeitos de mau contato é necessário o desenvolvimento do circuito em uma placa de circuito impresso.

8.6 Ampliação da área atendida

O tarifador desenvolvido só tem validade para números tanto de origem como de destino dentro do estado de Santa Catarina, a ampliação deste atendimento é um trabalho complexo e duradouro.

Isto por que a quantidade de dados é muito ampla, e para evitar dados desnecessários, o programa por eliminatória encontra os prefixos de degrau 4, pois quando não for nenhum dos demais degraus é o último. Sendo assim poderá ser desenvolvido um programa que prepara os arquivos dos prefixos, conforme o estado origem, eliminando todos os prefixos de degrau 4.

O desenvolvimento deste trabalho será muito duradouro devido a dificuldade do desenvolvedor de encontrar as tarifas, pois são diferentes de estado para estado, e de encontrar todos prefixos de telefones móveis, pois o custo das ligações locais fixo-móvel variam de acordo com a prestadora de destino.

8.7 Atender a todos os tipos de chamadas

O tarifador desenvolvido não implementa todos os número especiais existentes, como 102, 4004 e outros. E para que a tarifação seja total é necessário o desenvolvimento desta rotina.

8.8 Embarcar o programa

Este é um trabalho complexo, pois consiste em aprimorar o circuito e embarcar o programa, para que o tarifador se execute por completo separadamente do computador. Esse trabalho seria a implementação de um produto para a comercialização.

9 Referências Bibliográficas

ANATEL – **Apêndice a Resolução 424**. Disponível em: < http://www.anatel.gov.br/Portal/exibirPortalRedireciona.do?caminhoRel=Cidadao-Biblioteca-Acer-vo%20Documental&codigoDocumento=116059 > . Acesso em: 5 Mar. 2008.

ANATEL – **Prefixos**. Disponível em: < http://sistemas.anatel.gov.br/areaarea/N_Download/Tela.asp?varMod=Publico&SISQSmodulo=7179>. Acesso em: 10 Abr. 2008.

BRASIL TELECOM S.A. **- Tarifas do Plano Básico**. Disponível em: <http://www.brasiltelecom.com.br/site/frame.jsp?frame=http://www.brasiltelecom.com.br/site/> . Acesso em: 12 Abr. 2008.

BURGOS, L. C. - **Estudo de foto acopladores**. Disponível em : <http://www.burgoseletronica.net/fotoacopladores.htm >. Acesso em: 19 Abr. 2008 .

DEITEL, H. M. ;DEITEL, P. J**. Java, como programar**. 4ed. 2003- Bookman, Porto Alegre.

GEOCOMP SYSTEMS's – **Programa de cálculo de distância geodésica**. Disponível em: <http://www.geocomp.com.au/geocalc/ > . Acesso em: 5 Mai. 2008.

MESSIAS, A. R. – **Estudo sobre a Porta Paralela**. Disponível em : <http://www.rogercom.com/pparalela/introducao.htm >. Acesso em: 19 Abr. 2008.

PORTILLO, J. G. Del Cid – **Biblioteca Parport**. Disponível em: <http://www.geocities.com/Juanga69/parport/>. Acesso em: 19 Abr. 2008

ANATEL - **A norma de sinalização**. Disponível em: <http://sistemas.anatel.gov.br/sdt/PraticasTelebras/00117.pdf >. Acesso em: 23 Abr. 2008.

SOUZA, F. A. de; - **Apostila de Telefonia 1** - São José, 2007.

SUN MICROSYSTEMS, Inc. – **Classes do Java**. Disponível em: < http://java.sun.com/j2se/1.4.2/docs/api/allclasses-frame.html >. Acesso em: 5 Jun. 2008.

ANATEL - **A norma de temporzação.** Disponível em: <http://sistemas.anatel.gov.br/sdt/PraticasTelebras/220001701.pdf>. Acessoem: 25 Abr. 2008.

Apêndices

Apêndice A – Código `Autômato`

```
1  package parport;
2  import java.util.Vector;
3  import parport.Tarifador.tipo_chamada;
4
5  public class Automato extends Thread {
6
7  public enum tipo_estado {RECONHECENDO_CIRCUITO, AGUARDANDO_CHAMADA,
8  AGUARDANDO_DÍGITOS,SALVANDO_DÍGITOS, DETERMINANDO_TARIFA,
9  AGUARDANDO_ATENDIMENTO, VALIDANDO_ATENDIMENTO, TARIFANDO}
10
11 // Declaração das variaveis
12   tipo_estado estado;
13   char dígito_lido;
14   char Retorno_salva;
15   tipo_chamada chamada;
16   double tarifa;
17   Tarifador tarifador;
18   static Temporizador timer;
19   Exibe_Custo ExibeCusto;
20   static Vector<Character> vetor;
21
22 // Metodo run da thread Automato
23   public void run ()
24   {
25     timer = new Temporizador () ;
26     estado = tipo_estado.RECONHECENDO_CIRCUITO;
27 //   inicia a variavel estado como RECONHECENDO ESTADO
28     while (true)                          {
29 //   cria o objeto timer da classe temporizador
30       SalvaDígito salva = new SalvaDígito () ;
31 //   cria o objeto salva da classe SalvaDígito
32     switch (estado)
33     {
34     case RECONHECENDO_CIRCUITO :
35 // garante que o circuito foi iniciado corretamente
36     if (Paralela.estado_pino_15 () == 0)                  estado =
37 tipo_estado.AGUARDANDO_CHAMADA;           break;
38
39     case AGUARDANDO_CHAMADA :
40       if (Paralela.estado_pino_15 () != 0)                {
41
42         estado = tipo_estado.AGUARDANDO_DÍGITOS;
43         timer.Tempo (7000) ;                    vetor = new
44 Vector<Character> () ;
```

```
45          }
46          else
47            break;
48       case AGUARDANDO_DÍGITOS:                       if
49  (timer.RetornarCondicao () == 1 ||
50    Paralela.estado_pino_15 () == 0)
51          {                              System.out.println ("TIMEOUT OU
52  GANCHO NO AGUARDANDO_DÍGITOS") ;
53            estado = tipo_estado.RECONHECENDO_CIRCUITO;          }
54                       if ((dígito_lido = Paralela.dígito ()) != 120)
55            {
56            System.out.println ("AGUARDANDO_DÍGITOS") ;
57    Retorno_salva = salva.salvar_dígitos (dígito_lido) ;
58            vetor.addElement (Retorno_salva) ;
59            System.out.println ("dígito 1 "+vetor) ;            estado =
60  tipo_estado.SALVANDO_DÍGITOS;
61            if (timer.RetornarCondicao () == 0)
62            timer.cancel () ;
63            timer.Tempo (3000) ;                   }
64          break;
65
66       case SALVANDO_DÍGITOS :                       int DigMax = 6;
67
68          // número maximo de leituras
69          int Dig = 0;
70          // número de leituras realizadas
71          int x = 1;
72          // variavel para controle
73          int novo_dígito = 0;
74          // variavel para controle
75
76          while (estado == tipo_estado.SALVANDO_DÍGITOS)         {
77            while (Dig < DigMax)                        {
78              dígito_lido = Paralela.dígito () ;             if
79  (dígito_lido != 120 && novo_dígito == 1)
80            {                               Retorno_salva =
81  salva.salvar_dígitos (dígito_lido) ;
82            vetor.addElement (Retorno_salva) ;
83            novo_dígito = 0;
84            Dig++;
85            }                             if (dígito_lido == 120)
86                 {                              while (Paralela.dígito ()
87  == 120)
88  {                             novo_dígito = 0;
89  }
90              novo_dígito = 1;                        if
91  (timer.RetornarCondicao () == 0)
92              {
93              timer.cancel () ;
94              timer.Tempo (3000) ;
95              }
96              else
97              {
98              Dig = DigMax+1;
99              break;
100             }
101           }
102 }
103
104
```

```
105
106 if (Dig == DigMax && x == 1 && timer.RetornarCondicao () == 0)
107 {                       DigMax = 6 + Tarifador.número_dígitos (vetor) ;
108        chamada = Tarifador.tipo_chamada () ;          x = 0;
109             novo_dígito = 0;
110 }
111 else if (Dig == DigMax && x == 0 && timer.RetornarCondicao () == 0)
112
113        {                                estado =
114 tipo_estado.DETERMINANDO_TARIFA;
115        }                             else if (timer.RetornarCondicao ()
116 == 1 ||
117          Paralela.estado_pino_15 () != 0)             {
118          estado = tipo_estado.AGUARDANDO_CHAMADA;
119        }
120     }                             break;
121
122     case DETERMINANDO_TARIFA :                    if
123 (timer.RetornarCondicao () == 0)
124          timer.cancel () ;                     if
125 (Paralela.estado_pino_15 () == 0)
126 {                               estado =
127 tipo_estado.RECONHECENDO_CIRCUITO;           break;
128
129 }                             Tarifador.Prefixos (vetor) ;
130        tarifador = new Tarifador () ;
131        tarifador.start () ;
132        estado = tipo_estado.AGUARDANDO_ATENDIMENTO;          break;
133
134     case AGUARDANDO_ATENDIMENTO:                     if
135 (timer.RetornarCondicao () == 0)                timer.cancel () ;
136                    timer.Tempo (90000) ;
137        while (Paralela.estado_pino_15 () != 0)             {
138                      if (timer.RetornarCondicao () == 1)
139
140 {                               estado = tipo_estado.AGUARDANDO_CHAMADA;
141               break;
142 }                           }                               if
143 (Paralela.estado_pino_15 () == 0 &&
144          estado == tipo_estado.AGUARDANDO_ATENDIMENTO)
145        {                                estado =
146 tipo_estado.VALIDANDO_ATENDIMENTO;           if
147 (timer.RetornarCondicao () == 0)             timer.cancel () ;
148                 timer.Tempo (150) ;
149 }
150        else
151          break;
152
153     case VALIDANDO_ATENDIMENTO :
154        if (timer.RetornarCondicao () == 1)                  {
155                      if (Paralela.estado_pino_15 () != 0)
156
157 estado = tipo_estado.TARIFANDO;
158          else                            estado =
159 tipo_estado.AGUARDANDO_CHAMADA;          }
160   break;
161
162     case TARIFANDO :                          if
163 (Paralela.estado_pino_15 () != 0)
164 {
```

```
165       timer.cancel () ;
166       ExibeCusto = new Exibe_Custo () ;
167       ExibeCusto.start () ;
168       while (Paralela.estado_pino_15 () != 0)
169         {
170           estado = tipo_estado.TARIFANDO;                    }
171       ExibeCusto.stop () ;
172       estado = tipo_estado.AGUARDANDO_CHAMADA;
173       }                              else
174     estado = tipo_estado.AGUARANDO_CHAMADA;
175   break;
176   }
177   }
178   }
179
180   public static void main (String args[])
181   {
182     Automato teste ;
183     teste = new Automato () ;
184     teste.start () ;
185   }
186   }
```

Apêndice B – Código Tarifador

```
1   package parport;
2   import java.util.Vector;
3   import java.text.*;
4   import java.util.*;
5   import java.io.BufferedReader;
6   import java.io.File;
7   import java.io.FileReader;
8   import java.io.IOException;
9
10  public class Tarifador extends Thread{
11
12  public enum tipo_chamada{LOCAL_FIXO, VC1, VC2, VC3, LDN_FIXO, LDI,
13  ESPECIAL}
14    static tipo_chamada chamada;
15  public enum tipo_modulacao { BRT_NORMAL, BRT_REDUZIDO,
16  BRT_DIFERENCIADA,BRT_SUPERREDUZIDO,BRT_NORMAL_CEL,
17  BRT_REDUZIDO_CEL,GVT_NORMAL_CEL,GVT_REDUZIDO_CEL,
18  GVT_SUPERREDUZIDO_CEL,BRT_LDI,GVT_LDI,GVT_LDN,BRT_LOCAL_NORMAL,BRT_LOC
19  AL_REDUZIDO, GVT_LDN_NORMAL_CEL,
20  GVT_LDN_REDUZIDO_CEL,GVT_LOCAL_NORMAL, GVT_LOCAL_REDUZIDO}
21    public enum prestadora{BRT, tim, vivo, claro, nextel};
22    static prestadora prest;
23    static tipo_modulacao mod_horaria;
24    static Vector<Character> prefixo = new Vector<Character> () ;
25    public static double lat,log;
26    static double Distância = 1; // distância geodesica em km
27    static int Degrau;
28    static int operadora = 0;//invalida
29    static double tarifa;
30    public static char CA = '8'; // codigo de area
31    public static int P_usuario = 1; // prestadora do usuario
32
33  public static int número_dígitos (Vector<Character> vetor)
34    {
35      int DigMax = 0;
36      int operadora = Tarifador.Operadora (vetor) ;
37
38  //      CHAMADA INTERNACIONAL
39
40      if (vetor.get (0) == '0' && vetor.get (1) == '0')
41      {
42        chamada = tipo_chamada.LDI;
43        DigMax = 4;
44      }
45
46  //   CHAMADA PARA 0800 e 0300
47
48  else if ((vetor.get (1) == '8' || vetor.get (1) == '3') && (vetor.get
49  (0) == '0'))
50      {
51        chamada = tipo_chamada.ESPECIAL;
52        DigMax = 0;
53      }
```

```
54
55  //     CHAMADA LONGA DISTÂNCIA NACIONAL
56
57       else if (vetor.get (0) == '0' && operadora != '0')
58       {
59         DigMax = 0;
60         if (vetor.get (5) == '9' || vetor.get (5) == '8')
61         {
62           if (vetor.get (3) == '4' && vetor.get (4) != CA)
63           {
64             chamada = tipo_chamada.VC2;
65           }
66           else if (vetor.get (3) != '4')
67           {
68             chamada = tipo_chamada.VC3;
69           }
70         }
71         else
72         {
73           chamada = tipo_chamada.LDN_FIXO;
74           DigMax = 3;
75         }
76       }
77
78  //     CHAMADA LOCAL
79
80       else if (vetor.get (0) != 0)
81       {
82         chamada = tipo_chamada.LOCAL_FIXO;
83         DigMax = 0;
84  if ((vetor.get (0) == '9' || vetor.get (0) == '8' || vetor.get (0) ==
85  '7') && (vetor.get (1) != '0'))
86         {
87           chamada = tipo_chamada.VC1;
88  if (vetor.get (0) == '9' && (vetor.get (1) == '1' || vetor.get (1) ==
89  '2' || vetor.get (1) == '3' || vetor.get (1) == '4'))
90           {
91             prest = prestadora.vivo;
92           }
93           else if (vetor.get (0) == '8' && vetor.get (1) == '8')
94           {
95             prest = prestadora.claro;
96           }
97  else if (vetor.get (0) == '8' && (vetor.get (1) == '4' || vetor.get
98  (1) == '5'))
99           {
100            prest = prestadora.BRT;
101          }
102 else if (vetor.get (0) == '9' && (vetor.get (1) == '6' || vetor.get
103 (1) == '7' || vetor.get (1) == '8' || vetor.get (1) == '9'))
104          {
105            prest = prestadora.tim;
106          }
107          else if (vetor.get (0) == '7' && vetor.get (1) == '8')
108          {
109            prest = prestadora.nextel;
110          }
111        }
112        if (vetor.get (0) == '9' && vetor.get (1) == '0')
113        {
```

```
114             chamada = tipo_chamada.ESPECIAL;
115           }
116   System.out.println ("chamada dentro do tarifador "+chamada) ;
117       }
118       System.out.println ("acessando a função número_dígitos") ;
119       return DigMax;
120     }
121
122   private static int Operadora (Vector<Character> vetor)
123     {
124       String op = "desconhecida";
125       if (vetor.get (0) == '0')
126       {
127         if (vetor.get (1) == '1' && vetor.get (2) == '4')
128         {
129           operadora = 1; // brasil telecom
130           op = "brasil telecom";
131         }
132         if (vetor.get (1) == '2' && vetor.get (2) == '5')
133         {
134           operadora = 2; //gvt
135           op = "gvt";
136         }
137       }
138
139       System.out.println ("Operadora: "+op) ;
140       return operadora;
141     }
142
143   public static tipo_chamada tipo_chamada () {
144
145       return chamada;
146     }
147
148   public static void Prefixos (Vector<Character> vetor)
149     {
150       String pre;
151       //String pre = new String (vetor.toString ()) ;
152       if (chamada == tipo_chamada.LOCAL_FIXO)
153       {
154         pre = "04";
155         pre += CA;
156         pre += Character.toString (vetor.get (0)) ;
157         pre += Character.toString (vetor.get (1)) ;
158         pre += Character.toString (vetor.get (2)) ;
159         pre += Character.toString (vetor.get (3)) ;
160       }
161       else
162       {
163         pre = Character.toString (vetor.get (3)) ;
164         pre += Character.toString (vetor.get (4)) ;
165         pre += Character.toString (vetor.get (5)) ;
166         pre += Character.toString (vetor.get (6)) ;
167       }
168       if (chamada == tipo_chamada.LDN_FIXO)
169       {
170         pre += Character.toString (vetor.get (7)) ;
171         pre += Character.toString (vetor.get (8)) ;
172       }
173
```

```
174       try
175       {
176   File fArquivo = new File ("/home/oem/Documentos/CE_F_154912.TXT") ;
177         FileReader arquivo = new FileReader (fArquivo) ;
178         BufferedReader buffer = new BufferedReader (arquivo) ;
179         String LinhaArquivo = null;
180   String prefixo = pre; //.substring (3, 8) ;//"483254"; // prefixo
181   digitado pelo usuario
182         String n = null;
183         int ini = 116;
184         int fim = 122;
185         System.out.println ("prefixo alterado "+prefixo) ;
186         do{
187           if (prefixo.equals (n))
188           {
189             System.out.println ("prefixo "+n) ;
190             n = LinhaArquivo.substring (ini-56, fim-13) ;
191             System.out.println ("Nome da localidade "+n) ;
192             n = LinhaArquivo.substring (ini+45,fim+47) ;
193             lat = new Double (n) ;
194             System.out.println ("latitude "+lat) ;
195             n = LinhaArquivo.substring (ini+58,fim+60) ;
196             log = new Double (n) ;
197             System.out.println ("longitude "+log) ;
198             break;
199           }
200           else
201           {
202             LinhaArquivo = buffer.readLine () ;
203             if (LinhaArquivo != null) {
204               n = LinhaArquivo.substring (ini, fim) ;
205             }
206           }
207         }while (LinhaArquivo != null) ;
208         arquivo.close () ;
209         buffer.close () ;
210       }
211       catch (IOException ioe)
212       {
213       }
214     }
215
216   public static tipo_modulacao determina_horario () throws Exception{
217   SimpleDateFormat formatoH = new SimpleDateFormat ("HH:mm:ss") ;
218       SimpleDateFormat formatoS = new SimpleDateFormat ("E") ;
219       SimpleDateFormat formatoD = new SimpleDateFormat ("ddMM") ;
220
221       String hora1 = "06:00:00"; Date d1 = formatoH.parse (hora1) ;
222       String hora2 = "23:59:59"; Date d2 = formatoH.parse (hora2) ;
223       String hora3 = "00:00:00"; Date d3 = formatoH.parse (hora3) ;
224       String hora4 = "05:59:59"; Date d4 = formatoH.parse (hora4) ;
225       String hora5 = "06:59:59"; Date d5 = formatoH.parse (hora5) ;
226       String hora6 = "21:00:00"; Date d6 = formatoH.parse (hora6) ;
227       String hora7 = "14:00:00"; Date d7 = formatoH.parse (hora7) ;
228       String hora8 = "07:00:00"; Date d8 = formatoH.parse (hora8) ;
229       String hora9 = "08:59:59"; Date d9 = formatoH.parse (hora9) ;
230   String hora10 = "12:00:00"; Date d10 = formatoH.parse (hora10) ;
231   String hora11 = "13:59:59"; Date d11 = formatoH.parse (hora11) ;
232   String hora12 = "18:00:00"; Date d12 = formatoH.parse (hora12) ;
233   String hora13 = "20:59:59"; Date d13 = formatoH.parse (hora13) ;
```

```
String hora14 = "09:00:00"; Date d14 = formatoH.parse (hora14) ;
String hora15 = "11:59:59"; Date d15 = formatoH.parse (hora15) ;
String hora16 = "17:59:59"; Date d16 = formatoH.parse (hora16) ;
String hora17 = "08:00:00"; Date d17 = formatoH.parse (hora17) ;
String hora18 = "19:59:59"; Date d18 = formatoH.parse (hora18) ;
     String dom = "Dom";
     String sab = "Sáb";

     String H = formatoH.format (new Date ()) ;
     Date hora = formatoH.parse (H) ;
     String dia = formatoS.format (new Date ()) ;

//   01 de Janeiro ConfraternizaÃ§Ã£o Universal
//   21 de Abril   Tirádentes
//   01 de Maio  Dia do Trabalho
//   07 de Setembro  IndependÃªncia do Brasil
//   12 de Outubro N. Sra. de Aparecida
//   02 de Novembro  Finados
//   15 de Novembro  ProclamaÃ§Ã£o da RepÃºblica
//   25 de Dezembro  Natal

//   =-=-=-LDN-=-=-=-=-=-=-LDN-=-=-=-=-=-=-LDN-=-=-=-=-=-=-LDN
//   Fixo - Fixo - Domgingos e Feriados

     if (chamada == tipo_chamada.LDN_FIXO)
     {
if ("0101" == formatoD.format (new Date ()) || "2104" ==
formatoD.format (new Date ()) || "0105" == formatoD.format (new Date
()) || "0709" == formatoD.format (new Date ()) ||"1210" ==
formatoD.format (new Date ()) || "0211" == formatoD.format (new Date
()) || "1511" == formatoD.format (new Date ()) || "2512" ==
formatoD.format (new Date ()) || dom == dia)
       {
         if (operadora == 1) //brasil telecom
         {
if (-1 == d1.compareTo (hora) && 1 == d2.compareTo (hora))
//            hora entre 06:00 e 23:59:59

           {
mod_horaria = tipo_modulacao.BRT_REDUZIDO;

           }
else if (-1 == d3.compareTo (hora) && 1 == d4.compareTo (hora))

//            hora entre 00:00 e 05:59:59

           {
mod_horaria = tipo_modulacao.BRT_SUPERREDUZIDO;
           }
         }
         else
         {
           mod_horaria = tipo_modulacao.GVT_LDN;
         }

       }

//     Fixo - Fixo - Segunda a Sabado

       if (dom != dia)
```

```
294         {
295   if (operadora == 1 && "Sab" != formatoS.format (new Date ())) //brasil
296   telecom
297             {
298   if ((-1 == d8.compareTo (hora) && 1 == d9.compareTo (hora)) || (-1 ==
299   d10.compareTo (hora) && 1 == d11.compareTo (hora)) || (-1 ==
300   d12.compareTo (hora) && 1 == d13.compareTo (hora)))
301
302   //            hora entre 7:00 e 08:59:59
303   //            hora entre 12:00 e 13:59:59
304   //          hora entre 18:00 e 20:59:59
305
306   {
307   mod_horaria = tipo_modulacao.BRT_NORMAL;
308               }
309   else if ((-1 == d14.compareTo (hora) && 1 == d15.compareTo (hora)) ||
310       (-1 == d7.compareTo (hora) && 1 == d16.compareTo (hora)))
311   //          hora entre 9:00 e 11:59:59
312   //            hora entre 14:00 e 17:59:59
313   {
314   mod_horaria = tipo_modulacao.BRT_DIFERENCIADA;
315               }
316   else if ((-1 == d1.compareTo (hora) && 1 == d5.compareTo (hora)) || (-
317   1 == d6.compareTo (hora) && 1 == d2.compareTo (hora)))
318    //            hora entre 6:00 e 06:59:59
319    //            hora entre 21:00 e 23:59:59
320               {
321   mod_horaria = tipo_modulacao.BRT_REDUZIDO;
322               }
323   else if ((-1 == d3.compareTo (hora) && 1 == d5.compareTo (hora)))
324
325   //            hora entre 00:00 e 05:59:59
326   {
327   mod_horaria = tipo_modulacao.BRT_SUPERREDUZIDO;
328               }
329             }
330             else if (operadora == 1)
331             {
332   if ((-1 == d8.compareTo (hora) && 1 == d11.compareTo (hora)))
333
334   //            hora entre 7:00 e 13:59:59
335               {
336   mod_horaria = tipo_modulacao.BRT_NORMAL;
337               }
338   else if ((-1 == d1.compareTo (hora) && 1 == d5.compareTo (hora)) || (-
339   1 == d7.compareTo (hora) && 1 == d2.compareTo (hora)))
340   //            hora entre 6:00 e 6:59:59
341   //            hora entre 14:00 e 23:59:59
342               {
343   mod_horaria = tipo_modulacao.BRT_REDUZIDO;
344               }
345   else if (-1 == d3.compareTo (hora) && 1 == d5.compareTo (hora))
346
347   //            hora entre 00:00 e 05:59:59
348               {
349   mod_horaria = tipo_modulacao.BRT_SUPERREDUZIDO;
350               }
351             }
352             else
353             {
```

```
354             mod_horaria = tipo_modulacao.GVT_LDN;
355           }
356         }
357       }
358
359   //     Fixo – Móvel - Domgingos e Feriados
360
361   if (chamada == tipo_chamada.VC1 || chamada == tipo_chamada.VC2 ||
362   chamada == tipo_chamada.VC3)
363       {
364   if ("0101" == formatoD.format (new Date ()) || "2104" ==
365   formatoD.format (new Date ()) || "0105" == formatoD.format (new Date
366   ()) || "0709" == formatoD.format (new Date ()) || "1210" ==
367   formatoD.format (new Date ()) || "0211" == formatoD.format (new Date
368   ()) || "1511" == formatoD.format (new Date ()) || "2512" ==
369   formatoD.format (new Date ()) || dom == dia)
370   //       hora entre 00:00 e 23:59:59
371         {
372           if (operadora == 1) //brasil telecom
373           {
374   mod_horaria = tipo_modulacao.BRT_REDUZIDO_CEL;
375           }
376   else if (operadora == 2 && chamada != tipo_chamada.VC1)
377           {
378   if (-1 == d14.compareTo (hora) && 1 == d16.compareTo (hora))
379             {
380   mod_horaria = tipo_modulacao.GVT_LDN_NORMAL_CEL;
381   }
382             else
383             {
384   mod_horaria = tipo_modulacao.GVT_LDN_REDUZIDO_CEL;
385             }
386           }
387   else if (operadora == 2 && chamada == tipo_chamada.VC1)
388           {
389   mod_horaria = tipo_modulacao.GVT_REDUZIDO_CEL;
390           }
391         }
392
393   //     Fixo – Móvel - Segunda a Sabado
394
395         else
396         {
397   if (operadora == 2 && chamada != tipo_chamada.VC1) // gvt
398           {
399   if (-1 == d14.compareTo (hora) && 1 == d16.compareTo (hora))
400             {
401   mod_horaria = tipo_modulacao.GVT_LDN_NORMAL_CEL;
402             }
403             else
404             {
405   mod_horaria = tipo_modulacao.GVT_LDN_REDUZIDO_CEL;
406             }
407           }
408   else if (operadora == 2 && chamada == tipo_chamada.VC1)
409           {
410             if (operadora == 2 && sab == dia) // gvt
411             {
412   if (-1 == d8.compareTo (hora) && 1 == d6.compareTo (hora))
413   //              hora entre 07:00 e 20:59:59
```

```
414                 {
415   mod_horaria = tipo_modulacao.GVT_REDUZIDO_CEL;
416                 }
417                 else
418   //              hora entre 21:00 e 06:59:59
419                 {
420   mod_horaria = tipo_modulacao.GVT_SUPERREDUZIDO_CEL;
421                 }
422               }
423               else if (operadora == 2) // gvt
424               {
425   if (-1 == d8.compareTo (hora) && 1 == d16.compareTo (hora))
426   //              hora entre 07:00 e 17:59:59
427                 {
428   mod_horaria = tipo_modulacao.GVT_NORMAL_CEL;
429                 }
430   else if (-1 == d12.compareTo (hora) && 1 == d6.compareTo (hora))
431   //              hora entre 18:00 e 20:59:59
432                 {
433   mod_horaria = tipo_modulacao.GVT_REDUZIDO_CEL;
434                 }
435                 else
436   //              hora entre 21:00 e 06:59:59
437                 {
438   mod_horaria = tipo_modulacao.GVT_SUPERREDUZIDO_CEL;
439                 }
440               }
441             }
442   else if (operadora == 1 || chamada == tipo_chamada.VC1) // brasil
443   telecom
444             {
445   if (-1 == d8.compareTo (hora) && 1 == d6.compareTo (hora))
446   //              hora entre 07:00 e 20:59:59
447               {
448                 mod_horaria =
449     tipo_modulacao.BRT_NORMAL_CEL;
450               }
451               else
452   //              hora entre 00:00 e 06:59:59
453   {
454    mod_horaria =
455    tipo_modulacao.BRT_REDUZIDO_CEL;
456               }
457             }
458         }
459       }
460
461   //   =-LDI-=-=-=-=-=-=-LDI-=-=-=-=-=-=-=-LDI-=-=-=-=-=-=-=-LDI-=-=-=-
462
463       if (chamada == tipo_chamada.LDI)
464       {
465         if (operadora == 1) // brasil telecom
466         {
467           mod_horaria = tipo_modulacao.BRT_LDI;
468         }
469         else
470         {
471           mod_horaria = tipo_modulacao.GVT_LDI;
472         }
473       }
```

```
474
475 //   =-=-=-Local-=-=-=-=-=-=-Local-=-=-=-=-=-=-Local-=-=-=-=-=-=-=-
476
477       if (chamada == tipo_chamada.LOCAL_FIXO)
478       {
479         if (P_usuario == 1)
480         {
481 if (0 != dia.compareTo (sab) && 0 != dia.compareTo (dom))
482             {
483 if (-1 == d1.compareTo (hora) && 1 == d2.compareTo (hora))
484               {
485 mod_horaria = tipo_modulacao.BRT_LOCAL_NORMAL;
486               }
487 else if (-1 == d3.compareTo (hora) && 1 == d4.compareTo (hora))
488               {
489 mod_horaria = tipo_modulacao.BRT_LOCAL_REDUZIDO;
490               }
491             }
492             else if (0 == dia.compareTo (sab))
493             {
494 if (-1 == d1.compareTo (hora) && 1 == d11.compareTo (hora))
495               {
496 mod_horaria = tipo_modulacao.BRT_LOCAL_NORMAL;
497               }
498 else if ((-1 == d3.compareTo (hora) && 1 == d4.compareTo (hora)) || (-
499 1 == d7.compareTo (hora) && 1 == d2.compareTo (hora)))
500
501 //            hora entre 6:00 e 6:59:59
502 //            hora entre 14:00 e 23:59:59
503
504               {
505 mod_horaria = tipo_modulacao.BRT_LOCAL_REDUZIDO;
506               }
507             }
508             else if (0 == dia.compareTo (dom))
509             {
510 mod_horaria = tipo_modulacao.BRT_LOCAL_REDUZIDO;
511             }
512         }
513         else
514         {
515 if (0 != dia.compareTo (sab) && 0 != dia.compareTo (dom))
516             {
517 if (-1 == d17.compareTo (hora) && 1 == d18.compareTo (hora))
518               {
519 mod_horaria = tipo_modulacao.GVT_LOCAL_NORMAL;
520               }
521               else
522               {
523 mod_horaria = tipo_modulacao.GVT_LOCAL_REDUZIDO;
524               }
525             }
526             else
527             {
528 mod_horaria = tipo_modulacao.GVT_LOCAL_REDUZIDO;
529             }
530         }
531       }
532     return mod_horaria;
533 }
```

```
534
535   public static double geodesica (double lat, double log) {
536
537        double K1g = lat/1000000;
538        double L1g = log/1000000;
539        double K2g = 28.012422;
540   //     latitude da origem serÃ¡ lido de um arquivo
541        double L2g = 48.364531;
542   //     latitude do destino serÃ¡ lido de um arquivo
543        double K1 = Math.toRadians (K1g) ;
544        double L1 = Math.toRadians (L1g) ;
545        double K2 = Math.toRadians (K2g) ;
546        double L2 = Math.toRadians (L2g) ;
547        double X1 = Math.sin (K1) ;
548        double X2 = Math.sin (K2) ;
549        double Y1 = (Math.cos (K1)) * (Math.cos (K2)) ;
550   double Y2 = ((Math.sin (L1)) * (Math.sin (L2)) + (Math.cos (L1)) *
551   (Math.cos (L2))) ;
552        double Z = ((X1*X2) + (Y1*Y2)) ;
553        Distância = (111.18 * Math.acos (Z)) ;
554        Distância = Distância * 57.2957995;
555        if (Distância <= 50)
556        {
557          Degrau = 1;
558        }
559        else if (Distância > 50 && Distância <= 100)
560        {
561          Degrau = 2;
562        }
563        else if (Distância >100 && Distância <= 300)
564        {
565          Degrau = 3;
566        }
567        else
568        {
569          Degrau = 4;
570        }
571        System.out.println ("Distância: "+Distância) ;
572        System.out.println ("Degrau: "+Degrau) ;
573        return (Distância) ;
574      }
575      public static double determina_tarifa () {
576        String lido = null;
577        int loop = 1;
578        int VC = 0;
579        try {
580          Tarifador.determina_horario () ;
581        }
582   catch (Exception e)
583   {
584        }
585        switch (mod_horaria)
586        {
587        case BRT_NORMAL_CEL       :
588        {
589          switch (chamada)
590          {
591          case VC1 :
592          {
593            System.out.println ("akidgjjgj") ;
```

```
          System.out.println ("prestadora: "+prest) ;
          switch (prest)
          {
          case BRT:
          {
            try
            {
FileReader arquivo = new FileReader ("/home/oem/tarifas/BRT_vc1.txt")
;
BufferedReader buffer = new BufferedReader (arquivo) ;
               String LinhaArquivo = null;
               LinhaArquivo = buffer.readLine () ;
               if (LinhaArquivo != null)
{
lido = LinhaArquivo.substring (0,4) ;
               }
               tarifa = new Double (lido) ;
               arquivo.close () ;
               buffer.close () ;
            }
            catch (IOException ioe)
            {
            }
          }break;
          case tim:{
            try
            {
FileReader arquivo = new FileReader ("/home/oem/tarifas/tim_vc1.txt")
;
BufferedReader buffer = new BufferedReader (arquivo) ;
               String LinhaArquivo = null;
               LinhaArquivo = buffer.readLine () ;
               if (LinhaArquivo != null)
{
lido = LinhaArquivo.substring (0,4) ;
               }
               tarifa = new Double (lido) ;
               arquivo.close () ;
               buffer.close () ;
            }
            catch (IOException ioe)
            {
            }
          }break;
          case vivo:{
            try
            {
FileReader arquivo = new FileReader ("/home/oem/tarifas/vivo_vc1.txt")
;
BufferedReader buffer = new BufferedReader (arquivo) ;
               String LinhaArquivo = null;
               LinhaArquivo = buffer.readLine () ;
               if (LinhaArquivo != null)
{
lido = LinhaArquivo.substring (0,4) ;
               }
               tarifa = new Double (lido) ;
               arquivo.close () ;
               buffer.close () ;
            }
```

```
654             catch (IOException ioe)
655             {
656             }
657           }break;
658           case claro:{
659             try
660             {
661 FileReader arquivo = new FileReader
662 ("/home/oem/tarifas/claro_vc1.txt") ;
663 BufferedReader buffer = new BufferedReader (arquivo) ;
664               String LinhaArquivo = null;
665               LinhaArquivo = buffer.readLine () ;
666               if (LinhaArquivo != null)
667 {
668 lido = LinhaArquivo.substring (0,4) ;
669               }
670               tarifa = new Double (lido) ;
671               arquivo.close () ;
672               buffer.close () ;
673             }
674             catch (IOException ioe)
675             {
676             }
677           }break;
678           case nextel:{
679             try
680             {
681 FileReader arquivo = new FileReader
682 ("/home/oem/tarifas/nextel_vc1.txt") ;
683 BufferedReader buffer = new BufferedReader (arquivo) ;
684               String LinhaArquivo = null;
685               LinhaArquivo = buffer.readLine () ;
686               if (LinhaArquivo != null) {
687 lido = LinhaArquivo.substring (0,4) ;
688               }
689               tarifa = new Double (lido) ;
690               arquivo.close () ;
691               buffer.close () ;
692             }
693             catch (IOException ioe)
694             {
695             }
696           }break;
697           default:break;
698           }
699         }break;
700         case VC2 :VC = 2; break;
701         case VC3 :VC = 3; break;
702         default: break;
703         }
704       break;
705       }
706       case BRT_REDUZIDO_CEL   :
707       {
708         switch (chamada)
709         {
710         case VC1 :
711         {
712           System.out.println ("prestadora: "+prest) ;
713           switch (prest)
```

```
714             {
715             case BRT:
716             {
717               try
718               {
719   FileReader arquivo = new FileReader ("/home/oem/tarifas/BRT_vc1.txt")
720   ;
721   BufferedReader buffer = new BufferedReader (arquivo) ;
722                 String LinhaArquivo = null;
723                 LinhaArquivo = buffer.readLine () ;
724                 LinhaArquivo = buffer.readLine () ;
725                 if (LinhaArquivo != null) {
726   lido = LinhaArquivo.substring (0,4) ;
727                 }
728                 tarifa = new Double (lido) ;
729                 arquivo.close () ;
730                 buffer.close () ;
731               }
732               catch (IOException ioe)
733               {
734               }
735             }break;
736             case tim:{
737               try
738               {
739   FileReader arquivo = new FileReader ("/home/oem/tarifas/tim_vc1.txt")
740   ;
741   BufferedReader buffer = new BufferedReader (arquivo) ;
742                 String LinhaArquivo = null;
743                 LinhaArquivo = buffer.readLine () ;
744                 LinhaArquivo = buffer.readLine () ;
745                 if (LinhaArquivo != null) {
746   lido = LinhaArquivo.substring (0,4) ;
747                 }
748                 tarifa = new Double (lido) ;
749                 arquivo.close () ;
750                 buffer.close () ;
751               }
752               catch (IOException ioe)
753               {
754               }
755             }break;
756             case vivo:{
757               try
758               {
759   FileReader arquivo = new FileReader ("/home/oem/tarifas/vivo_vc1.txt")
760   ;
761   BufferedReader buffer = new BufferedReader (arquivo) ;
762                 String LinhaArquivo = null;
763                 LinhaArquivo = buffer.readLine () ;
764                 LinhaArquivo = buffer.readLine () ;
765                 if (LinhaArquivo != null) {
766   lido = LinhaArquivo.substring (0,4) ;
767                 }
768                 tarifa = new Double (lido) ;
769                 arquivo.close () ;
770                 buffer.close () ;
771               }
772               catch (IOException ioe)
773               {
```

```
774             }
775           }break;
776           case claro:{
777             try
778             {
779                FileReader arquivo = new                    FileReader
780  ("/home/oem/tarifas/claro_vc1.txt") ;
781  BufferedReader buffer = new BufferedReader (arquivo) ;
782                String LinhaArquivo = null;
783                LinhaArquivo = buffer.readLine () ;
784                LinhaArquivo = buffer.readLine () ;
785                if (LinhaArquivo != null) {
786  lido = LinhaArquivo.substring (0,4) ;
787                }
788                tarifa = new Double (lido) ;
789                arquivo.close () ;
790                buffer.close () ;
791             }
792             catch (IOException ioe)
793             {
794             }
795           }break;
796           case nextel:{
797             try
798             {
799  FileReader arquivo = new FileReader
800  ("/home/oem/tarifas/nextel_vc1.txt") ;
801  BufferedReader buffer = new BufferedReader (arquivo) ;
802                String LinhaArquivo = null;
803                LinhaArquivo = buffer.readLine () ;
804                LinhaArquivo = buffer.readLine () ;
805                if (LinhaArquivo != null) {
806  lido = LinhaArquivo.substring (0,4) ;
807                }
808                tarifa = new Double (lido) ;
809                arquivo.close () ;
810                buffer.close () ;
811             }
812             catch (IOException ioe)
813             {
814             }
815           }break;
816           default:break;
817           }
818         }break;
819         case VC2 :VC = 2; break;
820         case VC3 :VC = 3; break;
821         default: break;
822         }
823       break;
824       }
825       case BRT_LOCAL_NORMAL   :
826       {
827         try
828         {
829  FileReader arquivo = new FileReader
830  ("/home/oem/tarifas/BRT_local_normal.txt") ;
831  System.out.println ("leu arquivo BRT_local_normal.txt") ;
832  BufferedReader buffer = new BufferedReader (arquivo) ;
833           String LinhaArquivo = null;
```

```
834             LinhaArquivo = buffer.readLine () ;
835             if (LinhaArquivo != null) {
836               lido = LinhaArquivo.substring (0,4) ;
837             }
838             tarifa = new Double (lido) ;
839             arquivo.close () ;
840             buffer.close () ;
841           }
842           catch (IOException ioe)
843           {
844           }
845           break;
846         }
847         case BRT_LOCAL_REDUZIDO   :
848         {
849           try
850           {
851   FileReader arquivo = new
852   FileReader ("/home/oem/tarifas/BRT_local_reduzido.txt") ;
853   System.out.println ("leu arquivo BRT_local_reduzido.txt") ;
854   BufferedReader buffer = new BufferedReader (arquivo) ;
855             String LinhaArquivo = null;
856             LinhaArquivo = buffer.readLine () ;
857             if (LinhaArquivo != null) {
858               lido = LinhaArquivo.substring (0,4) ;
859             }
860             tarifa = new Double (lido) ;
861             arquivo.close () ;
862             buffer.close () ;
863           }
864           catch (IOException ioe)
865           {
866           }
867           break;
868         }
869         case BRT_LDI         :
870           break;
871         case BRT_NORMAL      :
872         {
873           Tarifador.geodesica (lat, log) ;
874           try
875           {
876   FileReader arquivo = new FileReader
877   ("/home/oem/tarifas/BRT_normal.txt") ;
878             System.out.println ("leu arquivo BRT_normal.txt") ;
879   BufferedReader buffer = new BufferedReader (arquivo) ;
880             String LinhaArquivo = null;
881             while (loop < Degrau+1)
882             {
883               loop ++;
884               LinhaArquivo = buffer.readLine () ;
885               if (LinhaArquivo != null) {
886                  lido = LinhaArquivo.substring (0,4) ;
887               }
888             }
889             tarifa = new Double (lido) ;
890             arquivo.close () ;
891             buffer.close () ;
892           }
893           catch (IOException ioe)
```

```
        {
        }
        break;
      }
      case BRT_REDUZIDO       :
      {
        Tarifador.geodesica (lat, log) ;
        try
        {
          FileReader arquivo = new FileReader
("/home/oem/tarifas/BRT_reduzido.txt") ;
System.out.println ("leu arquivo BRT_reduzido.txt") ;
BufferedReader buffer = new BufferedReader (arquivo) ;
          String LinhaArquivo = null;
          while (loop < Degrau+1)
          {
            loop ++;
            LinhaArquivo = buffer.readLine () ;
            if (LinhaArquivo != null) {
              lido = LinhaArquivo.substring (0,4) ;
            }
          }
          tarifa = new Double (lido) ;
          arquivo.close () ;
          buffer.close () ;
        }
        catch (IOException ioe)
        {
        }
        break;
      }
      case BRT_DIFERENCIADA   :
      {
        Tarifador.geodesica (lat, log) ;
        try
        {
FileReader arquivo = new FileReader
("/home/oem/tarifas/BRT_diferenciada.txt") ;
System.out.println ("leu arquivo BRT_diferenciada.txt") ;
BufferedReader buffer = new BufferedReader (arquivo) ;
          String LinhaArquivo = null;
          while (loop < Degrau+1)
          {
            loop ++;
            LinhaArquivo = buffer.readLine () ;
            if (LinhaArquivo != null) {
              lido = LinhaArquivo.substring (0,4) ;
            }
          }
          tarifa = new Double (lido) ;
          arquivo.close () ;
          buffer.close () ;
        }
        catch (IOException ioe)
        {
        }
        break;
      }
      case BRT_SUPERREDUZIDO    :
      {
```

```
954         Tarifador.geodesica (lat, log) ;
955         try
956         {
957   FileReader arquivo = new
958   FileReader ("/home/oem/tarifas/BRT_superreduzida.txt") ;
959   System.out.println ("leu arquivo BRT_superreduzida.txt") ;
960   BufferedReader buffer = new BufferedReader (arquivo) ;
961             String LinhaArquivo = null;
962             while (loop < Degrau+1)
963             {
964               loop ++;
965               LinhaArquivo = buffer.readLine () ;
966               if (LinhaArquivo != null) {
967                  lido = LinhaArquivo.substring (0,4) ;
968               }
969             }
970             tarifa = new Double (lido) ;
971             arquivo.close () ;
972             buffer.close () ;
973         }
974         catch (IOException ioe)
975         {
976       break;
977       }
978       default: break;
979       }
980       System.out.println ("tarifa: "+tarifa) ;
981       return tarifa;
982     }
983   public void run () {
984       System.out.println ("iníciou thread tarifador") ;
985       Tarifador.determina_tarifa () ;
986     }
987   }
988
```

Apêndice C - Código do Exibe Custo

```
1  package parport;
2  import java.text.DecimalFormat;
3  import parport.Tarifador.tipo_modulacao;
4  public class Exibe_Custo extends Thread{
5
6     static Temporizador timer_ec;
7     static Tarifador tarifador;
8     static double custo;
9     int x = 1;
10    Double preco = Tarifador.tarifa;
11    Double nova_tarifa;
12    tipo_modulacao mod = tarifador.mod_horaria;
13    tipo_modulacao mod_nova;
14    DecimalFormat real = new DecimalFormat ("0.00") ;
15
16    public void run () {
17       tarifador = new Tarifador () ;
18       timer_ec = new Temporizador () ;
19
20       timer_ec.Tempo (1000) ;
21       while (x == 1)
22       {
23          if (timer_ec.RetornarCondicao () == 1)
24          {
25             x = 0;
26             timer_ec.cancel () ;
27             timer_ec.Tempo (27000) ;
28             custo = (preco/2) ;
29             System.out.println ("CUSTO: "+real.format (custo) +) ;
30          }
31          else
32             x = 1;
33          while (x == 0)
34          {
35             if (timer_ec.RetornarCondicao () == 1)
36             {
37                timer_ec.cancel () ;
38                timer_ec.Tempo (6000) ;
39                try {
40                   Tarifador.determina_horario () ;
41                   mod_nova = tarifador.mod_horaria;
42                } catch (Exception e) {
43                   // TODO Auto-generated catch block
44                   e.printStackTrace () ;
45                }
46                if (mod != mod_nova)
47                {
48                   Tarifador.determina_tarifa () ;
49                   nova_tarifa = tarifador.tarifa;
50                   if (nova_tarifa != preco)
51                   {
52                      preco = nova_tarifa;
53                   }
```

```
54                }
55                custo = custo+ (preco/10) ;
56   System.out.println ("CUSTO: "+real.format (custo)) ;
57                tarifador.stop () ;
58             }
59          }
60       }
61     }
62   }
```

Apêndice D – Código da classe Paralela

```
1   package parport;
2
3   public class Paralela
4   {
5
6     static ParallelPort lpt1 = new ParallelPort (0x378) ; // 0x378 is
7   normally the base address for the LPT1 port
8     static int aByte=0;
9     static String binario;
10
11    /**public static String le_pinos ()
12    {
13      aByte = lpt1.read () ; // read a byte from the port's STATUS pins
14      binario = conversor.getbits (aByte) ;
15      return (binario) ;
16    }
17
18    public static char[] digitos ()
19    {
20      String pinos = Paralela.le_pinos () ;
21      StringBuffer buffer = new StringBuffer (pinos) ;
22      char digito[] = new char[4];
23      buffer.getChars (1,5,digito,0) ;
24      return (digito) ;
25
26    }**/
27    public static char digito ()
28    {
29      return (char) (lpt1.read () &0x78) ;
30    }
31    public static char estado_pino_15 ()
32    {
33    // System.out.print ("acessando a funÃ§Ã£o estado_pino_15 -> ") ;
34    // System.out.println (~lpt1.read () &0x80) ;
35      return (char) (~lpt1.read () &0x80) ;
36    }
37
38  }
```

Apêndice E – Código do Salva Dígito

```
1   package parport;
2   public class SalvaDigito {
3     public String numero;
4     public char salvar_digitos (char digito_lido)
5     {
6       char converte[]=
7   {'0','8','4','0','2','0','6','9','1','9','5','0','3','0','7','x'};
8       StringBuffer buffer = new StringBuffer () ;
9       buffer.append (digito_lido) ;
10      int x = buffer.charAt (0) ;
11      x = x >>3;
12      return converte[x];
13    }
14  }
```

Apêndice F – Código do Temporizador

```
1  package parport;
2  import java.util.Timer;
3  import java.util.TimerTask;
4
5  public class Temporizador {
6     Timer timer;
7     char Esgotado;
8     public Temporizador () {
9     }
10    public void Tempo (int mseg) {
11
12       timer = new Timer () ;
13       timer.schedule (new RemindTask () , mseg) ;
14       Esgotado = 0;
15    // System.out.println ("INICIOU TIMER DE "+ mseg) ;
16    }
17
18    public char RetornarCondicao () {return Esgotado;}
19    class RemindTask extends TimerTask {
20       public void run () {
21          System.out.println ("Time's up!") ;
22          Esgotado = 1;
23
24       }
25    }
26
27    public void cancel () {
28       timer.cancel () ;
29    }
30 }
```