DESENVOLVIMENTO DE INTERFACE GRÁFICA PARA O SISTEMA DE CONTROLE DE ANTENA DA ESTAÇÃO MULTIMISSÂO DE NATAL

RELATÓRIO FINAL DE PROJETO DE INICIAÇÃO CIENTÍFICA (PIBIC/CNPq/INPE)

Rafael Henrique Barboza da Silva (UFRN, Bolsista PIBIC/CNPq)
Email: rafael.rhbs@crn2.inpe.br

Manoel Jozeane Mafra de Carvalho (CCR/CRN/INPE, Orientador)
Email: manoel@crn.inpe.br

Junho de 2010.

PUBLICADO POR:

Instituto Nacional de Pesquisas Espaciais - INPE
Gabinete do Diretor (GB)
Serviço de Informação e Documentação (SID)
Caixa Postal 515 - CEP 12.245-970
São José dos Campos - SP - Brasil
Tel.:(012) 3945-6923/6921
Fax: (012) 3945-6919
E-mail: pubtc@sid.inpe.br

MINISTÉRIO DA CIÊNCIA E TECNOLOGIA
INSTITUTO NACIONAL DE PESQUISAS ESPACIAIS

DESENVOLVIMENTO DE INTERFACE GRÁFICA PARA O SISTEMA DE CONTROLE DE ANTENA DA ESTAÇÃO MULTIMISSÂO DE NATAL

RELATÓRIO FINAL DE PROJETO DE INICIAÇÃO CIENTÍFICA (PIBIC/CNPq/INPE)

Rafael Henrique Barboza da Silva (UFRN, Bolsista PIBIC/CNPq)
Email: rafael.rhbs@crn2.inpe.br

Manoel Jozeane Mafra de Carvalho (CCR/CRN/INPE, Orientador)
Email: manoel@crn.inpe.br

Junho de 2010.

Dados Internacionais de Catalogação na Publicação

Silva, Rafael Henrique Barboza da Silva.
Cutter Desenvolvimento de interface gráfica para o sistema de controle de antena da estação multimissâo de Natal- Relatório final de projeto de iniciação científica (PIBIC/CNPq/INPE) / Rafael Henrique Braboza da Silva.
Natal: INPE, 2010.
18p. ; (INPE-0000-TDI/00)

Grau (graduando) - Instituto
Nacional de Pesquisas Espaciais, São José dos Campos, 2010.
Orientador: Manoel Jozeane Mafra de Carvalho.

1. EMMN. 2. Interface gráfica. 3. Satélite. 4. Sistema
5. PCD. I. Título.

CDU

RESUMO

Este trabalho tem por objetivo fazer um breve resumo sobre o projeto de interface gráfica da EMMN, e das tecnologias utilizadas, assim como das atividades realizadas pelo bolsista Rafael Henrique, autor deste relatório, bem como uma breve descrição de como elas foram implementadas. A interface gráfica foi implementada com a linguagem C/C++ utilizando o *framework* Qt, sobre GNU/Linux.
A EMMN é uma estação que tem o objetivo de fazer rastreio de satélites e capturar seus dados. A interface gráfica tem a finalidade de controlar a antena da estação e fazer os agendamentos para o rastreio dos satélites.

ABSTRACT

This paper aims to give a brief summary about the interface design graphic EMMN, and technologies used, as well as the activities performed by the fellow RafaelHenrique, author of this report and a brief description of how they were implemented. The graphical interface was implemented with C / C + + using the Qt framework on GNU / Linux.
The EMMN is a station that aims to make satellite tracking and capture their data. The graphical interface is intended to control the antenna of the station and make appointments for screening of satellites.

LISTA DE FIGURAS

LISTA DE SIGLAS E ABREVIATURAS

FHS	*Filesystem Hierarchy Standard*
GNU	GNU *is Not* Unix
INPE	Instituto Nacional de Pesquisas Espaciais
PCD	Plataforma de Coleta de Dados
TCP	*Transmission Control Protocol*

SUMÁRIO

1 INTRODUÇÃO

Este projeto de iniciação científica em andamento desde 2007, para o desenvolvimento de uma interface gráfica para o sistema de controle de uma antena da estação de multimissão de Natal, teve como seus objetivos iniciais a definição de tecnologias a serem utilizadas, a modelagem de comunicação entre os módulos de controle e interface gráfica, e o início da implementação desses módulos.

O sistema foi desenvolvido com a linguagem C/C++, e utiliza-se da biblioteca Qt (*framework* multiplataforma para desenvolvimento de interfaces gráficas) no desenvolvimento da Interface Gráfica. Ele pode ser dividido em três módulos com funções bem distintas que são:

- O controlador, módulo responsável por se comunicar diretamente com o *hardware* da antena utilizando a biblioteca Comedi;
- A interface gráfica, módulo responsável por permitir o controle da antena de modo mais intuitivo, utilizando-se de comandos existentes no módulo do controlador;
- e o servidor TCP, módulo responsável por retirar a necessidade da interface gráfica permanecer no mesmo computador em que se encontra o módulo controlador.

Durante o desenvolvimento da interface gráfica por outros bolsistas, sentiu-se a necessidade de adicionar novas funcionalidades ao controlador, como a criação de um arquivo de configuração garantindo uma maior flexibilidade na troca de parâmetros utilizados pela antena, assim como algumas modificações para o seu funcionamento correto ou melhoramento de performance, como é caso da análise da efeméride que agora é concluída em um tempo menor.

Este relatório visa uma descrição geral do que se trata o projeto, e uma descrição e documentação das tarefas executadas pelo bolsista, autor deste relatório, durante o período de Fevereiro a Junho de 2010. A bolsa atual possui como objetivo as seguintes tarefas:

- fazer a ligação do controlador com a interface gráfica, utilizando-se do servidor TCP;
- implementar funcionalidades previstas porém não implementadas na interface gráfica, como é o caso da listagem das passagens agendadas;
- corrigir alguns comportamentos fora do esperado, como é caso da recriação das janelas mesmo quando estas já estão criadas;
- e adicionar novas funcionalidades visando automatização na recuperação e determinação das passagens do satélite, facilitando o agendamento dessas passagens.

Este relatório conterá a descrição e documentação da implementação da listagem das passagens agendadas, e da resolução do problema da recriação da janela quando esta encontra-se criada, que foram as tarefas realizadas pelo bolsista no período citado.

2 A ESTAÇÃO MULTIMISSÃO DE NATAL

A Estação Multimissão de Natal é um projeto do INPE que visa o desenvolvimento de uma estação de rastreio de satélites reconfigurável e definida por *software*.

Possui como origem a estação francesa do projeto SACI, que tinha por objetivo fazer funcionar um sistema educacional usando rádio, televisão e satélite. A estação encontrava-se avariada e teve sua reativação realizada e documentada por Queiroz (2006) em atividades anteriores.

Uma das aplicações da EMMN é atender a uma demanda do sistema brasileiro de coleta de dados ambientais, no qual possui como uma de suas fontes de dados as PCD's distribuídas por todo território nacional.

Um resumo de como acontece a captura de dados através da antena da estação flui da seguinte forma: as PCD's transmitem os dados para o satélite, este o captura e retransmite para a antena da estação. Os dados capturados deverão ser gravados e posteriormente processados.

Essa é apenas uma das aplicações da estação, que como já descrito, pode fazer o rastreamento de qualquer satélite e capturar seus dados desde que este transmita em uma faixa de frequência na qual a antena consiga capturar.

3 TECNOLOGIAS UTILIZADAS

Um dos pré-requisitos do sistema é que seja todo desenvolvido em código aberto visando a não utilização de tecnologias proprietárias, evitando assim custos que por ventura essa poderia vir cobrar, e ter total acesso ao código, caso fosse necessário alguma modificação ou visualização do código de como ele funciona.

Como consequência disso o software resultante também deverá ser livre, ou seja, por se está utilizando tecnologias livres, deverá obedecer à licença que esta utiliza, e uma de suas regras é que, o software que a utiliza também deverá ser disponibilizado sob licença livre.

Então todas as tecnologias escolhidas e apresentadas abaixo teve como um de seus parâmetros de escolha de ser livre.

3.1 GNU/Linux

O GNU/Linux é um sistema operacional do tipo UNIX, que teve seu núcleo criado por Linus Torvalds com ajuda de vários programadores voluntários, e teve sua primeira versão oficial lançada em outubro de 1991. Ele nada mais é que um núcleo monolítico com as funções de escalonamento de processos, gerenciamento de memória, operações de entrada e saída, acesso ao sistema de arquivos, unido ao um conjunto de softwares sob licença GNU.

Entre suas características o GNU/Linux é conhecido por ser multitarefa, multiusuário, estável, portável e seguro.

Possui diversas distribuições, em que cada distribuição é formada por um conjunto de softwares de aplicações mais o núcleo Linux. Cada distribuição se diferencia por possuir seleção de aplicações e pré-configuração delas, um sistema de pacotes, biblioteca básica, e uma estrutura de organização de diretório obediente ao padrão FHS, que é bastante flexível.

Para rodar o módulo de controle, a distribuição GNU/Linux escolhida foi a Slackware 10.2 com o núcleo Linux na versão 2.4.31.

3.2 C/C++

C/C++ são linguagens de programação de alto nível utilizadas no desenvolvimento do sistema. A linguagem C foi criada em 1972 por Dennis Ritchie, no AT&T Bell Labs e possui como alguns de seus pontos fortes a sua eficiência, e sua proximidade com o código de máquina. Possui como um de seus pontos fracos a falta de proteção contra incompatibilidades programadas pelos programadores, ou seja, normalmente consegue-se executar tudo que é programado, até coisas improváveis, o que dificulta o encontro de erros e geração resultados inesperados. Essa falta de proteção ocorre por C deixar sempre o programador no controle, o que oferece a possibilidade do programa executar mais eficiente que outras linguagens.

Já a linguagem C++ foi criada em 1983 por Bjarn Stroustrup, também nos laboratórios da AT&T Bell Labs, com o objetivo de integrar a programação orientado a objetos com o C. Nessa nova linguagem foi preservado a compatibilidade com a linguagem C.

No presente sistema em desenvolvimento tanto o controlador quanto o servidor TCP foram escritos predominantemente em C, já as interfaces gráficas foi utilizado em sua maior parte C++.

3.3 Comedi

Como é explicado em seu próprio site, Comedi (interface de controle de dispositivos de medição em linux) é um projeto que desenvolve *drivers open-source*, ferramentas e bibliotecas para aquisição de dados.

A sua biblioteca trabalha no espaço do usuário e fornece uma interface

amigável ao desenvolvedor. Foi originalmente concebido e escrito por David Schleef e teve várias contribuições por outros programadores da comunidade. Hoje os mantenedores do projeto são Frank Mori Hess e Ian Abbott.

No projeto da EMMN, foi a biblioteca escolhida para fazer comunicação e controle da antena que faz o rastreio dos satélites. O comedi é utilizada especificamente no módulo de controle do sistema de software em desenvolvimento, e dele é o utilizado o *driver* referente a placa conversora AD/DA de modelo PCI-6025E, fabricada pela National Intruments.

Uma outra opção seria fazer o controle da antena através do aplicativo LabView, que é a ferramenta para aquisição de dados desenvolvida pela National Intruments, porém, como desde o início do projeto da EMMN optou-se por utilizar tecnologias *open-source*, o comedi passou a ser a alternativa, uma vez que o LabView é uma ferramenta proprietária e exige a compra de licença para uso.

A versão do Comedi utilizada no sistema é a 0.7.73.

3.4 Qt

Qt é uma aplicação multiplataforma e um *framework* utilizado para a construção de interfaces gráficas para usuário. A sua biblioteca é disponibilizada em várias linguagens, e por ser multiplataforma, suas aplicações conseguem ser compiladas para muitas plataformas sem muitas alterações em seu código fonte. Foi originalmente desenvolvida por Haavard Nord e Eirik Chambe-Eng em 1991, e hoje pertence a Nokia na qual a disponibiliza na linguagem C++ sob a licença LGPL.

No sistema em desenvolvimento, a Qt é utilizada na implementação da interface gráfica em que o usuário final irá fazer agendamentos para o rastreio de satélites e ter o controle da antena. A Qt foi escolhida por ser uma ferramenta amplamente utilizada por empresas de desenvolvimento (o que a torna uma ferramenta validada

e confiável), por oferecer facilidades em seu uso (que é possibilitado pela vasta documentação publicada e por possuir ferramentas de desenvolvimento que a utilizam), e por gerar uma interface do usuário que pode ser facilmente adaptada para funcionar em outras plataformas. A versão da Qt utilizada no projeto é a 3.

4 TAREFAS REALIZADAS

Como a atual bolsa é continuação do projeto, a interface gráfica já se encontra toda construída. O que falta é a ligação dela com o servidor TCP, a implementação de algumas partes e ajustes de outras que não apresentam comportamento desejado. O início da implementação dela ocorreu com Peres (2007) onde em seu trabalho é mostrado como ela foi desenvolvida.

Neste projeto também houve contribuições de Souto (2009), que foi responsável pelo desenvolvimento e testes do analisador léxico encontrado no servidor TCP e pequena correção na conexão entre o servidor TCP e a interface gráfica.

Abaixo segue as tarefas realizadas pelo autor do relatório, durante o período de Fevereiro a Junho de 2010.

4.1 O problema das janelas

A interface gráfica é composta por uma janela principal na qual contém todas as funcionalidades disponíveis. Cada funcionalidade é aberta em uma janela dentro da principal, quando selecionada no menu ou na barra de ferramentas. Entre os objetivos desse trabalho se encontra o de identificar comportamentos inesperados na interface gráfica e corrigi-los. Um problema localizado e classificado como este, foi o das janelas que eram recriadas toda vez que se acessava novamente uma janela aberta anteriormente e não fechada. Esse problema fazia com que janela de mesma funcionalidade sempre fosse recriada, mesmo que se encontrasse aberta, o que poderia resultar em sua reaparição quando era fechada.

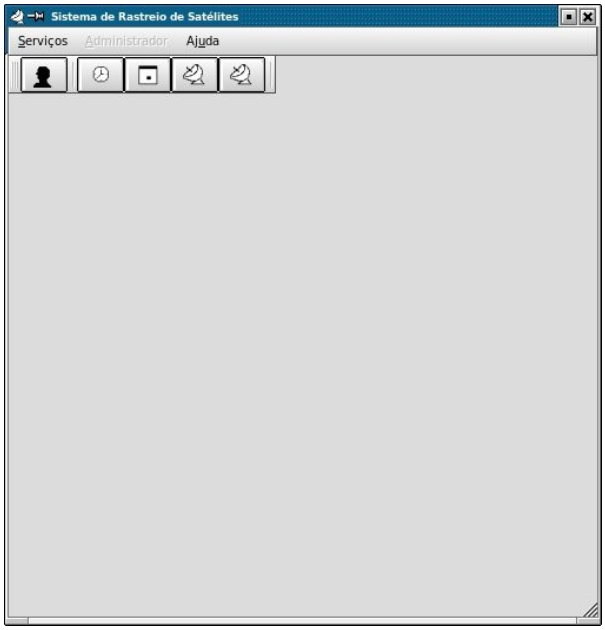

Figura 4.1 – Janela principal que contém todas as funcionalidades

Para solucionar esse problema, foram implementadas algumas funções para ter o controle de todas as janelas que encontravam-se abertas. Padronizou-se então a utilização dessas funções no momento de sua abertura e fechamento. Além disso, foi feita uma estrutura em que a ordem na qual as janelas eram abertas, passavam a ser exibidas a medida em que a janela em exibição fosse fechada.

Cada janela recebeu uma identificação que a representa, como é visto na figura 4.2. A identificação foi implementada através de constantes pois melhora a legibilidade do código e facilita no momento da manutenção. Essa identificação é utilizada para localizar a referencia de uma janela que está aberta.

```
//constanste que atribui um id para cada janela
const int static JANELA_CONECTAR_USUARIO = 1;
const int static JANELA_CONFIGURAR_CONEXAO = 2;
const int static JANELA_MOVER_ANTENA = 3;
const int static JANELA_VISUALIZAR_PASSAGEM = 4;
const int static JANELA_ADICIONAR_PASSAGEM = 5;
const int static JANELA_MONITORAR_PASSAGEM = 6;
const int static JANELA_GERENCIAR_USUARIO = 7;
const int static JANELA_CONFIGURAR_CONTROLADOR = 8;
const int static JANELA_EXIBIR_GAVETA = 9;
```

Figura 4.2 – Trecho de código com as constantes de identificação das janelas.

Como explicitado anteriormente, a interface gráfica foi implementada utilizando-se o *framework* Qt, e visando a reutilização de código e as vantagens da programação orientado a objetos, foi utilizado estruturas de dados disponíveis no *framework*. Dentre elas encontram-se as classes QMap e QValueVector.

O QMap utiliza o conceito da estrutura de dados chamada mapa. Essa estrutura funciona relacionando uma chave com um valor, montando coleções mapeadas através das chaves. Ela também oferece operações otimizadas de inserção desses pares (chave, valor) assim como de busca de valor através da chave. Tanto a chave como o valor podem ser definidos por qualquer tipo de variável. Na nossa implementação a chave foi definida como do tipo inteiro, representando a constante de identificação da janela, e o valor sendo do tipo QWidget, indicando a referencia da janela aberta propriamente dita , como é mostrado na figura 4.3. Esse mapa tem o objetivo de armazenar todas as janelas que forem abertas durante a execução do sistema.

```
//mapa que guarda todas as referencias das jenelas abertas
QMap <int, QWidget *> janelas;
QValueVector<int> ordemJanelas;
```

Figura 4.3 – Trecho de código com a declaração das variáveis do tipo QMap.

Porém isso apenas foi o primeiro passo para solução do problema. Foram criadas três funções na janela principal para controlar as aberturas das janelas. Os nomes dessas funções são: abrirJanela, criarJanela e fecharJanela.

A função abrirJanela tem o objetivo de verificar se a janela que se deseja abrir já está aberta. Ela recebe como paramento a identificação da janela a ser aberta. Caso ela esteja, a função esconde a janela atualmente em exibição e exibe a janela que se deseja visualizar. Ela faz isso acessando o mapa de janelas abertas, fazendo uma busca através da operação *find*, como é visto no trecho de código mostrado na figura 4.4. Caso a janela que se deseja abrir não se encontre aberta, a função chama outra de nome criarJanela, que cria a janela desejada, a armazenado dentro do mapa, e repete a ação de esconder a

janela que se encontra atualmente em exibição, e exibir a recém criada.

```
//função responsável por gerenciar janelas.
//Controla a criação de apenas uma janela de cada tipo,
//recuperando-a quando já tiver sido criada e a criando
//somente quando não existir.
void principal::abrirJanela(int idJanela) {

.       QMap <int, QWidget *>::iterator janela = janelas.find(idJanela);
.       QWidget * novaJanela;

.       //verifica se existe uma referencia de uma janela já aberta com o mesmo tipo do id passado como parametro
.       if (janela != janelas.end())   {
.       .       novaJanela = janela.data();
.       .       //remove ela da posição da pilha de ordemJanelas
.       .       removerDaPilha(idJanela);
.       .       //a coloca no topo da pilha
.       .       ordemJanelas.push_back(idJanela);

.       }//caso não exista uma referencia de janela aberta, é criado uma e guardada sua referencia no mapa
.       else {
.       .       novaJanela = criarJanela(idJanela);
.       .       janelas.insert(idJanela, novaJanela);
.       .       //coloca o id dela no topo da pilha de ordemJanela
.       .       ordemJanelas.push_back(idJanela);
.       }

.       //se tiver alguma janela aberta feche-a
.       if (centralWidget() && idJanelaAberta != idJanela) {
.       .       centralWidget()->hide();
.       }
.

.       //exibe a janela escolhida
.       idJanelaAberta = idJanela;

.       setCentralWidget( novaJanela );
.       centralWidget()->show();
}
```

Figura 4.4 – Código fonte da função abrirJanela.

Essa função deve ser chamada em todos os menus de funcionalidade da janela principal, assim como é visto em um exemplo na figura 4.5.

```
void principal::slotverpassagens() {
.       abrirJanela(JANELA_VISUALIZAR_PASSAGEM);
}
```

Figura 4.5 – Trecho de código que faz chamada a função abrirJanela a partir de uma funcionalidade do menu.

A função criarJanela é bem simples e poderia ser colocada dentro da função abrirJanela, e só não foi feito dessa forma com o objetivo de refatorar o código e deixá-lo com melhor legibilidade. Essa função recebe o identificador da janela e retorna uma instancia da janela criada.

Para encerrar a solução, a função fecharJanela foi criada com o objetivo de retirar a referencia das janelas fechadas do mapa de janelas abertas, otimizando o consumo de memória, uma vez que após serem fechadas todas

as referencias devem sair do sistema. Na figura 4.6 podemos observar a implementação desta função.

```
//função que deve ser chamada para fechar janela.
//O objetivo dessa função é manter o controle sobre o
//mapa removendo as referencias das janelas que são destruídas,
//e abrir a janela que se encontra no topo da pilha de ordemJanela
void principal::fecharJanela() {
        if (idJanelaAberta > 0) {
                QMap <int, QWidget *>::iterator janela = janelas.find(idJanelaAberta);
                //remove a janela a ser destruida do mapa janelas
                janelas.remove(janela);
                //destroi a janela
                centralWidget()->close(true);
                //remove da pilha de ordemJanelas o id da janela que está sendo fechada
                ordemJanelas.pop_back();

                if(ordemJanelas.size() > 0) {
                        QValueVector<int>::iterator it = ordemJanelas.end();
                        it--;
                        int idJanela = * it;
                        //recupera a referencia dessa janela
                        janela = janelas.find(idJanela);
                        //exibe a janela recuperada
                        idJanelaAberta = idJanela;
                        setCentralWidget( janela.data() );
                        centralWidget()->show();
                }
        }
}
```

Figura 4.6 – Trecho de código da função fechar Janela.

A função fecharJanela é chamada a partir de cada janela de funcionalidade aberta através do menu. Na figura 4.7 é mostrado a chamada para esta função a partir da janela de nome passagem.

```
void passagens::slotCancelar() {

        mainWin->fecharJanela();
}
```

Figura 4.7 – Chamada a função fecharJanela a partir da janela passagem.

Um detalhe da interface gráfica é que a janela principal sempre exibe uma janela, se alguma estiver aberta. Logo, no momento de fechamento de uma, outra será exibida. Com isso, foi adicionado nas funções de abrirJanela e fecharJanela, algumas linhas de códigos que determina a escolha de qual janela será aberta. A ideia implementada é que as ultimas janelas abertas sejam exibidas, a medida que aquela que encontra-se em exibição seja

fechada. Para isso foi utilizado um vetor do tipo QValueVector, que vai colocando os identificadores das janelas que são abertas na sua posição final, e as removendo no momento em que são fechadas. Se por acaso a janela que se deseja abrir já se encontre aberta, a identificação desta janela é removida da posição que se encontra no vetor, e é colocado no seu final.

E dessa forma foi resolvido o problema de recriação desnecessária de janelas e de suas aparições inesperadas.

4.2 A visualização do agendamento das passagens

Um outro problema encontrado contempla uma funcionalidade prevista porém não implementada e trata-se da listagem do agendamento das passagens. A interface gráfica possui a janela referente a esta funcionalidade porém os dados não eram pegos de um arquivo válido, e nem existia implementação de um filtro para exibição de agendamentos específicos por situação, como era previsto no projeto desta janela.

A solução adotada para concluir essa funcionalidade consistiu em definir um arquivo no padrão a ser utilizado pelo módulo controlador, que o geraria no momento do agendamento, e a criação de classes para leitura desse arquivo, com operações de filtragem por situação das passagens e estruturação dos dados advindos do arquivo.

O padrão adotado consistiu em passar os dados separados por tabulação na seguinte ordem: nome do arquivo com os dados da passagem, situação do agendamento, satélite cujo arquivo tem origem, data em que a passagem irá ocorrer, hora de início, hora final da passagem. E também de uma passagem para outra deverá possuir uma quebra de linha. Esse padrão pode ser observado na figura 4.8.

```
CBERS2B 08082010.txt, ATIVO   , CBERS2B, 08/08/2010, 15:10:00.000, 15:25:00.000
SCD2 09082010.txt,    INATIVO , SCD2,    09/08/2010, 16:20:00.000, 16:35:00.000
SCD1 09082010.txt,    ATIVO   , SCD1,    09/08/2010, 17:30:00.000, 17:45:00.000
CBERS2B 10082010.txt, CANCELADO, CBERS2B, 10/08/2010, 18:40:00.000, 18:55:00.000
SCD2 10082010.txt,    CANCELADO, SCD2,    10/08/2010, 19:00:00.000, 19:15:00.000
SCD2 11082010.txt,    INATIVO , SCD2,    11/08/2010, 20:10:00.000, 20:25:00.000
CBERS2B 11082010.txt, CANCELADO, CBERS2B, 11/08/2010, 21:20:00.000, 21:35:00.000
SCD1 12082010.txt,    INATIVO , SCD1,    12/08/2010, 22:30:00.000, 22:45:00.000
CBERS1B 12082010.txt, ATIVO   , CBERS1B, 12/08/2010, 23:40:00.000, 23:55:00.000
SCD2 13082010.txt,    ATIVO   , SCD2,    13/08/2010, 00:00:00.000, 00:15:00.000
SCD1 13082010.txt,    CANCELADO, SCD1,   13/08/2010, 01:10:00.000, 01:25:00.000
```

Figura 4.8 - Exemplo de um arquivo com os agendamentos das passagens

Quanto a criação de classes foram criadas duas, uma para representar uma linha do arquivo e agendamento, separando em variáveis distintas cada tipo de dado, chamada de ItemAgendamento, e outra para fazer a leitura de todo o arquivo, guardando em uma lista em memória, chamada de Agendamento.

A classe ItemAgendamento possui vários atributos que representam cada dado da linha do arquivo do agendamento. Em seu método construtor foi projetado e implementado para receber uma linha do arquivo e fazer a separação dos dados colocando-os em seus respectivos atributos. A criação dessa classe vem a facilitar a implementação dos filtros, uma vez que o conteúdo da linha fica todo separado, focando a codificação no atributo em que se deseja fazer o filtro.

A classe Agendamento é responsável por fazer a leitura do arquivo e a filtragem de seus dados. Em seu método construtor deve ser passado o caminho onde se encontra o arquivo. Ela possui como atributo uma lista do tipo ItemAgendamento que é preenchida quando o seu método carregarArquivo é chamado. O seu último método, filtrarPorStatus, como já foi descrito, é fazer a filtragem da lista retornando uma nova com os itens de agendamento contendo a situação passada como parâmetro.

Toda essa estrutura implementada é utilizada na classe “passagem”, que é a janela onde é exibida a lista das passagens agendadas e é mostrada na figura 4.9.

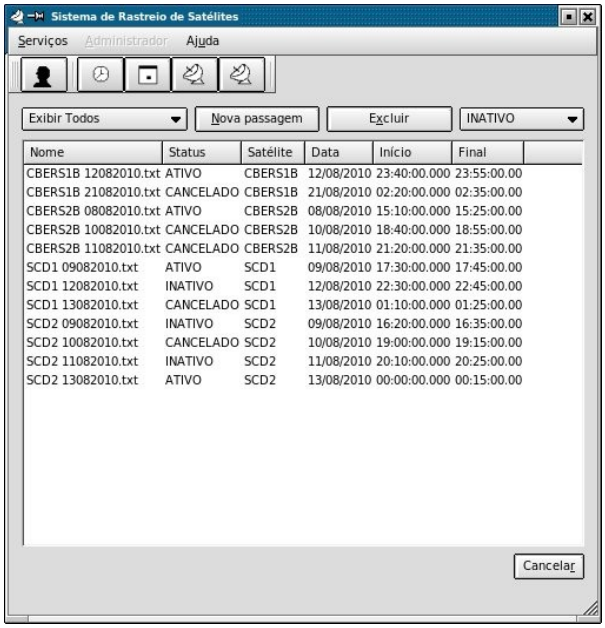

Figura 4.9 - Janela passagem listando todas as passagens agendadas.

No método construtor de passagens é instanciado a classe Agendamento e chamado o método carregarArquivo.

A lista com os itens do arquivo é percorrida e colocada no componente do Qt chamado QListView. Existe outro componente que compõe o QListView chamado de QListViewItem, e este é quem é de fato que armazena os dados contidos na lista de agendamento.

O resultado após a seleção da situação cancelado é visualizada na figura 4.10.

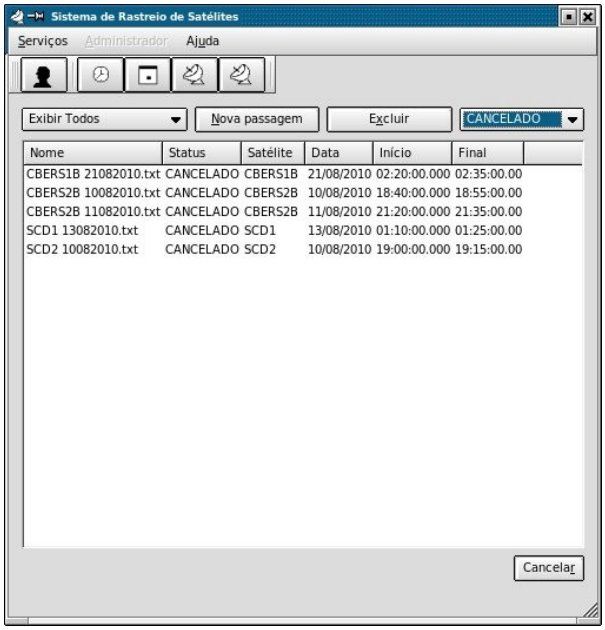

Figura 4.10 - Agendamentos filtrado pela situação cancelado.

5 CONCLUSÂO

As implementações realizadas supriram as necessidades dos problemas expostos. Nessa primeira parte muito tempo foi gasto para adaptação ao ambiente de desenvolvimento, no que diz respeito a utilização da linguagem C/C++ assim como da própria ferramenta de desenvolvimento. Outro tempo gasto foi no entendimento da problemática do sistema bem como dos artefatos produzidos anteriormente pelos bolsistas ligados a esse projeto. Por isso poucas coisas foram produzidas.

Atualmente estamos trabalhando na integração da interface gráfica com o servidor TCP e na otimização do módulo controlador, faltando ainda a automatização na determinação das passagens, funcionalidade que melhorará significativamente o processo de agendamento, e consequentemente os testes.

Como trabalhos futuros exitem algumas frentes interessantes que podem ser trabalhadas, como por exemplo a refatoração do módulo controlador, deixando-o com uma implementação mais voltada para o paradigma de orientação a objetos, facilitando dessa forma adaptações para outras estações. Outra frente seria aumentar o controle de acesso do sistema permitindo a utilização por mais usuários com perfis diferentes, e para isso utilizando banco de dados para esse controle.

Entretanto, por ainda não ter concluído todas as tarefas assumidas no início da bolsa e por ter várias frentes que podem ser trabalhadas para o melhoramento do sistema, seria muito interessante e importante a renovação da bolsa para a conclusão dos trabalhos.

REFERÊNCIAS BIBLIOGRÁFICAS

Queiroz, K. I. P. M. e Carvalho, M. J. M. Sistema de Controle de Apontamento para uma Antena da Estação TT&C de Natal. Natal, RN: INPE, 2006;

PERES, H. S. e CARVALHO, M. J. M. Desenvolvimento de aplicação para Linux utilizando Kdevelop e Qt. Natal, RN: INPE, 2007;

Souto, M. C. B. Desenvolvimento de uma interface gráfica para o sistema de controle da antena da estação Multimissão de Natal – EMMN. Natal, RN: INPE, 2009;

SCLEEF, D., HESS MORI, F. e ABBOTT, I. Comedi: linux control and measurement device interface. Disponível em <http://www.comedi.org/> Acesso em 18 de Junho de 2010;