UNIVERSIDADE FEDERAL DO PARÁ
INSTITUTO DE TECNOLOGIA
FACULDADE DE ENGENHARIA DA COMPUTAÇÃO

ALEX BARROS DOS SANTOS

ESTUDO E DESENVOLVIMENTO DE UM SOFTWARE PARA AUTOMATIZAÇÃO DO INTERROGADOR ÓPTICO FS2200

BELÉM/PA
DEZEMBRO – 2011

ALEX BARROS DOS SANTOS

ESTUDO E DESENVOLVIMENTO DE UM SOFTWARE PARA AUTOMATIZAÇÃO DO INTERROGADOR ÓPTICO FS2200

Trabalho de Conclusão de Curso apresentado como exigência parcial para obtenção do título de Bacharel em Engenharia da Computação, pela Universidade Federal do Pará.

Orientador: Prof. Dr. Marco José de Sousa

BELÉM/PA
DEZEMBRO – 2011

ESTUDO E DESENVOLVIMENTO DE UM SOFTWARE PARA AUTOMATIZAÇÃO DO INTERROGADOR ÓPTICO FS2200

Este trabalho foi julgado / / adequado para obtenção do Grau de Engenheiro da Computação, aprovado em sua forma pela banca examinadora que atribuiu o conceito. _____.

_____

Prof. Dr. Marco José de Sousa

Orientador

_____

Prof. Dr. João Crisóstomo Weyl Albuquerque Costa

Membro da banca examinadora

_____

Prof. Dr. Aldebaro Barreto da Rocha Klautau Júnior

Membro da banca examinadora

BELÉM/PA

DEZEMBRO - 2011

À minha família.

AGRADECIMENTOS

Primeiramente, agradeço a Deus por ter sido tão generoso comigo, tendo me concedido saúde e tanta sorte ao longo de todos esses anos. Agradeço aos meus pais, Abdias e Vânia Santos, pela educação, amor e dedicação que sempre me deram, acompanhando e torcendo por cada momento desta graduação. À minha irmã, Amanda, agradeço a doçura e a paciência. Vocês sempre foram os meus maiores motivadores, sem o apoio incondicional de vocês e a união, nós não teríamos ido tão longe. Desculpem-me as ausências.

Gostaria de agradecer também ao meu orientador nesse TCC, Prof. Marco Sousa, por ter me sugerido esse trabalho e orientado para que fosse possível a conclusão do mesmo, obrigado pelas correções e idéias. Ao Prof. João Costa, por ter me dado a oportunidade de ser membro do LEA desde o primeiro semestre, e também por ter me ajudado e apoiado todas às vezes que precisei. Ao Prof. Aldebaro Klautau, por sempre motivar e inspirar não só a mim, mas muitos dos meus colegas. A todos os colegas da turma de 2008.1 e 2008.2, em especial aqueles que estiveram mais próximos: Amaury, Danilo, Paulo Victor, Ilan, Pedro e Tasso.

Aos meus amigos Débora Madonado, Elton Moutinho, Lauro Américo, Leonardo Lira, Fernanda Leal, Joarley Moraes, Roberto Medeiros, Edicleiton Wanzeler e tantos outros que não pude mencionar aqui, como Antonio Costa, Karol Sapucaia, Sara Ribeiro e os ISAC´s , meu muito obrigado.

Aos meus amigos do Laboratório de Eletromagnetismo Aplicado e de toda EngComp, incluindo o pessoal de elétrica, obrigado pelos ótimos momentos de convivência e debates. Em especial ao Adam Dreyton pelas contribuições significativas neste trabalho.

Por fim, agradeço por todo o apoio, ajuda e orações que recebi dos familiares, meus avós, tios e primos. À Universidade Federal do Pará, UC, MTU, CNPQ e a CAPES. Muito obrigado a todos.

Alex Barros dos Santos.

"(...)Um homem precisa viajar. Por sua conta, não por meio de histórias, imagens, livros ou tv. Precisa viajar por si, com seus olhos e pés, para entender o que é seu. Para um dia plantar as suas próprias árvores e dar-lhes valor. Conhecer o frio para desfrutar do calor. E o oposto. Sentir a distância e o desabrigo para estar bem sob o próprio teto. Um homem precisa viajar para lugares que não conhece para quebrar essa arrogância que nos faz ver o mundo como o imaginamos, e não simplesmente como é ou pode ser; que nos faz professores e doutores do que não vimos, quando deveríamos ser alunos, e simplesmente ir ver".

("Mar sem fim"- Amyr Klink)

RESUMO

Sensores de fibra óptica tem ganho cada vez mais espaço na indústria de sensoriamento. Com o uso de Fibras com Grades de Bragg (FBG) é possível monitorar diversos tipos de estruturas e fenômenos. Um equipamento fundamental para o monitoramento utilizando sensores ópticos é o interrogador, responsável por excitar os sensores FBGs e extrair informações a partir dos mesmos. Dentre os diversos tipos, o mais consolidado na indústria é o baseado em laser de varredura, devido sua flexibilidade de utilização. Os interrogadores necessitam de um software para processar a aquisição de dados, e disponibilizá-los de forma inteligível para um usuário ou para conectá-lo, por exemplo, a um sistema de controle ou alerta de monitoramento. Contudo, os softwares já existentes são um pouco limitados, fazendo com que o uso dos equipamentos não atinja a potencialidade esperada. Neste trabalho, buscou-se desenvolver um novo software para se realizar a aquisição de dados a partir do interrogador FS2200 (equipamento da empresa FiberSensing). Possibilitando, dessa forma, que os dados brutos sejam usados para estudos mais aprofundados, permitindo uma análise completa do comportamento do espectro óptico dos FBGs, que é onde todas as informações de sensoriamento estão codificadas.

Palavras-chave: FBG, Monitoramento, Interrogador Óptico, Software de aquisição.

ABSTRACT

Fiber optic sensors have gained more space in the sensing industry. Due to several characteristics, Fibers Bragg Grating (FBG) can be used to monitor many types of structures and phenomena. One of the most important part of a monitoring system is the optical interrogator, it is responsible for exciting the FBGs sensors and extract information from them. Among the various types, the more consolidated in industry is one based on laser scanning, it is very flexible for its purpose. An interrogator needs a software to process data acquisition and provide this data in an intelligible way to users or to a control system, for instance. However, the software are limited, limiting the potential use which the equipment could achieve. In this monograph, It was developed a new software to perform data acquisition through the FS2200 interrogator (FiberSensing's equipment). This new software should acquire the raw data, becoming possible their use for further studies, allowing a complete analysis of the behavior of the optical spectrum of FBGs, which has the complete sensing information encoded in it.

Keywords: FBG, Optical interrogator, monitoring systems, Acquisition software.

Lista de Figuras

Lista de Tabelas

Lista de Abreviaturas e Siglas

CCD- Charge-Coupled Device

EMI – Eletromagnetic Interference

FBG – Fiber Bragg Grating

LPO – Linear Phase Operator

OC- Optical Channel

OSA – Optical Spectrum Analyzer

OPGW - Overhead Ground Wire

SDM-Spatial Division Multiplexing

SNR – Signal Noise Ratio

TDM- Time Division Multiplexing

WDM- Wavelength Division Multiplexing

Sumário

Capítulo 1

Introdução

1.1 Cenário atual dos sensores ópticos

Nos últimos 20 anos, o grande crescimento da optoeletrônica e das comunicações com fibras ópticas ocasionou duas grandes revoluções. A indústria da optoeletrônica trouxe produtos como leitores de CD, impressoras a laser, scanners de código de barra e ponteiros laser que mudaram significativamente a vida cotidiana das pessoas. Já o crescimento das comunicações com fibras ópticas, revolucionou a indústria de telecomunicações. Hoje em dia fibras ópticas são praticamente sinônimos da palavra “telecomunicações” [11], possibilitando um maior desempenho, links de comunicação mais confiáveis e com custo por largura de banda cada vez menor. Com essas expansões nesses mercados, gerou-se então uma grande disponibilidade de componentes ópticos e eletrônicos de alta qualidade e a preços cada vez mais competitivos. Isso beneficiou diretamente os sensores ópticos, que fazem uso tanto de tecnologias ligadas a optoeletrônica, como de fibras ópticas propriamente ditas. Grande parte dos componentes desenvolvidos por terem o seu uso diretamente ligados a área, contribuíram para o desenvolvimento e a viabilidade de aplicações com sensores de fibra óptica.

Assim, os sensores de fibra óptica, que estavam apenas nos laboratórios de pesquisa, puderam passar para outro estágio de desenvolvimento possuindo uma gama de aplicações muito ampla.

Dentre elas, o monitoramento de grandes estruturas merece um destaque especial, devido a maturidade que os equipamentos alcançaram. Hoje em dia, interrogadores ópticos baseados em varredura por laser (como o FS2200) são produtos bem estabelecidos comercialmente, especializados no monitoramento de estruturas com FBGs. Diversos são os fabricantes e modelos para esse propósito.

Apesar dessa capacidade tão ampla para sensoriamento, o alto custo permaneceu por muitos anos sendo o maior limitador para a sua expansão no mercado [11]. Mas com a queda dos preços de fabricação e as melhorias na

qualidade, assim como a produção em grande escala dos componentes, a capacidade dos sensores ópticos de substituir os sensores tradicionais foi potencialmente aumentada. Dentre os tipos desses sensores podemos mencionar os de rotação, aceleração, temperatura, pressão, acústica, vibração, posicionamento linear e angular, deformação, medidas químicas e uma série de outras aplicações. O custo que na década de 80 parecia inviabilizar economicamente um projeto com sensores ópticos, vem mostrando nos últimos anos [10] não ser mais tão limitador.

Mesmo com essa redução de preços, ainda hoje, os sensores ópticos tem atingido apenas marginalmente seu pleno sucesso comercial. Embora a tecnologia possua um grande potencial a ser explorado, possui ainda como entrave um custo relativo mais alto (devido a compra de novos equipamentos) e o fato do usuário final ainda não estar familiarizado. Mas apresentam inúmeras vantagens em relação ao sensores eletromecânicos, pois são: leves, pequenos, passivos, não sofrem interferência eletromagnética (EMI), possuem alta sensibilidade, boa largura de banda, robustez ambiental e durabilidade. Nos campos de atuação onde os sensores tradicionais estão estabelecidos há muitos anos o avanço tem continuado lento, porém há casos em que os sensores de fibra oferecem novas capacidades e aplicações, como por exemplo em sensoriamento distribuído. Sensores de fibra óptica com grades de Bragg (FBG - *Fiber Bragg Grating*) e outros dispositivos baseados em grades são exemplos do tipo de sensores que fornecem esse recurso [1]. Nesses tipos de sensores em específico e suas aplicações estará o foco deste trabalho.

As FBGs são elementos de sensoriamento intrínseco (o estímulo do meio ocorre diretamente na fibra) que podem ser foto-inscrito em uma fibra de sílica [2-7] possuindo todas as vantagens já atribuídas a sensores de fibra óptica, porém, na área de sensoriamento embarcado e distribuído em materiais, eles são capazes de criar estruturas inteligentes. Desse modo, sensores podem ser embarcados/incorporados em um determinado material para permitir a medição de parâmetros como carga, deformação, temperatura e vibração, a partir da qual o estado da estrutura pode ser avaliado e monitorado em tempo real [1]. Um detalhe importante a ser ressaltado é o tempo de vida de uma FBG comercializado atualmente, cerca de mais de 25 anos, durante esse tempo a refletividade pode reduzir somente 1% ou 2% [18].

Há vários exemplos de monitoramento que expõem sensores tradicionais à influência de interferência eletromagnética, como no monitoramento de linhas de transmissão (maior interesse do projeto em questão), na supervisão de máquinas elétricas em processos indústrias e em usinas hidroelétricas. Outros fatores como alta temperatura, umidade, alta tensão, corrente ou ruído intenso, podem fazer com que os sensores tradicionais não funcionem corretamente, ou mesmo nem funcionem [8].

As FBGs também podem ser usados para monitorar outras estruturas como pontes suspensas, plataformas de exploração no oceano e prédio arranha-céus [8]. Esse monitoramento completo em tempo real, com maior precisão e segurança demonstram um grande avanço que o sensoriamento distribuído com FBG pode trazer, principalmente no momento em que o Brasil apresenta um crescimento acentuado, com novas perspectivas nas indústrias e obras de infra-estrutura, especialmente as ligadas ao petróleo e pré-sal [9].

1.2 Motivação e objetivo

Para o sensoriamento óptico com FBGs é necessário toda uma estrutura de equipamentos e tecnologias envolvidas. Dentre eles, o mais importante e o mais caro é o interrogador óptico, sendo o responsável por realizar a aquisição dos dados propriamente dita. A teoria e tecnologias envolvidas nos interrogadores ópticos serão descritas no Capítulo 2. Além do interrogador óptico, equipamento físico, é necessário um software para o gerenciamento de suas atividades e configurações.

Infelizmente, dispositivos tradicionais de medição óptica geralmente fornecem apenas uma funcionalidade de software e uma interface de usuário fixa [27]. Esta falta de flexibilidade não permite que sensores convencionais sejam utilizados em aplicações não-convencionais. Poderíamos usar sensores de tração para monitorar a pressão de uma caldeira industrial por exemplo. Mas esse tipo de tratamento não é feito no software padrão dos fabricantes. Além disso, instrumentos de sensoriamento óptico tradicionais não são projetados para ter uma fácil integração com medições elétricas ou sistemas de controle, o que é muitas vezes necessário no mundo real em aplicações de monitoramento de estruturas, tanto para a indústria

como para pesquisas nas universidades. Outro exemplo seria o próprio projeto TECCON (Tecnologias de Sensores em Fibras Ópticas para Supervisão, Controle e Proteção de Sistemas de energia Elétrica), onde dados de sensores serão utilizados para alarmes online: o software da FiberSensing não possibilita o compartilhamento de dados de forma online.

Essa integração com sistemas de controle poderia ocorrer de modo mais suave com um software específico e conhecimento sobre os protocolos do equipamento. Visto que o interrogador óptico é um produto caro e de difícil fabricação devido à complexidade dos componentes, espera-se que seja possível usar suas capacidades como ferramenta ao máximo.

A Figura 1.1 mostra um Interrogador óptico modelo FS2200 da FiberSensing [31] que foi adquirido pelo grupo de pesquisa do Laboratório de Eletromagnetismo Aplicado (LEA). Ele possui a capacidade de capturar completamente o comportamento dos sensores FBGs em uma fibra óptica, através do espectro de potência dos mesmos, mas o software não disponibiliza tais dados para a aplicação. Porém, é possível desenvolver um software para se comunicar com o interrogador, respeitando o protocolo de comunicação existente, podendo “comandar” e capturar dados do equipamento.

Figura 1.1 Interrogador FS2200.

Outro fabricante consolidado como o National Instruments [43], também possui modelos especializados como o NI PXIe-4844 da Figura 1.2 [27], fornecendo integração completa com o LabView, permitindo que o cliente desenvolva seu próprio aplicativo, atendendo assim a umademanda específica.

Figura 1.2 – Interrogador óptico NI PXIe-4844.

A Micron Optics possui também modelos similares, como o Interrogador *sm225* (Figura 1.3 [44]). Esse fabricante inclusive enfatiza que o seu produto é ideal para customização, respondendo a comandos via Ethernet e a um protocolo adaptado.

Figura 1.3 – Interrogador sm225 da Micron Optics.

Assim, percebeu-se que devido à natureza variada das aplicações possíveis com sensoriamento a partir de FBGs, em alguns casos o próprio usuário precisa desenvolver o seu software, pois aparentemente a necessidade da maioria dos usuários é satisfeita. Porém em universidades, grande parte dos problemas estudados ainda não fazem parte do cotidiano de outros clientes e possuem cunho inovador e aplicabilidade bem específicas.

Desse modo, surge a necessidade de desenvolver aplicativos visando aprimorar as ferramentas já existentes para sensores ópticos e monitoramento. Por exemplo, seria interessante salvar o comportamento completo do espectro refletido pelos FBGs. Possibilitando que novas aplicações possam ser estudadas, como o uso de FBGs simples para medidas de corrente, aumento da largura do pico de reflexão quando sofrido determinado estímulo externo e novos algoritmos de detecção de picos adaptáveis, isto sem mencionar a integração com outros módulos

como um banco de dados e ferramentas de controle para permitir o monitoramento de estruturas via web. Infelizmente o software disponibilizado pela *FiberSensing*, o *iLog*, não nos permite isso, apesar de realizar a captura desses dados através do interrogador. Ele salva apenas o frame atual que está sendo visualizado pela interface gráfica ou somente uma imagem desse frame.

Este trabalho busca o desenvolvimento desse software próprio para realizar a aquisição de dados com o FS2200. Ele deve suprir as demandas existentes e servir de ponte para conectar novos trabalhos, essenciais para o desenvolvimento da pesquisa, tendo acesso direto aos dados no futuro será possível inclusive utilizar técnicas de inteligência computacional para por exemplo, prever comportamentos na redes e auxiliar na tomada de decisões. Sabendo que o grupo pretende desenvolver um sistema de monitoramento para linhas de transmissão (os únicos trabalhos encontrados nesse sentido foram [24] e [25], que não condizem com a realidade brasileira), o software desenvolvido nesse trabalho será a base para esse sistema maior. Realizando a aquisição e arquivamento completo do comportamento dos sensores, sendo possível visualizá-los e disponibilizar os dados brutos de potência e comprimentos de onda em formatos adequados para pós-processamento.

Com esse novo software próprio o aproveitamento do FS2200 será muito superior, possibilitando que haja uma interação e de fato um monitoramento ativo com ele, além do desenvolvimento de novas aplicações, pois com o software atual, é permitido apenas uma visualização simples, sem que o grupo possa atuar e realizar investigação de fato com sensores FBGs e técnicas ligadas a eles.

1.3 Organização do trabalho

O presente trabalho está organizado da seguinte forma:

O **Capítulo 1** já faz uma introdução um pouco mais detalhada do assunto abordado neste Trabalho de Conclusão de Curso, mostrando a motivação e objetivo geral.

O **Capítulo 2** descreve mais detalhadamente como é feito o monitoramento com sensores FBGs, além disso são detalhados os aspectos da teoria de sensores em fibra com Grade de Bragg e interrogadores ópticos, mostrando os princípios de

sensoriamento e técnicas utilizadas em equipamentos como os mostrados nas Figuras 1.1, 1.2 e 1.3. Também são mencionadas técnicas de multiplexação importantes no monitoramento de uma grande estrutura, onde a quantidade de sensores necessários pode alcançar até dezenas de milhares, e dessa forma, equipamentos com múltipla-demodulação se fazem necessários.

O **Capítulo 3** se reserva a mostrar detalhes mais específicos do equipamento adiquirido pelo grupo, o FS2200, e seu software. Ressaltando aspectos como o seu funcionamento, protocolo de comunicação e possibilidades, É detalhado também o software disponível e as suas falhas. Esta avaliação é de fundamental importância para a definição dos requisitos necessários para o novo software a ser construído.

O **Capítulo 4** mostra as contribuições deste trabalho, descrevendo as principais funcionalidades desenvolvidas para o software e os aspectos do desenvolvimento, tecnologias usadas e aspectos ligados a engenharia de software, como diagramas para detalhar as classes envolvidas e funcionamento completo.

O **capítulo 5** faz as considerações finais do trabalho, demonstrando o quanto se avançou com o desenvolvimento do novo software e faz uma coletânea geral sobre todos os trabalhos futuros possibilitados com a conclusão deste TCC em questão.

Capítulo 2

Monitoramento com Sensores em Fibra com Grade de Bragg

Ao longo de muitas décadas, os sensores elétricos vêm sendo usados para monitorar fenômenos físicos, porém algumas limitações existem,como perda na transmissão do sinal, interferência eletromagnética (provenientes de outros equipamentos e ruídos) e baixa durabilidade do sistema. Com o monitoramento de grandes estruturas com FBGs muitos desses problemas podem ser solucionados.

As redes de sensores ópticos possuem FBGs ao longo do percurso óptico e o interrogador em uma extremidade. O interrogador excita as FBGs e analisa a resposta, que é na verdade o espectro de potência óptica refletida por elas. Nessa resposta está codificado o comportamento sensorial dos FBGs, sendo algo fundamental no nosso estudo.

2.1 Uma estrutura de monitoramento com FBGs

Para realizar o monitoramento de estruturas com FBGs é necessário alguns componentes básicos. Dentre eles, como já mencionado, está o array de sensores ópticos e o interrogador óptico. Arranjos dos componentes podem ser feitos de diversas formas, no projeto sob o qual esse TCC foi desenvolvido temos uma proposta de estrutura para monitoramento. Na Figura 2.1 há um array de sensores FBG, que pode estar disperso no concreto de uma estrutura, possibilitando o monitoramento da sua estabilidade de uma maneira eficiente. Neste caso, a idéia é instalar os sensores nas linhas de transmissão de energia, em cabos conhecidos como OPGW (Overhead Ground Wire). Conectado aos sensores, pode-se notar o interrogador óptico, ele realiza a aquisição propriamente dita dos dados de sensores, necessitando de um software externo para comunicação e tratamento dos dados. O software é geralmente instalado em um computador e essa comunicação com o interrogador é feita geralmente através da rede ethernet.

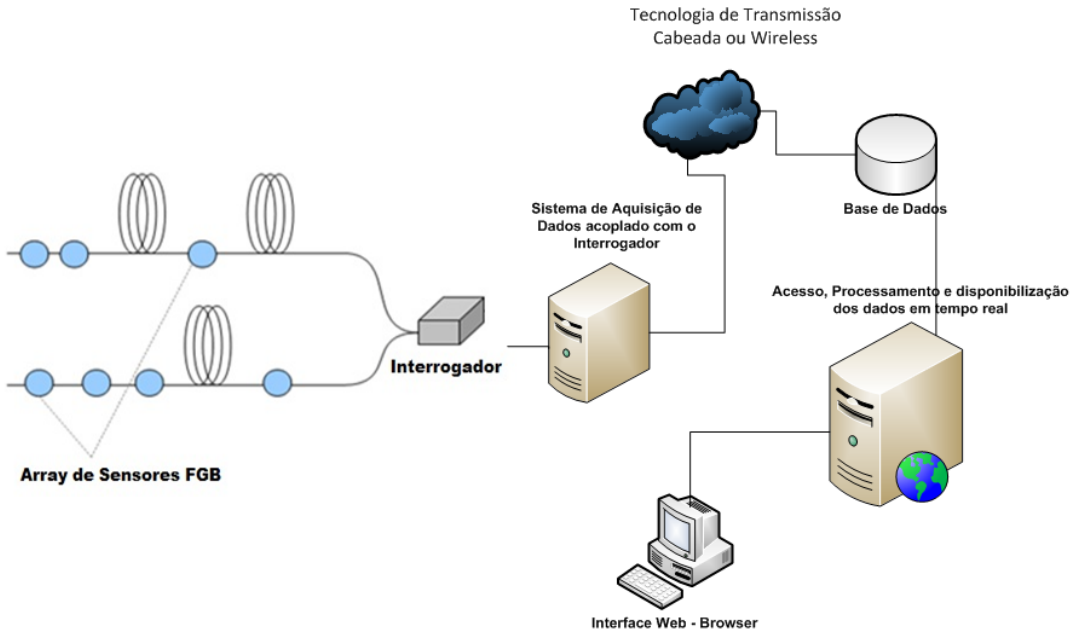


Figura 2.1: Esquema de uma estrutura de monitoramento.

Nesse projeto, deseja-se disponibilizar via plataforma web as condições das linhas de transmissão. Dessa forma, o software externo precisa se conectar a um banco de dados e tornar os dados coletados através do interrogador disponíveis para outros componentes.

Ainda na Figura 2.1 há um sistema web que processa os dados inseridos no banco de dados e os disponibiliza em uma página web. A outra ponta do sistema estaria em um dispositivo que acessaria essa plataforma web e conseguiria visualizar os dados. É importante monitorar linhas de transmissão, pois a sobrecarga de energia pode levar ao sobreaquecimento dos condutores, e a conseqeente interrupção do fornecimento de energia para uma região inteira do país.

A estrutura da Figura 2.1 é apenas um exemplo contextualizado por projeto da própria UFPA. Outro exemplo de aplicação que podemos citar está em [15], onde FBGs são usadas para monitoramento de segurança, avaliando aspectos como a temperatura em uma estação produtora de óleo e gás. E em [17], que é um caso de sucesso no monitoramento de pontes com sensores de FBG.

Porém, todos os exemplos vistos sempre necessitam do software externo ao interrogador. É onde se concentra o trabalho deste TCC. Desenvolver um software capaz de armazenar dados da luz refletida pelo sensor, capaz de fornecer

informações referentes a mudança de temperatura, pressão e deformação no conjunto de sensores [12]. Após realizar a aquisição dos dados, será preciso disponibilizá-los para outros processamentos. Possibilitando o desenvolvimento de ”agentes” inteligentes capazes de prever o comportamento da rede de sensores através do uso de técnicas de inteligência computacional e mineração de dados.

2.2 Teoria de Sensores em Fibras com Grade de Bragg

Um sensor FBG é basicamente uma estrutura física, do tipo refletor Bragg, construído no interior da fibra óptica, capaz de refletir uma faixa de comprimentos de onda, conservando-se relativamente transparente para o restante do espectro [45]. Ela é formada devido a incidência de radiação ultravioleta, ocasionando uma variaçào periódia no índice de refraçào do núcleo da fibra óptica, conferindo à FBG um coeficiente de reflexão em função do comprimento de onda incidente.

A estrutura é bastante sensível a diversos parâmetros físicos externos, eles induzem um deslocamento na reflexão do seu comprimento de onda, esse deslocamento pode ser detectado e usado para se calcular a intensidade de tais parâmetros.

Na Figura 2.2 [10] há a ilustração do princípio de funcionamento de um sensor FBG. Quando uma luz é injetada na fibra, geralmente com o uso de uma fonte band larga, a FBG reflete somente uma faixa bem estreita da luz em torno de um comprimento de onda específico, enquanto que transmite todos os demais. Esse pico de refletividade é conhecido como comprimento de onda de Bragg e notado por λ_B. Essa Luz refletida passa por um acoplador, responsável por encaminhá-la para uma outra fibra óptica onde será feita a detecção do sinal por um elemento fotodetector.

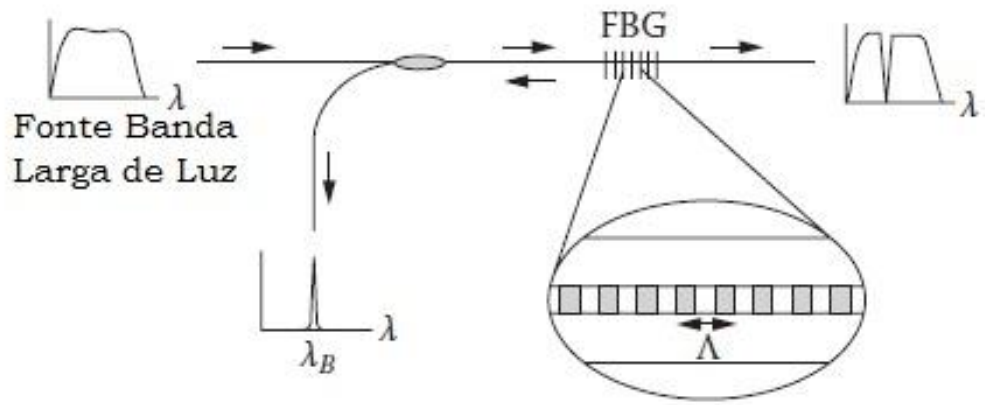


Figura 2.2: Esquemático com a estrutura e a resposta espectral da FBG.

Esse comprimento de onda de Bragg (λ_B) depende diretamente do período da foto-inscrição na fibra e é dado por:

$$\lambda_B = 2\eta_{eff}\Lambda \tag{2.1}$$

Onde η_{eff} representa o índice efetivo do núcleo da fibra e o Λ é o período de pertubação do índice de refração, essa pertubação periódica pode ser modelada da seguinte forma[45]:

$$n(z) = n_{eff} + \delta n[1 + \nu\cos(\frac{2\pi z}{\Lambda})] \tag{2.2}$$

Sendo que o δn é o incremento do índice de refração médio, *v* é a visibilidade de franjas ou razão de modulação e *z* representa a posição da grade. Esta estrutura permite que a luz possa passar de um modo da fibra para outro, possibilitando a resposta refletida em um modo contra-propagante,o que nos é de interesse. A Figura 2.3 [45] ilustra o que os parâmetros de (2.2) representam no perfil de índices de refração periódico de uma FBG de comprimento L, retratada como uma senóide.

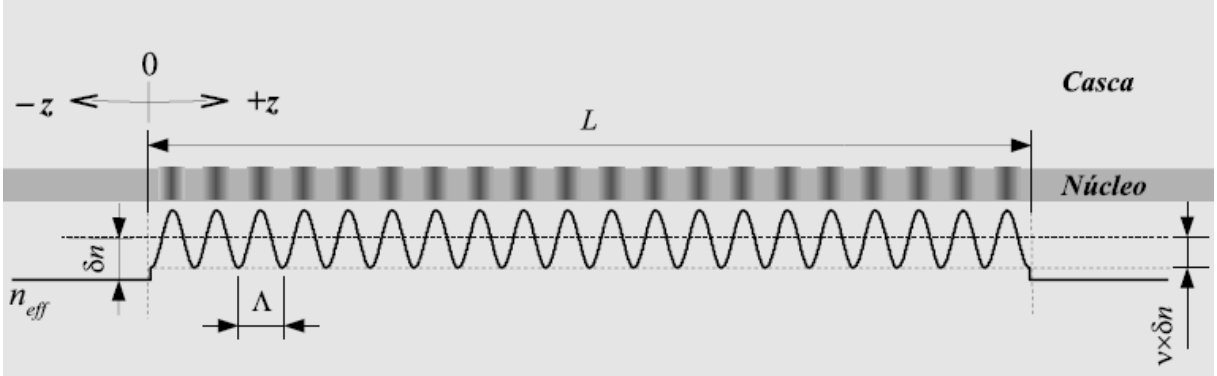


Figura 2.3 – Diagrama de uma FBG uniforme.

2.3 Príncipios de Sensoriamento para FBGs

Para o uso de FBGs em medições de grandezas, como de deformação e temperatura, por exemplo, é necessário entender como uma FBG tem capacidade de funcionar como um sensor. Isso acontece devido a sensibilidade dos parâmetros da grade, tais como índice de refração e período da grade, a uma pertubação externa mecânica ou termal.

Quando um sensor é iluminado por uma fonte banda larga (um LED, por exemplo, ou um laser sintonizável), este reflete uma porção do espectro que é determinado pela condição de Bragg. Pela equação 2.1 e 2.2 podemos notar que o mensurando, ao induzir uma perturbação no sensor, alterará o período da grade (Λ) no núcleo da fibra e o índice de refração efetivo do núcleo (η_{eff}). Originando uma variação no comprimento de onda de Bragg (λ_B), que está relacionado com a grandeza física que será medida através da equação da variação do comprimento de onda de Bragg.

A aplicação de uma deformação mecânica em uma FBG, iria alongá-la (ou comprimi-la no caso de uma deformação negativa), desse modo o período da grade é aumentado (ou reduzido) resultando em um deslocamento do comprimento de onda de Bragg para um comprimento maior ou menor. Portanto, o sensoriamento com FBG é baseado no efeito foto-elástico da fibra óptica, ou seja, a modificação induzida pela deformação da fibra altera o seu índice de refração. Já a sensibilidade a temperatura de uma FBG, depende da mudança no índice de refração induzido, e

em menor grau, do coeficiente de expansão termal da fibra que altera o período da grade.

Há uma sensibilidade cruzada, sendo difícil de distinguir uma perturbação por temperatura de uma por deformação mecânica, caso esteja sendo observado apenas o comportamento do espectro refletido, verifica-se isso até em relação a sensibilidade para os dois parâmetros [23]. Para se observar outro parâmetro diferente de temperatura, uma solução elementar para minimizar a dependência cruzada consiste em se monitorar a temperatura do sensor principal com outro sensor FGB, aproveitando assim o mesmo sistema de interrogação. Este tipo de técnica é usada para compensar a dependência da temperatura para a deformação.

2.4 Príncípios de Interrogação Óptica

O Interrogador óptico ou demoduladores são equipamentos que realizam necessariamente duas tarefas: excitar a rede de sensores ópticos e processar os sinais recebidos [20].

Em laboratórios, para o desenvolvimento de FBGs, analisadores de espectro óptico geralmente são usados para monitorar o espectro de reflexão e transmissão. Apesar de ser um método simples e prático para uso em laboratórios (os quais já possuem os equipamentos), não é atrativo em aplicações reais de sensoriamento devido ao seu tamanho, capacidade de resolução limitada, falta de robusteza e custo relativo alto.

Muitas técnicas diferentes para a interrogação de sensores foram propostas, alguns esquemas são detalhados nas seções seguintes.

2.4.1 Esquemas de Detecção Passivos – Filtros

O modo mais simples de medir a mudança no comprimento de onda refletido de um FBG é usando filtros ópticos linearmente dependentes a essa variação. De acordo com o alcance dessa resposta linear, esse tipo de filtro é chamado de *filtro*

de borda (o qual possui uma banda de transição estreita) ou *filtro banda larga* (o qual tem uma banda de transição bem mais larga) [10].

Esses tipos de interrogadores são baseados na medida de intensidade, isto é, a informação relativa a mudança no comprimento de onda é obtida pelo monitoramento da intensidade da luz no detector. Em vários modelos de interrogação baseados em intensidade, é preciso usar uma intensidade como referência, pois a intensidade recebida pode mudar devido não somente a reflexão do comprimento de onda do FGB (λ_B), mas também ao fenômeno de flutuação de potência na fonte de luz, a distúrbios no caminho da guia de onda, ou até mesmo da depêndencia da luz ao comprimento de onda.

Na Figura 2.4 [10] há o esquemático do sistema de sensor FBG adotando um demodulador baseado em filtros, nesse esquema a luz refletida é dividida em dois feixes, o elemento acoplador faz com que um deles passe através do filtro dependente do comprimento de onda (Figura 2.5 [10] na qual o pico pontilhado representa o espectro refletido pela FBG), enquanto que o outro é usado como referência. O isolador é usado para separa a fonte de luz, evitando que haja perdas de potência devido a ruídos provenientes do resto do sistema.

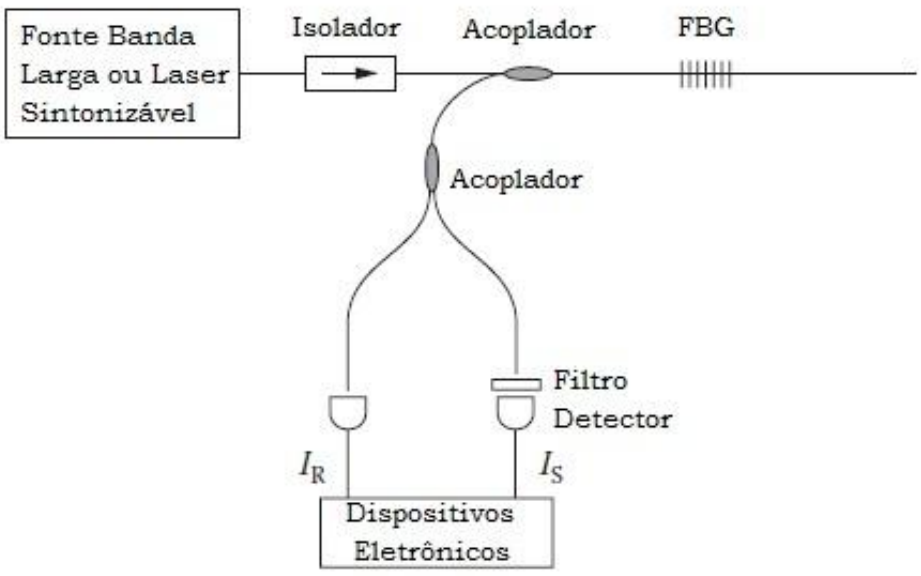


Figura 2.4:Sistema de Interrogação com filtro detector.

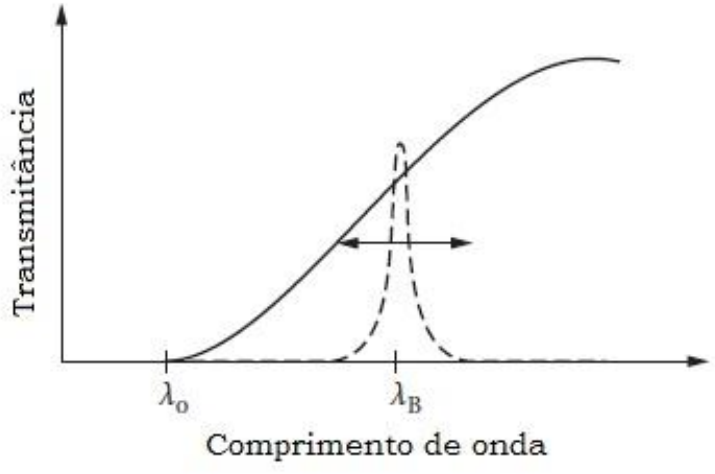


Figura 2.5: Transmitância de um filtro óptico linear.

Como uma FGB é basicamente um filtro com alta refletividade em torno de um comprimento de Bragg. Dessa forma, uma FBG poderia ser usada como filtro de borda. Nesse esquema, a vantagem está na simplicidade, e consequentemente o baixo custo, porém, ele não faz uso da maior vantagem do uso de FGBs, que é o fato da informação do mensurando está codificada no comprimento de onda refletido, e não na intensidade da sua potencia óptica. Além disso, o filtro deteriora a relação sinal ruído (SNR) devido à redução na potência óptica.

2.4.2 Esquemas de Detecção Passivos – CCD

É um método mais robusto que o anterior, realizando uma detecção em paralelo e envolve o uso de um dispositivo detector de carga acoplada (CCD-*charge-coupled device*), também é necessário um elemento fixo de dispersão (difrativo) para converter a posição do comprimento de onda e focar no CCD. A luz é incidida no dispersor, sendo o ângulo de difração dependente do comprimento de onda da luz, assim, a luz com diferentes comprimentos de onda irá iluminar diferentes áreas dos pixels. Essa mudança no comprimento de onda da luz resulta em um desvio da luz no detector em série do CCD. O esquema é mostrado na Figura 2.7 [27] abaixo:

Figura 2.6: Esquema para interrogador com sensor CCD.

Nessa abordagem, todo o espectro da luz é coletado ao mesmo tempo, ou seja, o sinal refletido de cada FBG é lida durante um único período de escaneamento do CCD. Dessa forma, esse sistema possui a vantagem de ser capaz de detectar um sinal mais fraco de luz refletida se comparado com outros interrogadores como o que usa um interferômetro de *Fabry-Perot*. Também é possível detectar-se nesse esquema luzes advindas de várias fibras ópticas, usando uma técnica de multiplexação espacial, na verdade a idéia primitiva do interrogador CCD foi para uso com sensores FBGs em várias linhas.

Esse método é efetivamente rápido, proporcionando medições simultâneas de todo o conjunto de FBGs, porém essa resolução é limitada e possui um baixo SNR. Por exemplo, a detecção de uma mudança de pico FBG de 1 pm num intervalo de 80nm, segundo [27], requer um CCD linear com pelo menos 20.000 pixels, um número muito maior do que os CCDs atualmente disponíveis no mercado possuem.

Interrogador com sensor CCD vem sendo usados com sucesso em aplicações para controle de produtos derivados do petróleo e sistemas de estudo biomédico, como em [29] e [30], respectivamente.

2.4.3 - Esquemas de Detecção Baseado em Varredura Espectral

Muitas configurações de interrogadores existentes fazem o uso de fontes banda larga e de dispositivos para filtragem adequada, capaz de detectar a mudança na luz refletida da grade. Contudo, a maioria dessas fontes banda larga

não são dispositivos de alta potência e a potência é consideravelmente reduzida após ser refletiva por um sensor de banda estreita (FBG), essa redução na potência nos leva a uma baixa SNR, a qual pode reduzir a confiabilidade e aumentar o tempo necessário de interrogação [10].

A alternativa é usar um esquema com uma fonte sintonizável de alta potência (geralmente um laser) e com uma faixa muito estreita. Com o conhecimento do comprimento de onda dessa fonte de linha, ao ser lançada na fibra um sinal, a potência óptica refletida pela grade indica a resposta espectral do sensor para aquele determinado comprimento de onda. Dessa forma, é possível interrogar completamente qualquer mudança no espectro do FBG, o laser sintonizável é sistematicamente sintonizado em comprimentos de ondas consecutivos dentro da faixa operacional. Para cada comprimento de onda o interrogador registra a potência recebida pelo fotodiodo, dessa forma, a resposta de cada comprimento de onda pode ser coletada e comparada, permitindo reconstruir a curva de refletividade da rede de sensores, o esquema é descrito na Figura 2.7 [20].

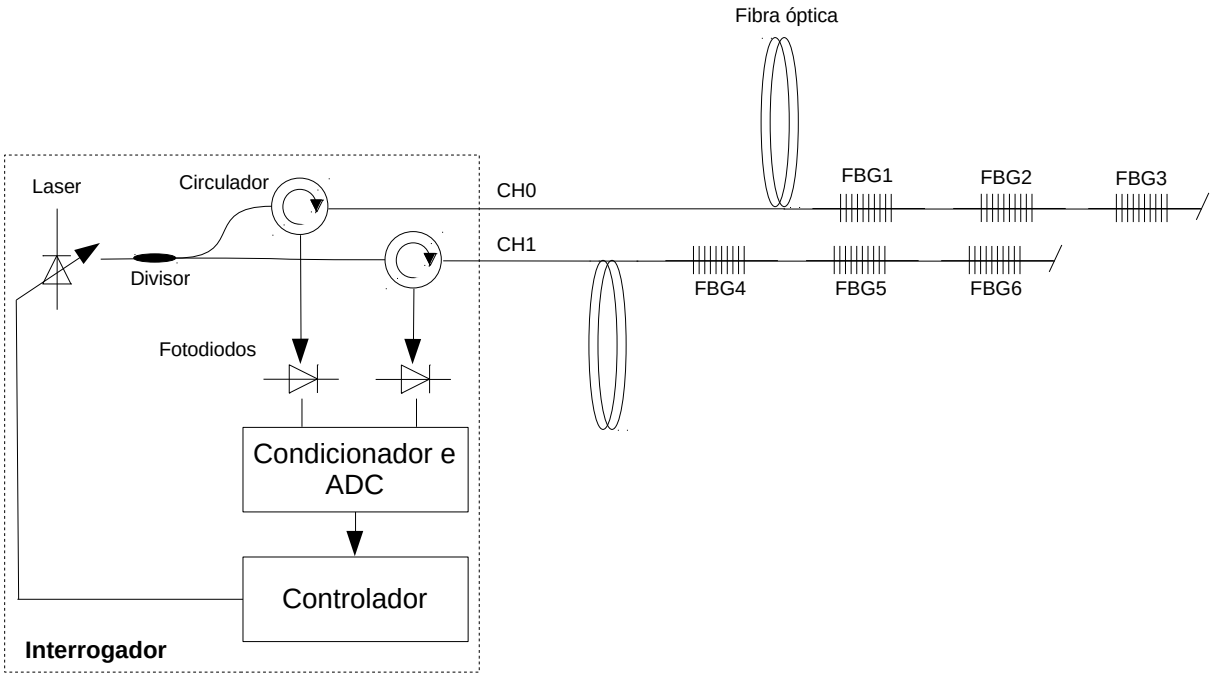


Figura 2.7: Esquema simplificado de um interrogador baseado em varredura espectral.

Uma das vantagens desse tipo de interrogador está na sua flexibilidade, uma vez que mais sensores ópticos podem ser adicionados à rede sem que seja

necessário alterar a estrutura do próprio interrogador. Este método também é muito preciso, oferecendo uma precisão de aproximadamente 1 pm, que se traduz em um FBG com precisão de aproximadamente 1,2 microstrain e aproximadamente 0,1 º C (dependendo do sensor). O uso do laser sintonizável permite a transmissão em comprimentos maiores de fibra (mais de 10 km) por causa de sua alta potência óptica em relação às outras alternativas [27].

Além disso, a alta potência gerada por essa arquitetura permite que uma única fonte de luz seja dividida entre canais, cada qual constituído por 1 fibra, o que reduz o custo e a complexidade de interrogadores multicanal. Compartilhando o mesmo laser sintonizável (a peça mais dispendiosa do equipamento).

Por outro lado, a desvantagem do interrogador de varredura espectral deriva basicamente da velocidade de varredura do laser sintonizável. O processo de sintonia e aquisição ainda demanda de um tempo apreciável de forma que os equipamentos comerciais operam geralmente a taxas de uma janela espectral (da ordem de 100 nm) por segundo. Como cada FBG terá seu espectro processado uma vez por segundo, a taxa de amostragem é de fato a mesma (o FS2200 possui essa taxa de operação): 1 amostra por segundo (frequência de operação de 1HZ) [46].

Em termos comparativos, os interrogadores baseados em filtro são de fato superiores neste aspecto, pois podem operar com taxas de amostragem apenas limitadas pelo sistema optoeletrônico e, principalmente, pelo conversor analógico-digital. Atualmente pode-se encontrar conversores com taxas de até 1 GSa/s (Sa/s é amostras por segundo), o que faz com que os interrogadores baseados em filtro sejam insubstituíveis em determinadas aplicações que exigem uma frequência de operação alta [20].

Por fim, todas as características específicas de cada esquema/método para interrogação podem ser melhoradas com a utilização de uma técnica de multiplexação adequada. Aumentando assim a sua eficiência e o número de sensores por sistema, reduzindo o custo e melhorando a confiabilidade do sistema.

Na Figura 2.7 poderiam ter sido usadas técnicas de multiplexação por comprimento de onda (WDM) ou por por divisão no tempo (TDM). Pois a multiplexação de sensores de fibra óptica permite que o custo por sensor seja reduzido, devido ao número reduzido de fontes ópticas, detectores, moduladores necessários para suportar certo número de sensores na rede.

O número de sensores que podem ser incorporados em uma única fibra depende da faixa de operação e do comprimento de onda de cada sensor e do comprimento de onda total disponível do interrogador. Os fabricantes costumam recomendar um número máximo de sensores na fibra e principalmente que não haja sobreposição entre os comprimentos de onda refletidos (linha tracejada), como na Figura 2.8 [27]. Isso limita bastante o número de sensores por fibra.

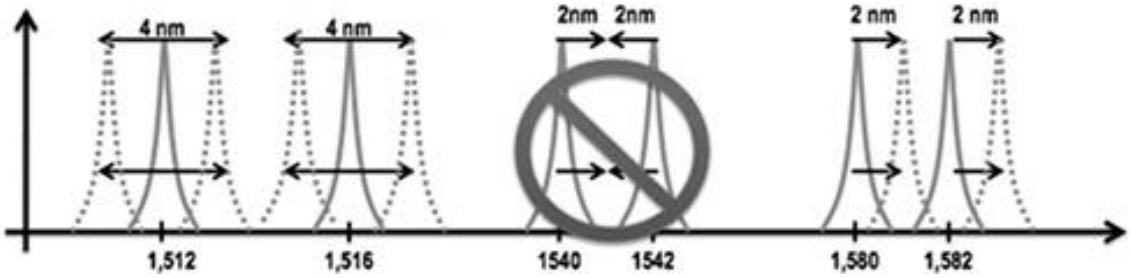


Figura 2. 8 – Faixa de operação de FBGs em WDM.

Contudo, um interrogador pode operar corretamente mesmo que haja sobreposição no espectro, ele precisa apenas utilizar algoritmos de detecção de pico mais sofisticados que os baseado em threshold presentes na maioria dos softwares.

Capítulo 3

Interrogador FS2200 - Software para Aquisição

Como foi visto nos capítulos anteriores, os interrogadores ópticos desempenham um papel fundamental para um sensoriamento de FBGs. Sendo responsáveis por medir corretamente as informações do meio, que chegam até ele codificada no espectro de potência óptica refletido pelos sensores FBGs. Em um sistema de monitoramento completo (como o proposto na Figura 2.1), o interrogador óptico FS2200 desempenharia um papel muito importante, pois é um equipamento capaz de realizar a coleta dos dados dos sensores dispostos ao longo de uma fibra óptica. A interpretação e processamento desses dados capturados é responsabilidade de um software, no caso do FS2200 o *iLog*, integrando o sistema de monitoramento e provendo informações detalhadas sobre a rede, que uma análise permitiria avaliar as condições da estrutura monitorada, infelizmente esse software apresenta algumas limitações.

Neste capítulo segue uma descrição do FS2200, assim como do seu protocolo de funcionamento e software fornecido pela FiberSensing. O novo software desenvolvimento precisa usar o protocolo para conseguir estabelecer uma comunicação segura com o equipamento e realizar a aquisição de dados desejada. Também são mostrados detalhes do software atual disponível, e inicia-se a discussão sobre o próximo capítulo.

3.1- O Interrogador Óptico FS2200

O Interrogador Óptico FS2200 (Figura 1.1) é uma unidade de medições baseado em um laser sintonizável de varredura para interrogação de sensores FBGs (seção 2.4.3). É capaz de operar durante um longo tempo sem necessitar de uma calibração, conferindo assim, uma precisão contínua para as medições.

A saída de alta potência do laser é então dividida em 4 canais, permitindo que até 4 fibras ópticas sejam conectadas simultaneamente no equipamento. Existem diversos modelos de configuração semelhante, alterando apenas a quantidade de canais. Os canais são lidos em paralelo e múltiplos sensores podem ser conectados

em série em cada um deles. Isso permite adquirir dados de um grande número de sensores, algo essencial para um sistema de monitoramento eficiente. O FS2200 é capaz de realizar uma aquisição completa do espectro, na faixa de 1500nm a 1600nm, em um segundo. A resolução alcançada é de 5*pm*. Apesar de ser suficiente para muitas aplicações, a taxa de 1 Hz é uma limitação, pois em casos que o estímulo externo ocorre a uma taxa superior a essa taxa (casos de vibração por exemplo), o interrogador não seria capaz de interpretar o fenômeno. Um estudo detalhado do espectro nessas condições pode trazer a uma formulação matemática para amenizar esse problema, típico de processamento digital de sinais, onde a taxa de amostragem é inferior a taxa de variação do sinal.

Segue na Tabela 3.1, um resumo dos dados técnicos do equipamento segundo o manual do fabricante [31]:

Faixa de operação	**100 *nm* (1500 a 1600 *nm*)**
Sensores por Canal	25 (Máximo recomendado)
Canais Ópticos	4
Taxa de amostragem	1 amostra por segundo
Potência de Saída Óptica	***-3 dBm***
Resolução do OSA	**5 *pm***
Interface de Controle	***Ethernet (TCP/IP)***
Umidade relativa	<90% a 40°C
Comandos de Controle	**SCPI (Strings em texto ASCII)**
Temperatura de Operação	10 a 40°C

Tabela 3.1 : Dados técnicos do FS2200.

O cuidado que se deve ter sobre o número de sensores por canal, é em relação a sobreposição das faixas de espectro entre os sensores vizinhos. Caso o software para processamento seja capaz de identificar FBGs com o espectro sobreposto, como foi discutido no Capítulo 2, esse problema é amenizado. Sabendo que a resolução é de 5 pm, por aquisição são capturados 20000 pontos em cada canal óptico (representando a potência refletida), essa larga faixa de operação e a

saída de alta potência permite uma alta resolução, sendo mantida mesmo para longas fibras ou com conectores que ocasionem perdas.

Como citado na Tabela 3.1, a unidade pode ser completamente controlada usando uma sintaxe de comunicação padrão com comandos ASCII (SCPI). Isso permite que um software seja desenvolvido para adaptar o uso da unidade de interrogação e determinadas aplicações, não sendo restringido o uso ao software padrão do fabricante.

O sistema é capaz de interrogar sensores que estejam a até 10 km de distância da unidade de medição, sem que interferências ou perdas significativas atrapalhem a transmissão na fibra óptica [31]. Diversos tipos de sensores (temperatura, deformação) podem ser usados simultaneamente. Outro fator importante que deve ser levado em consideração é o *threshold*, valor correspondente ao nível de potência a ser considerado na computação para o pico do FBG. Desse modo, picos menores, gerados muitas vezes devido ao ruído, são desconsiderados, como na Figura 3.1 [31] abaixo.

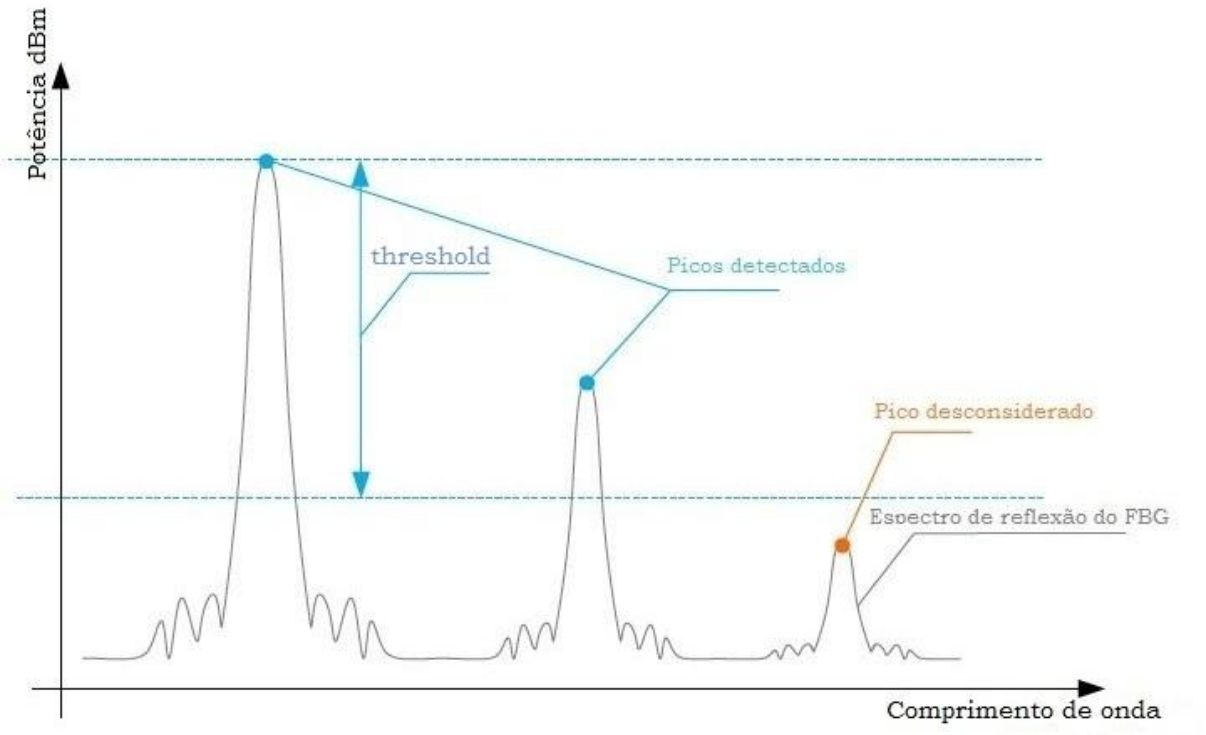


Figura 3.1 – Detecção de pico com Threshold.

3.2 Softwares para interrogadores ópticos

Nos interrogadores, podemos dividir os softwares em dois tipos; o software embarcado que funciona no interrogador, trabalhando diretamente com configurações físicas e controlando os dispositivos opto-eletrônicos que constituem o equipamento. E outro software, geralmente com interface gráfica, que funciona como um “cliente”, o qual acessa o software embarcado, nesse caso através da rede TCP/IP e faz a aquisição dos dados para pós-processamento, visualização e monitoramento. O primeiro tipo é restrito na fabricação do equipamento, mas o segundo está no nosso campo de trabalho podendo ser desenvolvido para aprimorar a utilização do interrogador óptico.

Esse segundo pode possuir as mais variadas funcionalidades, desde monitorar o processamento de dados, gerenciar as informações, mostrar gráficos, gerar relatórios e integração com plataforma web através do banco de dados. Além de fornecerem o suporte para funções mais complexas, como filtragem do sinal para reduzir o ruído, cálculo de picos com algoritmos sofisticados e demodulação do sinal para cálculo de temperatura, deformação, etc.

Para exemplificar, alguns detalhes do software para o interrogador FS2200, serão mostrados. O objetivo é que com a análise das funcionalidades e limitações dele seja mais eficiente a produção de um software específico para atender as demandas existentes, visando uma melhor eficiência para as nossas aplicações, e deste modo, colaborar para o desenvolvimento de um sistema de monitoramento completo.

3.3-Protocolo e funcionamento do FS2200

A respeito do funcionamento e operação, a unidade FS2200 funciona como uma máquina de estados que possui 6 estados operacionais identificados por um número inteiro: 0 (*error)*, 1 (*ready)*, 2 (*free acquisition)*, 3 (*continuous acquisition)*, 4 (*scheduled acquisition)* e 5 (*warming-up)*. Na Figura 3.2 [31] temos uma descrição completa do autômato para o FS2200, assim como os respectivos comandos necessários para a troca de estados.

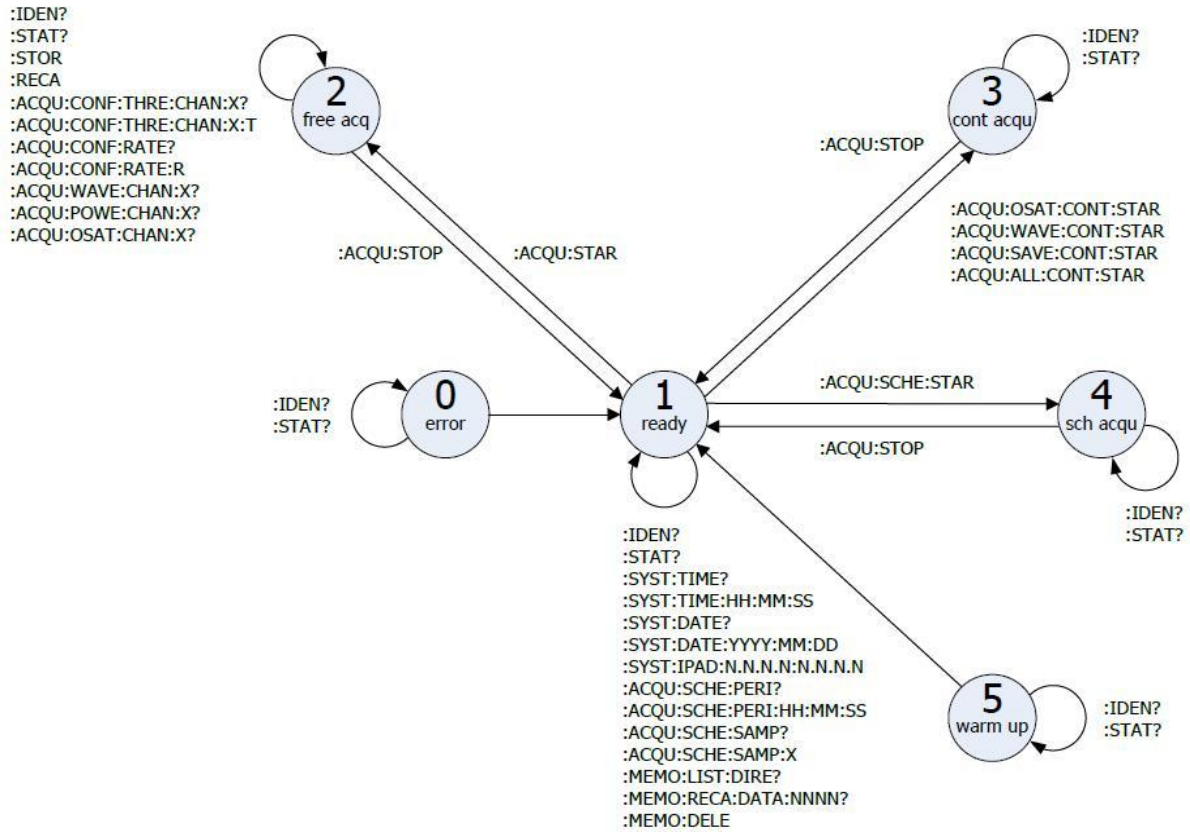


Figura 3.2 : Autômato operacional do FS2200.

A unidade interrogadora assim que é ligada fica alguns segundos no estado 5, o “warm up”, logo em seguida vai para o estado 1, o “ready. Estando nesse momento pronto para iniciar aquisição, condicionado a receber comandos que o determinem realizar tal operação, levando-o assim para um dos tipos de aquisições possíveis. Os comandos aceitos pelo interrogador são palavras no padrão ASCII com argumentos geralmente separados por “:”, ele os recebe de maneira simplificada por letras maiúsculas.

Dentre os possíveis estados de operação, destacamos o estado 2, *“free acquisition”, que* é usado para realizar medidas individuais em um canal óptico específico, quaisquer alterações nas configurações podem ser feitas nesse estado e armazenadas para futuras medições. Há também um estado de “*error”*, 0, correspondente a algum mal funcionamento do módulo optoeletrônico do FS2200.

Porém para este trabalho, o estado mais importante é o 3, *“continuous acquisition”,* o qual permite uma aquisição contínua em todos os canais ópticos e também realiza uma aquisição do espectro de potência óptica a taxa de 1 amostra

por segundo. Já o estado 4, *"scheduled acquisition"*, funciona para realizar uma aquisição periódica de dados em um período e número de amostras pré-definido, armazenando esses dados na memória interna do equipamento.

Todos os estados possuem comandos específicos, listados na Figura 3.2. Porém, antes de enviar esse comando, é necessário especificar o tamanho do texto (comando propriamente dito) através de um inteiro, o mesmo procedimento é feito para as respostas, ou seja, o interrogador também envia um inteiro que representa o tamanho do texto que enviará, inclusive quando estão sendo enviada as medidas no "*continuous acquisition*", um texto com caracteres representando 20.000 números de dupla precisão.

Por exemplo, a Tabela 3.2 abaixo, exemplifica a troca de informações entre o software e o interrogador, todas ocorrem na porta 3500:

Troca de mensagens via TCP/IP	
Software >> Interrogador :	>> 6 >> ":STAT?"
Interrogador >> Software :	>> 6 >> ":ACK:1"
Software >> Interrogador:	>> 20 >> ":ACQU:OSAT:CONT:STAR"
Interrogador >> Software:	>> 4 >> ":ACK"

Tabela 3.2: Troca de mensagens.

O exemplo mostrado, refere-se a quando o interrogador está no estado 1 (*ready*) e deseja-se iniciar uma aquisição contínua. Para isso é preciso enviar o comando em texto ":ACQU:OSAT:CONT:STAR" para a porta 3500, fazendo com que o interrogado vá para o estado 3 (*continuous*) e os dados correspondentes possam começar a serem recebidos através da porta 3365. A resposta é um array de 20001 pontos vezes o número de canais ópticos, esses valores correspondem aos valores de potência para todos os canais ópticos no comprimento de 1500nm a 1600nm com resolução de 5pm. Os valores de potência são enviados em dBm, separados por "**,**", e as medidas referentes a cada canal estão separadas por "**:**".

Como os comando são enviados através da rede Ethernet, uma falha na conexão poderia representar para o sistema de monitoramento uma falha grave de segurança, visto que o sistema ficaria fora de operação, por isso uma conexão segura é importante.

Ainda em relação ao autômato, para o novo software desenvolvido, apenas alguns estados da Figura 3.2 foram necessários, os comandos importantes para o seu funcionamento serão melhor detalhados no Capítulo 4, assim como a troca de estados no autômato e fluxo de atividades.

3.4-Software da FiberSensing

A maioria das fabricantes de interrogadores ópticos possui um software padrão com uma interface gráfica para visualizações básicas, o da FiberSensing é denominado de *"iLog"*. A aba de nosso interesse é o OSA (Optical Spectrum Analyzer), o seu propósito principal é prover informação gráfica sobre algumas características do espectro dos FBGs de cada canal óptico. Ela possui basicamente 2 áreas funcionais, uma barra de controle na parte de baixo, os quais estão sempre disponíveis e uma área superior a qual possui vários menus separados por abas, a tela inicial é mostrada na Figura 3.3 [32] abaixo.

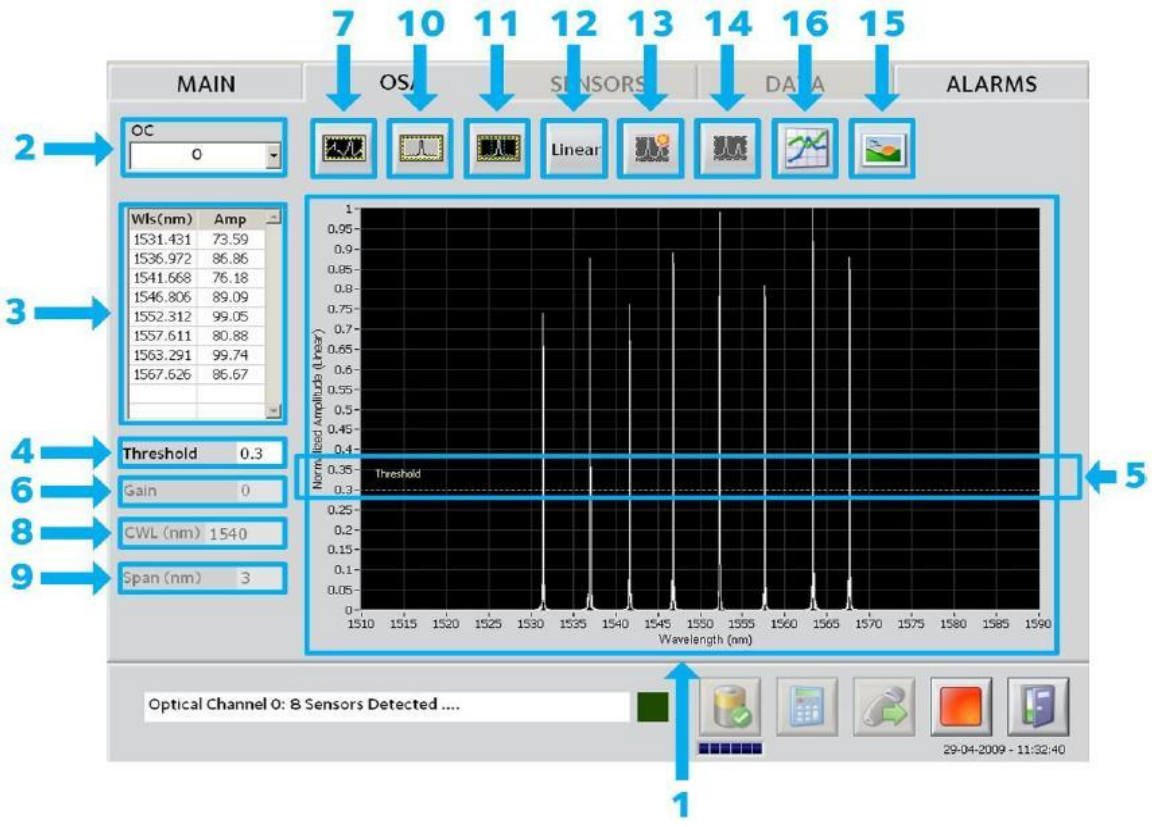


Figura 3.3: Interface gráfica do software iLog.

Sobre a Figura 3.3, o gráfico numerado pelo item 1, mostra a visualização da distribuição de potência espectral para cada canal. O item 2 permite a escolha para visualização dentre os canais possíveis (o número do canal pode variar dependendo do equipamento). Em 3 são listados os picos encontrados, assim como seus respectivos comprimentos de onda. Configurações de *threshold* para ser usado no algoritmos de detecção de pico estão em 4, a escolha adequada desse parâmetro faz com que o algoritmo seja mais eficiente e preciso. Os itens 6,8 e 9 dizem respeito a parâmetros de configuração que não estão disponíveis para alteração no software como ganho em relação ao laser emitido, comprimento de onda central e faixa de medidas.

Os ícones da parte superior da tela estão ligados a configurações de visualização do software, como ajuste automático de zoom (item 7), visualização baseado no alcance (item 10), alternar de escala logarítmica para linear (item 12) e os demais para faixas de escala horizontal e vertical.

É importante que no caso de ser alternada a visualização do modo linear para o modo logarítmico, o valor do *threshold* deve ser atualizado também para se adequar a mesma escala e continuar a ser funcional, como mostrado na Figura 3.4 [32].

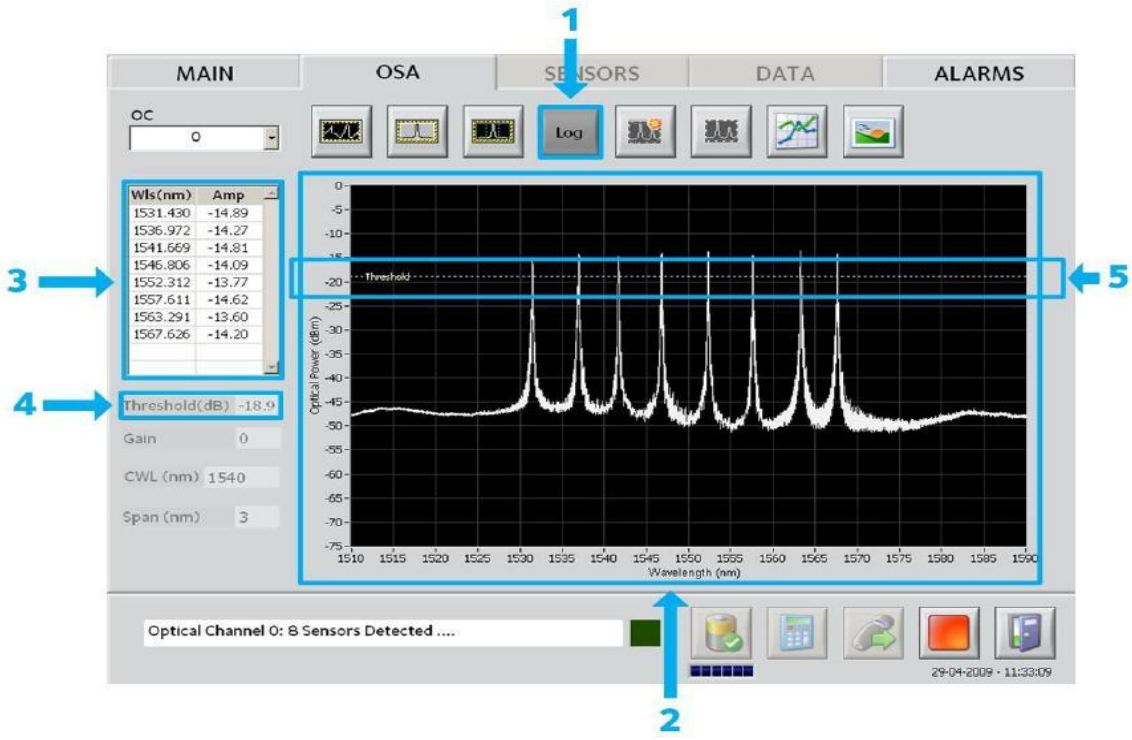


Figura 3.4 – OSA com visualização em escala logarítmica.

Algo confuso sobre o iLog, é que a versão v3.2.7 do software não permite a visualização em escala logarítmica para alguns modelos como o FS2100 e o FS2200 Industriais. Sendo que para isso, é necessário apenas alguns cálculos para realizar tal conversão.

Já com o pressionamento do botão 15 na Figura 3.3, é possível salvar uma figura do espectro atual que está sendo visualizado, o formato padrão é bitmap(bmp). Porém apenas uma figura do espectro do FBG não nos fornece dados suficientes para uma análise profunda do comportamento dos sensores. Elesservem para a ilustração do monitoramento.

Para o monitoramento através de FBG, os dados mais importantes, que são sobre o espectro óptico refletido, não podem ser manipulados pelo software atual. Além disso, o fato de não ser possível acessar dados brutos dos sensores, faz com que não seja possível integrar o software a um sistema de monitoramento ou de controle. Entende-se por dados brutos, os valores de potência refletida e seus respectivos comprimentos de onda. Isso reduz drasticamente as aplicações possíveis para o equipamento, por exemplo não há como usá-lo para estudar o comportamento dos sensores em condições específicas. Também não é possível o desenvolvimento de algoritmos para detecção de pico mais eficientes para determinadas condições de excitação dos sensores.

Por fim, fica evidente a necessidade de um software que possibilite o armazenamento completo do espectro dos FBGs durante um determinado tempo, salvando assim o comportamento dos sensores. Inclusive permita o carregamento de medições feitas anteriormente para um estudo e análise melhor detalhada de cenários que possam ocorrer constantemente para tipos de monitoramento específicos.

3.5-Requisitos para o novo software

O novo software deve permitir que equipamentos, como o FS2200, possam ser utilizados para outras funcionalidades, como testar novos métodos para se medir corrente ou técnicas mais eficientes para o uso de FBGs.

Dentre as necessidades para o novo software, as mais importantes são:

1. Conectar-se de forma estável e trocar comandos com o interrogador óptico.
2. Visualização do espectro de potência óptica em tempo real, taxa de 1 amostra por segundo e escolha entre os canais disponíveis.
3. Armazenamento dos dados coletados, permitindo que sejam realizados um pós processamento dos dados.
4. Carregamento de dados coletados anteriormente.
5. O software deve ser escalável, permitindo que processamentos futuros para os dados sejam facilmente adicionados, assim como integração com banco de dados.
6. Ser capaz de detectar os picos através da configuração de um *threshold.*
7. Permitir também a visualização tanto em escala linear como logarítmica.

Portanto, necessita-se prioritariamente que um novo software tenha a capacidade de extrair os dados brutos coletados diretamente da fibra óptica. Esse novo software precisa ser capaz de realizar esse interfaceamento com o interrogador óptico, realizando a aquisição dos dados, permitindo a visualização e ao mesmo tempo realizar o arquivamento dessas medições para trabalhos de pós processamento. Ele será a “ponte” inicial no desenvolvimento de outras ferramentas para um sistema completo de monitoramento através de FBGs, como mostrado na figura 1.3.

3.6-Possibilidades e vantagens de um software adaptável

Com um software desenvolvido de acordo com necessidades específicas,, há ganhos de conhecimento tanto em desenvolvimento de software, tanto como um melhor entendimento a respeito do funcionamento de um equipamento de alta tecnologia como o interrogador óptico FS2200.

Devido a utilização do software iLog, percebeu-se que o software até atende a necessidade da maioria dos clientes, que precisam apenas realizar um monitoramento simples e pontual, que não seria o caso de novas pesquisas em centros de pesquisa. Como deseja-se desenvolver aplicações para o monitoramento

de linhas de transmissão, com um software próprio seria possível desenvolver não só algoritmos de detecção de picos, como também avançar em uma linha de pesquisa sobre medição de corrente através do efeito Faraday, e para isso, é fundamental o acesso aos dados brutos coletados. Também é possível desenvolver técnicas matemáticas para cálculo de vibração dos sensores para os casos em que a velocidade de vibração é superior a velocidade de amostragem, ou seja, vários problemas relacionados a processamento de sinais podem ser explorados com a nova ferramenta, sendo fundamental para o avanço de novas pesquisas na mesma linha e fazendo com que o aproveitamento do interrogador óptico seja muito superior ao atual, retornando também, o investimento na sua aquisição.

Algumas das possibilidades de uso serão listados como trabalhos futuros, visto que com o desenvolvimento desse software muito mais implementações poderão ser feitas com o mesmo para a continuidade do projeto.

Capítulo 4

Novo Software Desenvolvido e Interface Gráfica

O uso de sensores FBG em rede conjuntamente com equipamentos modernos como os interrogadores ópticos e softwares de qualidade, possibilitam poderosos recursos de aquisição de dados, processamento e transmissão dessas informações. É o que vem sendo desenvolvido nos últimos anos em [12] por exemplo. Assim, para que possamos ter um sistema de qualidade, o desenvolvimento de nossos próprios componentes de software é de extrema importância. O software padrão não atende a demanda existente, deixando muito a desejar diante de todas as funcionalidades possíveis. Inclusive em ações básicas como armazenar o registro de todo o comportamento dos FBGs na fibra óptica.

Nesse capítulo serão discutidos as funcionalidades do novo software desenvolvido, e as implementações realizadas. Detalhando-o, para explicar como ele realiza algumas operações concorrentes, que exigiram cuidados especiais por parte do programador, principalmente com o uso de *threads* [40].

4.1 Funcionalidades Principais

Devido ao discutido nos capítulos anteriores, constatou-se a real necessidade do desenvolvimento de um software próprio, que estabelece uma comunicação com o interrogador óptico FS2200 via rede (Ethernet), comunicando-se através de comandos definidos no automato da Figura 3.2, a fim de realizar a aquisição de dados a partir do interrogador.

O software precisa inicialmente realizar as configurações em relação a rede TCP/IP. Por padrão o IP do equipamento é 10.0.0.10, porém esse valor pode ser alterado para se adequar a diversas redes, não sendo necessário alterar o IP de toda a rede, modificando apenas o do equipamento, para isso existe inclusive um comando que pode ser enviado quando o interrogador está no estado 1 (Ready) pela porta padrão 3500, “:SYStem:IPADress:N.N.N.N:N.N.N.N”, onde N são números que representam os octetos do endereçamento IP, esse comando faz com que o novo IP seja os primeiros 4 octetos e o segundo quadro de octetos se torne a

máscara. Como as comunicações são realizadas através de sockets, que é uma interface programada para trabalhar com a rede [35], é preciso também definir a porta para comunicação. O modelo FS2200 usa a porta 3500 para comunicação e comandos e geralmente a porta 3365 para o recebimento de dados, outros modelos podem usar portas diferentes.

Após as configurações de IP e de portas terem sido realizadas, o software se conecta com o interrogador óptico através da rede e começa a ser capaz de trocar mensagens com ele. O software precisa ser capaz de prover uma boa visualização do espectro de potência refletido pelos FBGs, a atualização dos dados é aproximadamente a mesma que a velocidade de varredura, 1 Hz. Já que o FS2200 possui mais de 1 canal, também deve ser permitada a escolha de qual canal se deseja visualizar.

Outro aspecto ligado a visualização, está a possibilidade de se alternar entre a escala logarítmica e linear, para isso faz a conversão de dBm para milliWatts através da seguinte fórmula [36] , onde M é o valor calculado em milliWatts:

$$M = 10^{\frac{dBm}{10}} \qquad \textbf{(4.1)}$$

É importante lembrar que o interrogador sempre faz a aquisição naturalmente em dBm, unidade de potência relativa a 1mWatt [36].

O ponto fundamental e diferencial do software é que ele permite o armazenamento completo do espectro de potência dos sensores em todos os canais, portanto, desde o início do recebimento dos dados, é armazenado frame por frame, entende-se por frame os valores dos 4 canais coletados em uma varredura espectral, ou seja, o software faz um registro completo do comportamento dos sensores, permitindo uma análise criteriosa e um pós processamento com os dados completos disponíveis.

Além disso, o software também permite que os dados armazenados sejam carregados para visualização no software, reconstruindo a simulação. Tanto no modo de carregamento quanto no modo de aquisição via rede, o software possibilita a detecção de pico e a configuração de um threshold adequado para isso. Caso o usuário ainda queira salvar uma imagem do espectro, isso também é possível, assim como o software iLog (Capítulo 3) faz. Também é possível definir uma faixa

específica para visualização, algo importante quando se sabe o comprimento de onda central do FBG e se deseja avaliar seu comportamento ao sofrer alguma excitação do ambiente.

4.2 Descrição do software

Primeiramente, foi necessário escolher uma linguagem de programação para o desenvolvimento do software. Alguns teste de comunicação com o equipamento via rede ethernet foram feitos em C++, porém, pretende-se que o software tenha um papel importante dentro do escopo maior do projeto o qual está inserido. Assim, ele foi desenvolvido em Java, dentre os motivos, está o fato de Java permitir que o software seja multi-plataforma, necessitando apenas que uma máquina virtual esteja instalada, não necessitando que o código seja compilado novamente para plataformas diferentes [34]. Além disso, outras ferramentas estão sendo desenvolvidas em Java, como um site para fazer a integração do equipamento com um sistema de monitoramento, e também pois a persistência em banco de dados é estável com Java, através do uso de Hibernate [38], dessa forma, todos os produtos de software desenvolvidos sob o mesmo projeto seguiriam um padrão de tecnologias.

Para a funcionalidade de visualização em tempo real do espectro refletido pelos FBGs, foi utilizada uma biblioteca para gráficos, chamada de JfreeChart [36], é uma biblioteca totalmente livre, liberada sob a *GNU Lesser General Public Licence.* Ela permite a criação de gráficos dos mais variados tipos, pizza, linha, de dispersão e de séries temporais. Ainda possui algumas características adicionais como: exportar para imagem em .PNG e .JPEG, dicas de ferramentas com o clique do mouse e zoom interativo.

A jFreeChart não realiza uma "atualização dos dados" propriamente dita, e sim uma reconstrução completa do gráfico, apagam-se os dados antigos e os novos dados são adicionados. É possível notar que o software possui alguns “gargalos”, pois é preciso esperar que os dados sejam recebidos pela rede para que possam começar a serem plotados no gráfico (independente de os valores terem sido alterados ou não).

Apesar disso, para a nossa aplicação a JFreeChart demonstrou ser suficiente, sendo capaz de realizar uma atualização por segundo, a qual não só é suficiente para o propósito, como inclusive é a mesma taxa de operação do FS2200, 1Hz. É importante mencionar que aplicações em tempo real de segurança precisam ser projetadas com cuidado, por exemplo, usar a aplicação para fechar uma válvula de emergência. O sistema operacional comum pode não ser capaz de realizar esse processamento em "tempo real".

Para se adicionar uma grande quantidade de pontos no gráfico no tempo de em 1 segundo, alguns conhecimentos de como trabalhar com texto em Java e interfaces gráficas foram necessários. Operações longas como as que serão explicadas no item 4.4 podem fazer com que a interface gráfica congele.

Fora a parte gráfica, o software ainda precisa realizar várias operações ao mesmo tempo, o que exige um controle adequado do fluxo de processos. É necessário estabelecer a comunicação com o equipamento e iniciar o recebimento dos dados eles são enviados continuamente no formato de palavras concatenadas, e armazenadas em variáveis do tipo *double* (dupla precisão – 8 bytes), são então adicionados no gráfico de acordo com a escolha do canal.

Dessa forma, para que fosse possível que essas ações acontecessem continuamente e a cada 1 segundo, foi necessário a utilização de threads [40]. Possibilitando que sempre houvesse um processo sendo executado pelo software.

O cálculo para transformação da escala linear para logarítimica também usa threads, caso contrário o software todo iria paralisar enquanto esse cálculo é feito, visto que é um vetor com muitos elementos (20.001), o mesmo ocorre para a busca de pico.

Para o cálculo do pico, é usado o valor de referência do *threshold*, a precisão depende basicamente da resolução do equipamento, no caso do FS2200 é de 5pm. O algoritmo implementado é o de busca simples do máximo, encontra-se o máximo dentro de um intervalo onde os valores de potência são maiores que o threshold, salva-se o ponto de busca, e inicia-se a busca de próximo pico a partir desse ponto.

Segue a descrição do algoritmo de pico baseado em *threshold*, onde vetor(i) é um vetor contendo todos os elementos capturados pelo interrogador, e i o índice desse vetor:

```
1- VARIÁVEIS: índice_pico, índice_busca, primeiro_pico.
2-ÍNICIO
3- SE primeiro_pico == true
4-       índice_pico, índice_busca =0;
5- PARA I <- índice_busca ATÉ fim do vetor  FAÇA:
6-       SE vetor(I) > threshold FAÇA:
7-                ENQUANTO  vetor(I) > threshold
8-                         índice_pico <- I e índice_busca <- I;
9-                         I<- I+1;
10-               FIM ENQUANTO;
11-               RETORNA ponto_de_busca e máximo;
12-      FIM SE;
13-     SE NÃO
14-          I <- I +1;
15-         índice_busca <- I;
16- FIM PARA;
17- FIM ROTINA
```

Figura 4.1 – Algoritmo para detecção de pico baseado em threshold

Os dados que estão sendo visualizados (recebidos via rede) são salvos automaticamente em um arquivo temporário, ação que também ocorre paralelamente, por isso é importante o cuidado com a consistência desses dados e variáveis. Caso deseje salvar o arquivo, o usuário precisa se desconectar (fazendo cessar o recebimento via rede), para poder renomear o arquivo temporário para um nome escolhido, apagando o arquivo temporário logo após.

Para que todas as ações do software aconteçam a cada 1 segundo, há um agendador de tarefas existente como recurso do Java, chamado de Timer [41]. Ele é responsável por invocar as tarefas da Tabela 4.1, uma lista das atividades em paralelo que ocorrem durante o funcionamento do software.

1-	**Leitura dos dados via rede/socket**
2-	**Processamento do texto e armazenamento em váriaveis**
3-	**Atualização dos dados no gráfico jFreeChart**
4-	**Escrita em arquivo temporário**

Table 4.1 – Atividades disparadas pelo Timer a cada 1 segundo.

Para que o software deixe de disparar as tarefas, é preciso desconectá-lo, realizando uma chamada a “ThreadDisconnect”. Ela possui um papel importante, pois além de parar o agendador de tarefas, ela precisa esperar os fluxos que estão em execução acabarem para então fechar a conexão, pois é importante garantir a consistência do arquivo que está sendo gravado garantindo o seu formato adequado.

Pode-se perceber que para cumprir as tarefas da Tabela 4.1, usou-se bastante Threads [40], operações como a concatenação de dados muito extensos e visualização on-line de dados transmitidos pela rede exigiram isso. Durante o desenvolvimento do software percebeu-se que a interface gráfica não conseguia atingir seus objetivos de desempenho com um fluxo contínuo, sendo necessário a divisão para fluxos semi-paralelos. Lembrando que apenas o uso de threads não faz o processo correr em paralelo, somente com mais de um processador isso seria possível.

4.2.1 Diagrama de Caso de Uso

No diagrama de casos de uso, podemos representar as funcionalidades e ações disponíveis para o usuário do software, como há apenas um tipo de usuário (o ator do processo), o diagrama fica relativamente simples como podemos ver na Figura 4.2:

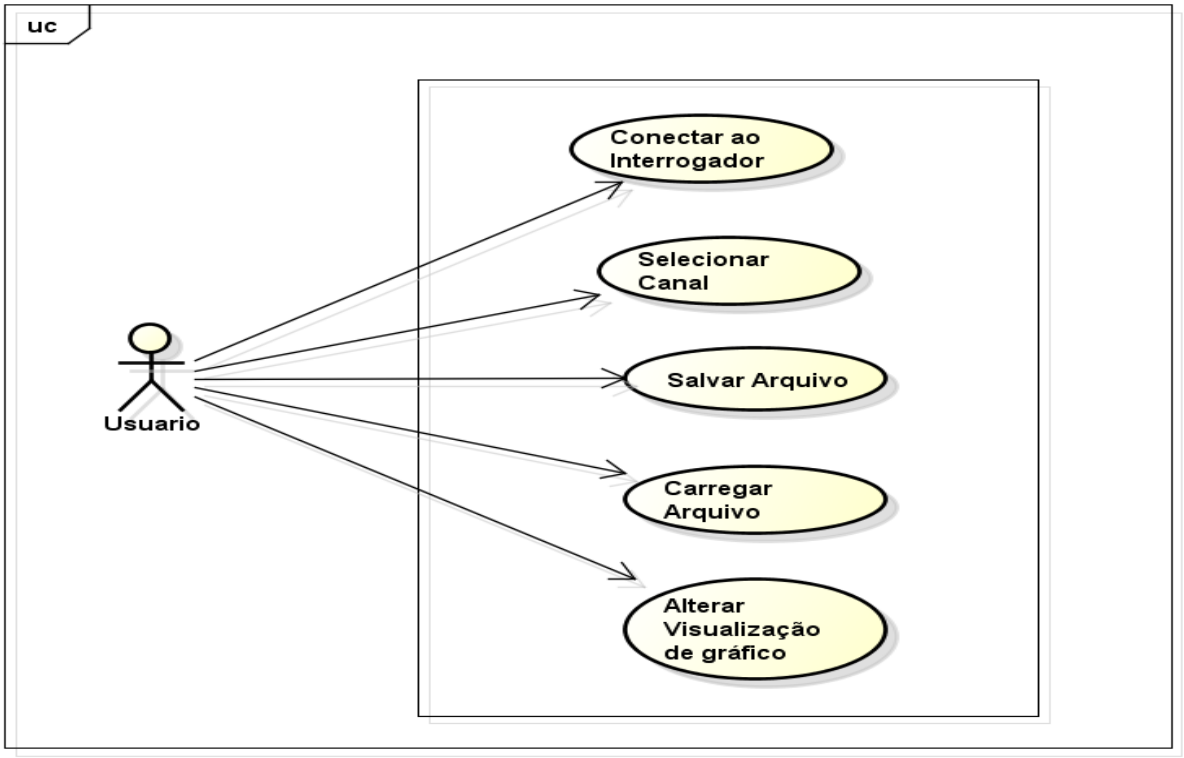


Figura 4.2 Diagrama de casos de uso.

Dessa forma, pode-se notar que o ator do sistema pode realizar a operação de se conectar ao interrogador, e assim como já discutimos, iniciando o funcionamento do software. Visto que o diagrama de caso de uso não mostra como o sistema realiza as ações de forma temporal, outros casos de uso são listados para o usuário, sem que o interrogador necessite estar enviando dados ao software.

Assim, o usuário pode Salvar os dados coletados em um arquivo específico e/ou carregar arquivos previamente salvos pelo software. A respeito de visualizações, ele pode optar por escolher qual canal prefere visualizar os dados, além de diversas outras opções relacionadas a faixa de visualização e grandezas, todas listadas no caso de uso “Alterar Visualização de gráfico”.

4.2.1 Diagrama de Classes

O diagrama de classes é usado para descrever os vários tipos de objetos no sistema e o relacionamento entre eles. Na Figura 4.3 está esquematizada a estrutura das classes do software. O diagrama de classes completo com o comportamento dos atributos e métodos está no apêndice A.

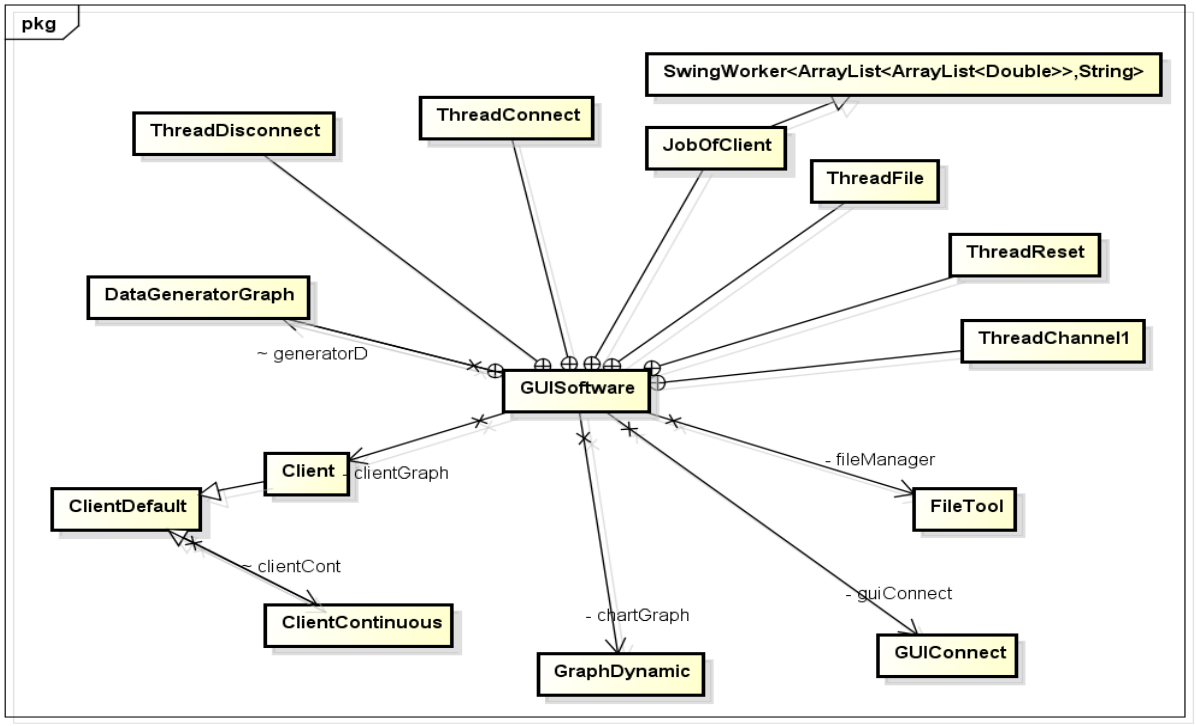


Figura 4.3 Diagrama de Classes do Software.

Podemos notar que a Classe GUISoftware é a principal do sistema, onde ocorre o fluxo principal do programa, ela é responsável por instanciar as outras classes e gerar a interface gráfica, além dos elementos para o funcionamento correto do sistema como um todo. Instanciando objetos das seguintes classes:

- Client
- FileTool
- GraphDynamic
- GUIConnect
- DataGeneratorGraph

4.2.1.1 Classes para conexão dos clientes

A classe ClientDefault está relacionada com a comunicação em rede via TCP/IP, ela é responsável por garantir que a conexão se estabeleça de acordo com o endereço IP e portas desejadas, ela instancia uma classe filha ClientContinuous, que é responsável por receber e processar os dados vindos do interrogador no estado continuous acquisition. A Classe Client também herda da classe ClientDefault, porém é responsável pela comunicação com o interrogador, ela por

exemplo faz a checagem do estado, e envia os comando adequados (listados no autômato da Figura 3.2) para preparar o equipamento para o envio do espectro de potência.

A figura 4.4 mostra o fluxograma do processo de recebimento de dados pela classe “Client”:

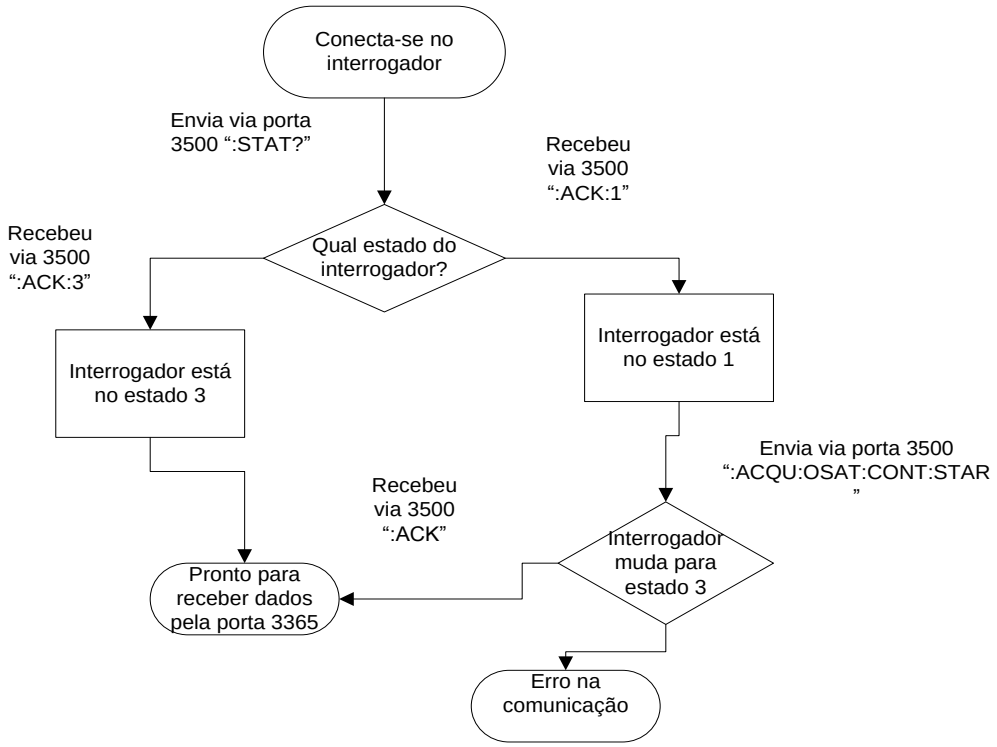


Figura 4.4: Fluxograma do cliente para receber dados.

A classe “GUIConnect” é a responsável pela interface gráfica de uma tela de configuração da rede, permitindo a conexão via rede TCP/IP com o interrogador. Ela também é responsável pelo recebimento dos dados, através de um Timer que agenda as tarefas da tabela 4.1 para ocorrem a cada 1 segundo.

4.2.1.2 Classes de arquivos

A Classe FileTool é a responsável por todas as funções ligadas a arquivos. Logo que se inicia o recebimento de dados pela porta 3365, ela cria um arquivo temporário para armazenar os dados, depois copia esses dados para o arquivo definido pelo usuário. Também possui os métodos para prover o carregando dos dados a partir de arquivos salvos. Desse modo o software faz a animação de dados

previamente capturados, simulando no gráfico como se a aquisição desses dados estivessem ocorrendo exatamente on-line. Isso permite que o comportamento gráfico do espectro possa ser analisado visualmente após o seu armazenamento.

Os métodos da classe ThreadFile ocorrem simultaneamente com as outras atividades do software, merecendo um cuidado especial, pois é preciso garantir que todos os dados são salvos no formato adequado, permitindo assim que os arquivos usados sejam usados com segurança depois. O formato dos arquivos é melhor detalhado na seção 4.3. Esse arquivo pode ser apagado e reiniciado a qualquer momento em que os métodos da classe ThreadReset forem chamados.

4.2.1.3 Classe para gráfico

A classe GraphDynamic faz uso da biblioteca jFreeChart mencionada anteriormente, criando um gráfico de linhas que é adicionado ao resto da interface gráfica na classe GUISoftware. Possui diversos objetos responsáveis por configurar a visualização do gráfico, alteradas dependendo da grandeza escolhida (logarítmica ou linear).

A classe DataGeneratorGraph, apenas configura o agendador para instanciar um objeto JobOfClient a cada 1 segundo. Ele implementa o SwingWorker [42], uma classe abstrata usada quando componentes gráficos (biblioteca Swing do JAVA) precisam realizar tarefas longas, caso contrário a interface gráfica congela enquanto espera a tarefa ser finalizada.

Nesse caso as tarefas longas são as mencionadas na tabela 4.1, como processar uma String com 4 linhas, cada linha com 20 mil campos que são convertidos para vetores de pontos-flutuantes de dupla precisão; E enviar esses dados para serem adicionados no gráfico (através da classe GraphDynamic), essa tarefa é chamada pela classe ThreadChannel, a qual adiciona os dados de apenas 1 canal ao gráfico, esse canal é determinado pela caixa de seleção de canais na interface gráfica principal. Exibir os dados de todos os canais simultaneamente comprometeria a visualização por parte do usuário.

4.3 Formato de dados

Quando o recebimento de dados é iniciado pela porta 3365, o interrogador envia um vetor com 20001 pontos vezes o número de canais ópticos. Os canais são separados por ":" e os valores são separados por ",". Por exemplo seria: **"-33.297,-33.303,-33.311,-33.291,-33.256, ...,-33.235:-33.285,-33.358.."**, onde cada valor representa a potência óptica recebida em dB. O interrogador permanece enviando esse texto continuamente, enquanto o software permanecer conectado na porta 3365.

O outro formato importante é o dos arquivos salvos, basicamente é usado um marcador para o frame, e no arquivo é salvo o número do frame, seguido de 4 linhas com os valores de potência separados por ",". Cada linha corresponde a um canal óptico do equipamento. Exemplo, lembrando que cada linha possui 20001 pontos:

```
1
-33.297,-33.303,-33.311,-33.291,-33.256,-33.228,-33.235,- 33.052,-32.967,…,
-33.092,-33.18,-33.23,-33.231,-33.214,-33.21,-33.247,-33.33,-33.434,-33.508,…,
-33.482,-3.424,-33.379,-33.356,-33.36,-33.378,-33.385,-33.381,-33.373,-33.376,…,
-33.388,-33.392,-33.392,-33.389,-33.364,-33.317,-33.254,-33.214,-33.224,…,
2
-33.274,-33.329,-33.348,-33.316,-33.23,-33.164,-33.158,-33.231,-33.36,-33.471,…,
-33.479,-33.389,-33.28,-33.195,-33.153,-33.165,-33.216,-33.267,-33.277,-33.236,…,
-33.169,-33.116,-33.102,-33.124,-33.172,-33.215,-33.235,-33.224,-33.168,-33.099,…,
-33.063,-33.071,-33.124,-33.186,-33.234,-33.252,-33.243,-33.226,-33.208,-33.183,…,
3
-33.151,-33.112,-33.083,-33.07,-33.1,-33.142,-33.17,-33.161,-33.106,-33.048,-33.023,…,
-33.121,-33.184,-33.196,-33.161,-33.115,-33.093,-33.108,-33.15,-33.198,-33.237,-33.274,…,
-33.299,-33.335,-33.357,-33.361,-33.351,-33.325,-33.306,-33.318,-33.361,-33.427,-33.488,…,
-33.526,-33.545,-33.546,-33.532,-33.498,-33.444,-33.368,-33.291,-33.251,-33.264,-33.329,…,
```

Com o formato de arquivo bem definido, é possível carregá-los com segurança, simulando um monitoramento realizado anteriormente.

4.4 A interface gráfica

A interface gráfica de um sistema, também conhecida como IHM (Interface Humano Máquina) ou GUI(*Grafic User Interface*) é uma das partes mais importante do software, pois é através dela que o usuário interage com a aplicação e obtém os resultados que espera.

Nesse trabalho buscou-se desenvolver as funcionalidades desejadas em uma interface gráfica simples que facilitasse o uso. Melhorias em design gráfico, cores e usabilidade podem ser realizadas posteriormente, o fundamental foi que atendesse as necessidades detectadas.

Com os diagramas de casos de uso e de classes, é possível ter uma idéia bem fundamentada do funcionamento e utilidade do software, porém a interface gráfica é sempre o maior resultado como produto final para o usuário.

A partir da tela inicial, mostrada na Figura 4.5, pode-se identificar as configurações básicas a respeito de visualização do gráfico, como faixa de comprimento de onda, threshold, ainda é possível definir qual canal deseja-se visualizar o espectro através de uma caixa de seleção.

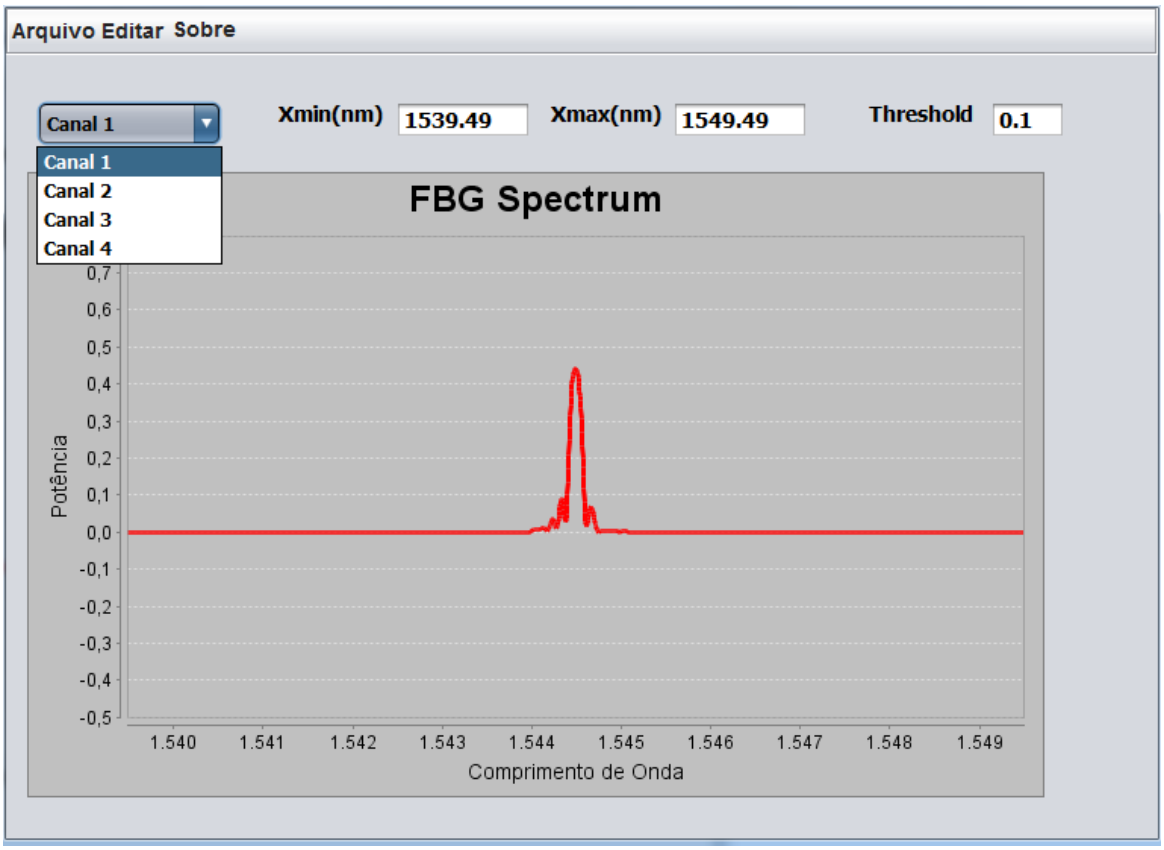


Figura 4.5: Tela inicial do software.

A interface gráfica possui 2 menus com funcionalidades, o menu Arquivo é mostrado na Figura 4.6. Ele possibilita o usuário carregar um arquivo e iniciar uma simulação, ou conectar-se ao interrogador para iniciar a aquisição de dados. Caso escolha por conectar terá que configurar o IP e as portas adequadas para isso (através da classe GUIConnect), uma janela pop-up simples faz essa função.

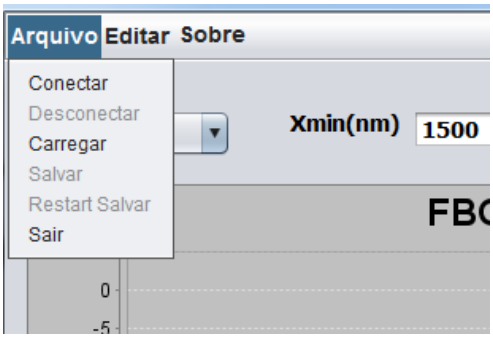


Figura 4.6 Opções do menu Arquivo.

Após conectar-se, o gráfico começa a ser atualizado com os dados provenientes da rede, e dependendo da escolha do canal. Pode-se realizar modificações visuais no gráfico, através do menu Editar, na Figura 4.7. Sendo possível escolher entre visualizar na escala logaritmica(dBm) ou linear (milliwatts).

Nesse menu também está implementado a busca por picos, baseada em threshold, que é automaticamente modificado no caso da mudança de escala. A opção Normal, faz com que o zoom do gráfico retorne para valores padrão(1500nm a 1600nm) o qual é possível visualizar todos os picos existentes. Ajustes dos mais variados podem ser feitos clicando-se com o botão direito do mouse sobre a área do gráfico. Estes são recursos providos pela biblioteca JfreeChart, inclusive salvar um imagem do frame mostrado no gráfico.

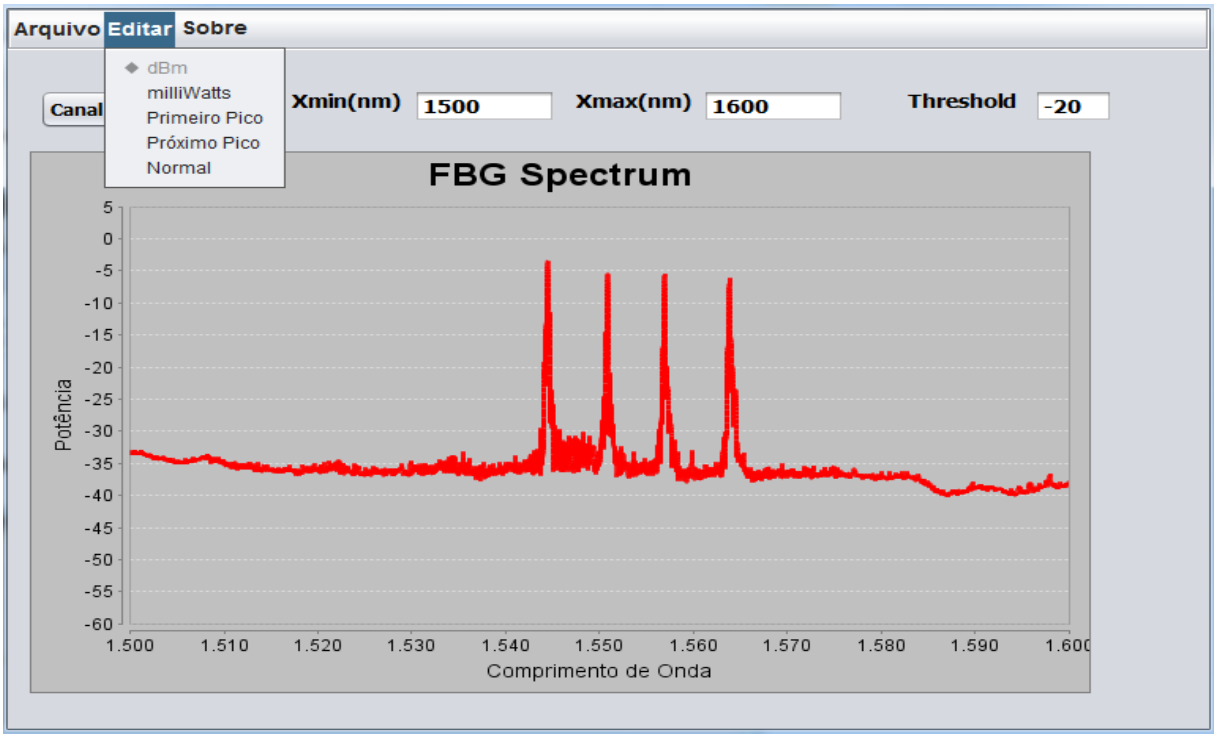


Figura 4.7- Menu Editar e dados em escala Logarítimica.

A qualquer momento, o usuário pode clicar em Desconectar no menu Arquivo, e interromper o recebimento de dados (disparando o evento para a classe ThreadDisconnect) , somente após isso é possível salvar um arquivo com os dados coletados. Caso o usuário clique em Restart Salvar no mesmo menu, o software desprezará os dados coletados anteriormente e começará a salvar a partir desse momento os dados visualizados. Escolhendo a opção Salvar ou Carregar, uma janela como a da Figura 4.8 se abrirá, permitindo a navegação e escolha do arquivo adequado.

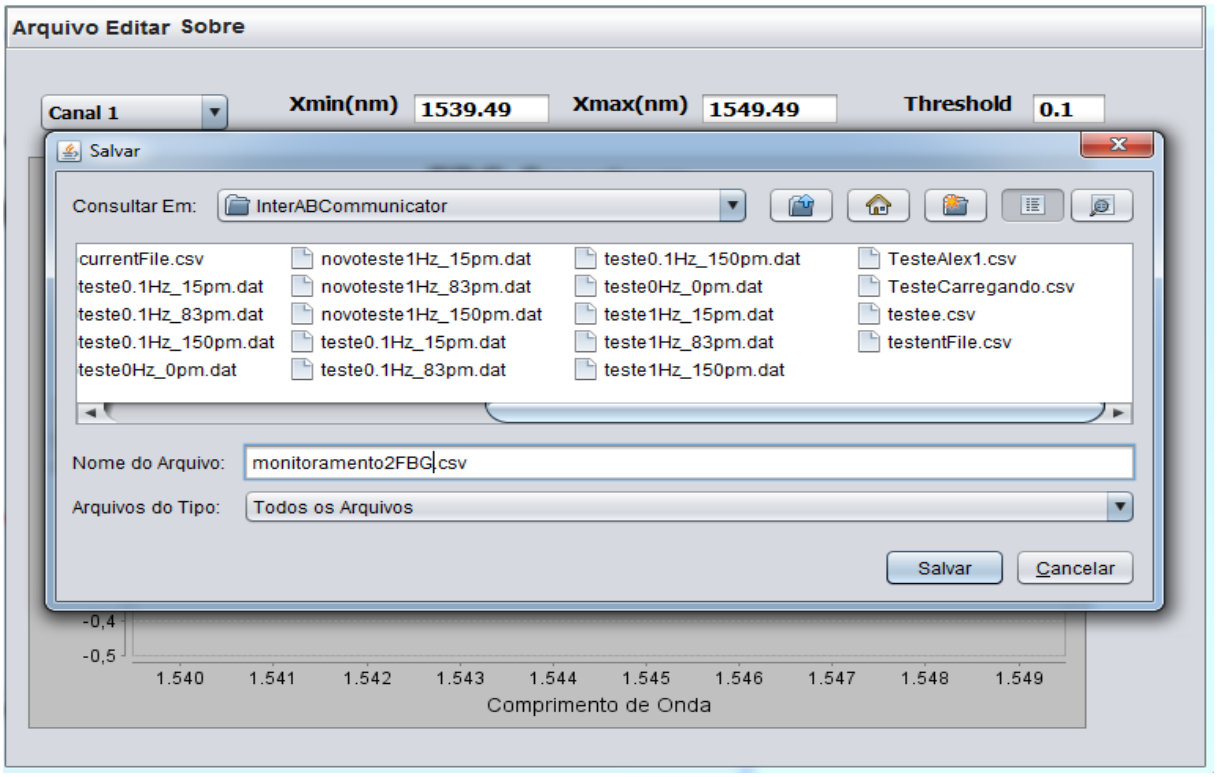


Figura 4.8- Opção Salvar Arquivo.

Por fim, clicando no item Sair do menu Arquivo, as conexões são verificadas e fechadas caso estejam abertas, e o arquivo temporário é apagado. Encerrando-se assim os fluxos existentes. Os novos arquivos gerados pelo software poderão ser usados por outros programas clássicos como o MatLab para uma análise de comportamentos mais detalhados de FBGs sob determinadas condições.

Capítulo 5

Considerações Finais

5.1 Resultados obtidos

Com o estudo da tecnologia de sensoriamento com FBGs e o desenvolvimento deste software, pode-se avançar bastante nesta área tão promissora, e que se espera crescer bastante devido as inúmeras vantagens citadas. A UFPA também vem avançando nessa área de sensoriamento e redes ópticas, espera-se que esse trabalho tenha contribuído para que novos trabalhos possam ser desenvolvidos nessa linha.

Por isso, era tão importante que pudéssemos ter um software capaz de fazer a aquisição dos dados brutos a partir do interrogador óptico. Permitindo testes e experimentos mais complexos, proporcionando assim um poderoso recurso para a pesquisa e desenvolvimento.

Também foi possível adquirir experiência em desenvolvimento com a linguagem Java, principalmente no que concerne a interfaces gráficas e comunicação via rede. A oportunidade de trabalhar e conhecer um equipamento como um interrogado róptico não seria possível sem a realização desse TCC.

5.2 Trabalhos futuros

Dentre os diversos trabalhos futuros, podemos mencionar o desenvolvimento de um simulador completo de um interrogador, através de modelos matemáticos para grades de Bragg. Possibilitando que grades de diferentes características sejam modeladas através de software. Tornando a pesquisa muito mais flexível e adaptável.

Neste software está implementado o cálculo de pico baseado em threshold, como explicado no capítulo 4, porém novos trabalhos podem ser feitos comparando diversos algoritmos de busca de picos para monitoramento real, como: o baseado

em Centróide; o ajuste gaussiano (*Gaussian Fit*); o ajuste quadrático (Q*uadratic Fit*) e o Operador de Fase Linear(*Linear Phase Operator-LPO*) [33] [28].

Também é possível implementar módulos para processamento dos dados e redução do ruído adicionado tanto pelo foto detector quanto por outros componentes. Como em [34], técnicas de processamento digital de sinais podem ser usadas para reduzir o ruído aditivo presente no sinal. Melhorando a precisão da demodulação para temperatura/deformação ou outra grandeza.

A nível de interface gráfica pode-se também aprimorar a animação de dados carregados, permitindo funcionalidades como pausar, adiantar ou atrasar a animação para o momento desejado. Melhorias no formato do arquivo salvo também são importantes, identificando um padrão que possa representar informações adicionais sobre o interrogador e tipos (modelos, número de canais, etc).

Assim, verifica-se que muitas outras aplicações podem ser desenvolvidas a partir do software fruto deste trabalho. Espera-se que ele possa contribuir para novas pesquisas como ferramenta fundamental e com o tempo novas funcionalidades sejam adicionadas a ele, tornando-o uma ferramenta útil para o monitoramento com sensores FBGs.

Referências Bibliográficas

[1] Fiber Grating Sensors, Alan D. Kersey, Michael A. Davis, Heather J. Patrick, Michel LeBlanc, K. P. Koo, JOURNAL OF LIGHTWAVE TECHNOLOGY, VOL. 15, NO. 8, AUGUST 1997.

[2] K. O. Hill, Y. Fujii, D. C. Johnson, and B. S. Kawasaki, “Photosensitivity in optical fiber waveguides: Application to reflection filter fabrication,” *Appl. Phys. Lett.,* vol. 32, p. 647, 1978.

[3] J. Stone, “Photorefractivity in GeO2-doped silica fibers,” *J. Appl. Phys.,* vol. 62, p. 4371, 1987.

[4] F. P. Payne, “Photorefractive gratings in single-mode optical fibers,” *Electron. Lett.,* vol. 25, p. 498, 1989.

[5] D. P. Hand and R. St. J. Russell, “Photoinduced refractive-index changes in germanosilicate fibers,” *Opt. Lett.,* vol. 15, p. 102, 1990.

[6] G. Meltz, W. W. Morey, and W. H. Glenn, “Formation of Bragg gratings in optical fibers by a transverse holographic method,” *Opt. Lett.,* vol. 14, p. 823, 1989.

[7] R. Kashyap, “Photosensitive optical fibers: Devices and applications,” *Optic. Fiber Technol.,* vol. 1, p. 17, 1994.

[8]Technical and Experimental Study of Fiber Bragg Grating Vibration Detection Based on Matching Demodulation Method, Proceedings of the IEEE International Conference on Automation and Logistics, 2007

[9] Relatório de investimentos para o PAC-2, Governo Federal do Brasil, 2010, acessado pela última vez em 18/12/2011. http://www.brasil.gov.br/pac/pac-2/pac-2-relatorio-1

[10]Fiber Optic Sensors, Second Edition, Shizhuo Yin, Paul B. Ruffin, Francis T. S. Yu 2008.

[11] Optical Fiber Sensor Technology: Advanced Applications - Bragg Gratings and Distributed Sensors, Springer; 1 edition, 2000.

[12]Real-Time Monitoring of the security for the Large Construction with wireless FBG Sensor Network, Yuanzhong Zhang, E -Business and E -Government (ICEE), 2011 International Conference.

[13]Modelo Teórico de Sensores Ópticos Baseados em Fibras com Grade de Bragg, José Renato Ferreira Alves da Cunha, Dissertação de Mestrado, 2007*, Programa de Pós Graduação em Física, UFPA.*

[14] Fu J. W., Xiao L. Z., Zhang Y. Z., “Progress of permanent fiber-optic sensor applications to oil and gas well”, Progress in Geophysics, 2004, v.19,n.3, pp 828-836(in Chinese)

[15] Zhang Y. Z., Xiao L. Z., Fu J. W., “The perspective of the permanent monitoring with an FBG sensor network in oil and gas production in China”, Proc. of SPIE, 2005, Vol.6041, pp186-191

[16]Zhang Yuan Zhong; Xiao Li Zhi; Wang Jie Yi; , "Oil Well Real-time Monitoring With Downhole Permanent FBG Sensor Network," *Control and Automation, 2007. ICCA 2007. IEEE International Conference on* , vol., no., pp.2591-2594, May 30 2007-June 1 2007 doi: 10.1109/ICCA.2007.4376830
URL: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=4376830&isnumber=4376307

[17] Taylor Bridge - A Bridge For the 21 st Century, Emile Shehatal, Rick Haldane-Wilsone1 , Doug Stewart, Gamil Tadros and Sami Rizkalla, *WORLD WISE'99-Winnipeg, Manitoba* http://www.ce.ncsu.edu/srizkal/linked_files/TaylorBridge-ABridgeForThe21stCentury_Emile_Dec99.pdf acessado em 08/10/2011

[18]S. Kannan, J. Z. Y. Guo, and P. J. Lemaire, Thermal stability analysis of UVinduced fiber Bragg gratings, *J. Lightwave Tech*., **15**, pp. 1478–1483, 1997.

[19]Ribeiro, A. B. L, Esquemas de multiplexagem de sensores de fibra óptica, departamento de física da faculdade de ciência da Universidade do Porto, tese de doutorado, outubro de 1996.

[20] Marco Sousa, Prospecção de tecnologias para interrogação de redes de sensores ópticos com aplicação em redes de transmissão de energia elétrica. Documento interno do Projeto TECCON

[21] Yunmiao Wang; Jianmin Gong; Wang, D.Y.; Bo Dong; Weihong Bi; Anbo Wang; , "A Quasi-Distributed Sensing Network With Time-Division-Multiplexed Fiber Bragg Gratings," *Photonics Technology Letters, IEEE* , vol.23, no.2, pp.70-72, Jan.15, 2011 doi: 10.1109/LPT.2010.2089676
URL: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=5610707&isnumber=5677493

[22] Govind P. Agrawal Fiber-Optic Communication Systems, 3rd Edition, 2002

[23] Y.-J. Rao, Fiber Bragg grating sensors: Principles and applications, in *Optical Fiber Sensor Technology,* Vol. 2, K. T. V. Grattan and B. T. Meggitt, eds., pp. 355–389, Chapman & Hall, London, 1998.

[24] Ma Guo-ming; Li Cheng-rong; Meng Chen-ping; Quan Jiang-tao; Cheng Yang-chun; Jiang Jian; Tian Xiao; , "Ice Monitoring on Overhead Transmission Lines with FBG Tension Sensor," *Power and Energy Engineering Conference (APPEEC), 2010 Asia-Pacific* , vol., no., pp.1-4, 28-31 March 2010 doi: 10.1109/APPEEC.2010.5448665
URL: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=5448665&isnumber=5448125

[25]Ma, G.; Li, C.; Quan, J.; Jiang, J.; Cheng, Y.; , "A Fiber Bragg Grating Tension and Tilt Sensor Applied to Icing Monitoring on Overhead Transmission Lines," *Power*

Delivery, IEEE Transactions on , vol.26, no.4, pp.2163-2170, Oct. 2011
doi: 10.1109/TPWRD.2011.2157947
URL: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=5942188&isnumber=6029804

[26] Yongqian Li; Yang Xie; Guozhen Yao; , "Comparison of Peak Searching Algorithms for Wavelength Demodulation in Fiber Bragg Grating Sensors," *Information Engineering and Computer Science (ICIECS), 2010 2nd International Conference on* , vol., no., pp.1-4, 25-26 Dec. 2010
doi: 10.1109/ICIECS.2010.5678281
URL: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=5678281&isnumber=5677633

[27] Fundamentos de detecção Óptica (FBG) no site da National Instruments, http://zone.ni.com/devzone/cda/tut/p/id/13466, acessado em 02/11/2011.

[28] Bodendorfer, T.; Muller, M.S.; Hirth, F.; Koch, A.W.; , "Comparison of different peak detection algorithms with regards to spectrometic fiber Bragg grating interrogation systems," *Optomechatronic Technologies, 2009. ISOT 2009. International Symposium on* , vol., no., pp.122-126, 21-23 Sept. 2009
doi: 10.1109/ISOT.2009.5326110
URL: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=5326110&isnumber=5326038

[29]. K. Usbeck, W. Ecke, V. Hagemann, R. Mueller, and R. Willsch, Temperature referenced fibre Bragg grating refractometer sensor for on-line quality control of petrol products, *Proc. 13th Int. Conf. Optical Fiber Sensors (OFS-13)*, Kyongju, Korea, *SPIE*, **3746**, pp. 163–166, 1999.

[30]. D. J. Webb, M. W. Hathaway, D. A. Jackson, S. Jones, L. Zhang, and I. Bennion, First in-vivo trials of a fiber Bragg grating based temperature profiling system, *J. Biomedical Optics,* **5**, 1, pp. 45–50, 2000.

[31] FS2100/2200 Rack-Mountable BraggMeter User Manual, FiberSensing, Novembro,2011.

[32] iLog Software User Manual, SW Version: 2010.11.08 – V3.2.6 User Manual Version: v1.8 .

[33] T. Zeh, "Optical Fiber Bragg Sensors - Measurement Systems and Signal Processing," Ph.D. dissertation, Technical University of Munich, 2005.

[34]Hu Wen-xia; Wan Sheng-peng; Zhang Hui-lian; , "Software Design of FBG Temperature Demodulation System Based on Wavelet Denoising," *Photonics and Optoelectronics (SOPO), 2011 Symposium on* , vol., no., pp.1-4, 16-18 May 2011 doi: 10.1109/SOPO.2011.5780384
URL: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=5780384&isnumber=5780376

[35] Larry Peterson, Bruce S. Davie Computer Networks: A Systems Approach, Fourth Edition, 2007.

[36]Gerd Keiser, Optical Fiber Communications, McGraw-Hill Science/Engineering/Math; 3rd edition (October 1, 1999)

[34]Deitel, H. M.; Deitel, P. J.; Java : Como programar, 6a. edição, 2005.

[38] Manual de refência do Hibernate 3.5 em Português
http://docs.jboss.org/hibernate/core/3.5/reference/pt-BR/pdf/hibernate_reference.pdf

[36] Biblioteca da JfreeChart , http://www.jfree.org/jfreechart/ acessado pela última vez em 17/12/2011

[40] Tutorial da Oracle em inglês sobre Threads, http://docs.oracle.com/javase/tutorial/essential/concurrency/procthread.html, acessado pela última vez em 17/12/2011

[41] API Java da Classe Timer, http://docs.oracle.com/javase/1.5.0/docs/api/java/util/Timer.html acessado pela última vez em 17/12/2011

[42]Tutorial da Oracle sobre o SwingWorker, http://docs.oracle.com/javase/tutorial/uiswing/concurrency/worker.html acessado pela última vez em 17/12/2011

[43] Nationals Instruments WebSite, http://brasil.ni.com/, acessado em 22/12/2011

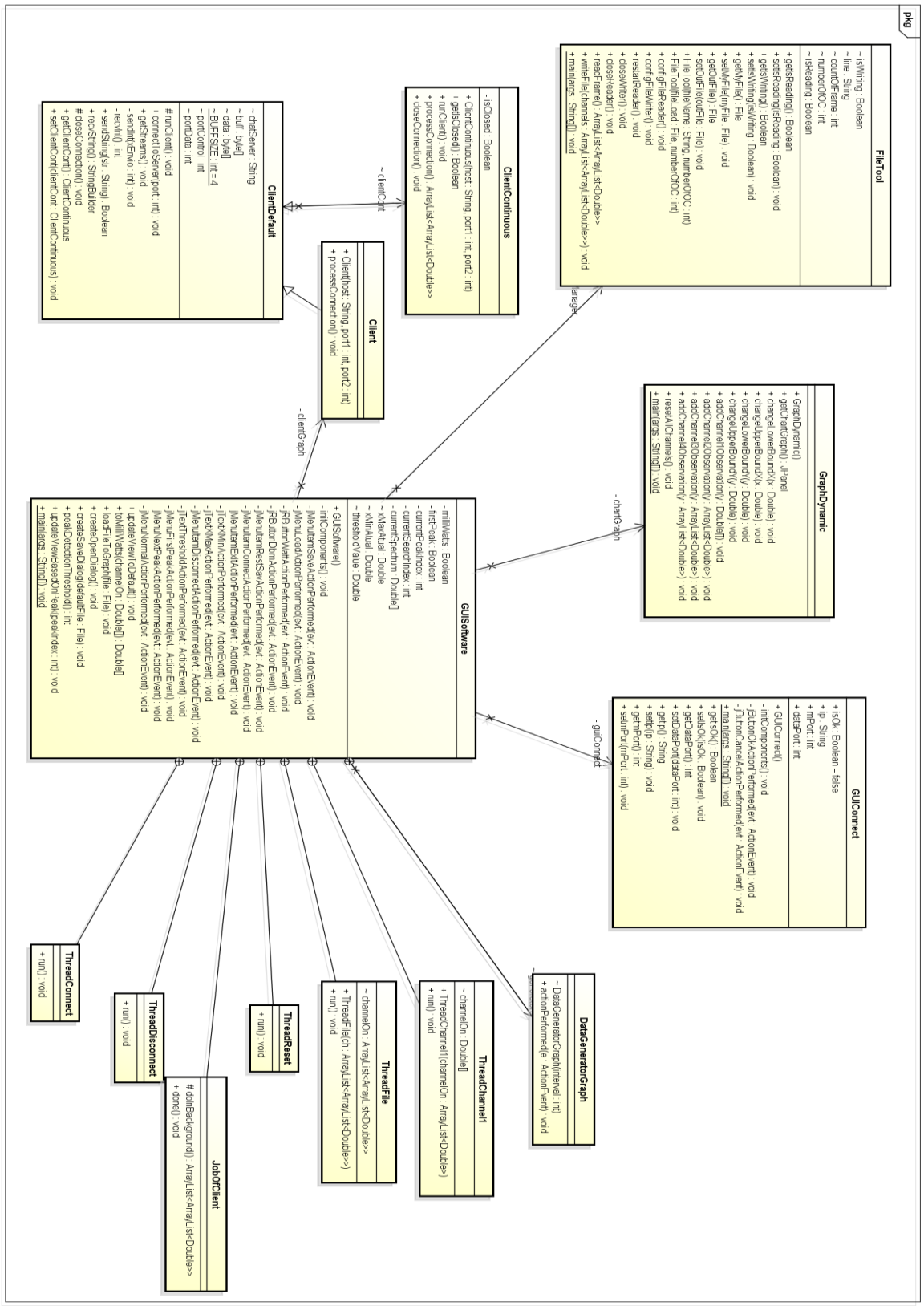
[44] Datasheet do Interrogador SM225 Rach Mountable da Micron Optics,http://www.micronoptics.com/uploads/library/documents/Datasheets/Micron%20Optics%20sm225.pdf , acessado em 22/12/2011

[45] Marco José de Souza, “Síntese de grades de Bragg em fibra: técnicas de aceleração e codificação para algoritmos evolucionários”, Tese de Doutorado, Universidade Federal do Pará, 2008.

[46] Müller *et al.* “Fiber-Optic Sensor Interrogation Based on a Widely Tunable Monolithic Laser Diode”, *IEEE transactions on instrumentation and measurement*, vol. 59, No. 3, March 2010.

Apêndice A

A.1: Diagrama de Classes:



Apêndice B – Código Fonte

```
1  /*
2   * GUISoftware.java
3   *
4   * Created on 03/12/2011, 10:28:11
5   */
6  package testing;
7
8  import interabcommunicator.FileTool;
9  import java.awt.event.ActionEvent;
10 import java.awt.event.ActionListener;
11 import java.io.File;
12 import java.io.FileInputStream;
13 import java.io.FileNotFoundException;
14 import java.io.FileOutputStream;
15 import java.io.IOException;
16 import java.nio.channels.FileChannel;
17 import java.util.ArrayList;
18 import java.util.concurrent.ExecutionException;
19 import java.util.logging.Level;
20 import java.util.logging.Logger;
21 import javax.swing.JFileChooser;
22 import javax.swing.JOptionPane;
23 import javax.swing.SwingWorker;
24 import javax.swing.Timer;
25
26 /**
27  *
28  * @author Alex Santos
29  */
30 public class GUISoftware extends javax.swing.JFrame {
31
32     /** Creates new form GUISoftware */
33     public GUISoftware() {
34         initComponents();
35     }
36
37     /** This method is called from within the constructor to
38      * initialize the form.
39      * WARNING: Do NOT modify this code. The content of this method is
40      * always regenerated by the Form Editor.
41      */
42     @SuppressWarnings("unchecked")
43     // <editor-fold defaultstate="collapsed" desc="Generated Code">
44     private void initComponents() {
45
46         jPanel1 = new javax.swing.JPanel();
47         jTextXMax = new javax.swing.JTextField();
48         jTextXMin = new javax.swing.JTextField();
49         jLabel2 = new javax.swing.JLabel();
50         jComboBox1 = new javax.swing.JComboBox();
51         jLabel1 = new javax.swing.JLabel();
52         jLabel3 = new javax.swing.JLabel();
53         jTextThreshold = new javax.swing.JTextField();
54         jMenuBar1 = new javax.swing.JMenuBar();
55         jMenuFile = new javax.swing.JMenu();
56         jMenuItemConnect = new javax.swing.JMenuItem();
```

```
57        jMenuItemDisconnect = new javax.swing.JMenuItem();
58        jMenuLoad = new javax.swing.JMenuItem();
59        jMenuItemSave = new javax.swing.JMenuItem();
60        jMenuItemRestSav = new javax.swing.JMenuItem();
61        jMenuItemExit = new javax.swing.JMenuItem();
62        jMenuEdit = new javax.swing.JMenu();
63        jRButtonDbm = new javax.swing.JRadioButtonMenuItem();
64        jRButtonWatt = new javax.swing.JRadioButtonMenuItem();
65        jMenuFirstPeak = new javax.swing.JMenuItem();
66        jMenuNextPeak = new javax.swing.JMenuItem();
67        jMenuNormal = new javax.swing.JMenuItem();
68        jMenuAbout = new javax.swing.JMenu();
69
70        setDefaultCloseOperation(javax.swing.WindowConstants.DISPOSE_ON_CLOSE);
71        setResizable(false);
72
73        jPanel1.setName("jPanel1"); // NOI18N
74        jPanel1.setLayout(new org.netbeans.lib.awtextra.AbsoluteLayout());
75
76        jTextXMax.setFont(new java.awt.Font("Tahoma", 1, 14));
77        jTextXMax.setText("1600");
78        jTextXMax.setName("jTextXMax"); // NOI18N
79        jTextXMax.setPreferredSize(new java.awt.Dimension(59, 25));
80        jTextXMax.addActionListener(new java.awt.event.ActionListener() {
81            public void actionPerformed(java.awt.event.ActionEvent evt) {
82                jTextXMaxActionPerformed(evt);
83            }
84        });
85        jPanel1.add(jTextXMax, new org.netbeans.lib.awtextra.AbsoluteConstraints(440, 30, 90, -1));
86
87        jTextXMin.setFont(new java.awt.Font("Tahoma", 1, 14));
88        jTextXMin.setText("1500");
89        jTextXMin.setName("jTextXMin"); // NOI18N
90        jTextXMin.setPreferredSize(new java.awt.Dimension(59, 25));
91        jTextXMin.addActionListener(new java.awt.event.ActionListener() {
92            public void actionPerformed(java.awt.event.ActionEvent evt) {
93                jTextXMinActionPerformed(evt);
94            }
95        });
96        jPanel1.add(jTextXMin, new org.netbeans.lib.awtextra.AbsoluteConstraints(257, 30, 90, -1));
97
98        jLabel2.setFont(new java.awt.Font("Tahoma", 1, 14));
99        jLabel2.setText("Xmax(nm)");
100        jLabel2.setName("jLabel2"); // NOI18N
101        jPanel1.add(jLabel2, new org.netbeans.lib.awtextra.AbsoluteConstraints(360, 30, -1, -1));
102
103        jComboBox1.setFont(new java.awt.Font("Tahoma", 1, 12));
104        jComboBox1.setModel(new javax.swing.DefaultComboBoxModel(new String[] { "Canal 1", "Canal
2", "Canal 3", "Canal 4" }));
105        jComboBox1.setName("jComboBox1"); // NOI18N
106        jComboBox1.setPreferredSize(new java.awt.Dimension(56, 30));
107        jPanel1.add(jComboBox1, new org.netbeans.lib.awtextra.AbsoluteConstraints(20, 30, 123, -1));
108
109        jLabel1.setFont(new java.awt.Font("Tahoma", 1, 14));
110        jLabel1.setText("Xmin(nm)");
111        jLabel1.setName("jLabel1"); // NOI18N
112        jPanel1.add(jLabel1, new org.netbeans.lib.awtextra.AbsoluteConstraints(180, 30, -1, -1));
113
114        jLabel3.setFont(new java.awt.Font("Tahoma", 1, 14));
115        jLabel3.setText("Threshold");
```

```
116         jLabel3.setName("jLabel3"); // NOI18N
117         jPanel1.add(jLabel3, new org.netbeans.lib.awtextra.AbsoluteConstraints(570, 30, -1, -1));
118
119         jTextThreshold.setFont(new java.awt.Font("Tahoma", 1, 14));
120         jTextThreshold.setText("-20");
121         jTextThreshold.setName("jTextThreshold"); // NOI18N
122         jTextThreshold.setPreferredSize(new java.awt.Dimension(59, 25));
123         jTextThreshold.addActionListener(new java.awt.event.ActionListener() {
124             public void actionPerformed(java.awt.event.ActionEvent evt) {
125                 jTextThresholdActionPerformed(evt);
126             }
127         });
128         jPanel1.add(jTextThreshold, new org.netbeans.lib.awtextra.AbsoluteConstraints(650, 30, 50, -1));
129
130         jMenuBar1.setBorder(javax.swing.BorderFactory.createEtchedBorder());
131         jMenuBar1.setName("jMenuBar1"); // NOI18N
132         jMenuBar1.setPreferredSize(new java.awt.Dimension(90, 31));
133
134         jMenuFile.setBorder(javax.swing.BorderFactory.createEtchedBorder());
135         jMenuFile.setText("Arquivo");
136         jMenuFile.setFont(new java.awt.Font("Arial", 1, 14));
137         jMenuFile.setName("jMenuFile"); // NOI18N
138
139         jMenuItemConnect.setText("Conectar");
140         jMenuItemConnect.setName("jMenuItemConnect"); // NOI18N
141         jMenuItemConnect.addActionListener(new java.awt.event.ActionListener() {
142             public void actionPerformed(java.awt.event.ActionEvent evt) {
143                 jMenuItemConnectActionPerformed(evt);
144             }
145         });
146         jMenuFile.add(jMenuItemConnect);
147
148         jMenuItemDisconnect.setText("Desconectar");
149         jMenuItemDisconnect.setEnabled(false);
150         jMenuItemDisconnect.setName("jMenuItemDisconnect"); // NOI18N
151         jMenuItemDisconnect.addActionListener(new java.awt.event.ActionListener() {
152             public void actionPerformed(java.awt.event.ActionEvent evt) {
153                 jMenuItemDisconnectActionPerformed(evt);
154             }
155         });
156         jMenuFile.add(jMenuItemDisconnect);
157
158         jMenuLoad.setText("Carregar");
159         jMenuLoad.setName("jMenuLoad"); // NOI18N
160         jMenuLoad.addActionListener(new java.awt.event.ActionListener() {
161             public void actionPerformed(java.awt.event.ActionEvent evt) {
162                 jMenuLoadActionPerformed(evt);
163             }
164         });
165         jMenuFile.add(jMenuLoad);
166
167         jMenuItemSave.setText("Salvar");
168         jMenuItemSave.setEnabled(false);
169         jMenuItemSave.setName("jMenuItemSave"); // NOI18N
170         jMenuItemSave.addActionListener(new java.awt.event.ActionListener() {
171             public void actionPerformed(java.awt.event.ActionEvent evt) {
172                 jMenuItemSaveActionPerformed(evt);
173             }
174         });
175         jMenuFile.add(jMenuItemSave);
```

```
176
177         jMenuItemRestSav.setText("Restart Salvar");
178         jMenuItemRestSav.setEnabled(false);
179         jMenuItemRestSav.setName("jMenuItemRestSav"); // NOI18N
180         jMenuItemRestSav.addActionListener(new java.awt.event.ActionListener() {
181             public void actionPerformed(java.awt.event.ActionEvent evt) {
182                 jMenuItemRestSavActionPerformed(evt);
183             }
184         });
185         jMenuFile.add(jMenuItemRestSav);
186
187         jMenuItemExit.setText("Sair");
188         jMenuItemExit.setName("jMenuItemExit"); // NOI18N
189         jMenuItemExit.addActionListener(new java.awt.event.ActionListener() {
190             public void actionPerformed(java.awt.event.ActionEvent evt) {
191                 jMenuItemExitActionPerformed(evt);
192             }
193         });
194         jMenuFile.add(jMenuItemExit);
195
196         jMenuBar1.add(jMenuFile);
197
198         jMenuEdit.setBorder(javax.swing.BorderFactory.createEtchedBorder());
199         jMenuEdit.setText("Editar");
200         jMenuEdit.setFont(new java.awt.Font("Arial", 1, 14)); // NOI18N
201         jMenuEdit.setName("jMenuEdit"); // NOI18N
202
203         jRButtonDbm.setSelected(true);
204         jRButtonDbm.setText("dBm");
205         jRButtonDbm.setName("jRButtonDbm"); // NOI18N
206         jRButtonDbm.addActionListener(new java.awt.event.ActionListener() {
207             public void actionPerformed(java.awt.event.ActionEvent evt) {
208                 jRButtonDbmActionPerformed(evt);
209             }
210         });
211         jMenuEdit.add(jRButtonDbm);
212
213         jRButtonWatt.setText("milliWatts");
214         jRButtonWatt.setName("jRButtonWatt"); // NOI18N
215         jRButtonWatt.addActionListener(new java.awt.event.ActionListener() {
216             public void actionPerformed(java.awt.event.ActionEvent evt) {
217                 jRButtonWattActionPerformed(evt);
218             }
219         });
220         jMenuEdit.add(jRButtonWatt);
221
222         jMenuFirstPeak.setText("Primeiro Pico");
223         jMenuFirstPeak.setEnabled(false);
224         jMenuFirstPeak.setName("jMenuFirstPeak"); // NOI18N
225         jMenuFirstPeak.addActionListener(new java.awt.event.ActionListener() {
226             public void actionPerformed(java.awt.event.ActionEvent evt) {
227                 jMenuFirstPeakActionPerformed(evt);
228             }
229         });
230         jMenuEdit.add(jMenuFirstPeak);
231
232         jMenuNextPeak.setText("Próximo Pico");
233         jMenuNextPeak.setEnabled(false);
234         jMenuNextPeak.setName("jMenuNextPeak"); // NOI18N
235         jMenuNextPeak.addActionListener(new java.awt.event.ActionListener() {
```

```
236             public void actionPerformed(java.awt.event.ActionEvent evt) {
237                 jMenuNextPeakActionPerformed(evt);
238             }
239         });
240         jMenuEdit.add(jMenuNextPeak);
241 
242         jMenuNormal.setText("Normal");
243         jMenuNormal.setEnabled(false);
244         jMenuNormal.setName("jMenuNormal"); // NOI18N
245         jMenuNormal.addActionListener(new java.awt.event.ActionListener() {
246             public void actionPerformed(java.awt.event.ActionEvent evt) {
247                 jMenuNormalActionPerformed(evt);
248             }
249         });
250         jMenuEdit.add(jMenuNormal);
251 
252         jMenuBar1.add(jMenuEdit);
253 
254         jMenuAbout.setText("Sobre");
255         jMenuAbout.setFont(new java.awt.Font("Arial", 1, 14)); // NOI18N
256         jMenuAbout.setName("jMenuAbout"); // NOI18N
257         jMenuBar1.add(jMenuAbout);
258 
259         setJMenuBar(jMenuBar1);
260 
261         javax.swing.GroupLayout layout = new javax.swing.GroupLayout(getContentPane());
262         getContentPane().setLayout(layout);
263         layout.setHorizontalGroup(
264             layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
265             .addGroup(layout.createSequentialGroup()
266                 .addComponent(jPanel1, javax.swing.GroupLayout.PREFERRED_SIZE, 749,
javax.swing.GroupLayout.PREFERRED_SIZE)
267                 .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))
268         );
269         layout.setVerticalGroup(
270             layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
271             .addGroup(layout.createSequentialGroup()
272                 .addComponent(jPanel1, javax.swing.GroupLayout.DEFAULT_SIZE, 495, Short.MAX_VALUE)
273                 .addContainerGap())
274         );
275 
276         chartGraph =new GraphDynamic();
277         //chartGraph.new DataGenerator(100).start();
278 
279         chartGraph.getChartGraph().setName("jPanelGraph");
280 
281         javax.swing.GroupLayout jPanelGraphLayout = new
javax.swing.GroupLayout(chartGraph.getChartGraph());
282         chartGraph.getChartGraph().setLayout(jPanelGraphLayout);
283         jPanelGraphLayout.setHorizontalGroup(
284             jPanelGraphLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
285             .addGap(0, 550, Short.MAX_VALUE)
286         );
287         jPanelGraphLayout.setVerticalGroup(
288             jPanelGraphLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
289             .addGap(0,270, Short.MAX_VALUE)
290         );
291 
292         jPanel1.add(chartGraph.getChartGraph(), new org.netbeans.lib.awtextra.AbsoluteConstraints(10, 73, -
1, -1));
```

```
293
294         jRButtonDbm.setSelected(true);
295         jRButtonDbm.setEnabled(false);
296         jRButtonWatt.setSelected(false);
297         jRButtonWatt.setEnabled(true);
298
299         xMinAtual=Double.parseDouble(jTextXMin.getText());
300         xMaxAtual=Double.parseDouble(jTextXMax.getText());
301         thresholdValue=Double.parseDouble(jTextThreshold.getText());
302         currentPeakIndex=0;
303         currentSearchIndex=0;
304         milliWatts=false;
305
306         java.awt.Dimension screenSize = java.awt.Toolkit.getDefaultToolkit().getScreenSize();
307         setBounds((screenSize.width-767)/2, (screenSize.height-575)/2, 767, 575);
308     }// </editor-fold>
309
310 private void jMenuItemSaveActionPerformed(java.awt.event.ActionEvent evt) {
311 // TODO add your handling code here:
312         createSaveDialog(fileManager.getOutFile());//Arquivo já existente
313
314 }
315
316 private void jMenuLoadActionPerformed(java.awt.event.ActionEvent evt) {
317 // TODO add your handling code here:
318         createOpenDialog();
319 }
320
321 private void jRButtonWattActionPerformed(java.awt.event.ActionEvent evt) {
322 // TODO add your handling code here:
323             jRButtonWatt.setSelected(true);
324             jRButtonWatt.setEnabled(false);
325             jRButtonDbm.setSelected(false);
326             jRButtonDbm.setEnabled(true);
327             //Mudar Escala do Gráfico para watt
328             milliWatts=true;
329             chartGraph.changeLowerBoundY(-0.5);
330             chartGraph.changeUpperBoundY(0.8);
331             thresholdValue=0.1;
332             this.jTextThreshold.setText(thresholdValue.toString());
333 }
334
335 private void jRButtonDbmActionPerformed(java.awt.event.ActionEvent evt) {
336 // TODO add your handling code here:
337             jRButtonDbm.setSelected(true);
338             jRButtonDbm.setEnabled(false);
339             jRButtonWatt.setSelected(false);
340             jRButtonWatt.setEnabled(true);
341             milliWatts=false;
342            //Mudar Escala do Gráfico para dBM
343             chartGraph.changeLowerBoundY(-55.0);
344             chartGraph.changeUpperBoundY(10.0);
345             thresholdValue=-20.0;
346             this.jTextThreshold.setText(thresholdValue.toString());
347 }
348
349 private void jMenuItemRestSavActionPerformed(java.awt.event.ActionEvent evt) {
350 // TODO add your handling code here:
351 //Restarta o Arquivo temporário que está sendo criado
352 //cria novamente o arquivo sobreescrevendo-o
```

```
353             ThreadReset tr1= new ThreadReset();
354             Thread ar1= new Thread(tr1);
355             ar1.start();
356 }
357
358 public class ThreadReset implements Runnable{//Thread para processar o ArrayGrande do Channel1
359         @Override
360         public void run() {
361             generatorD.stop();
362             while (fileManager.getIsWriting()) {
363                 double a = System.currentTimeMillis();
364                 double b, c = 0;
365                 while (c < 50) {
366                     b = System.currentTimeMillis();
367                     c = b - a;
368                 }
369             }
370             fileManager =new FileTool("currentFile.csv",4);
371             fileManager.configFileWriter();
372             generatorD.start();
373         }
374 }
375
376 private void jMenuItemConnectActionPerformed(java.awt.event.ActionEvent evt) {
377 // TODO add your handling code here:
378           //Configura o Client para acessar o servido e portas
379             ThreadConnect t1= new ThreadConnect();
380             Thread a1= new Thread(t1);
381             a1.start();
382 }
383
384 public class ThreadConnect implements Runnable{//Thread para processar o ArrayGrande do Channel1
385         @Override
386         public void run() {
387             guiConnect= new GUIConnect();
388             guiConnect.setVisible(true);
389          // System.out.print("Cheguei no data generator1");
390             while(true){
391             // System.out.println("Dentro do while");
392               if(guiConnect.getIsOk() && (!guiConnect.isVisible())){
393                 // System.out.print("Cheguei no data generator1");
394                     clientGraph= new Client(guiConnect.getIp(),
guiConnect.getmPort(),guiConnect.getDataPort());
395                     clientGraph.processConnection();//Inicio o Cliente
396                     jMenuItemConnect.setEnabled(false);
397                      jMenuItemDisconnect.setEnabled(true);
398                      jMenuItemSave.setEnabled(false);
399                      jMenuLoad.setEnabled(false);
400                      jMenuNextPeak.setEnabled(true);
401                      jMenuFirstPeak.setEnabled(true);
402                      jMenuNormal.setEnabled(true);
403                      jMenuItemRestSav.setEnabled(true);
404         /*########### CONFIGURA O ARQUIVO*/
405                      fileManager =new FileTool("currentFile.csv",4);
406                      fileManager.configFileWriter();
407         /*##########Faz a coleta de dados a cada 1 segundo*/
408                       generatorD=new DataGeneratorGraph(1000);
409                       generatorD.start();
410                      break;
411             }
```

```
412             else if(!guiConnect.getIsOk() && (!guiConnect.isVisible()) ){
413                 //fecha a guiConnect com o cancel
414                 break;
415             }
416           }//fecha while
417         }
418 }
419
420 class DataGeneratorGraph extends Timer implements ActionListener{
421 /**
422 * Constructor.
423 *@param interval the interval (in milliseconds)
424 */
425   DataGeneratorGraph(int interval) {
426         super(interval, null);
427         addActionListener(this);
428       }
429       @Override
430       public void actionPerformed(ActionEvent e) {
431         //Configurar para adicionar o dado de cada canal
432          JobOfClient job = new JobOfClient();
433          job.execute();
434       }
435 }
436
437 public class JobOfClient extends SwingWorker<ArrayList<ArrayList<Double>>, String>{
438       @Override
439       protected ArrayList<ArrayList<Double>> doInBackground() throws Exception {
440       //primeira funçao a ser processada quando chamado o método execute()
441         ArrayList<ArrayList<Double>> channelOn=new ArrayList<ArrayList<Double>>();
442         if(fileManager.getIsReading()){
443           channelOn= fileManager.readFrame();
444           if(channelOn == null){
445             generatorD.stop();
446             jMenuItemDisconnect.setEnabled(false);
447             jMenuItemConnect.setEnabled(true);
448             jMenuItemSave.setEnabled(true);
449             jMenuLoad.setEnabled(true);
450             jMenuNextPeak.setEnabled(false);
451             jMenuFirstPeak.setEnabled(false);
452             jMenuNormal.setEnabled(false);
453             jMenuItemRestSav.setEnabled(false);
454           }
455         }
456         else{
457           channelOn=clientGraph.getClientCont().processConnection();
458         }
459         return channelOn;
460       }
461
462       @Override
463       public void done() {
464         ArrayList<ArrayList<Double>> channelOn=new ArrayList<ArrayList<Double>>();
465         try {
466           channelOn=get();
467         } catch (InterruptedException ex) {
468           Logger.getLogger(GUISoftware.class.getName()).log(Level.SEVERE, null, ex);
469         } catch (ExecutionException ex) {
470           Logger.getLogger(GUISoftware.class.getName()).log(Level.SEVERE, null, ex);
471         }
```

```
472             if(channelOn==null){
473             //Interface com o fim de arquivo
474                 JOptionPane.showMessageDialog(null,"Fim do arquivo de simulação",
475                     "Fim", JOptionPane.INFORMATION_MESSAGE);
476                 fileManager.setIsReading(false);
477                 return;
478             }
479
480             /*####SALVA O FRAME EM ARQUIVO*/
481             //Casa não esteja lendo, entra nesse IF
482             if(!fileManager.getIsReading()){
483                 ThreadFile tf1= new ThreadFile(channelOn);
484                 Thread af1= new Thread(tf1);
485                 af1.start();
486             }
487             /*############################*/
488             //chartGraph.resetAllChannels();
489            if(jComboBox1.getSelectedIndex()==0){
490              ThreadChannel t1= new ThreadChannel(channelOn.get(0));
491              Thread a1= new Thread(t1);
492              a1.start();
493               System.out.println("Canal 1 OK");
494             }
495             if(jComboBox1.getSelectedIndex()==1){
496              ThreadChannel t2= new ThreadChannel(channelOn.get(1));
497             Thread a2= new Thread(t2);
498             a2.start();
499             System.out.println("Canal 2 OK ");
500             }
501             if(jComboBox1.getSelectedIndex()==2){
502              ThreadChannel t3= new ThreadChannel(channelOn.get(2));
503             Thread a3= new Thread(t3);
504             a3.start();
505              System.out.println("Canal 3 OK");
506             }
507             if(jComboBox1.getSelectedIndex()==3){
508              ThreadChannel t4= new ThreadChannel(channelOn.get(3));
509             Thread a4= new Thread(t4);
510             a4.start();
511              System.out.println("Canal 4 OK");
512             }
513
514         }
515 }
516
517 public class ThreadChannel implements Runnable{//Thread para processar o ArrayGrande do Channel1
518
519     Double [] channelOn;
520     public ThreadChannel(ArrayList<Double> channelOn){
521     this.channelOn= new Double[channelOn.size()];
522     channelOn.toArray(this.channelOn);
523     }
524         @Override
525         public void run() {
526             currentSpectrum=this.channelOn;
527             if(milliWatts){
528             this.channelOn= toMilliWatts(this.channelOn);
529             }
530            chartGraph.addChannelObservation(this.channelOn);
531        }
```

```
532 }
533
534 public class ThreadFile implements Runnable{
535
536     ArrayList<ArrayList<Double>> channelOn;
537     public ThreadFile (ArrayList<ArrayList<Double>> ch){
538         this.channelOn=ch;
539     }
540       @Override
541         public void run() {
542             //trava o fechamento enquanto estiver escrevendo em arquivo
543             fileManager.writeFile(channelOn);
544             //libera o fechamento
545         }
546 }
547
548 private void jMenuItemExitActionPerformed(java.awt.event.ActionEvent evt) {
549 // TODO add your handling code here:
550     //Preciso fechar todos os arquivos e fluxos abertos, além de conexões e avisar o Interrogador que fechei
551     //Apenas para zerar o arquivo e economizar espaço
552     fileManager =new FileTool("currentFile.csv",4);
553     fileManager.configFileWriter();
554     this.dispose();
555     System.exit(1);
556 }
557
558 private void jTextXMinActionPerformed(java.awt.event.ActionEvent evt) {
559 // TODO add your handling code here:
560     Double auxMin = Double.parseDouble(jTextXMin.getText());
561     if (auxMin < xMaxAtual) {
562         chartGraph.changeLowerBoundX(auxMin);
563         xMinAtual = auxMin;
564     } else {
565         jTextXMin.setText(String.valueOf(xMinAtual));
566     }
567 }
568
569 private void jTextXMaxActionPerformed(java.awt.event.ActionEvent evt) {
570 // TODO add your handling code here:
571     Double auxMax=Double.parseDouble(jTextXMax.getText());
572     if(xMinAtual<auxMax){
573         chartGraph.changeUpperBoundX(auxMax);
574         xMaxAtual=auxMax;
575     }
576      else{
577         jTextXMax.setText(String.valueOf(xMaxAtual));
578     }
579 }
580
581 private void jMenuItemDisconnectActionPerformed(java.awt.event.ActionEvent evt) {
582 // TODO add your handling code here:
583 /*##FECHA TODAS AS CONEXÕES E SOCKETS E PARA O GRÁFICO E STOP NO DATA
GENERATOR*/
584     jMenuItemDisconnect.setEnabled(false);
585     jMenuItemConnect.setEnabled(true);
586     jMenuItemSave.setEnabled(true);
587     jMenuLoad.setEnabled(true);
588     jMenuNextPeak.setEnabled(false);
589     jMenuFirstPeak.setEnabled(false);
590     jMenuNormal.setEnabled(false);
```

```
591    jMenuItemRestSav.setEnabled(false);
592
593     generatorD.stop();
594
595     if(!clientGraph.getClientCont().getIsClosed() ){
596         clientGraph.getClientCont().closeConnection();
597     }
598       ThreadDisconnect d= new ThreadDisconnect();
599       Thread td= new Thread(d);
600       td.start();
601 }
602
603 public class ThreadDisconnect implements Runnable{
604         @Override
605         public void run() {
606           JOptionPane.showMessageDialog(null,"Software Desconectado do Interrogador!",
607                   "Desconectado", JOptionPane.INFORMATION_MESSAGE);
608            while (fileManager.getIsWriting()) {
609               double a = System.currentTimeMillis();
610               double b, c = 10;
611               while (c < 100) {
612                  b = System.currentTimeMillis();
613                  c = b - a;
614               }
615            }//espero acabar de salvar em arquivo para fechar
616         if(fileManager.getIsReading())
617             fileManager.closeReader();
618         else
619            fileManager.closeWriter();
620     }
621 }//fim da Thread Desconnect
622
623 private void jTextThresholdActionPerformed(java.awt.event.ActionEvent evt) {
624 // TODO add your handling code here:
625 /*#### Atualiza o valor do Threshold usado no cálculo de pico*/
626         thresholdValue= Double.parseDouble(jTextThreshold.getText());
627 }
628
629 private void jMenuFirstPeakActionPerformed(java.awt.event.ActionEvent evt) {
630 // TODO add your handling code here:
631 /*PROCURAR O PICO A PARTIR DO INICIO*/
632     firstPeak=true;
633     updateViewBasedOnPeak(peakDetectionThreshold());//procura o pico e atualiza o gráfico
634 }
635
636 private void jMenuNextPeakActionPerformed(java.awt.event.ActionEvent evt) {
637 // TODO add your handling code here:
638     firstPeak=false;
639     updateViewBasedOnPeak(peakDetectionThreshold());//procura o pico e atualiza o gráfico
640 }
641
642 private void jMenuNormalActionPerformed(java.awt.event.ActionEvent evt) {
643     // TODO add your handling code here:
644     updateViewToDefault();
645 }
646
647 public void updateViewToDefault(){
648         if(!milliWatts){
649               xMaxAtual=1600.0;
650               xMinAtual=1500.0;
```

```
651             jTextXMax.setText( Double.toString(xMaxAtual));//Calcula o range
652             jTextXMin.setText( Double.toString(xMinAtual));//Calcula o range
653             chartGraph.changeLowerBoundX(xMinAtual);
654             chartGraph.changeUpperBoundX(xMaxAtual);
655             chartGraph.changeLowerBoundY(-60.0);
656             chartGraph.changeUpperBoundY(5.0);
657         }
658         else{
659             xMaxAtual=1580.0;
660             xMinAtual=1520.0;
661             jTextXMax.setText( Double.toString(xMaxAtual));//Calcula o range
662             jTextXMin.setText( Double.toString(xMinAtual));//Calcula o range
663             chartGraph.changeLowerBoundX(xMinAtual);
664             chartGraph.changeUpperBoundX(xMaxAtual);
665             chartGraph.changeLowerBoundY(-1.0);
666             chartGraph.changeUpperBoundY(1.0);
667     }
668
669 }
670 //milliWatts = 10^(dBm/10)
671 public Double[] toMilliWatts(Double [] channelOn){
672
673     for(int i=0; i<channelOn.length; i++){//converte para milliwatts
674         channelOn[i]= Math.pow(10, (channelOn[i]/10));
675     }
676     milliWatts=true;
677     return channelOn;
678 }
679
680 public void loadFileToGraph(File file){
681 /*CARREGA O ARQUIVO SELECIONADO PARA ALIMENTAR O GRÁFICO*/
682     System.out.println(file.getName());
683     fileManager = new FileTool(file,4);
684     fileManager.configFileReader();
685     /*######## Configuração de tela*/
686     jMenuItemConnect.setEnabled(false);
687     jMenuItemDisconnect.setEnabled(true);
688     jMenuItemSave.setEnabled(false);
689     jMenuLoad.setEnabled(false);
690     jMenuNextPeak.setEnabled(true);
691     jMenuFirstPeak.setEnabled(true);
692     jMenuNormal.setEnabled(true);
693     jMenuItemRestSav.setEnabled(true);
694
695     generatorD=new DataGeneratorGraph(1000);
696     generatorD.start();
697 }
698
699 public void createOpenDialog() { // Método que cria o FileChooser e recebe o arquivo inicial
700         JFileChooser saveFile = new JFileChooser();
701         File file= new File("tempFile.dat");
702         saveFile.changeToParentDirectory();
703         int resultado = saveFile.showOpenDialog(this);
704
705         if(resultado == JFileChooser.APPROVE_OPTION){
706         //Aqui a aplicação deve abrir o arquivo e realizar as operações
707             file = saveFile.getSelectedFile();
708         }
709     //Método que carrega de um File Salvo os dados para o Gráfico
710         loadFileToGraph(file);
```

```
711 }
712
713 public void createSaveDialog(File defaultFile){ // Método que cria o FileChooser e recebe o arquivo inicial
714        JFileChooser saveFile = new JFileChooser();
715        saveFile.changeToParentDirectory();
716        if (defaultFile != null) { // Se o arquivo inicial foi especificado, seta no FileChooser
717          saveFile.setSelectedFile(defaultFile);
718        }
719       int resultado = saveFile.showSaveDialog(this);
720       String nomeArquivo = saveFile.getName(saveFile.getSelectedFile());
721       //System.out.println("Nome do arquivo selecionado: "+nomeArquivo+" e "
+fileManager.getOutFile().getName());
722
723       if(resultado == JFileChooser.APPROVE_OPTION){
724             File arquivoNovo= new File(nomeArquivo);
725             FileInputStream origem;
726             FileOutputStream destino;
727             //canais
728             FileChannel fcOrigem;
729             FileChannel fcDestino;
730           try {
731              origem = new FileInputStream(fileManager.getOutFile().getName());
732              destino = new FileOutputStream(nomeArquivo);
733              fcOrigem = origem.getChannel();
734              fcDestino = destino.getChannel();
735              try {
736                 fcOrigem.transferTo(0, fcOrigem.size(), fcDestino);
737                 origem.close();
738                 destino.close();
739              } catch (IOException ex) {
740                 Logger.getLogger(GUISoftware.class.getName()).log(Level.SEVERE, null, ex);
741              }
742
743           } catch (FileNotFoundException ex) {
744              Logger.getLogger(GUISoftware.class.getName()).log(Level.SEVERE, null, ex);
745           }
746
747       }
748 }
749
750 public int peakDetectionThreshold(){
751
752    if(firstPeak.booleanValue()){
753 //Caso o firstPeak esteja setado como true, deseja-se encontrar o primeiro pico
754        currentPeakIndex=0;
755        currentSearchIndex=0;
756    }
757 //Inicia-se a busca, apartir do pico atual
758    for(int i= currentSearchIndex; i< currentSpectrum.length; i++){
759
760        if(currentSpectrum[i]> thresholdValue){//É um pico válido
761             currentPeakIndex=i;
762           for(int j= i; j< currentSpectrum.length; j++){
763              if(currentSpectrum[j]> currentSpectrum[currentPeakIndex]){
764                 currentPeakIndex=j;
765              }
766              else if(currentSpectrum[j] < thresholdValue){
767                        currentSearchIndex=j;
768                        break;//return currentPeakIndex;
769              }
```

```
770             }
771             break;
772             //fecha o for interno
773         }//fecha o if do Threshold
774         if(i >(currentSpectrum.length-20)){//retorna para o inicio
775             currentPeakIndex=0;
776             currentSearchIndex=0;
777             i=currentSearchIndex;
778         }
779     }//fecha o for da busca
780 
781 return currentPeakIndex;
782 }
783 
784 public void updateViewBasedOnPeak(int peakIndex){
785 //Atualiza o gráfico, valores nos campos de xMax e xMin
786     int range=1000;//Range em relacao ao pico
787     xMaxAtual= (peakIndex+1+range)*0.005+1500;//atualiza variaveis de controle
788     xMinAtual=(peakIndex+1-range)*0.005+1500;//atualiza variaveis de controle
789     jTextXMax.setText( Double.toString(xMaxAtual));//Calcula o range
790     jTextXMin.setText( Double.toString(xMinAtual));//Calcula o range
791     chartGraph.changeLowerBoundX(xMinAtual);
792     chartGraph.changeUpperBoundX(xMaxAtual);
793     currentSearchIndex=currentSearchIndex+20;
794 
795 }
796 
797 
798 
799     /**
800      * @param args the command line arguments
801      */
802     public static void main(String args[]) {
803         /* Set the Nimbus look and feel */
804         //<editor-fold defaultstate="collapsed" desc=" Look and feel setting code (optional) ">
805         /* If Nimbus (introduced in Java SE 6) is not available, stay with the default look and feel.
806          * For details see http://download.oracle.com/javase/tutorial/uiswing/lookandfeel/plaf.html
807          */
808         try {
809             for (javax.swing.UIManager.LookAndFeelInfo info :
javax.swing.UIManager.getInstalledLookAndFeels()) {
810                 if ("Nimbus".equals(info.getName())) {
811                     javax.swing.UIManager.setLookAndFeel(info.getClassName());
812                     break;
813                 }
814             }
815         } catch (ClassNotFoundException ex) {
816 
java.util.logging.Logger.getLogger(GUISoftware.class.getName()).log(java.util.logging.Level.SEVERE, null,
ex);
817         } catch (InstantiationException ex) {
818 
java.util.logging.Logger.getLogger(GUISoftware.class.getName()).log(java.util.logging.Level.SEVERE, null,
ex);
819         } catch (IllegalAccessException ex) {
820 
java.util.logging.Logger.getLogger(GUISoftware.class.getName()).log(java.util.logging.Level.SEVERE, null,
ex);
821         } catch (javax.swing.UnsupportedLookAndFeelException ex) {
```

```
822
java.util.logging.Logger.getLogger(GUISoftware.class.getName()).log(java.util.logging.Level.SEVERE, null,
ex);
823         }
824         //</editor-fold>
825
826         /* Create and display the form */
827         java.awt.EventQueue.invokeLater(new Runnable() {
828
829             public void run() {
830                 GUISoftware guiSoftware= new GUISoftware();
831                 guiSoftware.setVisible(true);
832             }
833         });
834     }
835
836     private FileTool fileManager;
837     private Boolean milliWatts; //inicia como false
838     DataGeneratorGraph generatorD;
839     private GUIConnect guiConnect;
840     private Boolean firstPeak;
841     private int currentPeakIndex, currentSearchIndex;//Armazena o indice do Pico atual na visualizaçao
842     private Double[] currentSpectrum;//ArrayList usado para cálculos de Pico
843     private Client clientGraph;//processConnection() para iniciar
844     Double xMaxAtual, xMinAtual, thresholdValue;
845     private GraphDynamic chartGraph;
846
847     // Variables declaration - do not modify
848     private javax.swing.JComboBox jComboBox1;
849     private javax.swing.JLabel jLabel1;
850     private javax.swing.JLabel jLabel2;
851     private javax.swing.JLabel jLabel3;
852     private javax.swing.JMenu jMenuAbout;
853     private javax.swing.JMenuBar jMenuBar1;
854     private javax.swing.JMenu jMenuEdit;
855     private javax.swing.JMenu jMenuFile;
856     private javax.swing.JMenuItem jMenuFirstPeak;
857     private javax.swing.JMenuItem jMenuItemConnect;
858     private javax.swing.JMenuItem jMenuItemDisconnect;
859     private javax.swing.JMenuItem jMenuItemExit;
860     private javax.swing.JMenuItem jMenuItemRestSav;
861     private javax.swing.JMenuItem jMenuItemSave;
862     private javax.swing.JMenuItem jMenuLoad;
863     private javax.swing.JMenuItem jMenuNextPeak;
864     private javax.swing.JMenuItem jMenuNormal;
865     private javax.swing.JPanel jPanel1;
866     private javax.swing.JRadioButtonMenuItem jRButtonDbm;
867     private javax.swing.JRadioButtonMenuItem jRButtonWatt;
868     private javax.swing.JTextField jTextThreshold;
869     private javax.swing.JTextField jTextXMax;
870     private javax.swing.JTextField jTextXMin;
871     // End of variables declaration
872 }
873
874
```