**AUTARQUIA ASSOCIADA À UNIVERSIDADE DE SÃO
PAULO**

**Sistema de automação e controle para aquisição de espectros de correlação
angular gama-gama**

ADEMIR FRANCISCO ABADE STACHOWSKI

**Dissertação apresentada como parte dos
requisitos para obtenção do Grau de Mestre
em Ciências na Área de Tecnologia
Nuclear - Aplicações**

**Orientador:
Prof. Dr. Artur Wilson Carbonari**

São Paulo

2019

INSTITUTO DE PESQUISAS ENERGÉTICAS E NUCLEARES
Autarquia associada à Universidade de São Paulo

**Sistema de automação e controle para aquisição de espectros de correlação
angular gama-gama**

ADEMIR FRANCISCO ABADE STACHOWSKI

**Dissertação apresentada como parte
dos requisitos para obtenção do
Grau de Mestre em Ciências na
Área de Tecnologia Nuclear -
Aplicações**

Orientador:
Prof. Dr. Artur Wilson Carbonari

Versão Corrigida
Versão original disponível no IPEN

São Paulo

2019

Autorizo a reprodução e divulgação total ou parcial deste trabalho, para fins de estudo e pesquisa, desde que citada a fonte

Como citar:

STACHOWSKI, A. F. A. . ***Sistema de automação e controle para aquisição de espectros de correlação angular gama-gama***. 2019. 63 p. Dissertação (Mestrado em Tecnologia Nuclear), Instituto de Pesquisas Energéticas e Nucleares, IPEN-CNEN/SP, São Paulo.

Disponível em: <www.teses.usp.br> (data de consulta no formato: dd/mm/aaaa)

Ficha catalográfica elaborada pelo Sistema de geração automática da Biblioteca IPEN/USP, com os dados fornecidos pelo(a) autor(a)

Stachowski, Ademir Francisco Abade
Sistema de automação e controle para aquisição de espectros de correlação angular gama-gama / Ademir Francisco Abade Stachowski; orientador Artur Wilson Carbonari. -- São Paulo, 2019.
63 p.

Dissertação (Mestrado) - Programa de Pós-Graduação em Tecnologia Nuclear (Aplicações) -- Instituto de Pesquisas Energéticas e Nucleares, São Paulo, 2019.

1. PAC. 2. Engenharia de Software. 3. Java. I. Carbonari, Artur Wilson, orient. II. Título.

AGRADECIMENTOS

Ao meu orientador Dr. Artur Wilson Carbonari pela oportunidade, competência e principalmente pela paciência.

Ao Dr. André Luís Lapolli também pela oportunidade, competência e constante acompanhamento e orientação na elaboração do projeto e por ter me apresentado o Laboratório de Interações Hiperfinas (LIH).

SISTEMA DE AUTOMAÇÃO E CONTROLE PARA AQUISIÇÃO DE ESPECTROS DE CORRELAÇÃO ANGULAR GAMA-GAMA

ADEMIR FRANCISCO ABADE STACHOWSKI

RESUMO

Este trabalho relata os resultados de um projeto de automação de aquisição de dados de um dos espectrômetros de correlação angular perturbada gama-gama (PAC) do laboratório de interações hiperfinas (LIH) do IPEN. O espectrômetro é constituído de quatro detectores de cintilação (BaF_3). Basicamente registra-se a diferença de temporal entre a detecção de quaisquer dois raios gamas emitidos em cascata (do núcleo de prova) incidentes em quaisquer dois detectores do espectrômetro. Dependendo das condições, o núcleo de prova ocupa determinados sítios de uma rede cristalina ou de uma macromolécula onde se pretende estudar as características elétricas e/ou magnéticas do meio nas áreas de física da matéria condensada ou biofísica.

Para realização deste trabalho utilizou-se a linguagem Java por se tratar de uma linguagem de programação multiplataforma e orientada a objetos, possuindo, portanto, portabilidade e maior tempo de vida do software e devido à facilidade de implementação dos periféricos. Por fim, o software possui uma interface amigável e intuitiva, com o usuário podendo operar o sistema manualmente ou automaticamente, com aquisições de tempo de até 100 horas consecutivas. No modo automático, Antes de cada aquisição, o software seta a temperatura (do controlador), previamente definida pelo usuário, aguarda a estabilidade, processa a aquisição, registra os dados em arquivos diferentes (para temperaturas diferentes). Além disso, o software também salva em um arquivo todas as informações de parâmetros relevantes durante a aquisição.

Palavras-chave: PAC. Engenharia de Software. Java.

CONTROL SYSTEM AND AUTOMATION OF THE SPECTRUM ACQUISITION FOR A PERTURBED GAMMA-GAMMA ANGULAR CORRELATION SPECTROMETER

ADEMIR FRANCISCO ABADE STACHOWSKI

ABSTRACT

This work reports results from the automation of data acquisition for a gamma-gamma perturbed angular correlation (PAC) spectrometer located in the hyperfine interactions laboratory (LIH) of IPEN. A PAC spectrometer comprises usually four scintillation detectors (BaF_3). Basically, the time difference between the detection of any two gamma emitted rays (of the probe nuclei) incident on any two spectrometer detectors is recorded. Depending on the conditions, the probe nuclei occupies certain sites of a crystalline lattice or a macromolecule where it is intended to study the electrical and / or magnetic characteristics of the medium in the areas of condensed or biophysical matter physics. In order to perform this work, the Java language was used because it is a multiplatform and object-oriented programming language, thus having portability and longer software life and due to the ease of implementation of the peripherals. Finally, the software has a friendly and intuitive interface with the user being able to operate the hardware manually or automatically with time acquisitions up to 100 consecutive hours. In automatic mode, the software sets the temperature (controller) previously set by the user, awaits stability, processes the acquisition, and records the data in different files (for different temperatures) before each acquisition. In addition, the software also saves to a file all relevant parameter information during acquisition.

Keywords: PAC. Software engineering. Java.

LISTA DE SIGLAS

UML *Unified Modeling Language*

API *Application Programming Interface*

DCE *Data Communications Equipment*

DTE *Data Terminal Equipment*

COM *Component Object Model*

JVM *Java Virtual Machine*

JNI *Java Native Interface*

UHCI *Universal Host Controller Interface*

OHCI *Open Host Controller Interface*

EHCI *Enhanced Host Controller Interface*

OSI *Open System Interconnection*

EPS *Encapsulated PostScript*

SVG *Scalable Vector Graphics*

PNG *Portable Network Graphics*

ISA *Industry Standard Architecture*

SUMÁRIO

1- INTRODUÇÃO	1
2- LABORATÓRIO DE INTERAÇÕES HIPERFINAS (LIH).....	3
2.1-CORRELAÇÃO ANGULAR PERTURBADA.....	5
2.2-INTERAÇÕES QUADROPOLARES.....	6
2.3- INTERAÇÕES MAGNÉTICAS.....	7
3- FUNDAMENTAÇÃO TEÓRICA.....	9
3.1-CICLO DE VIDA CLÁSSICO.....	9
3.2-PROTOTIPAÇÃO.....	10
3.3-MODELO INCREMENTAL.....	11
3.4-MODELO ITERATIVO.....	11
3.5-ITERATIVO E INCREMENTAL.....	12
3.6-COMBINANDO PARADIGMAS.....	12
4- REQUISITOS PARA O DESENVOLVIMENTO DO PROJETO.....	17
4.1 -INTERFACE SERIAL.....	19
4.1.1-RS 232.....	20
4.1.2-RS 485.....	22
4.1.2.1- PROTOCOLO MODBUS.....	23
4.2-USB.....	28
5.-O PROJETO DESENVOLVIDO.....	30
5.1- CONTROLADORES DE TEMPERATURA.....	35
5.1.1 CPM45.....	35
5.1.2 LAKESHORE.....	38
5.2 MULTICANAL.....	40
6. O SOFTWARE CONCLUSO.....	47
7- CONCLUSÕES.....	50
8-REFERÊNCIAS BIBLIOGRÁFICAS.....	52

Lista de Figuras

Figura 1- Representação da Correlação Angular Perturbada.....	5
Figura 2- Modelo Iterativo Incremental.....	13
Figura 3- Níveis de Abstração.....	14
Figura 4- Visão de Casos de Uso.....	15
Figura 5-Diagrama de Classes.....	15
Figura 6-Diagrama de Atividades.....	16
Figura 7-Diagrama de Implementação.....	16
Figura 8-Diagrama de Componentes.....	17
Figura 9-Representação dos Dispositivos.....	18
Figura 10-Transmissão de Caractere.....	19
Figura 11-Transmissão de Sinais.....	22
Figura 12-Driver de Comunicação.....	22
Figura 13-Representação do Modelo OSI.....	24
Figura 14-Fluxograma CRC 16.....	27
Figura 15-Diagrama de Classes.....	32
Figura 16-Estrutura Geral do Projeto.....	34
Figura 17-CPM 45.....	36
Figura 18-Controlador de Temperatura Lakeshore.....	39
Figura 19-Multicanal.....	41
Figura 20-Interface do Multicanal.....	45
Figura 21-Representação Conceitual da API JFreeChart.....	46
Figura 22-Interface de Acesso de Usuários.....	48
Figura 23- Espectro Gerado durante Aquisição de Dados.....	49
Figura 24-Diagrama de Atividades.....	50

1-INTRODUÇÃO

A ciência da computação revolucionou a pesquisa científica sendo hoje reconhecida como o “terceiro pilar” de sua sustentação, juntamente com a teoria e a experimentação, permeando os avanços em todas as áreas do conhecimento. Novas formas de interação entre as ciências, em vários níveis e escalas, são mediadas pela tecnologia da informação, que é a simbiose da ciência da computação com diferentes domínios do conhecimento. Muitas das grandes descobertas científicas recentes são resultados do trabalho de equipes multidisciplinares que envolvem cientistas da computação, sendo ela um componente indispensável para a implementação e o fortalecimento dos objetivos econômicos, tecnológicos e sociais. [1]

Com a evolução tecnológica, eletrônica e informática, o processo de aquisição e análise de dados experimentais, em todas as áreas da ciência, vêm se aprimorando no sentido de se obter resultados cada vez mais confiáveis e precisos. Neste sentido, a interferência humana é reduzida durante a aquisição de dados, evitando-se erros grosseiros, devido ao fato do conjunto hardware e software se integrarem de forma cada vez mais inteligente baseado em algoritmos específicos desenvolvidos para este fim. Por outro lado, os computadores com capacidade de memória e processamento cada vez maior permitem realização de análise de uma maior quantidade de dados de determinado fenômeno e, portanto, a obtenção de resultados com maior precisão. [2]

A aquisição de dados de interação hiperfina, necessita de detectores de radiação gama e eletrônica associada sendo necessário a variação de temperatura durante o processo de aquisição dos dados. Neste laboratório, há dois sistemas de controle de temperatura, um com variação da temperatura ambiente até 1200°C e outro da ambiente até 8 k. Neste contexto, para o estudo de interações hiperfinas, é necessário uma análise de dados em que a amostra seja submetida à diferentes temperaturas variando em um intervalo de aproximadamente 10 K à 1500 K e para cada temperatura seja necessário a gravação de espectros em tempos dependentes da meia vida do núcleo de prova utilizado na amostra. Desta forma, há a necessidade de interferência do operador para a troca de temperatura e nova aquisição dos dados até que o plano experimental seja satisfeito.

Todo este processo é cansativo pois depende da presença do operador bem como a mudança corretas dos parâmetros necessários para a aquisição.

Desta forma, a utilização de um sistema computacional que controle os processos de aquisições dados torna o plano de trabalho mais eficiente e reduz a probabilidade de erros operacionais durante a rotina de trabalho.

Por outro lado, computadores, controladores (de temperatura), módulos de aquisição, etc, evoluem e a planta de aquisição deve ser renovada periodicamente.

De acordo com o destacado acima, este trabalho propõe o desenvolvimento de um sistema que torna o trabalho de pesquisa mais eficiente e que satisfaça de forma adequada todo o contexto de evolução tecnológica possuindo um ciclo de vida longo, prevendo a alteração de apenas partes do mesmo sem o desenvolvimento de um novo sistema para cada renovação de hardware. Desta forma, optou-se pelo paradigma de programação orientada ao objeto onde a linguagem adotada foi a Java que permitiu a criação de interfaces e métodos específicos para controlar cada um dos periféricos que envolvem o sistema, permitindo ainda uma implementação constante. Além disso, a linguagem Java, devido as suas características, é uma linguagem multiplataforma, podendo ser utilizada em diversos sistemas operacionais sem a necessidade de se reescrever o código.

Neste sentido, a proposta deste trabalho é o desenvolvimento de um sistema de aquisição de dados para o estudo de interações hiperfinais em compostos intermetálicos, nanoestruturas, macromoléculas, biomoléculas, etc, no Laboratório de Interações Hiperfinais do IPEN.

Entretanto, para satisfazer os requisitos de desenvolvimento de software baseado em ciência da computação, este trabalho utiliza engenharia de software, que é uma área da informática focada na especificação, desenvolvimento e manutenção de sistemas de software que aplicam tecnologias e práticas de informática, gerenciamento de projetos e outras disciplinas, visando organização, produtividade e qualidade [1].

Finalmente, o trabalho resultou em software onde se permite a aquisição dos dados automaticamente/manualmente, dependendo da necessidade. Assim, no Capítulo 2 descreve-se sobre o Laboratório de Interações Hiperfinais; A fundamentação teórica, Capítulo 3; no Capítulo 4 são descritos os Requisitos Para o Desenvolvimento do Projeto; Os detalhes do projeto estão nos Capítulos 5 e 6 e no Capítulo 7 as Conclusões.

2- LABORATÓRIO DE INTERAÇÕES HIPERFINAS(LIH)

O Laboratório e Interações Hiperfinas (LIH) do Instituto de Pesquisas Energéticas e Nucleares (IPEN) investiga este fenômeno através da técnica de Correlação Angular gama-gama Perturbada há mais de 20 anos realizando investigações em diversos compostos como ligas de heusler, óxidos perovskitas [3], óxidos delafossites [4], compostos contendo terras raras (RAg, RCo₂, RNiIn, RPdIn), defeitos em Si e estudos em semicondutores magnéticos utilizando os núcleos de prova ¹⁸¹Hf, ¹¹¹In e ¹⁴⁰La. Esta técnica consiste na observação da perturbação, devido às interações hiperfinas, do estado intermediário de um núcleo radioativo que decai emitindo de duas radiações- γ sucessivas. Se o núcleo ocupar um local, numa estrutura cristalina ou molecular, é possível obter informações dessa estrutura e da distribuição de cargas em torno deste.

Além dos trabalhos experimentais, com o objetivo de interpretar parâmetros hiperfinos mais complexos, para compreender melhor os fenômenos na matéria condensada, o LIH vem desenvolvendo cálculos teóricos utilizando-se os métodos de primeiros princípios “*Real Space Linear Muffin-Tin Orbital in the Atomic Sphere Approximation*”(RS-LMTO-ASA) e “*Full-Potential Linear Augmented Plane Wave*” (FP-LAPW)

Para manter o sistema de aquisição de dados atualizado, além de investir na aquisição de novos equipamentos, há o investimento para o desenvolvimento de sistemas eletrônicos que possibilitam uma aquisição de dados de forma mais eficiente.

A infraestrutura para realização dos seus diversos trabalhos o LIH é a seguinte:

- 01 Espectrômetro de quatro detectores de BaF₂ com a eletrônica associada;
- 01 mini forno tubular para aquecimento das amostras durante a medida com um controlador eletrônico de temperatura;
- 02 sistemas criogênicos para resfriamento de amostras com controlador de temperatura Lakeshore;
- 01 criostato de pirex para nitrogênio líquido para medidas das amostras a temperatura de 77 K;
- Programa de análise de difração de Raios X;

- Programa TDPAC para tratamento dos dados das medidas de coincidências e obtenção dos espectros PAC;
- Programas de análise dos espectros PAC para ajuste de funções de interações hiperfinas. Hoje são os (FITLAST e PacFIT), DEPACK;
- Sistemas de estratégias de análise desenvolvidas pelos próprios pesquisadores: uso de processo manual, planilhas eletrônicas, editores gráficos, etc;
- Programa WIEN2K para cálculos de primeiros princípios.

Todos estes sistemas possuem suas características próprias e são operacionalizados da seguinte forma:

1. Os sistemas que submetem as amostras radioativas a temperaturas abaixo ou acima da temperatura ambiente são montados e operados manualmente durante o processo de aquisição de dados;
2. Para realização da aquisição de dados é preciso que o operador realize a seguinte rotina: parada da aquisição, gravação do arquivo, troca da temperatura (quando necessário), reinício de aquisição. Necessitando da presença física do pesquisador para realização do procedimento;
3. A pré-análise (redução) dos dados, realizada pelo TDPAC, é feita espectro a espectro e opera na plataforma Windows.
4. Para análise final utilizam-se os programas FITLAST ou DEPAK que operam em plataforma DOS e PacFIT que roda na plataforma Windows..
5. A pesquisa que utiliza o código FP-LAPW opera na plataforma LINUX.

Como pode ser observado acima, a operacionalização da rotina de trabalho é obsoleta para os dias de hoje. Neste sentido destacam-se os seguintes problemas:

- No processo de aquisição de dados não há nenhuma padronização tanto no que se refere aos módulos eletrônicos (hardware) quanto de manipulação dos dados (software), ocupando de forma desnecessária e inadequada o pesquisador;
- No processo de análise os programas utilizados operam em sistemas operacionais diferentes dificultando a interatividade entre os mesmos;

- De acordo com a estrutura de interligação dos computadores via internet, é possível, realizar-se o controle de aquisição de dados de forma remota, o que reduz a ociosidade do sistema, aumentando consequentemente sua eficiência na obtenção dos dados experimentais. O que não é possível se fazer hoje.

2.1-CORRELAÇÃO ANGULAR PERTURBADA

A Correlação Angular Perturbada consiste na verificação da perturbação do momento angular total de um núcleo (núcleo de prova), em um estado intermediário (com vida média da ordem de nano-segundos) a partir do decaimento de duas radiações gama em cascata devido a interação dos momentos de quadrupolo elétrico e momento de dipolo magnético com os elétricos e magnéticos em torno deste núcleo. A vida média do estado intermediário (da ordem de nano-segundos) é suficiente para que se observe a precessão do momento angular devido a alteração do padrão de emissão da segunda radiação gama em relação à primeira.

Figura 1: Representação da Correlação Angular Perturbada

A influência de um campo extra nuclear sobre a correlação angular de uma cascata gama depende, em particular, da magnitude e da duração da interação. Para perturbações estáticas, a magnitude pode ser descrita pela frequência de precessão ω . Para interações magnéticas, ω é chamada de Frequência de Larmor ω_L , que é proporcional ao momento dipolo magnético nuclear μ e ao campo magnético $\mathbf{B}$. No caso de interações quadrupolares, a frequência quadrupolar ω_Q é proporcional ao momento quadrupolo elétrico nuclear Q e ao gradiente de campo elétrico GCE. Assim, a função correlação angular $W(\theta)$ é escrita da seguinte forma [7]:

$$W(\theta, t) = \sum_k G_{kk}(t) A_{kk} P_k(\cos \theta)$$

onde $G_{kk}(t)$ é o fator de perturbação que contém informações detalhadas sobre a interação nuclear do nível intermediário. Portanto, a função correlação angular não tem a sua forma genérica alterada pela perturbação; apenas foi introduzido um fator de perturbação $G_{kk}(t)$.

2.2-INTERAÇÕES QUADROPOLARES

A interação quadrupolo elétrico é o resultado da interação entre o momento de quadrupolo elétrico nuclear Q e o Gradiente de Campo Elétrico GCE criado pelas cargas ao redor do núcleo. O Gradiente de Campo Elétrico é definido por dois parâmetros: V_{zz} e η , onde V_{zz} é a maior componente do tensor gradiente de campo elétrico no seu eixo principal e η é o parâmetro de assimetria definido como:

$$\eta = \frac{V_{zz} - V_{yy}}{V_{zz}} \quad \text{onde} \quad |V_{xx}| \leq |V_{yy}| \leq |V_{zz}|, \text{ que restringe } \eta \text{ ao intervalo: } 0 \leq \eta \leq 1,$$

A frequência quadrupolar elétrica é dada por:

$$\omega_Q = \frac{eQV_{zz}}{4I(2I-1)\hbar}$$

onde I é o spin do nível intermediário e $\hbar = h/2\pi$ onde h é a constante de Planck. O fator de perturbação para o caso da interação quadrupolar estática é dado por [8]:

$$G_{KK}(t) = \sum_{n=0}^{n_{max}} S_{kn} \cos(n\omega_0 t)$$

sendo $\omega_0 = 3\omega_Q$ para I inteiro e $\omega_0 = 6\omega_Q$ para I semi-inteiro. Os coeficientes S_{kn} dependem do parâmetro de assimetria η . O índice η é determinado pelo valor do spin I do estado intermediário. Conhecendo-se previamente o valor de Q, é possível determinar o valor de V_{zz} através da expressão da frequência quadrupolar.

2.3- INTERAÇÕES MAGNÉTICAS

Quando um campo magnético $\vec{B}$ age sobre um núcleo (seja um campo hiperfino em uma amostra ferromagnética ou um campo magnético externo aplicado) que decai com emissão de radiações gama sucessivas, a interação do campo magnético com o momento magnético nuclear $\vec{\mu}$ provoca uma precessão do spin nuclear em torno da direção do campo magnético $\vec{B}$, com uma frequência ω_L . Por consequência, após um tempo t, o spin nuclear sofre uma rotação dada pelo ângulo $\omega_L t$. Se a permanência do núcleo no estado intermediário da cascata gama for suficientemente longa ($> 10^{-9}$ s) a interação magnética provoca uma mudança na direção do spin do estado intermediário e consequentemente altera a direção de emissão da segunda radiação γ_2 da cascata. A frequência de Larmor é dada por:

$$\omega_L = -\frac{g\mu_N B}{\hbar}$$

onde g é o fator giromagnético do estado intermediário μ_N é o magnéton nuclear ($\mu_N = 5,05 \times 10^{-31}$ J/Gauss). A função correlação angular, quando o plano dos detectores está perpendicular ao campo magnético, é dada por [8]:

$$W(\theta, t, B) = \sum_K A_{KK} P_K[\cos(\theta - \omega_L t)]$$

Se considerarmos a amostra ferromagnética na ausência de campo magnético externo, seus domínios magnéticos estarão orientados ao acaso e cada domínio tem uma função correlação angular dada pela equação:

$$W(\theta, t) = \sum_K G_{KK}(t) A_{KK} P_K(\cos\theta)$$

Nesta situação, embora não exista uma rotação resultante da correlação angular, ocorrem oscilações harmônicas no padrão da correlação angular em função do tempo com mais de uma frequência, Para este caso, o fator de perturbação G_{KK} é escrito por:

$$G_{KK}(t) = \frac{1}{2k+1} \left[1 + 2 \sum_{n=1}^k \cos(n\omega_L t) \right]$$

A partir da medida experimental de correlação angular perturbada é possível extrair a frequência de Larmor ω_L , e, conseqüentemente, o campo hiperfino magnético $\mathbf{B}$ desde que se conheça o valor do fator g- nuclear.

3- FUNDAMENTAÇÃO TEÓRICA

A Engenharia de Software é o conjunto de técnicas, métodos, processos e ferramentas utilizados na especificação, construção, implementação e manutenção de um software destinado a garantir o gerenciamento, controle e qualidade de artefatos gerados através de recursos humanos. Tem como objetivo primário o aprimoramento da qualidade dos produtos de software e o aumento da produtividade dos engenheiros de software, além do atendimento aos requisitos de eficácia e eficiência, ou seja, efetividade [9].

A Engenharia de Software visa sistematizar a produção, a manutenção, a evolução e a recuperação de produtos de software, de modo que ocorra dentro de prazos e custos estimados, com progresso controlado e utilizando princípios, métodos, tecnologia e processos em contínuo aprimoramento. Os produtos desenvolvidos e mantidos, seguindo um processo efetivo e segundo preceitos da Engenharia de Software asseguram, por construção, qualidade satisfatória, apoiando adequadamente os seus usuários na realização de suas tarefas, operam satisfatoriamente em ambientes reais e podem evoluir continuamente, adaptando-se a um mundo em constante evolução [9].

Associando a esses objetivos, o termo engenharia pretende indicar que o desenvolvimento de software deve submeter-se a leis similares às que governam a manufatura de produtos industriais em engenharias tradicionais, pois ambos são metodológicos [10].

Com base nos objetivos da Engenharia de Software, fica demonstrada a necessidade da adoção de um modelo sistêmico para padronizar e gerenciar os processos de desenvolvimento de software com o intuito de buscar qualidade nos processos e produtos de software, sendo que os principais modelos serão descritos.

3.1- CICLO DE VIDA CLÁSSICO

É o modelo mais antigo utilizado. É um método sistemático e sequencial, em que o resultado de uma fase se constitui na entrada da outra fase. Foi modelado de acordo com o ciclo da engenharia convencional e abrange as seguintes fases [9]:

- **Análise de Engenharia de Sistema:** conhecer o sistema e através dele estabelecer os requisitos que devam fazer parte do software;

- **Análise de Requisitos de Software:** revisão de informações e requisitos e especificações das funcionalidades, desempenho e interface;
- **Projeto:** constituído de pontos distintos: estrutura de dados, arquitetura do software, detalhes procedimentais e caracterização de interface. Traduz os requisitos levantados para avaliação e qualidade do software antes da codificação;
- **Codificação:** transformação do projeto para que possa ser interpretado pela máquina. Se o projeto for bem detalhado a codificação torna-se praticamente mecânica;
- **Testes:** etapa onde são verificados os erros e se o código produz o resultado desejado;
- **Manutenção:** correção de erros e adaptação do software ao ambiente onde será instalado.

Dentre as vantagens que podemos citar é que o modelo tem uma abordagem disciplinada e dirigida a documentação, tendo como problemas a dificuldade de seguir o fluxo, a determinação de requisitos iniciais (incerteza natural) e o resultado efetivo aparece apenas no final.

O ciclo de vida clássico continua sendo o modelo procedimental mais amplamente usado pela Engenharia de Software. Embora tenha fragilidade, ele é significativamente melhor do que uma abordagem casual ao desenvolvimento de software. [9]

3.2- PROTOTIPAÇÃO

A prototipação é usada para identificar os requisitos e pode ser a melhor escolha quando estão definidos apenas objetivos gerais para o software, sem requisitos detalhados de entrada, processamento ou saída, ou o desenvolvedor pode não ter certeza daquilo que precisa ser desenvolvido. A prototipação tem início com a coleta dos requisitos, onde desenvolvedor e o cliente definem os objetivos gerais do software. Um projeto é elaborado e a partir dele é construído um protótipo para avaliação, proporcionando ao cliente e desenvolvedor a compreensão daquilo que precisa ser feito.

A vantagem da prototipação é a facilidade para se determinar os requisitos iniciais e a garantia de atingir as necessidades do cliente, enquanto que as desvantagens são a implementação rápida do protótipo comprometido, a determinação do fim do desenvolvimento e tendência a utilizar o protótipo como produto final.

A Prototipação é um ciclo de vida eficiente quando as regras entre cliente e desenvolvedor são definidas no início do processo e há concordância em que o protótipo, por ser um protótipo, deve ser descartado [9].

3.3- MODELO INCREMENTAL

Esta abordagem está baseada no incremento contínuo de funcionalidades do software ao longo do desenvolvimento, onde é inicialmente particionado em funcionalidades de acordo com um prévio levantamento de requisitos e as funcionalidades são construídas uma de cada vez, atravessando todas as atividades de desenvolvimento.

Este modelo representa um ganho enorme em relação às abordagens anteriores. De forma gradativa elimina os riscos do projeto e apresenta resultado rápido à comunidade usuária do software em construção. Tendo em vista que nenhuma empresa ou usuário irá esperar um ano para iniciar sua interação com o software. Os erros também são continuamente eliminados e a familiaridade dos usuários com o software é constante e produtiva [10].

3.4- MODELO ITERATIVO

O Modelo Iterativo tem como princípio a aplicação contínua de todas as atividades de um ciclo de vida ao longo de todo desenvolvimento do projeto. Uma forma relativamente fácil de entender esta abordagem é compararmos a pintura de uma casa.

- A modelo cascata só iniciaria a pintura quando toda casa estivesse pronta e no mesmo estágio.
- O modelo Incremental iniciaria a pintura quando a primeira parede já estivesse pronta e iria pintá-la completamente com a tinta alvo.
- O Modelo Iterativo iniciaria a pintura quando algumas paredes já estivessem construídas. Contudo, não aplicaria a tinta alvo. Iniciaria pintando com cal, pois tem consciência de que o muito trabalhar na casa iria sujar as paredes que já estariam prontas.

Em resumo, sempre quando possível, a abordagem iterativa deveria ser utilizada, o usuário irá interagir desde cedo com o software, diminuindo consideravelmente a recusa ao software durante a implantação, além do efeito treinamento [10].

3.5 -ITERATIVO E INCREMENTAL

A ideia básica por trás do desenvolvimento iterativo e incremental é desenvolver um produto através de ciclos repetidos (iterativos) e em pequenas porções de cada vez (incremental). Aproveitando o que foi aprendido no passado, o cliente e os desenvolvedores melhoram iterativamente o produto até que este esteja completo. Em cada iteração, são feitas modificações a componentes (desenvolvimento iterativo) e novas capacidades funcionais são adicionadas (desenvolvimento incremental).

Possui as seguintes vantagens:

- Disponibilidade de partes prontas do sistema mais cedo;
- Facilidade nos testes: Geralmente, testar cada incremento é mais fácil do que testar o software pronto e tudo de uma vez só;
- Feedback do cliente a cada incremento feito;
- A aprendizagem do desenvolvedor numa linguagem é favorecida: Pode se optar em resolver as partes mais fáceis antes, enquanto ele aprende a linguagem, e deixar as partes mais complexas do sistema para depois.

Na prática o processo iterativo acaba sendo também incremental, e vice-versa, pois ao desenvolver as partes no processo incremental, iremos fazer uma iteração [10].

3.6- COMBINANDO PARADIGMAS

Os paradigmas podem e devem ser combinados de forma que as potencialidades de cada um possam ser obtidas num único projeto. Os paradigmas da Engenharia de Software podem ser combinados num único esforço de desenvolvimento de software. O trabalho inicia-se com a definição dos objetivos, alternativas e restrições. A partir desse ponto qualquer caminho pode ser seguido [9].

Neste contexto, o software é um artefato que leva em consideração os modelos de ciclo de vida. Com base em um estudo bibliográfico, estruturas, especificidades, implementações do sistema ora desenvolvido, o modelo Iterativo Incremental é o que melhor se adequa a esses requisitos . Neste modelo, as funcionalidades são especificadas de acordo com as regras de coesão e acoplamento, respeitando a regra de não interagir de forma exaustiva em uma funcionalidade antes de prosseguir com outra. É importante

enfatizar que esse modelo é menos vulnerável às mudanças nas especificações ao longo do próprio desenvolvimento. A Figura 1 representa o processo do modelo Iterativo Incremental.

Figura 2: Modelo Iterativo Incremental.

Para desenvolver software usando a engenharia de software, é necessário adotar um paradigma de programação, e o paradigma escolhido foi orientação a objetos, que é uma forma de programação de software adotada pela maioria das linguagens de programação modernas. Ele busca expressar as coisas de uma maneira mais próxima da vida real, com base na composição e interação entre objetos [9].

A orientação a objetos nasceu como uma linguagem de programação. A ideia básica inicial por trás da orientação a objetos era unificar em módulos tanto os dados quanto as funções que operavam os dados posteriormente este conceito foi denominado encapsulamento de objetos.

Contudo, o conceito chave de uma linguagem orientada a objetos é denominado polimorfismo, que é o princípio pelo qual duas ou mais classes derivadas de uma mesma superclasse podem invocar métodos que têm a mesma identificação (assinatura), mas comportamentos distintos, especializados para cada classe derivada, usando para tanto uma referência a um objeto do tipo da superclasse. Esse mecanismo é fundamental na programação orientada a objetos, permitindo definir funcionalidades que operem genericamente com objetos, abstraindo-se de seus detalhes particulares quando esses não forem necessários [9-10].

Posteriormente, ficou claro que o benefício de uma linguagem orientada a objetos existia, porém não era suficiente para atender aos requisitos de engenharia de

software. Neste contexto, a modelagem orientada a objetos ganhou forças. Com esta abordagem a orientação a objetos foi impulsionada, os verdadeiros benefícios promulgados pela orientação a objetos começaram a se fazer sentir com mais facilidade. No entanto, não existia nenhum padrão estabelecido do que seria uma modelagem orientada a objetos, e desta necessidade nasceu a UML (*Unified Modeling Language*) [10].

Visando a facilidade de comunicação do projeto, bem como aos requisitos científicos de desenvolvimento, a modelagem foi realizada em UML, que é uma linguagem de diagramação para especificar, visualizar e documentar sistemas de software orientados a objeto auxiliando na visualização de desenho e comunicação entre objetos [10].

Finalmente, salienta-se que a orientação a objetos é uma tentativa de aumentar ainda mais o nível de abstração em relação ao funcionamento da máquina (*hardware*) observando-se na Figura 2 este aumento no nível de abstração [10].

Figura 3: Níveis de abstração.

A UML permite representar o sistema em cinco visões:

- Visão de Casos de Uso: Apresenta uma visão do ponto de vista externo ao sistema, descrevendo seu comportamento sob a perspectiva dos usuários finais e projetistas, capturando os requisitos funcionais que definirão a estrutura do projeto (Figura 4).

Figura 4 – Visão de Casos de Uso

- Visão Lógica: Apresenta as classes, interfaces, subsistemas, e as colaborações, e os diagramas UML usados para representa-la são: Diagrama de Classes, Objetos, Interação, Sequência, Estados e Atividades (Figura 5).

Figura 5 – Diagrama de Classes

- Visão de Processo: Exibe o processamento simultâneo, através de processos, além de mecanismos de concorrência e sincronização entre sistemas. Os diagramas usados são os mesmos da Visão Lógica com foco nas classes ativas (Figura 6).

Figura 6 – Diagrama de Atividades

- Visão de Implantação: Define a alocação dos componentes do sistema nos pontos da rede que formam a topologia de hardware onde o sistema será instalado. Os diagramas UML utilizados para mostrar o aspecto do sistema são: Diagrama de Implantação, Interação, Sequência, Estados e Atividades (Figura 7).

Figura 7 – Diagrama de Implantação

- Visão de Implementação: Corresponde aos componentes, subsistemas e arquivos que serão usados na composição do sistema. Os diagramas UML utilizados para mostrar a estrutura do sistema são: Diagrama de Componentes, Interação, Sequência, Estados e Atividades (Figura 8).

Figura 8 – Diagrama de Componentes

4. REQUISITOS PARA O DESENVOLVIMENTO DO PROJETO

Para o desenvolvimento do trabalho foi necessário integrar a automação e o controle dos seguintes dispositivos:

- Multicanal: multicanal ORTEC (modelo 918-A) para aquisição de dados PAC que se comunicam via interface Universal Serial Bus (USB).
- Sistemas criogênicos: controle de temperatura Lakeshore (modelo 331) com alcance de 8 K a 295 K, que se comunicam através de interface serial RS 232 e RS 485.
- Sistema de aquecimento: controlador de temperatura Contemp (modelo CPM45) com faixa de temperatura de 20°C a 1200°C, que se comunica através de interface serial RS 485.

Esses dispositivos foram automatizados e controlados por um computador que usa o sistema operacional Windows XP, onde o acionamento das funcionalidades destes dispositivos estão centralizadas em um único software, conforme a Figura 9 .

Figura 9: Representação dos dispositivos.

Nestas condições, o sistema foi desenvolvido constituído com módulos de controle utilizando as interfaces de comunicação para o multicanal e os sistemas de controle de temperatura que estão detalhados a seguir.

4.1 Interface Serial

Antes de 1960 a comunicação de dados compreendia a troca de dados digitais entre um computador central (*mainframe*) e terminais de computador remoto, ou entre dois terminais sem o envolvimento do computador. Tais dispositivos poderiam ser conectados através de linha telefônica, e conseqüentemente necessitavam de um modem em cada lado para fazer a decodificação dos sinais.

Na comunicação serial a troca de dados binários no qual os caracteres são enviados um a um como um conjunto de bits. A velocidade transmissão de dados seriais costuma ser expressa como bps (*bits-per-second*) ou baud (*baud rate*). Este método permite que seja utilizado um único fio para a transmissão (*TX*) e recepção (*RX*).

Na Fig. 10, observa-se o exemplo de um caractere sendo enviado por transmissão serial com um bit de início, seguido por sete bits de dados, possivelmente um bit de paridade e um bit de paragem sendo então necessários pelo menos 10 bits para enviar um único caractere. Vale notar que o estado alto e baixo é dado neste caso respectivamente pela tensão de 12 (V) e de -12 (V) [12].

Figura 10: Transmissão de caractere.

Quanto ao sentido de fluxo de dados, as comunicações se dividem em três tipos: *simplex*, *half-duplex* e *duplex (full-duplex)*. Estes conceitos se aplicam tanto as transmissões seriais quanto paralelas.

- *Simplex*: O tipo de comunicação *simplex* de interligação de dispositivos é aquele no qual se tem um elemento que apenas transmite e outro que somente recebe.
- *Half-duplex* ou *semi-duplex*. Neste modo de comunicação, têm-se elementos que recebem e transmitem dados, embora as duas operações não possam ocorrer simultaneamente.

- *Full-duplex* ou simplesmente *duplex*, que consiste em um modo pelo qual os sistemas podem transmitir e receber dados simultaneamente, permitindo maior agilidade das operações de comunicação de dados.

Abaixo uma tabela comparativa com as principais características das interfaces seriais utilizadas no presente trabalho:

Especificações	Padrão RS232	Padrão RS485	Padrão USB 2.0
Modo de operação:	Ponto a ponto	Multiponto	Multiponto
Tipo de transmissão :	Full-duplex	Half-duplex	Half -duplex
Nº total de <i>Drivers</i> e <i>Receivers</i> :	1 Driver e 1Receivers	1 Driver e 32 Receivers	1 Driver e 127 Receivers
Comprimento máximo da rede :	12 metros	1200 metros	5 metros
Taxa de transmissão :	20 kbps	10 Mbps	480 Mbps

4.1.1 - RS 232

RS é uma abreviação de “*Recommended Standard*”. Ela relata uma padronização de uma interface comum para comunicação de dados entre equipamentos, criada no início dos anos 60, por um comitê conhecido atualmente como “*Electronic Industries Association*” (EIA). Naquele tempo, a comunicação de dados compreendia a troca de dados digitais entre um computador central (mainframe) e terminais de computador remotos, ou entre dois terminais sem o envolvimento do computador. Estes dispositivos poderiam ser conectados através de linha telefônica, e consequentemente necessitavam um modem em cada lado para fazer a decodificação dos sinais. Dessas idéias nasceu o padrão RS232. Ele especifica as tensões, temporizações e funções dos

sinais, um protocolo para troca de informações, e as conexões mecânicas [12].

A mais de 30 anos desde que essa padronização foi desenvolvida, a EIA publicou três modificações. A mais recente, EIA232E, foi introduzida em 1991. Ao lado da mudança de nome de RS232 para EIA232, algumas linhas de sinais foram renomeadas e várias linhas novas foram definidas. Embora tenha sofrido poucas alterações, muitos fabricantes adotaram diversas soluções mais simplificadas que tornaram impossível a simplificação da padronização proposta. As maiores dificuldades encontradas pelos usuários na utilização da interface RS232 incluem pelo menos um dos seguintes fatores [12]:

- A ausência ou conexão errada de sinais de controle, resultam em estouro do buffer (“*overflow*”) ou travamento da comunicação.
- Função incorreta de comunicação para o cabo em uso, resultam em inversão das linhas de Transmissão e Recepção, bem como a inversão de uma ou mais linhas de controle (“*handshaking*”).

Felizmente, os drivers utilizados são bastante tolerantes aos abusos cometidos, e o equipamento normalmente sobrevive. Diversos sinais são necessários para conexões onde o dispositivo DCE (*Data Circuit-Terminating Equipment*) é um modem, e eles são utilizados apenas quando o protocolo de software os emprega. Para dispositivos DCE que não são modem, ou quando dois dispositivos DTE (*Data Terminal Equipment* – usualmente um computador ou terminal) são conectados diretamente, poucos sinais são necessários. Deve-se notar que existe um segundo canal que inclui um conjunto de sinais de controle duplicados. Este canal secundário fornece sinais de gerenciamento do modem remoto, habilitando a mudança de taxa de transmissão durante a comunicação, efetuando um pedido de retransmissão se erros de paridade forem detectados, e outras funções de controle. Os sinais de temporização de transmissão e recepção são utilizados somente quando o protocolo de transmissão utilizado for síncrono. Para protocolos assíncronos, padrão 8 bits, os sinais de temporização externos são desnecessários.

Os nomes dos sinais que implicam em uma direção. Como “*Transmit Data*” e “*Receive Data*”, são nomeados do ponto de vista dos dispositivos DTE. Se a norma

EIA232 for seguida a risca, estes sinais terão o mesmo nome e o mesmo número de pino do lado do DCE. Infelizmente, isto não é feito na prática, provavelmente porque em alguns casos torna-se difícil definir quem é o DTE e quem é o DCE. A figura a seguir apresenta a convenção utilizada para os sinais mais comuns [12].

Figura 11: Transmissão de sinais.

4.1.2-RS 485

No protocolo RS-485 há apenas um par de fios para transmissão e recepção que deve ser compartilhado. Esta estratégia possui algumas vantagens e desvantagens. A grande vantagem, sem dúvida, é a possibilidade de se interligar vários equipamentos que podem se comunicar entre si através do mesmo cabo, ao contrário dos padrões nos quais só pode haver um dispositivo em cada uma das extremidades do cabo. O alcance da transmissão RS-485 é compatível com o padrão RS-422. Por isso, esse padrão está sendo cada vez mais utilizado em ambientes industriais onde há sistemas automatizados ligados em rede. Na Figura 12 é apresentado o driver do padrão de comunicação RS-485.

Figura 12: Driver de comunicação

Como desvantagem, pode-se citar que se trata de uma comunicação *half-duplex*. O software de comunicação deve gerenciar a habilitação de transmissão e

recepção para evitar uma confusão de dados. Além disso, deve haver um meio de se evitar ou resolver o conflito de dados que pode ocorrer quando dois equipamentos tentam transmitir ao mesmo tempo. Neste caso a solução é permitir que cada equipamento só inicie a transmissão quando verificar se a linha está livre. No caso de dois equipamentos iniciarem a comunicação no mesmo instante, estes devem identificar a colisão de dados, interromper o processo e tentar novamente após tempos diferentes.

Outra solução é ter um gerenciador de rede, que é o único a poder começar uma transmissão e é quem autoriza os demais componentes da rede a iniciar uma comunicação. Desse modo, evita-se todo conflito, porém há a desvantagem de que componentes da rede não podem comunicar entre si livremente, tendo que aguardar pelo gerenciador para desencadear o processo [14].

4.1.2.1- PROTOCOLO MODBUS

O Protocolo Modbus foi desenvolvido na década de 1970 pela Modicon Inc. é um protocolo de transferência de mensagens da camada de aplicação, localizado no nível 7 do modelo de referência OSI (*Open System Interconnection*) que provê comunicação no modo mestre/escravo entre dispositivos conectados em diferentes tipos de redes e barramentos [15].

O modelo de referência OSI possui 7 níveis sendo o primeiro responsável pelo meio físico, o segundo nível responsável pelo link de dados e finalmente o sétimo ocupado pelo protocolo Modbus. A representação do modelo pode ser visto na figura 13.

Figura 13: Representação do modelo OSI.

O protocolo Modbus é de domínio público, o que significa que pode ser usado no desenvolvimento de novos projetos, sem a necessidade do pagamento de *royalties* a quaisquer empresas. O protocolo Modbus utiliza modo de transmissão ASCII ou RTU .

O interesse do presente item está sobre o modo de transmissão RTU, o qual define que para cada palavra de dados da mensagem é enviado apenas um caractere no padrão hexadecimal. A principal vantagem do modo RTU em relação ao ASCII é a maior densidade de caracteres que é enviada numa mesma mensagem, aumentando o desempenho da comunicação. Modbus é um protocolo do tipo requisição/resposta e oferece serviços especificados por códigos de funções. O protocolo Modbus define uma estrutura de mensagens composta por bytes, a qual os dispositivos são capazes de reconhecer, independentemente do tipo de rede utilizada.

Durante o processo de comunicação, o protocolo Modbus determina como cada dispositivo [15]:

- Identifica seu próprio endereço na rede;
- Reconhece se uma mensagem é endereçada a ele;
- Determina qual o tipo de ação a ser executada;
- Obtém as informações necessárias para a execução da ação.

A comunicação é feita através da técnica mestre-escravo, onde apenas um

dispositivo (mestre) pode iniciar a comunicação (*query*). Os outros dispositivos (escravos) respondem enviando os dados solicitados. O mestre pode endereçar individualmente um escravo ou acessar a todos em uma rede através de mensagens em cadeia (*broadcast*). Entretanto, apenas o escravo endereçado retorna uma resposta (*response*) a uma *query* e nunca são geradas respostas quando uma *query* for do tipo *broadcast*. No formato estabelecido pelo Modbus para as mensagens de solicitação, são definidos da seguinte maneira [15]:

- O endereço do escravo (ou código em caso de acesso *broadcast*);
- A função solicitada, com eventuais parâmetros;
- O campo de *checksum*, para verificar a integridade da mensagem.

De maneira similar, é gerada a resposta do escravo correspondente à função solicitada pelo mestre, onde se define:

- A confirmação da função realizada;
- Resultado da função realizada;
- O campo de *checksum*, para verificar a integridade da mensagem.

O protocolo Modbus possui seis campos básicos que definem a estrutura de um frame de comunicação. O campo INÍCIO e o campo FIM definem o início e o final do *framing* de comunicação Modbus. Durante o processo de transmissão de uma mensagem Modbus há identificadores que delimitam o início e o fim de quadro, de acordo com cada modo de transmissão. Desta forma, os dispositivos da rede são capazes de perceber o início da mensagem e ler todos os dados contidos nela até o seu final. Para identificar os campos de início e fim, não deve haver nenhuma transmissão de dados por um período mínimo, equivalente a 3.5 vezes o tamanho da palavra de dados (*silent*). Os dispositivos ficam monitorando a rede e checando o intervalo *silent*, que depois de detectados, dá início à recepção da mensagem. Após a recepção da mensagem, deve ser gerado pelo mestre um intervalo *silent* similar ao início da mensagem, caracterizando o fim da mensagem. As mensagens devem ser transmitidas de modo contínuo. Se um intervalo maior que 1.5 vezes o tamanho da palavra de dados for detectado, o escravo descarta os dados já recebidos e assume que o próximo caractere será o campo de endereço de uma nova mensagem.

O campo ENDEREÇO define o endereço a quem é destinado o *framing* de comunicação. A faixa de endereços válidos para os escravos é de 0 a 247, sendo o 0

reservado para acesso tipo broadcast, e o 1 reservado para o mestre da rede. O campo FUNÇÃO define a função de comunicação Modbus. A faixa de valores válidos para este campo varia de 1 a 255. Quando um escravo recebe uma mensagem, este campo indica qual ação deve ser tomada por ele, como por exemplo: ler e/ou programar uma ou mais saídas, ler e/ou programar um ou mais registradores, ler o estado do escravo, etc. Na resposta do escravo ao mestre, este campo é utilizado para informar se a ação foi executada com sucesso ou se ocorreu algum tipo de erro (*exception*). O campo CRC define o valor de cálculo CRC, para verificar a integridade (*checksum*) de dados usados pelo protocolo Modbus [15]:

- verificação de paridade, que pode ou não ser aplicada para cada caractere da mensagem;
- verificação do quadro, que pode ser do tipo LRC, aplicado a todos os campos de uma mensagem no modo ASCII, ou do tipo CRC, aplicado ao conteúdo de toda a mensagem no modo RTU.

O CRC é calculado através das operações da aritmética módulo 2. Com efeito, é o resto da divisão polinomial entre os dados a enviar, e um polinômio gerador adequadamente escolhido. Para efetuar esta divisão, cada posição dos bits no bloco de dados ou arquivo é considerado como uma potência de x no polinômio, e o valor do bit é o coeficiente correspondente (0 ou 1). O resultado do cálculo é então colocado para fazer parte da mensagem a ser enviada. Ao receber a mensagem, o escravo recalcula o CRC e compara com o valor recebido, se o resultado for igual, a mensagem é aceita caso contrário ela é descartada. O cálculo de CRC é o método geralmente utilizado para checagem da integridade das informações em uma rede Modbus. Todos os escravos da rede, são implementados para verificarem a integridade das mensagens através do cálculo de CRC [15].

O cálculo do CRC é iniciado carregando-se um registrador de 16 bits com todos os bits 1's (65535 decimais). Então um processo se inicia aplicando-se sucessivamente bytes de 8 bits ao conteúdo deste registrador. Somente os 8 bits de dados são utilizados para a geração do CRC. Os bits de início, parada e paridade não participam do cálculo do CRC. Durante a geração do CRC, cada caracter de 8 bits passa por uma operação de “ou exclusivo” com o conteúdo do registrador de 16 bits.

Então o resultado desta operação é deslocado no sentido do bit menos significativo com o bit mais significativo sendo preenchido por um zero. O bit menos significativo é extraído e examinado. Se este bit for 1, o conteúdo do registrador sofre nova operação de “ou exclusivo” com o polinômio

Se o bit for 0, nenhuma ação é executada. Este processo é repetido até que 8 deslocamentos tenham sido realizados. Depois do último deslocamento, o próximo byte de 8 bits sofre o mesmo processo de “ou exclusivo” e deslocamentos descrito acima.

O conteúdo final do registrador, depois de todos os bytes da mensagem terem passado por este processo, é o valor do CRC. Quando o CRC é anexado a mensagem, o byte de menor ordem será anexado primeiro, seguido pelo byte de maior ordem. Na figura 14 abaixo é apresentado o fluxograma do cálculo do CRC para o quadro Modbus.

Figura 14: Fluxograma CRC16.

Para a programação no microcontrolador foi utilizado o esquema lógico descrito pela própria fabricante conforme fluxograma indicado acima, as seguintes

considerações também são necessárias para análise e entendimento do fluxograma:

- XOR = ou exclusivo.
- N = número de bits de informação.
- POLY = polinômio de geração do CRC 16 (1010 0000 0000 0001).
- Polinômio de geração utilizado = $1 + X^2 + X^{15} + X^{16}$.
- No CRC 16 o primeiro byte transmitido é o de menor significado;

4.2-USB

O padrão USB (*Universal SerialBus*) surgiu em 1995 de uma parceria entre várias companhias de tecnologia (Compaq, Hewlett-Packard, Intel, Lucent, Microsoft, NEC e Philips). Uma das primeiras versões foi a 1.0 com velocidade de 1,5 Mbps (*Low-speed*). Logo em seguida foi concebida a 1.1 com velocidades que vão de 1,5 Mbps a 12Mbps. Essa versão é conhecida com *Full-speed*. No final de 2000 foi lançada a versão 2.0 (*High-speed*), compatível com as versões anteriores, mas com alguns aperfeiçoamentos que vão desde a topologia à velocidade de tráfego de dados, chegando ao limite de 480 Mbps, equivalente a cerca de 60 MBps (60 milhões de bytes por segundo). As primeiras versões do bus USB utilizavam/utilizam os Controladores Host UHCI (*Universal Host Controller Interface*) e OHCI (*Open Host Controller Interface*). O USB 2.0 utiliza o Controlador Host EHCI (*Enhanced Host Controller Interface*). O barramento USB permite a conexão máxima de até 127 dispositivos em uma única porta. Para que isso seja possível utilizam-se de *hubs* conectados em cascata [16].

O cabo USB é composto por quatro fios e uma malha para eliminação de ruídos. Desses, dois são fios de alimentação dos dispositivos. A alimentação é fornecida pela controladora e gerenciada pelo *Driver* do *Controlador Host*.

Diferentemente da Porta Serial ou Paralela, nas quais por meio dos sinais elétricos dos pinos e um programa básico, é possível se comunicar com um dispositivo, no USB isso só é possível se o dispositivo carregar o protocolo USB num *chip set* ou

mesmo dentro de um microcontrolador [16].

O protocolo USB tem vários recursos como: CRC (*Cyclic Redundancy Check*), detecção e correção de erros, detecção de conexão e remoção de dispositivos, controle de fluxo de dados assegurando transmissões isossíncronas (tráfego contínuo de dados), disponibilidade assegurada de largura de banda, entre outros [16].

No presente trabalho o padrão USB é utilizado para comunicação com o multicanal, que transmitirá conjuntos de 8192 ou 12288 números resultante da aquisição de espectros de correlação angular perturbada.

5. O PROJETO DESENVOLVIDO

Para colocar em prática o projeto foi utilizada a IDE Eclipse, que é uma plataforma de desenvolvimento de software livre extensível, baseada em Java. Por si só, é simplesmente uma estrutura e um conjunto de serviços para desenvolvimento de aplicativos de componentes de plug-in. Felizmente, o Eclipse vem com um conjunto padrão de plug-ins, incluindo as amplamente conhecidas Ferramentas de Desenvolvimento Java (JDT) [18].

Embora a maioria dos usuários esteja satisfeita em usar o Eclipse como um ambiente de desenvolvimento integrado (IDE) Java, a plataforma ambiciona mais. O Eclipse também inclui o Plug-in Development Environment (PDE), que é de interesse principalmente daqueles que desejam estender o Eclipse, visto que ele permite desenvolver ferramentas que se integram perfeitamente ao ambiente do Eclipse. Como tudo no Eclipse é plug-in, todos os desenvolvedores de ferramentas têm um campo de ação nivelado para oferecer extensões ao Eclipse e fornecer um IDE unificado e consistente aos usuários.

Essa paridade e consistência não se limitam às ferramentas de desenvolvimento Java. Embora o Eclipse seja escrito na linguagem de programação Java, seu uso não se limita à linguagem Java. Por exemplo, estão disponíveis ou planejados plug-ins que incluem suporte para linguagens de programação como C/C++ e COBOL. A estrutura do Eclipse também pode ser usada como base para outros tipos de aplicativos não relacionados ao desenvolvimento de software, com sistemas de gerenciamento de conteúdo [18].

Já para a confecção dos diagramas previstos na UML foi usado a ferramenta CASE Dia é aproximadamente inspirado pelo programa comercial do Windows "Visio", embora mais voltado para diagramas informais para uso casual. Pode ser usado para desenhar muitos tipos diferentes de diagramas. Atualmente, ele possui objetos especiais para ajudar a desenhar diagramas de relacionamento de entidade, diagramas UML, fluxogramas, diagramas de rede e muitos outros diagramas. Também é possível adicionar suporte a novas formas, escrevendo arquivos XML simples.

Ele pode carregar e salvar diagramas em um formato XML personalizado (compactado por padrão, para economizar espaço), pode exportar diagramas para vários formatos, incluindo EPS, SVG, XFIG, WMF e PNG, e pode imprimir múltiplos

diagramas [19]

O sistema foi projetado de forma que o usuário/pesquisador pudesse acessar de um único software todas as funcionalidades necessárias para acompanhar e salvar as medidas PAC em um arquivo para posterior análise. Sendo assim o usuário/pesquisador pode usar o modo manual onde existe a possibilidade de estipular a temperatura da amostra via controladores (CPM45 ou Lakeshore), iniciando a medida quando atingir a temperatura especificada observando o monitor gráfico de temperatura implementado no sistema e posteriormente salvando o resultado da medida em arquivo. Na figura 15 pode-se ver o diagrama de classes do projeto.

Figura 15:Diagrama de Classes.

Já no modo automático o usuário/pesquisador poderá programar até vinte sessões de medida, cada qual com o seu tempo de duração, temperatura e medidas parciais salvas em arquivo no lapso temporal definido pelo usuário/pesquisador, salientando que em ambos os modos (manual e automático) é possível salvar manualmente em arquivo o resultado da medida em andamento e acompanhar a evolução da medida pela representação gráfica do espectro, que é atualizada periodicamente.

A estrutura geral do sistema descrito acima é demonstrada na Figura 16 usando o diagrama de casos de uso constante na UML, que demonstra como os atores interagirão com o software [11].

Figura 16: Estrutura Geral do Projeto.

5.1- CONTROLADORES DE TEMPERATURA

As interfaces de comunicação dos controladores de temperatura são:

- Sistemas criogênicos: RS 232.
- Sistema de aquecimento: RS 485 (protocolo ModBus-RTU).

Para isso foi usado RXTX, que é uma biblioteca nativa que fornece comunicação serial e paralela (com possibilidade de comunicação via Universal Serial Bus - USB) para o Java Development Kit [20].

Além de ajustar a temperatura desejada, ele também retorna o valor atual da temperatura do processo. A API Java RXTX é idealizada através da classe de `CommPort` abstrata e duas subclasses, `SerialPort` e `ParallelPort`, que descreve os dois tipos de comunicação habilitados, que são as portas mais comuns em desktops. Não há possibilidade de instanciar uma dessas classes diretamente, porque seus construtores são intencionalmente não públicos. Para obtê-los, deve-se usar o método estático **`CommPortIdentifier.getPortIdentifier()`** para capturar as portas disponíveis, dando ao usuário a oportunidade de escolher a porta a ser utilizada. Desta forma, através do **`CommPortIdentifier`**, pode-se chamar o método **`open()`** para receber uma instância do objeto **`CommPort`**. Para usar o objeto como uma porta serial, é necessário apenas lançar um elenco para qualquer uma das subclasses derivadas de **`CommPort`**.

Cada uma dessas subclasses tem suas características e métodos. Que permitem que o usuário use eventos Java padrão para receber a notificação de dados disponíveis ou mesmo tipos específicos de eventos como um indicador de sinal (na comunicação de dados) [20].

5.1.1 CPM45

O controlador de temperatura utilizado quando a temperatura da amostra necessita estar entre 25°C e 1200°C foi o modelo CPM45, da CONTEMP. Este dispositivo funciona como um controlador PID (Proporcional, Integral, Derivativo), que atua sobre a variável de processo (*process variable* - PV) de acordo com três parâmetros. O controlador é acessado pelo usuário por duas vias: um painel digital no próprio equipamento, e por comunicação RS-485 com o computador. Os parâmetros adotados pelo controlador para atuar sobre o sistema de controle são os seguintes: Banda proporcional (P), que corresponde à diferença instantânea entre a temperatura medida (*process variable* - PV) e a temperatura de interesse (*setpoint* - SP); Banda Integral (I), que corresponde ao somatório da diferença entre SP e PV ao longo de um determinado intervalo de tempo; banda derivativa (D), que corresponde à taxa de variação da diferença entre SP e PV em um dado instante. O controlador CPM45 pode ser visto na figura 17.

Figura 17: CPM45.

No controlador utilizado, esses parâmetros podem ser ajustados manual ou automaticamente. O método automático consiste em um processo de auto-sintonia, no qual o controlador opera no modo "liga/desliga". Neste modo, o aquecedor é mantido ligado abaixo do SP, e desligado acima do SP. Naturalmente, este modo "liga/desliga" não proporciona controle algum de temperatura, tendo meramente a finalidade de fazer uma varredura de uma ampla faixa de temperatura. Essa varredura permite que o controlador encontre os parâmetros ótimos de acordo com o comportamento de um sistema em particular. Ao longo da auto-sintonia, o dispositivo registra o comportamento

da temperatura do sistema

A chave utilizada pelo controlador para ligar ou desligar o sistema de aquecimento é um relê, que opera através de ciclos liga/desliga. A duração de cada ciclo liga/desliga é configurada pelo usuário, e o tempo em que o relê permanece ligado ou desligado corresponde à função de saída do controlador. Ciclos liga/desliga e o modo "liga/desliga" mencionado anteriormente são duas operações distintas do controlador. Os ciclos se referem ao modo de operação do controlador funcionando de acordo com os parâmetros PID; o modo liga/desliga é maneira de atuação do dispositivo durante o processo de auto-sintonia previamente descrito.

No modo RTU (*Remote Terminal Unit*), cada byte da mensagem contém dois caracteres de 4-bits hexadecimal. A principal vantagem deste modo é que a grande densidade de caracteres permite uma melhor taxa de transferência do que o modo ASCII para a mesma taxa de transmissão. As mensagens são transmitidas em um fluxo contínuo de caracteres para assegurar uma correta transação, e como será visto a frente, o intervalo entre bytes para que a mensagem seja considerada incompleta ou perdida é determinado de acordo com a velocidade de transmissão escolhida. O formato define que para cada 8 bits de informação RTU, são utilizados 3 bits adicionais para formalizar a comunicação serial. Deste modo, cada byte RTU conterá em suma 11 bits:

- 1 start bit;
- 8 bits de dados (menos significativo enviado primeiro);
- 1 bit de paridade;
- 1 stop bit.

Isto garante um aproveitamento de 72% do quadro completo para transmissão da informação útil, sendo os outros 28% bytes de controle. Para efeito de comparação, este aproveitamento gira em torno de 40% para o modo de transmissão Modbus ASCII. A paridade par é utilizada como padrão, mas outras variações como paridade ímpar e sem paridade podem ser implementadas. No envio sem paridade são necessários 2 stop-bits, de modo que não há mudança na relação de bits enviados por bits úteis.

Uma mensagem Modbus-RTU é colocada pelo dispositivo transmissor em um quadro que tem seu início e fim conhecidos. Isto permite aos dispositivos que recebem um novo quadro descobrir quando uma mensagem começa e quando ela é completada. No modo RTU, os quadros de mensagem são separados por um silêncio de pelo menos

3,5 caracteres de tempo. Porém, para evitar carregamento desnecessário da CPU, estes valores devem ser respeitados somente para transmissões de baixa velocidade, de até 19200bps. Para valores maiores é recomendado pelo protocolo que se utilize valores fixos de 750µs para tempo entre caracteres (t1.5) e 1,750 ms (t3.5) para tempo entre mensagens.

Após a detecção do fim do quadro, o cálculo de CRC é realizado. Após o campo de endereço é analisado para determinar se o quadro é destinado a este dispositivo. Se não for, o quadro é descartado. Para reduzir o tempo de processamento da recepção, o campo do endereço pode ser analisado assim que for recebido sem esperar o fim do quadro. Neste caso o CRC será calculado e verificado somente se o quadro é destinado a este escravo (quadros broadcast inclusive). O modo RTU inclui um campo de verificação de erro que é baseado em um método de Verificação cíclica de Redundância (CRC) incluído no conteúdo da mensagem.

O campo de CRC verifica o conteúdo da mensagem inteira. O campo de CRC contém um valor de 16-bits implementado como dois bytes de 8 bits. O campo de CRC é incluído como último campo da mensagem. Quando isto é realizado, o byte de menor ordem do campo é anexado primeiro, seguido pelo byte de maior ordem. O byte CRC de maior ordem é o último byte enviado na mensagem.

O valor do CRC é calculado pelo dispositivo que transmitiu o quadro. O dispositivo que recebe o quadro recalcula o CRC durante a recepção da mensagem, e compara o campo de CRC recebido com o CRC calculado. Se os valores não são iguais, o dispositivo retorna um erro.

Para implementar a automatização do referido controlador de temperatura foram criadas duas classes (uma pra enviar o valor da temperatura desejada e outra para obter o valor da temperatura atual do processo), sendo necessário este artifício para evitar choque durante a transmissão de dados e após a definição da temperatura do processo obtêm-se constantemente o valor da temperatura atual.

5.1.2 LAKESHORE

Para as medições onde a temperatura necessária esteja entre 8k e 320k é usado

o controlador de temperatura Lakeshore Modelo 331, visto na figura 18.

Figura 18: Controlador de temperatura Lakeshore

A interface serial usada no Modelo 331 é a RS-232, que descreve uma das interfaces mais comuns entre computadores e equipamentos eletrônicos. O padrão RS-232 é bastante flexível e permite muitas configurações diferentes. No entanto, quaisquer dois dispositivos que solicitem compatibilidade RS-232 não podem necessariamente ser conectados juntos sem a configuração da interface.

O instrumento usa drivers para gerar os níveis de tensão de transmissão exigidos pelo padrão RS-232. Essas tensões são consideradas seguras sob condições normais de operação devido a seus limites relativamente baixos de tensão e corrente. Os drivers são projetados para trabalhar com cabos de até 50 pés de comprimento.

A temporização assíncrona é usada para os dados de bit individuais em um caractere. Esse tempo requer os bits de início e parada como parte de cada caractere, para que o transmissor e o receptor possam ressincronizar entre cada caractere. A transmissão *half-duplex* permite que o instrumento seja um transmissor ou um receptor de dados, mas não ao mesmo tempo.

Velocidades de comunicação de 300, 1200 ou 9600 bps são suportadas. A taxa de transmissão é o único parâmetro de interface que pode ser alterado pelo usuário.

O *handshaking* de hardware não é suportado pelo instrumento. *Handshaking* é frequentemente usado para garantir que as *strings* de mensagem de dados não colidam e que nenhum dado seja transmitido antes que o receptor esteja pronto. Neste

instrumento, os temporizadores de software apropriados substituem o *handshaking* por hardware. Os programas do usuário devem assumir total responsabilidade pelo controle de fluxo e tempo.

Um caractere é a menor informação que pode ser transmitida pela interface. Cada caractere tem 10 bits e contém bits de dados, bits para temporização de caracteres e um bit de detecção de erros. O instrumento usa 7 bits para dados no formato ASCII. Um bit inicial e um bit de parada são necessários para sincronizar caracteres consecutivos. Paridade é um método de detecção de erros. Um bit de paridade configurado para paridade ímpar é incluído em cada caractere.

Caracteres de letras e números ASCII são usados com mais frequência como dados de caractere. Caracteres de pontuação são usados como delimitadores para separar diferentes comandos ou partes de dados. Dois caracteres ASCII especiais, retorno de carro (CR 0DH) e avanço de linha (LF 0AH), são usados para indicar o final de uma sequência de mensagens.

O controlador de temperatura Modelo 331 usa um algoritmo de controle PID. A equação de controle para o algoritmo PID possui três termos variáveis: proporcional (P), integral (I) e derivativo (D). [17]

Também foram criadas duas classes (uma para enviar o valor da temperatura desejada e outra para obter o valor da temperatura atual do processo), sendo necessário este artifício para evitar choque durante a transmissão de dados e após a definição da temperatura do processo obtêm-se constantemente o valor da temperatura atual.

O *handshaking* foi providenciado no software, sendo definido o intervalo de 100 milissegundos entre as transmissões com o intuito de evitar choque de dados.

5.2 MULTICANAL

O Analisador Multicanal é uma das principais ferramentas usadas na ciência nuclear para registrar os espectros de energia ou tempo disponíveis de detectores de radiação nuclear. É muitas vezes referido como um MCA (MultiChannel Analyzer) ou um MCB (MultiChannel Buffer). Para o desenvolvimento deste trabalho foi utilizado o ADCAM Multichannel Buffer (Model 916-A) da Ortec que pode ser visto na figura 19. [22]

Figura 19: Multicanal

Até o início da década de 1990 função de gravação do espectro estava totalmente contida em um instrumento autônomo. Com o uso do computador pessoal, as funções auxiliares de memória e exibição são transferidas para um computador e o hardware especializado para a histogramação de altura de pulso foi incorporada em um cartão conectado ao painel traseiro do computador ou interfaceado ao computador via USB ou algum outro cabo de interface.

Assim, o MCA com interface de computador é às vezes chamado de MultiChannel Buffer[22].

Considere o caso em que um detector, oferecendo uma resposta linear ao a energia dos raios gama produz um pulso de carga elétrica para cada fóton de raios gama que é detectado. Em termos mais simples a quantidade de carga no pulso produzido pelo detector é proporcional à energia do fóton. O pré-amplificador coleta carga, adicionando o menor ruído possível, converte a carga em um pulso de tensão e transmite o resultado

ao longo da distância ao amplificador de suporte. O amplificador aplica filtros *low-pass* e *high-pass* ao sinal para melhorar o sinal e aumenta a amplitude do pulso. Na saída do amplificador, cada pulso tem uma duração da ordem de microssegundos, e uma amplitude ou altura de pulso que é proporcional à energia do fóton detectado. Medir essa altura do pulso revela energia do fóton. Manter o controle de quantos pulsos de cada altura foram detectados durante um período de tempo registra a energia espectro dos raios gama detectados.

O espectro de energias de raios gama poderia ser gravado usando um analisador de altura de pulso de canal único, um contador e um cronômetro. Esse processo envolveria o uso de uma janela de voltagem muito estreita no analisador de canal único. A janela percorreria todo o intervalo de altura de pulso de 0 a 10V. Em cada etapa, as contagens por unidade de tempo seriam registradas com o contador e timer. Plotar a taxa de contagem medida versus a tensão de nível inferior da janela geraria o espectro de energia.

Antes da invenção do MCA, é assim que os físicos nucleares mediram o espectro de energia. Comparado com o que pode ser realizado com o MCA, era um processo terrivelmente inconveniente [22].

Há uma variedade de projetos de circuitos eletrônicos empregados em analisadores de altura de pulso multicanal. Mas o processo básico pode ser entendido como segue: A chegada de um pulso de entrada válido (derivado da saída do amplificador descrita anteriormente) deve ser reconhecida para iniciar o processo. Para evitar desperdício de tempo de processamento o ruído que está sempre presente entre os pulsos, o MCA usa um discriminador de nível mais baixo com seu limite de tensão definido logo acima da amplitude máxima do ruído. Quando um pulso válido de um raio gama excede a tensão limite, o circuito subsequente é acionado para procurar a altura máxima do pulso. Para proteger contra contaminação de pulsos sucessivos, o *linear gate* (Um *linear gate* transmite o sinal analógico quando o *gate* está aberto e bloqueia o sinal analógico quando o *gate* é fechado. A temporização e as durações dos estados aberto/fechado são controladas pela largura de um pulso lógico) na entrada do ADC fecha assim que o máximo de amplitude é capturada. Essa altura máxima do pulso é apresentada a um conversor analógico-digital (ADC) para ser codificado como representação da tensão analógica. Essencialmente, o ADC compara a altura do pulso analógico a uma escala de

níveis de voltagem. A escala estende-se de 0 a 10 V com um espaço uniforme entre os degraus. Cada degrau representa uma voltagem definida. Quando um pulso chega para medição, o ADC analisa a tensão do pulso e decide em quais dos graus ele pertence. Os intervalos entre os graus são numerados de zero a $N_{\max}$. Se o pulso cair no enésimo espaço entre os graus, uma contagem é adicionada às contagens existentes na enésima localização da memória de histogramação. Uma vez concluída a análise do pulso, o MCA abre seu *linear gate* e aguarda que o próximo pulso disponível chegue. O processo é repetido pulso por pulso durante o tempo de contagem estabelecido pelo operador do instrumento. No final do tempo de contagem, a memória de histograma contém um registro de contagens versus localização que pode ser exibida para apresentar o mesmo espectro de energia que foi plotado a partir do SCA no processo descrito no parágrafo anterior [22].

Os dados na memória de histogramagem dividem a escala de energia do espectro em intervalos digitais discretos. Esses intervalos são tipicamente chamados canais, em analogia com o uso do analisador de canal único descrito acima. Se o ADC tiver 1000 canais, o 0 V até 10 V de largura de pulso é digitalizada em canais com 10 mV de largura. Resoluções ADC estão geralmente disponíveis de 500 a 32.000 canais.

Detectores que possuem uma resolução de energia mais precisa requerem mais canais, enquanto é mais produtivo usar um número menor de canais com detectores com resolução menor.

MCAs que são suportados por computadores geralmente usam este para exibir o espectro, analisar o conteúdo espectral e selecionar os parâmetros operacionais para o MCA. É importante desenvolver familiaridade e agilidade com esses controles. Porque o número do canal é proporcional à energia do raio gama (de primeira ordem), a escala horizontal pode ser calibrada para ler apropriadas unidades de energia. Normalmente, o mouse do computador pode ser usado para clicar no topo de um pico de raios gama no espectro para identificar a energia do raio gama. O mouse também pode ser empregado para definir uma região de interesse (ROI) em um pico. Isso permite convenientes como a soma do número total de contagens em um pico e a subtração do segundo plano abaixo do pico. ROIs podem ser usados com software embarcado para obter uma estimativa mais precisa do pico de energia para fins de calibração de energia.

Uma das características mais úteis de um MCA é o relógio de tempo. Este é um esquema que conta o tempo decorrido apenas quando o sistema está não está morto para aceitar um pulso adicional. Assim, se o experimento é executado por um tempo pré-definido, há uma correção automática para o tempo morto. As contagens observadas divididas pelo tempo decorrido representam a taxa de contagem verdadeira na entrada do detector.

A coleta de contagens de um tempo de vida predefinido não apenas compensa o erro sistemático causado pelo tempo morto, mas também elimina distorção do tempo morto das estatísticas de contagem. Consequentemente, a raiz quadrada do número de contagens obtidas via predefinição em tempo real.

O tempo pode ser usado para estimar o desvio padrão esperado nas contagens.

Outras características importantes são controles para iniciar / parar a aquisição de dados, um botão para limpar o conteúdo da memória, capacidade de selecionar a resolução digital (número de canais em escala completa), discriminadores de nível superior e inferior para limitar as alturas de pulso que são aceitos, e uma função de *linear gate*.

O *linear gate* incorporada no MCA permite que o usuário exerça o controle de entrada para selecionar preferencialmente quais pulsos analógicos serão processados e quais pulsos serão rejeitados. Esta é uma função útil para experimentos de coincidência [22].

MCAs e MCBs geralmente têm o ADC acoplado à memória de histogramação. Este modo de histogramação é uma eficiente configuração quando o instrumento é usado para analisar o espectro de um único detector. Experimentos mais sofisticados em ciência nuclear envolvem múltiplos detectores que medem vários aspectos do decaimento de um núcleo excitado. Neste caso, é mais produtivo que se separe os ADCs da memória de histogramação. O conjunto de eventos coincidentes de todos os detectores para a decadência de um único núcleo deve ser concatenado como um evento multiparâmetros isolado para um processamento de dados mais complexo.

Para essa situação, as saídas de ADC concatenadas são salvas como eventos multiparâmetros sequenciais em uma lista na memória do computador. Isto é referido

como o modo de lista. A mineração de dados úteis do modo de lista requer software complexo e para fins especiais [22].

No painel posterior do módulo ADCAM há o conector DUAL PORT MEMORY (Figura 20) onde foi adaptado (adquirido da ORTEC) um cabo que interliga à interface USB do computador. Juntamente com o cabo foi adquirido o software (driver) que determina a comunicação entre o computador e o multicanal.

Anteriormente esta ligação era feita através de uma interface própria adquirida com este módulo. Esta modificação possibilitou com que o analisador multicanal se comunica-se com o microcomputador com a mesma eficiência e eficácia da interface antiga, que por ser padrão ISA(*Industry Standard Architecture*) não se adapta aos computadores mais recentes.

Figure 20: Interface do Multicanal

Nestas condições, o processo de comunicação computador multicanal foi determinada como uso do Component Object Model (COM), que é uma plataforma da Microsoft para componentes de software lançados em 1993. Ele é usado para ativar a comunicação entre processos e a criação de objetos dinâmicos em qualquer linguagem de programação que suporte a tecnologia [21].

Utilizou-se a biblioteca JACOB (Java COM bridge), que permite chamar componentes de automação COM a partir do Java. Ele usa o JNI (Java Native Interface) para fazer chamadas nativas para as bibliotecas COM e Win32 [23,24].

Neste trabalho foi necessário o desenvolvimento de uma DLL auxiliar que permite o acesso aos envios de comando ao multicanal e a obtenção dos dados retornados

por este. Esses comandos podem ser: "START", "STOP", "SAVE" e "CLEAR" o espectro, enfatizando que o comando "SAVE" irá gravar o espectro em um arquivo a qualquer momento da medição, tanto em modo automático como manual.

Outra característica integrada ao sistema é a possibilidade de representar graficamente o espectro, que é atualizado periodicamente. Esta funcionalidade foi utilizada na biblioteca JFreeChart, que permite a criação de uma grande variedade de gráficos em linguagem Java [25].

Para que os dados gerados possam ser analisados graficamente, a ferramenta utiliza uma biblioteca de classes Java conhecida como JFreeChart. Esta biblioteca gera gráficos, tais como: gráficos de pizza, linhas, barras, áreas, combinações, entre outros. Todo o código-fonte é disponibilizado sob a licença GPL (General Public License), então não há problemas em alterá-la para fins pessoais ou comerciais, fato importante dentro do contexto da ferramenta, pois não será usada apenas para fins pessoais.

Como outras facilidades que são disponibilizadas, podemos contar também com: zoom, impressão, exportar para figuras JPEG e PNG, exportar para PDF e documentação em Javadoc (documentação padrão de classes Java).

Figura 21: Representação conceitual API JFreeChart.

Existem duas famílias principais de gráficos no JFreeChart: de categorias e XY. Para o software em tela, foi utilizada apenas a categoria XY.

Para a construção de gráficos com o JFreeChart deve-se seguir alguns passos:

- definição de um conjunto de dados a serem plotados (Dataset);
- criação do modelo do gráfico

- alteração de propriedades de plotagem (opcional)
- geração da imagem do gráfico a partir do seu modelo.

Esta API oferece vários tipos de séries de dados e essas são instanciadas através de classes como por exemplo XYSeries. Cada série tem um buffer de dados que são vinculados a um plotador. A classe XYPlot pode ser utilizada como plotadora, e nesse caso os dados são inseridos aos pares em coordenadas (x,y)[25].

6. O SOFTWARE CONCLUSO

A interface gráfica principal pode ser vista na figura 22A. A partir desta todas as funções do sistema são acessadas, tais como: a definição da pasta de trabalho, comando Directory do menu; controlador de temperatura, caixa Temperature Controller; número de segmentos, input; se medida será no modo automático ou manual, comandos Manual e Automatic do menu, dentre outras funcionalidades.

É importante ressaltar que as funções vão sendo habilitadas em determinada ordem (para evitar acionamento acidental de algum dispositivo) a saber: Primeiro escolhe-se a pasta de trabalho, número de segmentos do multicanal; qual a porta do controlador de temperatura. Após o que, são habilitados os outros comandos do menu principal (Manual, Automático) e a caixa de escolha de temperatura (Temperature Controller) que podem ser utilizados.

O comando Help, possui todas as orientações para o uso do programa, sendo que a janela Help, não apresentada inicia junto com a janela principal e pode ser acionada a qualquer momento durante a execução do programa.

A utilização do sistema de controle de temperatura é de forma exclusiva onde, independentemente da forma de aquisição escolhe-se uma das formas de submissão de temperatura da amostra de acordo com a necessidade do trabalho.

A janela, Figura-22B, é apresentada após a determinação do modo de medida de temperatura. Para os controladores CPM ou Lakeshore, após a definição manual da temperatura, o monitor de temperatura em tempo real, Figura-22E, aparece para que o operador acompanhe sua variação pela janela gráfica. Quando se escolhe temperatura constante, o operador pode escolher a temperatura em Kelvin ou Celsius acionando-se

um dos botões da janela de unidade de temperatura (Figura 22D).

O uso do sistema em modo manual é feito através do comando Manual do menu, janela principal, Figura-22A. Através deste, o usuário pode iniciar ou parar a aquisição além de salvar o espectro a qualquer tempo.

Figura 22: Interfaces de Acesso dos Usuários.

Já, no modo automático, comando Automatic do menu, após escolha do controlador de temperatura, Em uma janela própria (vista na Figura 22C) existe a opção de se programar até 20 medidas, cada qual com seus respectivos tempos de duração e temperaturas (definidos pelo usuário/pesquisador), obtendo-se também medidas parciais (salvas em arquivos próprios) em intervalos pré-definidos, esclarecendo que antes do início das medidas há uma verificação para que as temperaturas digitadas não estejam fora do intervalo aceito pelos controladores de temperatura (20°C-1200°C no CPM45 e 8 K-320 K no Lakeshore) e cada medida é iniciada automaticamente após a estabilização da temperatura definida.

Acionando-se o botão [SPECTER], Figura 23, pode ser observado em tempo real pelo usuário, permitindo verificação do andamento da medida, possibilitando, desta forma a interferência do operador a qualquer tempo durante a aquisição dos dados.

Durante a aquisição, de forma automática, o programa gera um arquivo de monitoração onde o usuário, durante ou após a aquisição do conjunto de dados, pode verificar os parâmetros definidos por ele definido na programação, bem como, o processo de gravação automático dos espectros de periodicamente. Havendo qualquer ocorrência durante a medida, é possível, com base neste arquivo, realizar-se um diagnóstico de falhas.

Figura 23: Espectro gerado durante a aquisição dos dados.

Todas as funcionalidades acima descritas podem ser vistas no Diagrama de Atividades (em conformidade com a UML) representado na Figura 24.

Figura 24:Diagrama de Atividades.

7- CONCLUSÕES

O software foi testado em vários experimentos PAC e foi avaliado a partir do ponto de vista da engenharia de software e pelos usuários pesquisadores, e foi considerado aprovado.

Devido a restrições impostas pelo fabricante do multicanal (ORTEC), a comunicação só é compatível com várias versões da plataforma Windows (XP ou superior) de 32 ou 64 Bits, porém futuramente com novas implementações tais restrições poderão ser eliminadas tornando a comunicação do multicanal totalmente multiplataforma. No módulo de comunicação dos controladores de temperatura, essa limitação não existe, sendo compatível com plataformas Linux e Mac, bem como o Windows.

Também quando o gráfico está acionado, durante seu processo de atualização existe consumo muito grande de memória do computador, o que não pode ser evitado pois é uma característica da Linguagem Java que procura desalocar as posições de memória automaticamente (processo por vezes deficiente) diferentemente de outras linguagens de programação onde a limpeza de posições de memória é feita

explicitamente como o C/C++. É somente uma pequena desvantagem que em quase nada desabona a Linguagem Java diante das vantagens descritas anteriormente neste trabalho.

8- REFERÊNCIAS BIBLIOGRÁFICAS

1. Grandes Desafios da Pesquisa em Computação no Brasil – 2006 – 2016. Relatório sobre o Seminário realizado em 8 de maio de 2006. Disponível em <<http://www.sbc.org.br/documentos-da-sbc/send/141-grandes-desafios/798-grandesdesafios-portugues>>
2. Computational Science: Ensuring America's Competitiveness. PITAC Report to the President. EUA. Junho de 2005, Disponível em <http://www.nitrd.gov/pitac/reports/20050609_computational/computational.pdf>
3. Junqueira, Astrogildo de C.. Estudo de interações hiperfinas em óxidos perovskitas do tipo $\text{La}(\text{MT})\text{O}_{\text{sub}(3)}$ (MT=metais de transição Fe, Cr, Mn e Co). 2004. Tese (Doutoramento) - Instituto de Pesquisas Energéticas e Nucleares - IPEN/CNEN-SP, São Paulo. 112 p. Orientador: Artur Wilson Carbonari.
4. Attili, R.N.; SAXENA, R.N.; CARBONARI, A.W.; MESTNIK FILHO, J.; UHRMACHER, M.; LIEB, K.P. Delafossite oxides $\text{ABO}_{\text{sub}(2)}$ (A=Ag, Cu; B=Al, Cr, Fe, In, Nd, Y) studied by perturbed-angular-correlation spectroscopy using a $\text{sup}(111)\text{Ag}(\text{Beta sup}(-)) \text{sup}(111)\text{Cd}$ probe. Phys. Rev. B, , v. 58, n. 5, p. 2563-2569, 1998.
5. Cavalcante, Fabio H. de M. Estudo de interações hiperfinas magneticas em sistemas intermetalicos do tipo RAg (R=terra rara). 2004. Dissertacao (Mestrado) - Instituto de Pesquisas Energeticas e Nucleares - IPEN/CNEN-SP, Sao Paulo. 77 p. Orientador: Artur Wilson Carbonari
6. Lapolli, André L. Estudo de interações hiperfinas em compostos intermetálicos $\text{Gd}(\text{nI}, \text{Pd})\text{In}$ e $\text{Ho}(\text{ni}, \text{Pd})\text{In}$. 2006. Tese (Doutoramento) - Instituto de Pesquisas Energéticas e Nucleares - IPEN-CNEN/SP, São Paulo. Orientador: Arthur Wilson Carbonari.

7. FRAUENFELDER, H.; STEFFEN, R.M. Angular Distribution of Nuclear Radiation: an Angular Correlation. In SIEGBAHN, K. (Ed.) Alpha-, Beta-, and Gamma-ray Spectroscopy. North-Holland Publishing, Amsterdam, N.Y., 1965, v.2, p.997-1198
8. DOMIENIKAN, Claudio. Interface eletrônica para aquisição de 12 espectros de coincidências gama-gama atrasadas. 2001. Dissertação (Mestrado) - Instituto de Pesquisas Energéticas e Nucleares - IPEN/CNEN-SP, São Paulo. 86 p. Orientador:Rajendra Narain Saxena.
9. Pressman, Roger S., *Engenharia de Software*, McGraw-Hill, São Paulo Brasil (2006).
10. Magela, R., *Engenharia de Software Aplicada: Fundamentos*, Alta Books, Rio de Janeiro (2006).
11. Lima, A.S., *UML 2.0 Do Requisito à Solução*, Editora Érica Ltda, São Paulo Brasil (2005).
12. RABELLO, L. M, *Programa em Linguagem JAVA para comunicação Serial*, Embrapa Comunicado Técnico, São Paulo Brasil (2009).
13. TEXAS INSTRUMENTS. *The RS-232 Design Guide*, Texas, USA (2002)
14. TEXAS INSTRUMENTS. *The RS-485 Design Guide*, Texas, USA (2008)
15. <<http://www.modbus.org/>>
16. <<http://www.usb.org/home>>
17. ASTRÖM, K. J & T. HÄGGLUND, — PID Controllers: Theory, Design and Tuning, ISA – Instrumentation, Systems and Automation Society, RTP, NC, 1995.
18. “Introdução à Plataforma Eclipse”
<<https://www.ibm.com/developerworks/br/library/os-eclipse-platform/index.html>>
(2012).

19. <<https://wiki.gnome.org/Apps/Dia/Documentation>>
20. “RXTX Wiki”, http://rxtx.qbang.org/wiki/index.php/Using_RXTX (2015).
21. “The Component Object Model”, <[https://msdn.microsoft.com/pt-br/library/windows/desktop/ms694363\(v=vs.85\).aspx](https://msdn.microsoft.com/pt-br/library/windows/desktop/ms694363(v=vs.85).aspx)> (2015).
22. <<https://www.ortec-online.com/>>
23. “The JACOB project: A Java-COM bridge”, <http://danadler.com/jacob/> (2015).
24. “Java-COM integration with JACOB using XML wrappers”, <http://users.jyu.fi/~minurmin/javacom/javacom_report.pdf> (2015).
25. “JFreeChart”, <<http://www.jfree.org/jfreechart/>> (2016).