André de Oliveira Águila Favoto

Desenvolvimento de uma arquitetura de controle distribuído para um criostato vertical

Uberlândia, MG

2024

André de Oliveira Águila Favoto

Desenvolvimento de uma arquitetura de controle distribuído para um criostato vertical

Trabalho de Conclusão de Curso da Engenharia de Controle e Automação da Universidade Federal de Uberlândia - UFU - Campus Santa Mônica, como requisito para a obtenção do título de Graduação em Engenharia de Controle e Automação.

Universidade Federal de Uberlândia - UFU
Faculdade de Engenharia Elétrica - FEELT

Orientador Prof. Dr. Renato Ferreira Fernandes Junior

Uberlândia, MG
2024

André de Oliveira Águila Favoto

Desenvolvimento de uma arquitetura de controle distribuído para um criostato vertical

Trabalho de Conclusão de Curso da Engenharia de Controle e Automação da Universidade Federal de Uberlândia - UFU - Campus Santa Mônica, como requisito para a obtenção do título de Graduação em Engenharia de Controle e Automação.

Trabalho aprovado em de de 2024.

COMISSÃO EXAMINADORA

Prof. Dr. Renato Ferreira Fernandes Junior
Orientador

Professor Dr. Renato Santos Carrijo
Membro Avaliador

Professor Dr. Márcio José da Cunha
Membro Avaliador

Uberlândia, MG
2024

Agradecimentos

Agradeço à minha família, em especial meus pais, Anderson e Érica e à minha irmã Amanda, pelo apoio e incentivo durante toda a minha vida e acreditar em mim quando nem eu mesmo acreditei.

Agradeço à minha noiva, Flávia, pela companhia, apoio e compreensão incondicionais durante toda a minha graduação, mesmo nos momentos mais difíceis e nas longas horas de estudo.

Agradeço também aos meus amigos da 95 FEELT, que sem dúvidas tornaram essa jornada mais leve e divertida, além das experiências acadêmicas e pessoais compartilhadas que levarei para a vida toda.

Agradeço ao Grupo de Automação e Robótica do CNPEM, pela oportunidade de aprender com profissionais de alto nível e conhecer tecnologias que nunca imaginei que conheceria de perto. Destaco o reconhecimento aos co-autores deste projeto, Caio Marcilio, Ranieri Santos e em especial Andrei Guinancio, meu orientador de estágio, pela paciência, ensinamentos e profissionalismo ao longo de meus quase dois anos no CNPEM.

Agradeço aos professores da Universidade Federal de Uberlândia, especialmente ao Prof. Dr. Renato Ferreira Fernandes Junior, pela orientação e apoio durante a realização deste trabalho.

Por fim, agradeço a mim mesmo pela resiliência e dedicação durante toda a graduação, apesar dos obstáculos e dificuldades, e por ter chegado até aqui.

Resumo

Este documento apresenta o sistema de controle desenvolvido para um criostato vertical, um componente essencial nas validações do projeto de um dispositivo de inserção supercondutor para o Sirius, a fonte de luz síncrotron brasileira. Usado para testar bobinas supercondutoras, condutividade térmica de materiais e validar desenvolvimentos relacionados ao projeto, este criostato depende de cinco principais subsistemas: criogenia, vácuo, fontes de corrente, e proteção pessoal e do dispositivo. Dadas as suas interdependências e os aspectos únicos de cada um, a estratégia desenvolvida permite gerenciar cada subsistema de forma eficaz, garantindo uma operação confiável e intuitiva. Este trabalho discute a integração dos subsistemas do dispositivo, a arquitetura de controle definida, as estratégias adotadas e a interface de operação desenvolvida.

Palavras-chaves: Integração de sistemas; Operação remota; Sistemas distribuídos; EPICS; Supercondutividade

Abstract

This document presents the control system developed for a vertical cryostat, an essential component in the validation of the project for a superconducting insertion device for Sirius, the Brazilian synchrotron light source. Used to test superconducting coils, thermal conductivity of materials, and validate developments related to the project, this cryostat relies on five main subsystems: cryogenics, vacuum, current sources, and personal and device protection. Given their interdependencies and the unique aspects of each, the developed strategy allows for effective management of each subsystem, ensuring reliable and intuitive operation. This work discusses the integration of the device's subsystems, the defined control architecture, the adopted strategies, and the developed operation interface.

Keywords: Systems integration; Remote operation; Distributed systems; EPICS; Superconductivity

Lista de Figuras

Lista de tabelas

Lista de Fragmentos de Código

Lista de abreviaturas e siglas

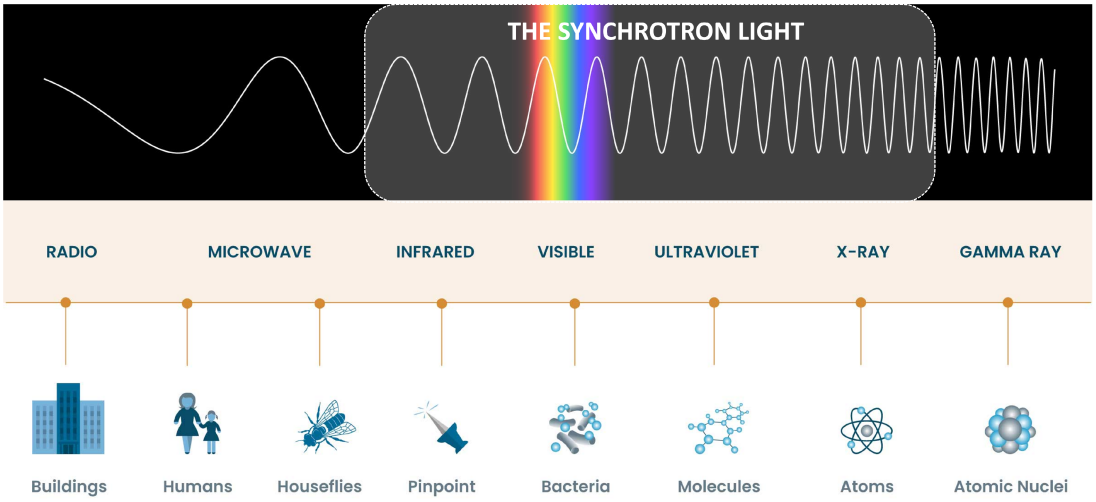
ARO	Grupo de Automação e Robótica - CNPEM
BSMP	Basic Small Messages Protocol
CA	Channel Access
COP	Grupo Conversores de Potência - CNPEM
CLP	Controlador Lógico Programável
CNPEM	Centro Nacional de Pesquisa em Energia e Materiais
CSS	Cascading Style Sheets
EPICS	Experimental Physics and Industrial Control System
HTTP	Hypertext Transfer Protocol
HTTPS	Hypertext Transfer Protocol Secure
ID	Dispositivo de Inserção
IOC	Input/Output Controller
IP	Internet Protocol
JS	JavaScript
PV	Variável de Processo
SPA	Single Page Application
SPOF	Ponto Único de Falha
SWLS	Superconducting Wavelength Shifter
TCP	Transmission Control Protocol
UDP	User Datagram Protocol

Sumário

1. Introdução

Nos aceleradores de partículas emissores de luz, a operação e suas aplicações são baseados na utilização da luz síncrotron. A luz síncrotron é um tipo de radiação eletromagnética de alto brilho e de amplo espectro, composta por comprimentos de onda que variam do infravermelho até as ondas de raios X. Sua emissão se dá a partir de desvios na trajetória de feixes de elétrons submetidos a velocidades próximas à da luz (LNLS, 2022).

Figura 1 – O espectro eletromagnético da luz síncrotron



Fonte: Adaptado de (HUBBLESITE, 2022)

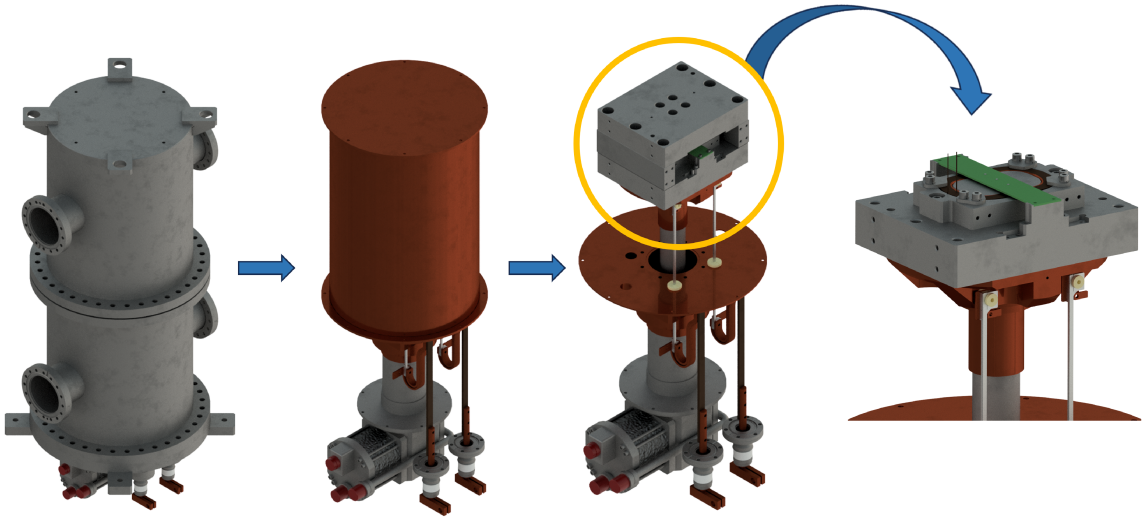
De acordo com LNLS (2023), com a manipulação adequada dessa radiação é possível revelar características moleculares e atômicas de todo tipo de material, com aplicações diretas no desenvolvimento da indústria farmacêutica, química, metalúrgica, energética, alimentícia, entre diversas outras.

Em síncrotrons de 4ª geração como o Sirius, localizado em Campinas (SP) no Centro Nacional de Pesquisa em Energia e Materiais (CNPEM), essa radiação é emitida preferencialmente através de dispositivos de inserção (IDs). Através de sucessões de campos magnéticos de polos alternados, os dispositivos de inserção são equipamentos responsáveis por provocar desvios ou oscilações sucessivos na trajetória do feixe de elétrons, e, consequentemente, emissão de luz síncrotron para as chamadas “linhas de luz”. As propriedades do feixe de luz síncrotron gerado para cada linha de luz são controladas pelo período, ângulo e outras características dessas oscilações provocadas pelo respectivo ID da linha (NIST, 2021).

Dessa forma, com a demanda da instalação de uma linha de luz de tomografia de Raios-X de alta energia no Sirius, e, devido às características necessárias para o ID que fosse capaz de atender a tais requisitos, propôs-se o projeto de um dispositivo de

inserção supercondutor do tipo wavelength shifter, chamado de SWLS (Superconducting Wavelength Shifter). Desenvolvido por uma equipe multidisciplinar do CNPEM, o projeto detalhado em Galvez et al. (2023) conta com três dipolos de Nióbio-Titânio (NbTi) que, para atingir a supercondutividade, operam em temperaturas inferiores a 5K (-268,15 ºC). Devido às condições extremas de operação, a validação dos subsistemas e do projeto eletromagnético não podem ser feitas em bancadas de testes convencionais. Assim, a fim de permitir os testes das bobinas supercondutoras, bem como a validação dos subsistemas relacionados ao projeto, a equipe propôs-se a desenvolver um dispositivo auxiliar capaz de chegar às condições de temperatura e pressão especificados para a operação do SWLS: o criostato vertical.

Figura 2 – Estrutura do criostato vertical

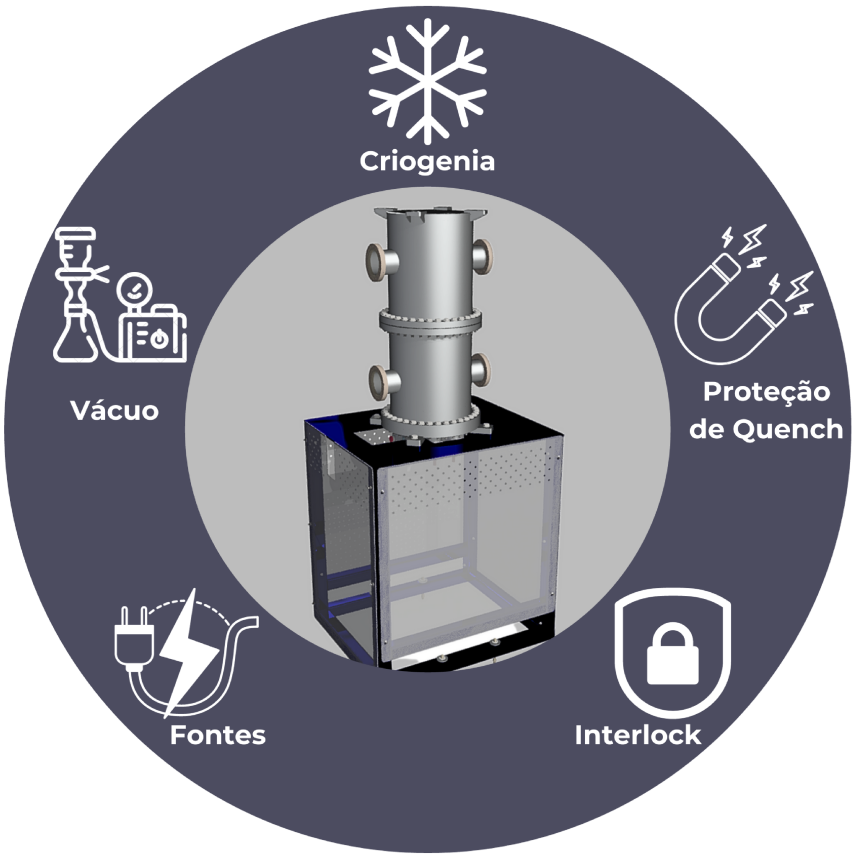


Fonte: Adaptado de Grupo de Análise e Projeto de Sistemas (APS) - CNPEM (2024)

Em geral, o processo de validação de bobinas ou materiais começa com a fixação da amostra no interior do dispositivo, na região destacada na Figura 2. Em seguida, o dispositivo é fechado e inicia-se o processo de resfriamento em vácuo, que, em regime permanente, atinge temperaturas em torno de 4 Kelvin na região do material a ser testado. A seguir, dadas as condições adequadas de operação, dá-se início aos testes. Para as bobinas supercondutoras, uma corrente elétrica conhecida é aplicada ao material através de condutores especiais denominados *current leads*. Dessa forma, por meio da interação desta corrente com a bobina em estudo, são analisados efeitos como a geração de campo magnético, variações de temperatura, entre outros. Além disso, através de diferentes montagens é possível realizar diversos outros testes em temperaturas criogênicas, como testes de resistências de contato, vedação de gases, condutividade térmica, entre outros.

Para que todo este processo ocorra de forma adequada e segura, o criostato vertical conta com cinco principais subsistemas, ilustrados na Figura 3.

Figura 3 – Subsistemas relacionados ao criostato vertical



Fonte: O autor (2024)

Ao longo dos testes, diversas grandezas relativas a cada um deles devem ser monitoradas para garantir a integridade do dispositivo e a segurança dos usuários, como limiares de pressão, temperaturas, tensões e correntes. Além disso, exclusivamente para supercondutores, monitora-se a ocorrência do quench — a transição de uma região do material supercondutor para o estado "normal condutor" —, que pode trazer danos ou até a destruição do material em teste caso não seja propriamente administrado (KATE, 2013).

Nesse sentido, a gestão eficaz desses subsistemas e suas interdependências torna-se um desafio. A falta de uma estratégia de controle adequada pode impossibilitar a operação do sistema de forma eficiente, colocando em risco tanto a integridade do dispositivo quanto a segurança dos usuários. Assim, desenvolveu-se uma arquitetura de controle distribuído para o criostato vertical a fim de solucionar este problema. Associada à interface gráfica desenvolvida, essa arquitetura permite a eficiência, segurança e robustez dos testes relacionados ao SWLS, facilitando o desenvolvimento do projeto e a obtenção de resultados. Este trabalho aborda os aspectos relacionados à integração dos subsistemas, detalhes da arquitetura de controle definida, estratégias adotadas e a interface web de operação desenvolvida.

1.1. Justificativas

Além de permitir a comunicação entre diferentes dispositivos, a integração de sistemas viabiliza o desenvolvimento de interfaces de controle unificadas e torna a operação mais prática, o que facilita a análise e otimiza os procedimentos experimentais. Para o criostato vertical, a adoção de um sistema de controle distribuído traz diversas vantagens, dentre elas:

- **Confiabilidade:** Com o controle distribuído são reduzidos os pontos únicos de falha (SPOFs), ou seja, partes do sistema cuja falha implicaria na queda de todo o restante. Em outras palavras, com a presença de unidades de controle independentes é possível reduzir os pontos onde a falha de um subsistema implica diretamente na falha dos demais, aumentando a confiabilidade da aplicação. Vale ressaltar, contudo, que do ponto de vista da operação, o criostato vertical possui sistemas intrinsecamente interdependentes (como a dependência direta do sistema de criogenia em relação ao sistema de vácuo, por exemplo). Para contornar esse problema, os equipamentos relacionados aos SPOFs intrínsecos devem possuir boa robusteza para garantir a disponibilidade necessária do dispositivo.
- **Flexibilidade:** Com esta arquitetura, é possível adaptar os sistemas de controle de forma independente, ajustando-os às demandas específicas do processo como um todo;
- **Modularidade:** Em um sistema distribuído, cada subsistema pode ser tratado como um módulo independente, simplificando o processo de manutenção, adição ou substituição de equipamentos, caso seja necessário.

1.2. Objetivos

O objetivo deste trabalho é detalhar o processo de desenvolvimento da arquitetura de controle distribuído definida para o criostato vertical projetado e desenvolvido entre os anos de 2022 e 2023 para realização de testes relacionados ao projeto do SWLS. Ao longo do estudo são detalhados os protocolos e estratégias de comunicação utilizados para cada equipamento relacionado ao dispositivo, as estratégias de software adotadas para a integração dos sistemas, frameworks, bibliotecas e linguagens utilizados para a interface de operação, bem como os resultados obtidos.

1.2.1. Objetivos Específicos

- Detalhar os diferentes protocolos de comunicação utilizados para cada equipamento relacionado ao criostato vertical, evidenciando a estratégia definida para a integração

de cada um deles com o sistema completo;

- Detalhar o desenvolvimento das camadas de software de comunicação baseadas no *toolkit* EPICS (Experimental Physics and Industrial Control System);
- Evidenciar as estratégias, linguagens e bibliotecas utilizadas no desenvolvimento da interface de operação web;
- Apresentar a interface de operação em comunicação com o sistema real e em operação, exibindo as detecções de falha e alarmes implementados.

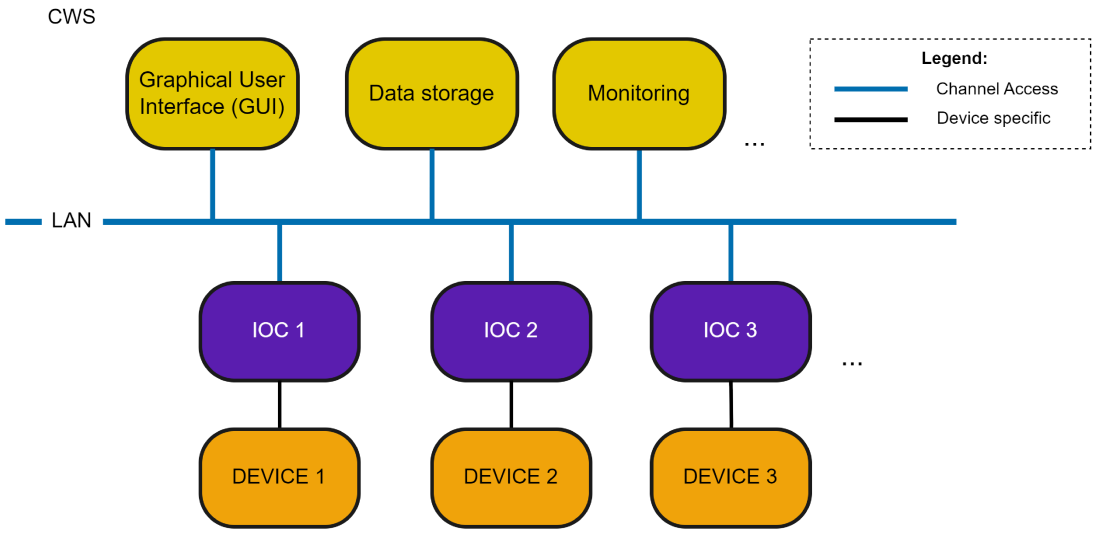
2. Referenciais Teóricos

2.1. EPICS - Experimental Physics and Industrial Control System

Um dos principais componentes de software utilizados no contexto da engenharia do Sirius e em diversos outros laboratórios (EPICS, 2024b) é o EPICS (Experimental Physics and Industrial Control System). Desenvolvido para integração e controle de sistemas distribuídos, o EPICS é um conjunto de ferramentas de código aberto, escrito majoritariamente nas linguagens C e C++. Amplamente difundido na comunidade de experimentos científicos de grande porte, é comumente utilizado para aquisição de dados, monitoramento e controle (EPICS, 2024a).

De forma geral, esse *toolkit* opera através dos modelos cliente/servidor e produtor/consumidor, possibilitando a comunicação entre computadores, dispositivos e sistemas. A Figura 4 representa a arquitetura básica do EPICS 3, versão utilizada ao longo deste trabalho.

Figura 4 – Arquitetura do EPICS 3 representada em diagrama de blocos



Fonte: O autor (2024)

Além dos dispositivos com os quais se deseja estabelecer a comunicação, a arquitetura exibida na Figura 4 pode ser dividida em três elementos principais:

- **IOCs (Input/Output Controllers)**
  Os IOCs (Input/Output Controllers) são unidades de software que funcionam como servidores de entradas e saídas dos dispositivos. Em suas bases de dados os IOCs são formados por múltiplos *records*, estruturas de vários tipos responsáveis por receber ou enviar as informações aos equipamentos, geralmente através de *pooling* ou interrupções. Os nomes atribuídos a essas estruturas e seus campos compõem as chamadas PVs (Variáveis de Processo), nomes únicos por meio das quais são

acessadas as informações ou comandos do dispositivo de interesse. Após a realização da tarefa de leitura ou escrita através do protocolo específico do dispositivo, os IOCs são responsáveis por publicar essas informações na rede local por meio do protocolo Channel Access, descrito na seção 2.1.1. Devido a seu baixo custo de processamento, os IOCs podem ser executados na maioria das plataformas computacionais com suporte a comunicação de rede e bases de dados, podendo o mesmo host executar simultaneamente múltiplos IOCs ligados a diferentes dispositivos, a depender apenas de sua capacidade de processamento (EPICS, 2019c). Alguns exemplos de hosts utilizados no CNPEM são sistemas embarcados como RaspberryPis, BeagleBones, e principalmente computadores comuns e servidores.

- **LAN (Local Area Network)**
  Trata-se da rede de comunicação por meio da qual os IOCs se comunicam com os clientes. Por meio do acesso à LAN, os clientes podem acessar as PVs de todos os IOCs visíveis naquela rede através do Channel Access (ver seção 2.1.1). Nesta configuração, os IOCs são servidores que podem ser acessados por múltiplos clientes, assim como os clientes podem acessar múltiplos IOCs.

- **CWS (Client Workstations)**
  As CWS são os clientes dos IOCs. A partir delas é possível acessar as PVs de interesse, colhendo ou enviando informações, e inseri-las em interfaces gráficas para monitoramento, interfaces de comando, bancos de dados, entre outros. Do ponto de vista da aplicação, é possível combinar a arquitetura descrita com diversas outras ferramentas de desenvolvimento, o que viabiliza a entrega de soluções rápidas e escaláveis para controle de diversos sistemas.

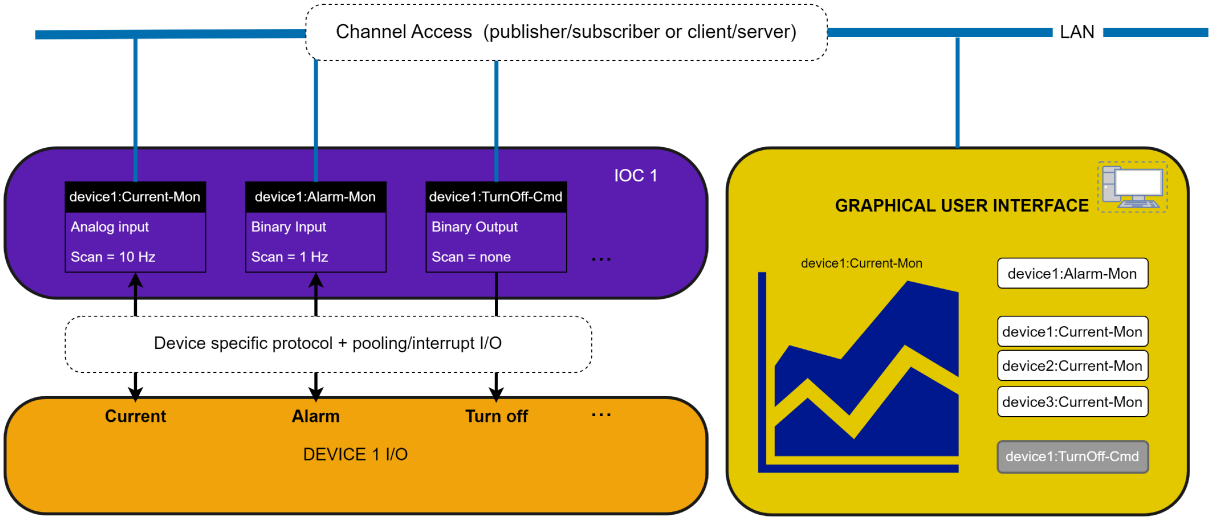
A Figura 5 mostra em maiores detalhes a relação entre as variáveis dos dispositivos, os records, as PVs e as interfaces gráficas de usuário (GUI).

2.1.1. O protocolo Channel Access

No contexto das aplicações baseadas no EPICS 3, o principal protocolo de comunicação entre os servidores (IOCs) e os clientes (CWSs) é o Channel Access, ou EPICS Channel Access (CA). Com as informações providas em EPICS (2019a), é possível elencar as principais características deste protocolo conforme abaixo:

- **Arquitetura:** Em primeira instância, o Channel Access segue a arquitetura cliente/-servidor. Nesta configuração, os servidores são os IOCs, e os clientes são as CWS, conforme descrito na seção 2.1. O modelo de comunicação é baseado em pergunta e resposta, onde o cliente solicita a informação ou escrita desejados, e o servidor CA é responsável por tomar a ação correspondente. Além disso, a operação do

Figura 5 – Detalhes da relação entre IOCs, dispositivos e GUIs



Fonte: O autor (2024)

tipo *monitor*, detalhada a seguir, introduz também o modelo produtor/consumidor (publisher/subscriber), onde o cliente pode se inscrever e "monitorar" mudanças de forma assíncrona.

- **Formato de conexão:** O cliente localiza o servidor CA e inicia a comunicação utilizando UDP/IP e TCP/IP, criando assim um **circuito virtual**, conforme descrito no tópico **Fluxo de Informações** (a seguir). Segundo EPICS (2019a), o circuito virtual é uma "conexão TCP reutilizável por meio da qual todas as PVs fornecidas pelo servidor podem ser levadas ao cliente". A partir disso, dois conceitos são importantes para a troca de informações: o canal e a PV.
  - **PV (Variável de processo):** Trata-se de um nome por meio do qual pode-se acessar a informação desejada utilizando o Channel Access. Em uma rede de controle, cada PV deve ser única e deve ser servida por um único IOC. No exemplo ilustrado na Figura 5, as PVs servidas pelo "IOC 1" seriam device1:Current-Mon, device1:Alarm-Mon, device1:TurnOff-Cmd, contendo as respectivas informações do "DEVICE 1".
  - **Canal:** O canal é a associação em baixo nível entre um circuito virtual e uma PV específica. Em termos gerais, o canal é uma forma de abstração do nome da PV, que é uma string, para identificadores inteiros. Um circuito virtual pode ter múltiplos canais, que, assim como as PVs, são únicos em uma mesma rede de controle.
- **Operações do cliente:** No contexto do Channel Access, três operações são possíveis:
  - **Leitura:** O cliente solicita do servidor o valor atual de uma PV.
  - **Escrita:** O cliente envia novos valores ao servidor, atualizando a PV.

  - **Monitoramento:** Através da operação do tipo ***monitor***, o cliente pode se inscrever em um canal específico, através do nome da PV, e receber notificações quando o valor mudar. Devido à sua característica assíncrona, esta operação se assemelha ao formato produtor/consumidor.

  Note que, para todos os casos, desde que esteja conectado à mesma rede, basta que o cliente saiba o nome da PV de interesse. Isso pode se tornar uma grande vantagem em sistemas de maior escala, já que os endereços de IP e portas onde os IOCs estão hospedados deixam de ser uma preocupação do ponto de vista de quem consome as informações.

- **Fluxo de informações:** De forma geral, ao iniciar uma conexão, o cliente envia um *broadcast* na rede local (podendo também ser configurado para outra subrede) com o nome da PV de interesse. Os servidores CA, por sua vez, são responsáveis por ouvir por broadcasts de pesquisa através de um socket UDP, e retornar uma mensagem do tipo *unicast* para a origem da solicitação em caso de correspondência. Essa mensagem carrega o número da porta de um socket TCP, também aberto pelo servidor CA, através da qual é estabelecida a criação do circuito virtual mencionado anteriormente. Ao passo em que este circuito é aberto, o canal correspondente é criado, e então todas as operações de leitura, escrita e monitoramento da respectiva PV são feitas através dele até que ele seja fechado (EPICS, 2019a).

- **Administração de erros e segurança:** O Channel Access possui mecanismos de administração de erros e de segurança para garantir uma comunicação confiável entre cliente e servidor. Os principais pontos podem ser destacados abaixo:

  - **Códigos de erro:** O EPICS CA utiliza códigos de erro numéricos para transmitir informações sobre o sucesso ou falha de uma operação. Na documentação oficial do protocolo, referenciada neste documento, é possível encontrar a tabela contendo todos os códigos e seus respectivos significados.
  - **Relato Assíncrono:** Erros podem ser relatados de forma assíncrona, especialmente no contexto da operação do tipo *monitor*. Clientes podem receber notificações de erro quando ocorrem problemas, permitindo identificação e resolução rápidas.
  - **Status de Conexão:** Clientes podem verificar o status de sua conexão com um servidor, permitindo reação a mudanças na conectividade. Valores comuns de status de conexão incluem conectado, desconectado ou em estado de erro.
  - **Tratamento de Alarmes:** O EPICS CA suporta o conceito de alarmes, que indicam condições anormais associadas a uma PV. Os alarmes são configurados na definição dos records dos IOCs, e são categorizados com base na gravidade

(por exemplo, menor, maior ou inválido). Clientes podem monitorar alarmes junto com os valores das PVs para responder a condições anormais.

- **Tratamento de Timeout:** Erros de timeout podem ocorrer quando um cliente espera muito tempo por uma resposta de um servidor. Clientes podem implementar o tratamento de timeout para lidar com situações em que atrasos na comunicação podem impactar o desempenho do sistema.
- **Logging e Debugging:** O EPICS CA permite o registro e depuração detalhados, auxiliando engenheiros no diagnóstico de problemas de comunicação. Registros podem capturar mensagens de erro, timestamps e detalhes relevantes para facilitar a solução de problemas. Além disso, com o uso de módulos adicionais, é possível registrar por exemplo o IP de origem de comandos de escrita, o que melhora a segurança e facilita a depuração de possíveis problemas.
- **Controle de Acesso:** Através do chamado "Access Security", o EPICS permite o controle de acesso às PVs, restringindo operações de leitura e escrita. Os administradores podem configurar políticas de acesso a cada PV, definindo permissões específicas para diferentes usuários ou grupos de usuários de acordo com os IPs de origem das conexões. Vale ressaltar, contudo, que essa segurança deve ser utilizada apenas para prevenir manipulações não autorizadas ou não intencionais provenientes da mesma rede do IOC, não protegendo o sistema contra ataques maliciosos ou mais sofisticados. De acordo com EPICS (2019b), métodos convencionais de segurança de redes ainda devem ser utilizados para o controle de acesso à rede onde o IOC é hospedado.

2.1.2. Módulos adicionais: asynDriver e StreamDevice

Módulos adicionais ao EPICS são desenvolvidos para estender as capacidades do sistema principal, fornecendo funcionalidades específicas para diferentes necessidades de controle e aquisição de dados. Dentre diversos módulos, um dos mais utilizados no contexto da aplicação em estudo são o asynDriver e o StreamDevice.

O Asyn é um módulo que permite o desenvolvimento de drivers de comunicação entre os IOCs EPICS e dispositivos externos com protocolos específicos em baixo nível. A partir de seu uso é possível a troca de informações com uma ampla variedade de dispositivos, utilizando interfaces como GPIB, RS-232, RS-485, USB, Ethernet, entre outras (EPICS, 2018). O StreamDevice, por sua vez, é um módulo suplementar ao Asyn, utilizado para facilitar a integração de dispositivos de aquisição de dados baseados em envio e recebimento de mensagens como uma sequência de caracteres (PSI, 2018). A partir de um arquivo de protocolo definido junto ao IOC, é possível especificar o formato de envio e recebimento de strings do dispositivo de interesse, criando drivers de comunicação associados diretamente às PVs.

2.2. Linguagens e ferramentas

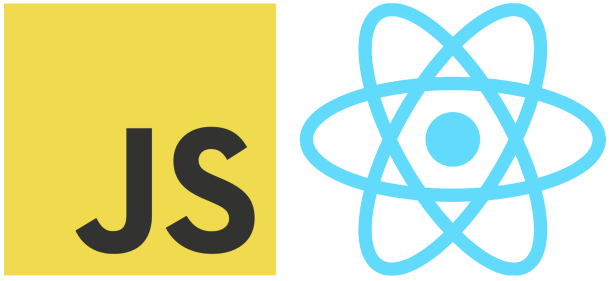
Nesta seção, será detalhada a linguagem utilizada para a elaboração da interface de usuário, além de ferramentas de software utilizados para o desenvolvimento das aplicações tanto no contexto de alto nível quanto na produção dos IOCs.

2.2.1. JavaScript + React

Fundado em 1995, o JavaScript (JS) é uma linguagem de alto nível, interpretada e orientada a objetos. Utilizada principalmente para desenvolvimento web, ela permite a criação de interfaces interativas e dinâmicas em navegadores. No contexto do *front-end*, o JavaScript roda no *client side* da web, o que permite a definição de comportamentos de uma página web a partir da occorência de um evento, como um clique ou o digitar de alguma tecla, por exemplo. Os padrões para a linguagem são definidos pela ECMAScript Language Specification (ECMA-262) e a ECMAScript Internationalization API specification (ECMA-402) (FOUNDATION, 2024).

O React, por sua vez, é uma das diversas bibliotecas que podem ser associadas ao JS para a construção de interfaces de usuário. Desenvolvido pelo Facebook, o React é especialmente eficiente na criação de componentes reutilizáveis e na gestão eficiente do estado da aplicação, o que facilita a atualização de elementos da interface em tempo real. Suas aplicações incluem o desenvolvimento de interfaces de usuário complexas, como em aplicações de página única (SPA), onde oferece um modelo de programação declarativo e eficiente para construir UIs interativas e escaláveis (META, 2024). Exemplos notáveis de aplicações React incluem o Facebook, Instagram e WhatsApp.

Figura 6 – Logomarcas Javascript e React



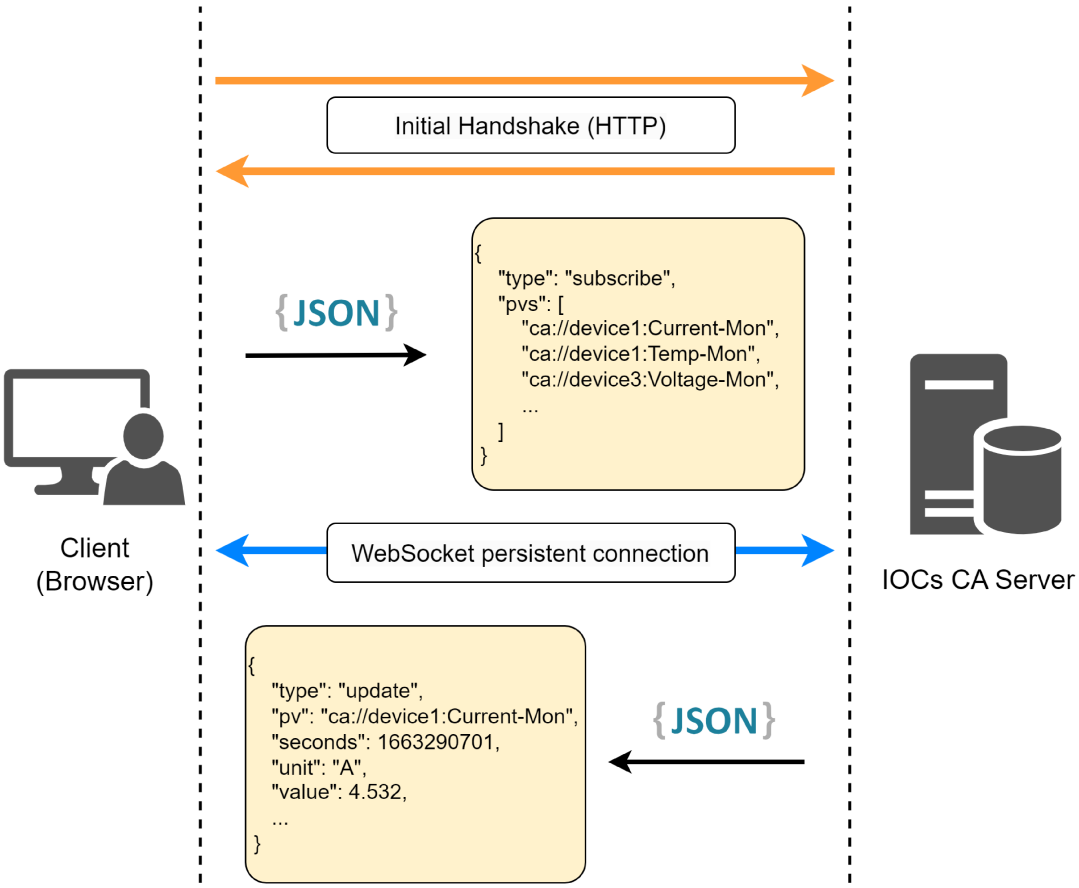
Fonte: (FOUNDATION, 2024; META, 2024)

2.2.2. PVWS

O PVWS (PV Web Socket) é uma ferramenta que oferece uma interface de comunicação baseada em WebSocket para o EPICS, permitindo a leitura e escrita de PVs de forma eficiente e em tempo real através de aplicações web (LABORATORY, s.d.). Uma vez aberta a comunicação, basta o envio de subscrições ao servidor com as PVs

de interesse, e o PVWS se encarrega de adquirir as informações através do protocolo especificado (neste caso o Channel Access) e enviar as atualizações para o cliente web. Nesta ferramenta as mensagens são enviadas e recebidas no formato JSON, podendo ser do tipo "subscribe", "unsubscribe" ou "write" para as mensagens enviadas pelo cliente e "update" para mensagens recebidas. A Figura 7 ilustra o funcionamento do módulo. Seu código-fonte está disponível no GitHub para colaboração e desenvolvimento contínuo.

Figura 7 – Funcionamento do PVWS



Fonte: O autor (2024)

2.2.3. Docker

Contêineres são uma tecnologia poderosa usada para encapsular aplicativos e suas dependências, fornecendo um ambiente isolado e consistente. Eles são independentes do sistema operacional do host, o que facilita a portabilidade das aplicações, permitindo que elas sejam executadas de maneira uniforme em diferentes ambientes ou máquinas. Isso contrasta com as máquinas virtuais, que requerem a replicação de um sistema operacional completo para cada aplicação, resultando em maior uso de recursos.

Ao adotar contêineres, a necessidade de instalar e configurar manualmente dependências é eliminada. Isso é especialmente benéfico no contexto do desenvolvimento, onde o tempo entre a escrita do código e sua implantação é significativamente reduzido.

Docker, uma ferramenta de conteinerização escolhida por muitos, exemplifica esses benefícios. Ele permite a criação, distribuição e execução de aplicativos em contêineres de maneira eficiente (DOCKER, 2024b).

Figura 8 – Logomarca Docker

Fonte: (DOCKER, 2024b)

Além disso, quando integrado a outras ferramentas, a tecnologia de contêineres oferece um ambiente altamente escalável e gerenciável para aplicações distribuídas. Essa abordagem modular e baseada em contêineres promove a flexibilidade, permitindo respostas rápidas às demandas de escalabilidade ou mudanças de requisitos dos softwares. O Docker, por exemplo, é comumente utilizado em conjunto com o docker-compose, uma ferramenta que permite a definição e execução de aplicativos multi-contêiner em um único arquivo de configuração (DOCKER, 2024a).

2.2.4. NGINX

O NGINX (*engine X*) é um servidor web de código aberto conhecido por sua alta performance, escalabilidade e capacidade de lidar com grandes volumes de tráfego.

Figura 9 – Logomarca NGINX

Fonte: (F5, s.d.)

Ele pode ser utilizado como um servidor web para hospedar sites e aplicativos da web, bem como um proxy reverso para distribuir o tráfego entre servidores de back-end. Além disso, o NGINX também é comumente usado como um balanceador de carga para distribuir solicitações entre vários servidores para otimizar a utilização dos recursos e melhorar a disponibilidade e confiabilidade dos aplicativos (F5, s.d.).

3. Metodologia e Desenvolvimento

3.1. Características do sistema

Inicialmente, para a elaboração do sistema proposto, é necessária a compreensão das características do dispositivo e de seus componentes. A Figura 10 mostra o criostato vertical em escala real. Conforme mencionado no Capítulo 1, sua operação é composta por cinco principais subsistemas. Os aspectos mais relevantes de cada um deles do ponto de vista dos sistemas de controle e automação são mencionados a seguir.

Figura 10 – O criostato vertical

Fonte: O autor (2024)

- **Criogenia**: Por meio da atuação do compressor Sumitomo CSW-71 e do Cryocooler RDK-415D2 4K, este sistema é responsável por refrigerar o interior do dispositivo às temperaturas criogênicas citadas. Para a aquisição e acompanhamento da evolução da temperatura em diferentes regiões, utiliza-se o monitor de temperatura Lakeshore 224, que possui canais para até 12 sensores e dois contatos para sinalização externa de alarmes. A comunicação deste equipamento se dá via Ethernet, com um protocolo definido pelo próprio fabricante e detalhado no manual do equipamento. O compressor, por sua vez, disponibiliza sinais de status e possibilita o comando remoto por meio de contatos auxiliares disponíveis a partir de um conector externo.

Figura 11 – Principais componentes do sistema criogênico

Fonte: (CRYOSRV, 2020; CRYOGENICS, 2024; CRYOTRONICS, 2023)

- **Vácuo**: Para que o sistema criogênico opere adequadamente, é necessário que todo o interior do criostato seja submetido à condição de alto vácuo. Dentre outros fatores, a eliminação de gases no ambiente de testes é necessária principalmente para impedir as trocas de calor por convecção, maximizar o isolamento térmico e evitar a condensação de gases durante a operação do dispositivo. Para isso, utiliza-se uma bomba de vácuo em conjunto com o medidor e controlador HPS 937A, da MKS Instruments. Além da aquisição e controle das leituras dos sensores e emissão de alarmes caso necessário, este dispositivo conta com relés de sinalização e utiliza um protocolo proprietário de comunicação, com meio físico RS485, sendo ligado à rede local através de um conversor Serial - Ethernet.

Figura 12 – Medidor e controlador HPS 937A - MKS Instruments

Fonte: (ARTISAN, s.d.)

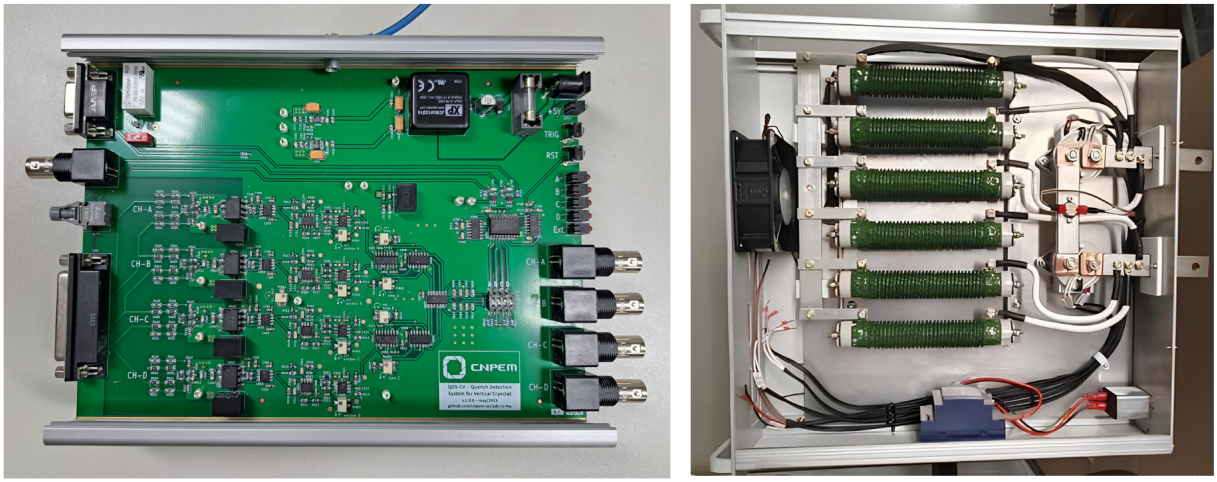
- **Fonte de corrente**: Considerando a principal aplicação para a qual este criostato foi projetado - o teste de bobinas supercondutoras - as fontes de corrente desempenham um papel crucial, já que, além das características construtivas da bobina a ser testada, o campo magnético gerado pode ser regulado através do controle da corrente elétrica aplicada ao enrolamento. Assim, o sistema de fontes é utilizado para garantir o controle do campo magnético a partir da definição de rampas e setpoints de corrente desejados. Para a produção do campo especificado para o projeto do SWLS são utilizadas correntes de até 300A, fornecidas através de um conversor ressonante projetado pelo Grupo Conversores de Potência (COP) do CNPEM (CONTESINI et al., 2023), ilustrado na Figura 13. Este conversor possui um controlador dedicado que conta com relés de sinalização, entradas para interlock externo e uma interface RS485. Para a comunicação, utiliza-se o protocolo BSMP (Basic Small Messages Protocol), também desenvolvido no CNPEM e amplamente utilizado nas fontes de corrente utilizadas no Sirius (LNLS, 2018). A conexão do controlador da fonte à rede é feita através de uma Beaglebone atuando como conversor Serial-Ethernet.

Figura 13 – Sistema de fontes montado em bastidor

Fonte: Grupo Conversores de Potência (COP) - CNPEM (2024)

- **Detecção de quench e extração de energia**: O quench é um fenômeno físico descrito pela transição de uma região de um material do estado supercondutor para o "normal-condutor". Quando isso ocorre, a maior parte da energia armazenada no material supercondutor - neste caso a bobina -, passa a ser dissipada em forma de calor, evento pode levar ao sobreaquecimento e trazer danos ou até a sua destruição caso não seja propriamente controlado. Por esse motivo, é necessário um sistema de proteção que detecte a ocorrência de um quench e extraia a energia armazenada o mais rápido possível, evitando danos ao material em teste ou ao equipamento como um todo (KATE, 2013). No criostato vertical, a extração é feita redirecionando a energia da bobina a um banco de resistores de descarga através do acionamento de um contator de potência. Devido à sua natureza crítica, este subsistema conta com uma eletrônica dedicada, sendo o hardware e arquitetura projetados integralmente no CNPEM. A Figura 14 mostra a eletrônica de detecção e a gaveta de extração de energia. O sistema disponibiliza saídas digitais para sinalização e uma entrada digital para trigger externo. Para as análises, a aquisição de dados é feita de forma indireta através do osciloscópio Tektronix TDS3014B, por meio do barramento Ethernet e protocolo também proprietário.

Figura 14 – Sistema de detecção de quench com 4 canais e gaveta de extração



Fonte: Grupo de Eletrônica e Microeletrônica (EMI), Grupo Conversores de Potência (COP) - CNPEM (2024)

- **Sistema de intertravamento**: Para garantir a segurança tanto dos equipamentos quanto dos usuários durante a operação, é necessária a atuação de um sistema que gerencie os eventos e alarmes de cada subsistema, coordenando desligamentos quando necessário e impedindo a operação em condições não ideais a nível de hardware. Dessa forma, as sinalizações digitais e alarmes descritos anteriormente são conectadas ao painel de automação que, a partir de uma lógica de intertravamento definida e executada em um Controlador Lógico Programável (CLP), permite ou não a operação de cada subsistema. Para o criostato vertical, o CLP Rockwell Micro820 (AUTOMATION, s.d.) cumpre este papel. Para supervisão, a comunicação com o controlador é feita através do protocolo Modbus TCP/IP. A Figura 15 mostra o painel de intertravamento acoplado ao mini-rack onde são montados os equipamentos de controle e instrumentação mencionados anteriormente. Ao lado é exibida a outra face do mini-rack, onde podem ser vistos os equipamentos montados em campo, com exceção do compressor, fontes e gaveta de extração, que são instalados em ambientes separados devido à suas dimensões e características específicas. É possível ver o criostato vertical da Figura 10 ao fundo da segunda imagem.

Figura 15 – Painel de intertravamento acoplado ao mini-rack de instrumentação

Fonte: Grupo de Automação e Robótica (ARO) - CNPEM (2024)

Assim, considerando as características particulares de cada dispositivo e suas interdependências, a necessidade de um sistema de controle que permita a gestão dos subsistemas e facilite a operação do criostato vertical torna-se essencial. A seguir, serão detalhados aspectos relacionados à integração dos subsistemas descritos, passando pela arquitetura de controle definida, estratégias adotadas e a interface de operação desenvolvida.

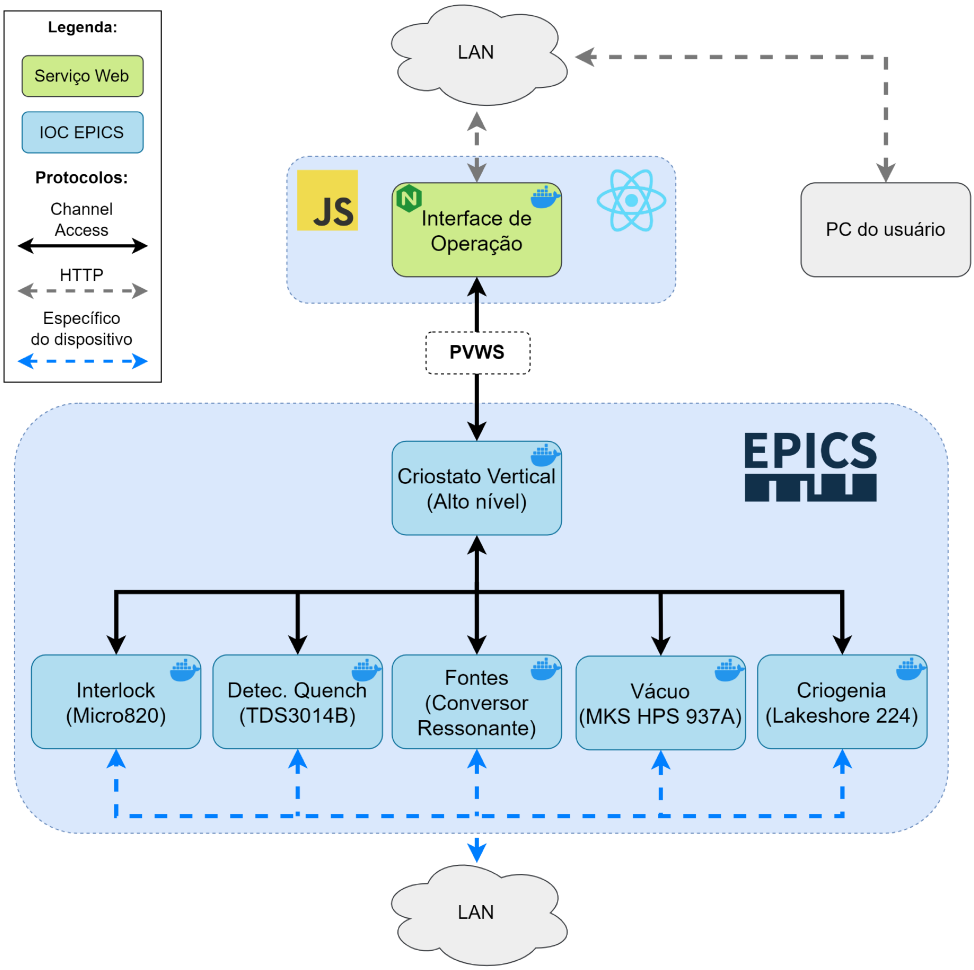
3.2. Definição da arquitetura

No contexto da engenharia no CNPEM, grande parte dos sistemas de controle distribuído são baseados em EPICS (Experimental Physics and Industrial Control System), descrito na seção 2.1. Dessa forma, visando manter o padrão das demais aplicações e considerando as características do EPICS apresentadas, optou-se por utilizar o mesmo framework, baseando as unidades de controle em IOCs tratados como módulos individuais e integrados através do Channel Access. Além disso, optou-se também por definir um IOC de alto nível, responsável por intermediar a comunicação entre os IOCs dos dispositivos e a interface de operação a ser elaborada, executando subrotinas de software quando necessário.

Em seguida, foi necessário definir o tipo de interface de operação a ser utilizado. Para isso, foram consideradas principalmente as durações de alguns dos processos (como o resfriamento, por exemplo), que podem levar horas até atingir o regime permanente. Desse ponto de vista, seria conveniente aos usuários poder monitorar e comandar o dispositivo

remotamente, dadas as devidas autenticações, sem a necessidade de se deslocar ou instalar programas e ferramentas adicionais. Assim, optou-se pela adoção de uma interface de operação web, facilmente acessível na rede local através de navegadores. Do ponto de vista da comunicação, para obter as informações dos IOCs e disponibilizá-los na interface web foi adotado o uso do módulo PVWS (LABORATORY, s.d.), detalhado na seção 2.2.2. Por fim, para eliminar dependências do sistema e isolar o ambiente de cada software de seu host, foi adotada a inserção dos IOCs e o serviço web em seus próprios contêineres Docker, obtendo, assim, sistemas portáveis e de fácil instalação. Ao final, a arquitetura de software definida pode ser representada pelo diagrama da Figura 16.

Figura 16 – Arquitetura de software definida para o criostato vertical



Fonte: O autor (2024)

3.3. Desenvolvimento dos IOCs

Conforme descrito na seção 3.1, muitos dos instrumentos relacionados aos subsistemas possuem protocolo de comunicação definido pelo próprio fabricante, o que exige a definição dos comandos e respostas específicos esperados para cada um deles. Como exemplo, a Tabela 1 mostra a tabela resumida de comandos ASCII para o monitor de

temperaturas criogênicas, Lakeshore 224, extraída do manual do equipamento.

Tabela 1 – Comandos e queries ASCII para o Lakeshore 224

Command	Function	Page	Command	Function	Page
✱CLS	Clear Interface Cmd	81	IEEE	IEEE-488 Parameter Cmd	87
✱ESE	Event Status Enable Register Cmd	81	IEEE?	IEEE-488 Interface Parameter Query	88
✱ESE?	Event Status Enable Register Query	81	INCRV	Input Curve Number Cmd	88
✱ESR?	Standard Event Status Register Query	81	INCRV?	Input Curve Number Query	88
✱IDN?	Identification Query	82	INNAME	Sensor Input Name Cmd	88
✱OPC	Operation Complete Cmd	82	INNAME?	Sensor Input Name Query	88
✱OPC?	Operation Complete Query	82	INTSEL	Interface Select Cmd	89
✱RST	Reset Instrument Cmd	82	INTSEL?	Interface Select Query	89
✱SRE	Service Request Enable Register Cmd	82	INTYPE	Input Type Parameter Cmd	89
✱SRE?	Service Request Enable Register Query	82	INTYPE?	Input Type Parameter Query	90
✱STB?	Status Byte Query	83	KRDG?	Kelvin Reading Query	90
✱TST?	Self-Test Query	83	LEDS	Front Panel LEDS Cmd	90
✱WAI	Wait-to-Continue Cmd	83	LEDS?	Front Panel LEDS Query	90
ALARM	Input Alarm Parameter Cmd	83	LOCK	Front Panel Keyboard Lock Cmd	90
ALARM?	Input Alarm Parameter Query	84	LOCK?	Front Panel Keyboard Lock Query	91
ALARMST?	Input Alarm Status Query	84	MDAT?	Minimum/Maximum Data Query	91
ALMRST	Reset Alarm Status Cmd	84	MNMXRST	Minimum and Maximum Function Reset Cmd	91
BRIGT	Display Contrast Cmd	84	MODE	Remote Interface Mode Cmd	91
BRIGT?	Display Contrast Query	84	MODE?	Remote Interface Mode Query	91
CRDG?	Celsius Reading Query	84	NET	Network Settings Cmd	91
CRVDEL	Curve Delete Cmd	85	NET?	Network Settings Query	92
CRVHDR	Curve Header Cmd	85	NETID?	Network Configuration Query	92
CRVHDR?	Curve Header Query	85	OPST?	Operational Status Query	92
CRVPT	Curve Data Point Cmd	85	OPSTE	Operational Status Enable Cmd	92
CRVPT?	Curve Data Point Query	85	OPSTE	Operational Status Enable Query	92
DFLT	Factory Defaults Cmd	86	OPSTR?	Operational Status Register Query	93
DIOCUR	Diode Excitation Current Parameter Cmd	86	RDGST?	Input Reading Status Query	93
DIOCUR?	Diode Excitation Current Parameter Query	86	RELAY	Relay Control Parameter Cmd	93
DISPFLD	Custom ModeDisplay Field Cmd	86	RELAY?	Relay Control Parameter Query	93
DISPFLD?	Custom Mode Display Field Query	86	RELAYST?	Relay Status Query	93
DISPLAY	Display Setup Cmd	87	SCAL	Generate SoftCal Curve Cmd	94
DISPLAY?	Display Setup Query	87	SRDG?	Sensor Units Input Reading Query	94
FILTER	Input Filter Parameter Cmd	87	WEBLOG	Website Login Parameters	94
FILTER?	Input Filter Parameter Query	87	WEBLOG?	Website Login Parameter Query	94

Fonte: Adaptado de (CRYOTRONICS, 2015)

Além das mensagens a serem enviadas, os manuais fornecem informações detalhadas do formato de resposta para cada mensagem, bem como suas características, conforme ilustra a Figura 17, para um dos comandos do mesmo equipamento.

Figura 17 – Detalhes do comando “KRDG?” para o Lakeshore 224

KRDG?	**Kelvin Reading Query**
Input Format	**KRDG? <input>[term]** a <input> Specifies input to query: A, B, C1–C5, D1–D5
Returned Format	<kelvin value> +nnnnn[term] Or if all inputs are queried: <A value>,<B value>,<C1 value>,<C2 value>,<C3 value>,<C4 value>,<C5 value>,<D1 value>, <D2 value>,<D3 value>,<D4 value,<D5 value>. Format: +nnnnn,+nnnnn,+nnnnn,+nnnnn,+nnnnn,+nnnnn,+nnnnn,+nnnnn,+nnnnn, +nnnnn,+nnnnn,+nnnnn[term]
Remarks	Returns the kelvin reading for a single input or all inputs. <input> specifies which input(s) to query. 0 = all inputs.

Fonte: Adaptado de (CRYOTRONICS, 2015)

Dessa forma, a partir do uso do StreamDevice descrito na seção 2.1.2, é possível criar um arquivo de protocolo que define o formato de envio e recebimento das mensagens de interesse, conforme o Fragmento de Código 1, que mostra alguns comandos da Tabela 1 implementados.

Fragmento de Código 1 – Arquivo de protocolo para o Lakeshore 224

```
1  ##################################################
2  # Protocol file for Lakeshore 224 - ls224.proto
3  ##################################################
4  #INPUTS:A, B, C1-C5, D1-D5
5  ##################################################
6  Terminator = "\r\n";
7  ReplyTimeout = 1500;
8  ##################################################
9  getID                 {out "*IDN?"; in "LSCI,%s";}
10 deviceModel           {in "LSCI,%8c,%*23c,%*s";}
11 deviceSerialNumber    {in "LSCI,%*8c,%23c,%*s";}
12 deviceFirmwareVersion {in "LSCI,%*8c,%*23c,%s";}
13 getInpVoltage         {out "SRDG? \$1"; in "%f";}
14 getRelayStatus        {out "RELAYST? \$1"; in "%d";}
15 getKelvinTemp         {out "KRDG? \$1"; in "%f";}
16 getInpReadingSts      {out "RDGST? \$1"; in "%d";}
17 getInpCrvNumber       {out "INCRV? \$1"; in "%d";}
18 resetAlarms           {out "ALARMRST";}
19 getInputName          {out "INNAME? \$1"; in "%s";}
20 setInputName          {out "INNAME \$1, \"%s\"";}
21 # Continue ...
22
```

Fonte: O autor (2024).

No arquivo de protocolo, a diretiva “out” precede a mensagem ou comando a

ser enviado, e, caso seja esperada uma resposta, a diretiva "in" aponta seu formato. Os símbolos \$1, \$2 e \$n representam o primeiro, segundo e n-ésimo argumento passado na chamada de cada comando. Por exemplo, para o comando "KRDG?" detalhado na Figura 17, deve-se enviar o comando seguido da entrada referente à consulta (A, B, C1-C5 ou D1-D5). Como resposta, espera-se um valor numérico. Assim, no arquivo de protocolo foi definido este comando conforme visto na linha 17 do Fragmento de Código 1. Com o arquivo de protocolo definido, para associar os comandos às PVs do IOC, é possível incluí-los diretamente no arquivo de definição dos records (database):

Fragmento de Código 2 – Arquivo database para Lakeshore 224

```
1  ########################################################
2  # DB file for Lakeshore 224 inputs
3  ########################################################
4  #DEVICE: Device name (PV prefix)
5  #INPUTS ={A, B, C1-C5, D1-D5}
6  #PORT = communication bus ID
7  ########################################################
8  record(ai, "$(DEVICE)$(INPUT)Temp-Mon") {
9    field(DESC, "$(INPUT) Temperature")
10   field(DTYP, "stream")
11   field(EGU, "K")
12   field(SCAN, "5 second")
13   field(INP,  "@ls224.proto getKelvinTemp($(INPUT)) $(PORT)")
14 }
15 record(ai, "$(DEVICE)$(INPUT)InpVolt-Mon") {
16   field(DESC, "get $(INPUT) input voltage")
17   field(DTYP, "stream")
18   field(EGU, "V")
19   field(SCAN, "5 second")
20   field(INP, "@ls224.proto getInpVoltage($(INPUT)) $(PORT)")
21 }
22 record(longin, "$(DEVICE)$(INPUT)InpCrv-Sts") {
23   field(DESC, "get curve number defined for $(INPUT) sensor")
24   field(DTYP, "stream")
25   field(SCAN, "5 second")
26   field(INP, "@ls224.proto getInpCrvNumber($(INPUT)) $(PORT)")
27 }
28 # Continue ...
29
```

Fonte: O autor (2024).

Neste trecho, o primeiro parâmetro passado a cada *record* é o seu tipo, podendo ser de entradas ou saídas analógicas (ai, ao), entradas ou saídas digitais (bi, bo), strings (stringin, stringout), entre diversos outros (EPICS, s.d.). Em seguida, tem-se a definição dos nomes das PVs, que podem conter prefixos de identificação (nesse caso, $(DEVICE),

um nome de identificação do dispositivo ao qual o IOC se comunica, seguido de $(INPUT), representando uma dentre as 12 entradas do dispositivo). Caso o usuáro escolha DEVICE="mydevice", por exemplo, as PVs obtidas seriam "mydevice:ATemp-Mon", "mydevice:BTemp-Mon", "mydevice:C1Temp-Mon", "mydevice:C2Temp-Mon", e assim por diante. As variáveis de inicialização do sistema são definidas nos arquivos de startup do IOC, que não serão objetos de estudo deste trabalho. Ainda nos records, há também a definição de alguns campos:

- **DESC**: Descrição da PV;
- **DTYP**: Tipo de dispositivo com o qual o record deve se comunicar;
- **EGU**: Unidades de engenharia;
- **SCAN**: Período de atualização do valor da PV;
- **INP**: Origem do valor da PV, ou seja, endereço da entrada.

Vale ressaltar também que, além desses, existem diversos campos que determinam diferentes propriedades e comportamentos da PV, e podem ser consultados em EPICS (s.d.).
Dessa forma, por meio da estratégia descrita é possível associar todos os comandos definidos no arquivo de protocolo às PVs de interesse, trazendo forma ao IOC do dispositivo. Através do esforço da equipe de software envolvida no projeto e também do aproveitamento de softwares disponíveis na comunidade, foram implementados os IOCs para o Lakeshore 224, o MKS 937A e o TDS3014B por meio dessa estratégia.

Para o CLP Rockwell, por outro lado, foi utilizado o módulo EPICS Modbus, construído como uma camada suplementar ao asynDriver (RIVERS, 2019) para comunicações com o protocolo Modbus. Para este, após as configurações de conexão Modbus no script de inicialização do IOC, é possível definir as entradas como mostra o Fragmento de Código 3. No exemplo, os campos são parecidos com os anteriores, com a adição dos campos ZNAM (Zero Name) e ONAM (One Name), específicos para variáveis binárias, que associam strings ao estado dos bits. Para o campo INP, seguiu-se o padrão "@asynMask(portName,offset,mask,timeout)drvUser", formato utilizado para comunicações utilizando o módulo EPICS Modbus e definido na documentação do software. Como a mesma conexão é utilizada para todas as PVs, varia-se somente o parâmetro "offset", que representa a posição do bit que se associa à variável de interesse a ser lida na memória do CLP em relação ao início do registrador.

Fragmento de Código 3 – Arquivo database para Rockwell Micro820

```
1  ########################################################
2  # DB file for CLP Interlock Inputs
3  ########################################################
4  # $(DEVICE) - PV prefixes passed on startup
5  # $(DTYP) - PVs Modbus Datatype
6  # $(RO) - Modbus Asyn Config name
7  ########################################################
8  record(bi, "$(DEVICE)VacPressure1Intlk-Mon") {
9      field(DESC, "Vacuum pressure interlock contact")
10     field(DTYP, "$(DTYP)")
11     field(INP, "@asynMask($(RO),0,1,1000) MODBUS_DATA")
12     field(SCAN, "1 second")
13     field(ZNAM, "Off")
14     field(ONAM, "On")
15 }
16 record(bi, "$(DEVICE)CompPressureIntlk-Mon") {
17     field(DESC, "Compressor pressure interlock contact")
18     field(DTYP, "$(DTYP)")
19     field(INP, "@asynMask($(RO),2,1,1000) MODBUS_DATA")
20     field(SCAN, "1 second")
21     field(ZNAM, "Off")
22     field(ONAM, "On")
23 }
24 record(bi,"$(DEVICE)TempCtrlIntlk1-Mon") {
25     field(DESC, "Magnets temperature interlock contact 1")
26     field(DTYP, "$(DTYP)")
27     field(INP, "@asynMask($(RO),5,1,1000) MODBUS_DATA")
28     field(SCAN, "1 second")
29     field(ZNAM, "Off")
30     field(ONAM, "On")
31 }
32 # Continue ...
33
```

Fonte: Grupo de Automação e Robótica - ARO (2024).

Para o IOC da fonte de corrente, por sua vez, uma abordagem diferente foi utilizada. Devido à existência prévia de uma biblioteca em Python para a comunicação com o controlador da fonte utilizando o protocolo BSMP, o PyDRS (CNPEM, 2024), o IOC foi desenvolvido em Python utilizando a biblioteca PCASpy (PSI, 2017). Além da criação de servidores CA, esta biblioteca permite a definição das PVs através de dicionários, como visto no Fragmento de Código 4, e permite a manipulação dos dados e criação de rotinas através de scripts Python comuns. Dessa forma, foi possível reaproveitar o módulo PyDRS, amplamante utilizado nas aplicações relacionadas ao Sirius, otimizando o processo de desenvolvimento da aplicação.

Fragmento de Código 4 – Arquivo database em Python para Conversor Ressonante

```
############## Resonant Converter DB ###############
SCAN = 0.1 # Seconds
#################### Readbacks ####################
RBS = {

  # Previous records ...

  "Alrms-RB": {
    "type": "string",
    "bsmp_id": 33,
    "variable": "ps_alarms",
    "scan": SCAN
  },
  "MeanDCCTCurrent-Mon": {
      "type": "float",
      "unit": "A",
      "prec": "3",
      "bsmp_id": 34,
      "variable": "i_load_mean",
      "scan": SCAN
  },
  "DCCT1Current-Mon": {
      "type": "float",
      "unit": "A",
      "prec": "3",
      "bsmp_id": 35,
      "variable": "i_load_1",
      "scan": SCAN
  },
  "DCCT2Current-Mon": {
      "type": "float",
      "unit": "A",
      "prec": "3",
      "bsmp_id": 36,
      "variable": "i_load_2",
      "scan": SCAN
  },
  # Continue ...
}

```

Fonte: Grupo de Automação e Robótica (ARO) - CNPEM (2024).

Por fim, o IOC de alto nível foi desenvolvido a partir do EPICS base conforme os demais, porém sem a adição de módulos externos de comunicação. Como esta unidade

de controle é responsável apenas pela comunicação com os demais IOCs e a execução de subrotinas, é possível que sua comunicação seja diretamente vinculada apenas ao Channel Access, bastando declarar os nomes das PVs de origem e destino das informações desejadas na definição dos records.

Com o desenvolvimento de cada IOC, foram criadas também imagens Docker para cada um deles, de forma a garantir que as aplicações fossem executadas em um ambiente consistente e de fácil portabilidade. Em posse das imagens, a criação dos contêineres foi feita através de um arquivo docker-compose, facilitando a execução dos IOCs e a passagem de parâmetros de execução de cada sistema, como endereços de IP, portas, entre outros. O Fragmento de Código 5 mostra parte do arquivo utilizado para deploy.

Fragmento de Código 5 – Arquivo docker-compose para execução dos IOCs

```
version: "3.7"
# Vertical Cryostat services specification
services:
  # Temperature monitor
  lakeshore224-ioc:
    stdin_open: true
    tty: true
    image: "gitregistry.cnpem.br/aro/lakeshore224-epics-ioc:v0.1.0"
    network_mode: host
    restart: always
    # Other container and deployment settings ...
  # Vacuum monitor
  mks937a-ioc:
    stdin_open: true
    tty: true
    image: "gitregistry.cnpem.br/aro/mks937a-epics-ioc:v0.1.0"
    network_mode: host
    restart: always
    # Other container and deployment settings ...
  # Power supply
  resonant-converter-ioc:
    stdin_open: true
    tty: true
    image: "gitregistry.cnpem.br/aro/resonant-converter-pcaspy-ioc:v0
    .1.0"
    network_mode: host
    restart: always
    # Other container and deployment settings ...
  # Continue ...

```

Fonte: Grupo de Automação e Robótica (ARO) - CNPEM (2024)

3.4. Desenvolvimento da interface gráfica

A partir da definição dos IOCs e da arquitetura de software, foi possível iniciar o desenvolvimento da interface de operação. Antes do início do desenvolvimento foram realizadas reuniões com os usuários finais para a definição dos requisitos e funcionalidades desejadas. Dentre as funcionalidades mais relevantes, destacam-se:

- **Monitoramento de variáveis**: Possibilidade de visualização em tempo real das variáveis de interesse, como temperaturas, pressões e correntes;
- **Alarmes**: Sinalização visual de alarmes por subsistema, com possibilidade de reconhecimento e reset;
- **Controle de subsistemas**: Capacidade de acionamento e desligamento de subsistemas, como o compressor e o sistema de quench, além do controle completo da fonte de corrente;
- **Configurações de interlock**: Definição dos limiares de intertravamento dos subsistemas de temperatura e vácuo;
- **Histórico de variáveis**: Possibilidade de visualização de gráficos históricos das variáveis monitoradas;

Com base nesses requisitos, a interface foi estruturada em uma página principal, contendo os principais controles e informações, e páginas secundárias para configurações e visualizações mais detalhadas de cada subsistema.

Devido à grande quantidade de informações a serem exibidas, foram criados arquivos JSON para cada página, agrupando os subsistemas e as PVs de interesse em arrays de objetos com valores iniciais. O Fragmento de Código 6 mostra trechos do arquivo JSON utilizado na página principal, onde os subsistemas e as PVs associadas a cada um foram definidos. A partir da leitura desses arquivos na inicialização da aplicação, é possível criar uma estrutura base de objetos para os dados que alimentam a interface e, com o auxílio do PVWS (ver seção 2.2.2), atualizar os respectivos campos de cada objeto com informações dos IOCs em tempo real. Esta estrutura se mostrou eficiente na organização dos dados e na manutenção da interface, permitindo a adição de novas PVs e subsistemas de forma simples e rápida.

Fragmento de Código 6 – Exemplo de arquivo JSON para a página principal

```
{
"subsystems": {
  "intlk":[
    {
      "pv":"UVX-PSRoom:VC-HL:GeneralIntlkStatus-Mon",
      "name": "General Status",
      "value": null,
      "unit": null,
      "type":"bi",
      "text": "loading..."
    },
    {
      "pv":"UVX-PSRoom:VC-HL:EStopBtn-Mon",
      "name": "Emergency Button",
      "value": null,
      "unit": null,
      "type":"bi",
      "text": "loading..."
    },
    ...
  ],
  "temp":[
    {
      "pv":"UVX-PSRoom:VC-HL:TempCtrlIntlk-Mon",
      "name": "Temperature Interlock",
      "value": null,
      "unit": null,
      "type":"bi",
      "text": "loading..."
    },
    {
      "pv":"UVX-PSRoom:VC-HL:LSATemp-Mon",
      "namepv": "UVX-PSRoom:VC-HL:LSAName-RB",
      "name": "Input A",
      "value": null,
      "unit": null,
      "type":"ai",
      "text": "loading..."
    },
    ...
  ],
  "compressor": [ ... ],
  "vacuum": [ ... ],
  ...
  }
}
```

Fonte: Grupo de Automação e Robótica (ARO) - CNPEM (2024)

Por meio do React, foi possível criar componentes reutilizáveis que recebem os objetos JSON como argumentos, gerenciando as renderizações das atualizações em tempo real através do gerenciamento de estados da aplicação. O Fragmento de Código 7 mostra a múltipla utilização do componente "SubsystemCard", criado para a exibição de informações de um subsistema em formato de cartão. Além dos componentes personalizados, foi utilizada também a biblioteca Material-UI (MATERIAL-UI, 2024), que fornece componentes como botões e caixas de diálogo prontos para uso, facilitando a construção da interface.

Fragmento de Código 7 – Exemplo de utilização de componente ReactJS

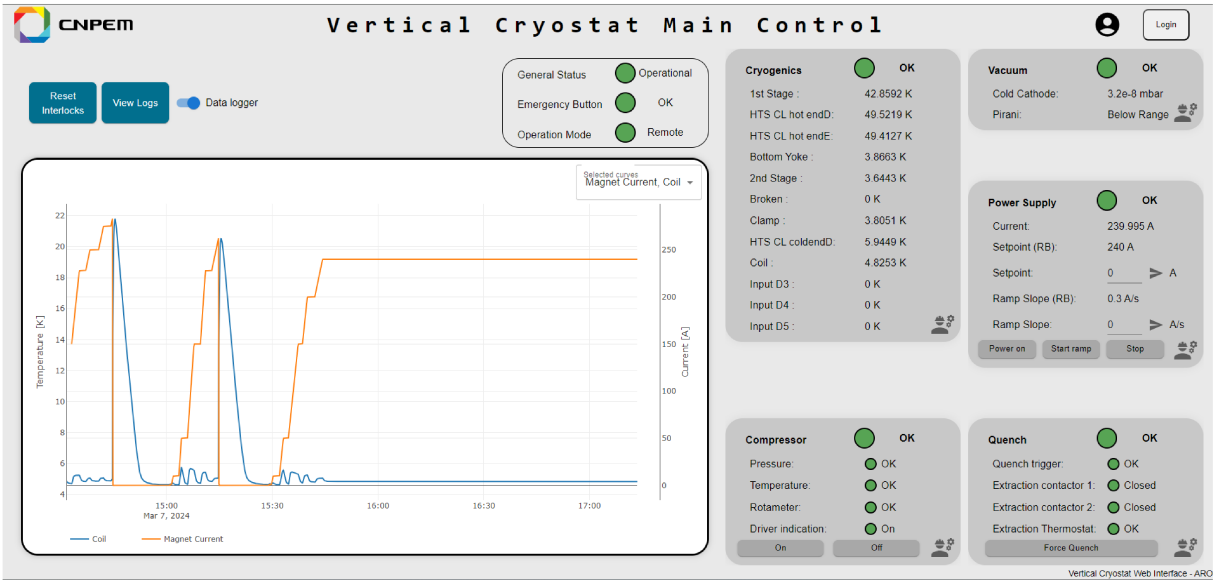
```
1  //Previous code...
2  <div className="column" id="column2">
3    <SubsystemCard
4      cardName="Cryogenics"
5      pvs={this.state.pvs.subsystems.temp}
6      link="/cryogenics"
7    />
8    <SubsystemCard
9      cardName="Compressor"
10     pvs={this.state.pvs.subsystems.compressor}
11     buttons={[
12       {
13         name: "On",
14         func: functions.startCompressor.bind(null, this.props.socket),
15       },
16       {
17         name: "Off",
18         func: functions.stopCompressor.bind(null, this.props.socket),
19       },
20     ]}
21   />
22 </div>
23 <div className="column" id="column3">
24   <SubsystemCard
25     cardName="Vacuum"
26     pvs={this.state.pvs.subsystems.vacuum}
27     link="/vacuum"
28   />
29   //Continue ...
30
```

Fonte: O autor (2024)

Para a geração de gráficos foi utilizada a biblioteca Plotly (PLOTLY, 2024), que permite a criação de gráficos interativos e responsivos. A definição de um objeto de PVs a

serem plotadas permitiu o desenvolvimento de um componente da interface com o qual o usuário pode selecionar uma ou mais variáveis de interesse e visualizar seus valores ao longo do tempo. Foram adicionados também estados do sistema de intertravamento e botões para reset de falhas e acesso aos logs da operação e avisos em caso de desconexão com os subsistemas ou os IOCs. A Figura 18 mostra a tela principal da interface de operação desenvolvida, exibindo os principais controles e informações visualizados durante o teste de uma das bobinas do SWLS. Mais detalhes a respeito dos resultados obtidos serão discutidos na seção 4. É possível perceber a reutilização dos componentes como o "SubsystemCard", por meio da semelhança entre os cartões de cada subsistema. Destaca-se também que os componentes foram criados de forma a permitir a visualização de diferentes tipos de PV, renderizando, por exemplo, leds para variáveis binárias, valores numéricos para variáveis analógicas e campos de entrada para variáveis de controle. A estilização da interface foi feita com o auxílio de CSS e da Material-UI, permitindo a criação de uma interface responsiva e de fácil utilização.

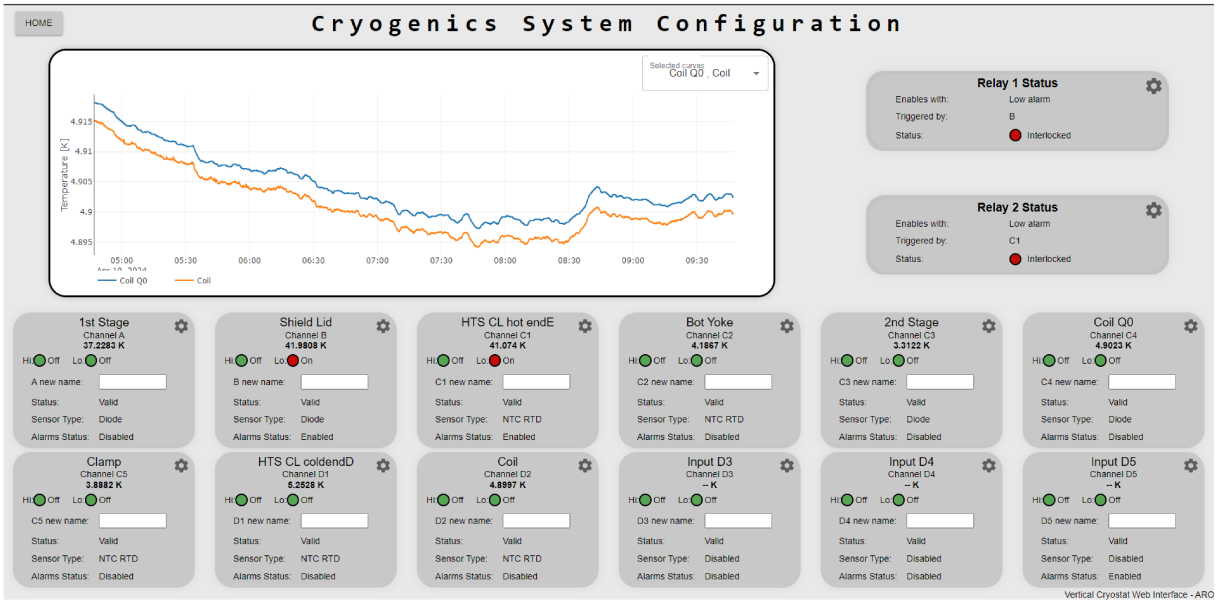
Figura 18 – Página principal da interface de operação



Fonte: Grupo de Automação e Robótica (ARO) - CNPEM (2024)

Além disso, a partir do uso da biblioteca React Router (ROUTER, 2024), foi possível criar rotas para as páginas secundárias, permitindo a navegação entre as diferentes telas da aplicação. Assim, para cada subsistema foram criadas páginas com informações mais detalhadas e configurações específicas, acessíveis através do botão de engenharia no canto inferior direito de cada card, conforme visto na Figura 18. Como exemplo, a Figura 19 mostra a página de controle do sistema criogênico, com a possibilidade de ajuste de limiares de intertravamento e parâmetros específicos da operação, além da visualização de informações adicionais deste subsistema. O acesso às páginas secundárias é protegido por senha, garantindo que somente usuários autorizados possam realizar alterações.

Figura 19 – Página de controle do sistema criogênico



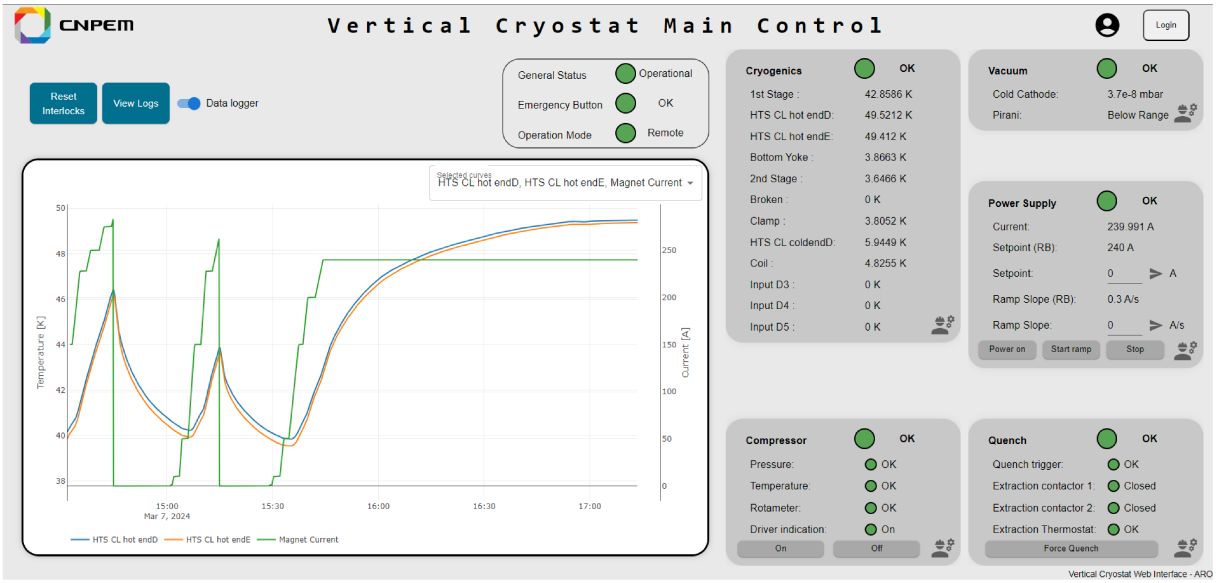
Fonte: Grupo de Automação e Robótica (ARO) - CNPEM (2024)

Por fim, para deploy da aplicação foi utilizado o NGINX, que atua como servidor web, redirecionando as requisições HTTP para o serviço web em execução no contêiner Docker criado para a interface. O NGINX também foi configurado para servir a aplicação em HTTPS, utilizando os certificados gerados para o domínio local do CNPEM. Com isso, a interface de operação foi disponibilizada na rede local do centro através da URL https://verticalcryostat.cnpem.br, permitindo o acesso remoto e seguro à aplicação.

4. Resultados obtidos

Com a disponibilização da interface de controle do criostato vertical foi possível realizar testes de integração e validação dos subsistemas do dispositivo. Na seção anterior, a Figura 18 ilustra a interface durante o teste de uma das bobinas do SWLS, com o sistema totalmente operacional. Na Figura é possível observar as temperaturas criogênicas (entre 3.8 e 50K) medidas em diferentes pontos, as pressões do sistema de vácuo, a corrente entregue pela fonte (240A), bem como o estado dos subsistemas e o histórico de variáveis através dos gráficos. O plot simultâneo de diferentes variáveis, como visto no exemplo, permite que os usuários identifiquem correlações entre as medidas e obtenham informações relevantes sobre o comportamento do sistema. A Figura 20 exibe novamente a interface com o sistema em condição de operação. Nos gráficos, são vistas as temperaturas das *current leads* e a corrente da fonte.

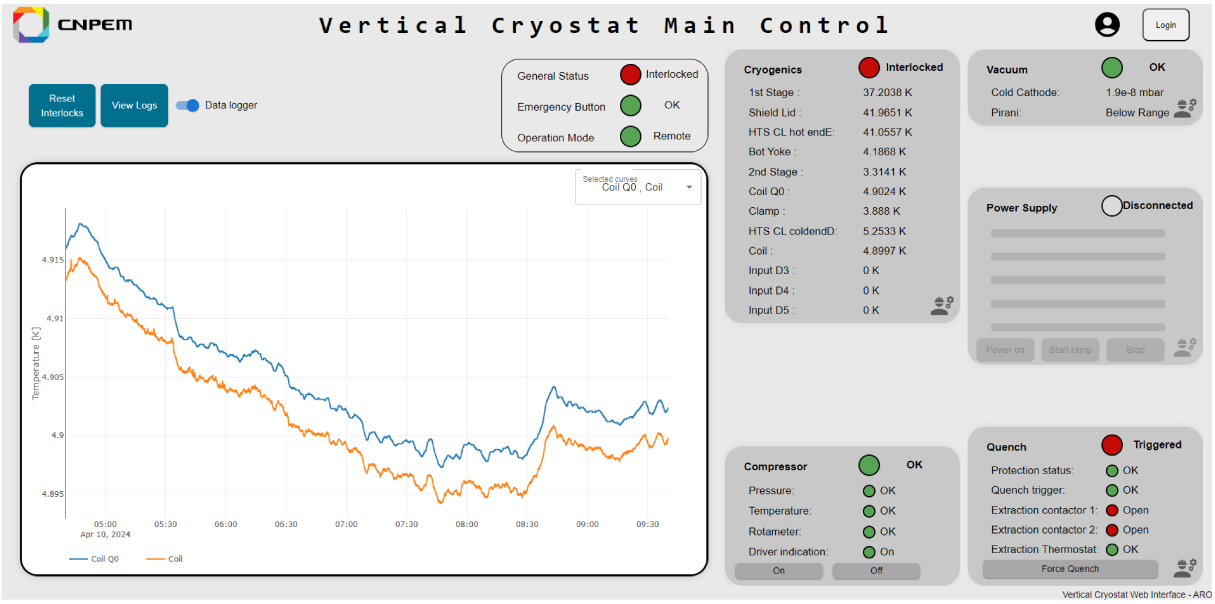
Figura 20 – Interface de controle com dispositivo em pleno funcionamento



Fonte: Grupo de Automação e Robótica (ARO) - CNPEM (2024)

Além disso, foram realizados testes em variadas condições de operação. A Figura 21 exibe o sistema no fim de uma rampa de resfriamento, conforme visto através dos gráficos. Apesar de o compressor encontrar-se em pleno funcionamento e os sensores detectarem temperaturas criogênicas já em torno de 4K no segundo estágio, o sistema encontra-se intertravado devido à desconexão da fonte de corrente e alarmes de temperatura não reconhecidos. É possível observar que os contatores de extração também encontram-se abertos (ou seja, desviando a energia da bobina), como consequência do interlock.

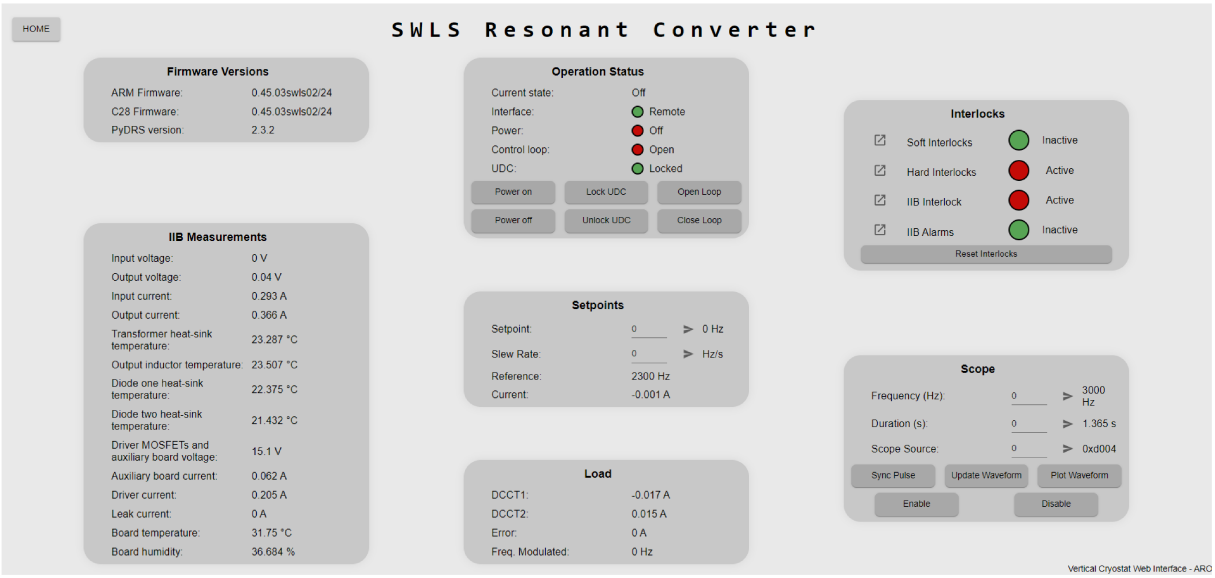
Figura 21 – Interface exibindo interlock ao fim de um resfriamento



Fonte: Grupo de Automação e Robótica (ARO) - CNPEM (2024)

Através das janelas específicas para cada subsistema, o usuário também pode visualizar detalhes sobre o equipamento associado e obter um diagnóstico mais apurado dos possíveis problemas, como visto na Figura 19 na seção anterior. A Figura 22 exibe a tela específica da fonte de corrente, onde, além de parâmetros mais detalhados, podem ser visualizados os tipos de alarmes ativos.

Figura 22 – Interface exibindo detalhes da fonte de corrente

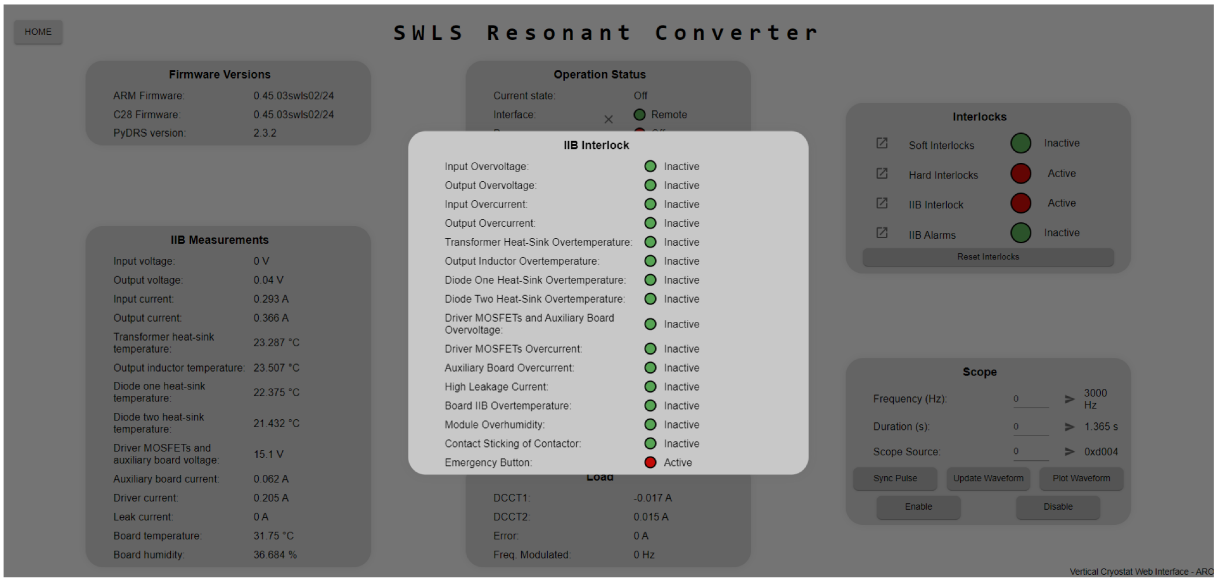


Fonte: Grupo de Automação e Robótica (ARO) - CNPEM (2024)

Ao expandir a janela de alarmes, pode ser visualizado exatamente qual bit do controlador da fonte está ativo, garantindo uma rápida identificação do problema. A Figura

23 exibe a janela de alarmes detalhados, onde é possível ver que o botão de emergência da fonte de corrente foi acionado, impedindo a operação.

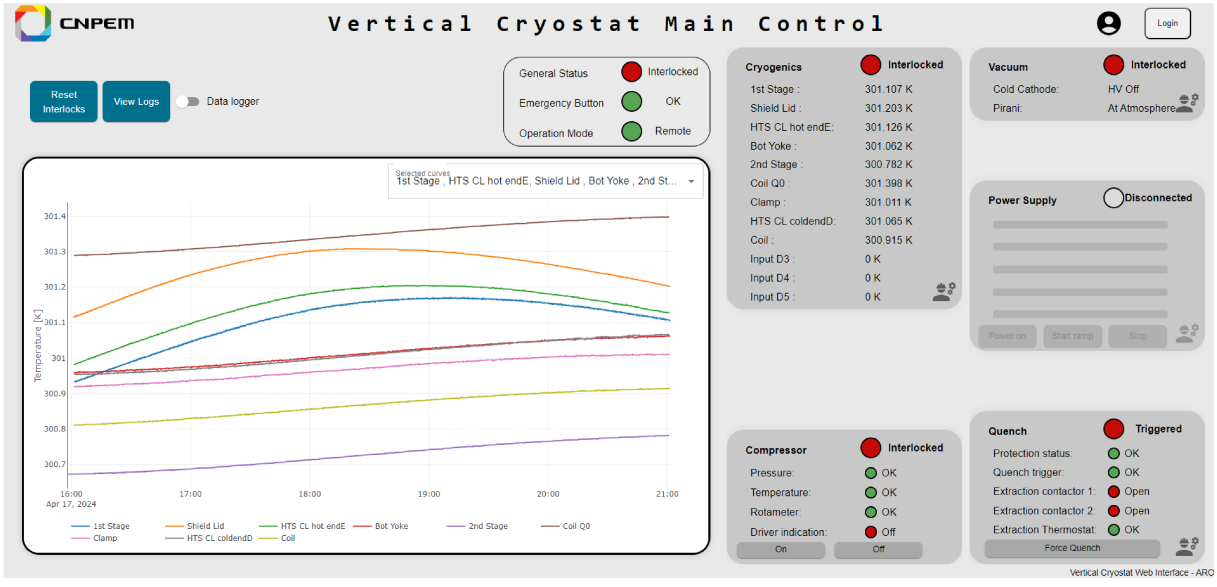
Figura 23 – Alarmes detalhados da fonte de corrente



Fonte: Grupo de Automação e Robótica (ARO) - CNPEM (2024)

É possível observar o comportamento do sistema também fora de operação, como na Figura 24, onde é possível visualizar o sistema totalmente desligado e todos os subsistemas intertravados. Neste caso, as temperaturas estão em torno de 300K, e foram plotadas simultaneamente as leituras de todos os sensores do sistema criogênico, possibilitando a observação de seu comportamento ao longo do tempo. Por segurança, quando fora de operação a fonte de corrente é totalmente desconectada, evento também indicado na interface.

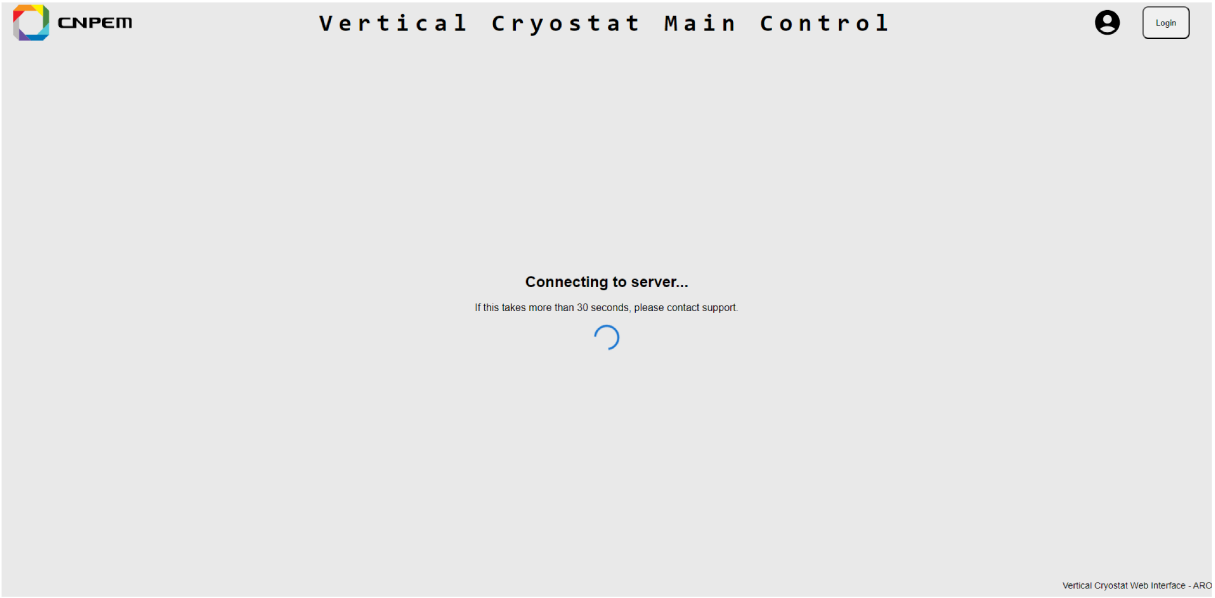
Figura 24 – Interface exibindo sistema desligado



Fonte: Grupo de Automação e Robótica (ARO) - CNPEM (2024)

Além disso, em caso de desconexão entre a interface e o PVWS, a fim de evitar a exibição de dados incorretos, a interface exibe uma tela de carregamento, durante a qual o usuário é informado sobre a tentativa de reconexão e a impossibilidade de operação. A Figura 25 exibe a interface nesta condição.

Figura 25 – Interface exibindo tela de carregamento



Fonte: Grupo de Automação e Robótica (ARO) - CNPEM (2024)

Assim, observou-se que a arquitetura de controle distribuído proposta permitiu a visualização e controle dos subsistemas de forma simplificada e intuitiva, facilitando a operação e a obtenção de resultados, além de auxiliar na identificação de problemas e na tomada de decisões. Na próxima seção serão feitas considerações finais sobre o trabalho desenvolvido e as perspectivas para trabalhos futuros.

5. Conclusão

A partir do desenvolvimento deste trabalho, observou-se a relevância dos sistemas de controle distribuído em aplicações com múltiplos subsistemas. A arquitetura proposta possibilitou a integração e controle simplificado e intuitivo dos subsistemas do criostato vertical, otimizando a operação do dispositivo e realização dos experimentos propostos. A implementação dos IOCs permitiu o controle modular de cada equipamento, alinhado ao conceito de controle distribuído. O protocolo Channel Access possibilitou a comunicação e a aquisição de dados dos dispositivos de instrumentação e controle utilizados. O uso do PVWS facilitou a comunicação entre a rede de IOCs e a interface gráfica, desenvolvida em JavaScript. Ferramentas auxiliares como bibliotecas, Docker e NGINX simplificaram o desenvolvimento da aplicação e proporcionaram uma execução e disponibilização de serviços mais robusta.

A interface gráfica desenvolvida consolidou as informações de diferentes dispositivos em uma única tela, simplificando a visualização e o controle do sistema completo. Devido à sua natureza web, a interface pode ser acessada remotamente, permitindo o monitoramento do sistema a partir de qualquer dispositivo com acesso à rede local do campus, proporcionando maior flexibilidade aos usuários. Ela permite o monitoramento de variáveis como temperaturas, pressões, correntes e estados dos subsistemas, além de seu histórico por meio de gráficos. Assim, pode ser observado o comportamento do sistema em diferentes condições de operação, como durante uma rampa de resfriamento, interlocks e fora de operação. A visualização simultânea de diferentes variáveis no gráfico possibilita aos usuários identificar correlações entre as medidas e obter informações relevantes sobre o comportamento do sistema em teste, enquanto mensagens de erro e alertas na interface facilitam o diagnóstico de falhas e a tomada de ações corretivas, tanto na janela principal quanto nas janelas específicas de cada subsistema.

Para implementações futuras, pretende-se adicionar à arquitetura a um banco de dados dedicado, permitindo o armazenamento e a análise de dados de forma mais eficiente e em intervalos maiores de tempo sem a dependência de ferramentas externas para geração de logs. Adicionalmente, pretende-se vincular o acesso à página às contas internas do CNPEM, baseadas em logins da Microsoft, limitando o acesso a usuários autorizados. Por meio do banco de dados e do controle de acesso, será possível definir permissões específicas para grupos de usuários, permitindo, por exemplo, que somente especialistas com a devida autenticação possam realizar a alteração de parâmetros críticos do sistema. Com estas medidas, pretende-se melhorar a segurança e rastreabilidade do sistema, bem como a eficiência na análise de dados.

Referências Bibliográficas

ARTISAN. HPS 937A MKS Instruments. s.d. Disponível em: <https://www.artisantg.com/Scientific/55093-1/MKS-Instruments-HPS-937A-Gauge-Controller>. Acesso em: 09 de março de 2024. Citado na página 26.

AUTOMATION, R. Micro820 20 I/O EtherNet/IP Controller. s.d. Disponível em: <https://www.rockwellautomation.com/pt-br/products/details.2080-LC20-20QBB.html>. Acesso em: 12 de março de 2024. Citado na página 28.

CNPEM. PyDRS - Sirius Power Supplies communication. 2024. Disponível em: <https://pypi.org/project/pydrs/>. Acesso em: 15 de março de 2024. Citado na página 35.

CONTESINI, W. et al. Prototype of a Resonant Converter for a Superconducting Magnet with 300 A and 10 V Output Operating in Discontinuous Conduction Mode. In: . [S.l.: s.n.], 2023. p. 1–8. Citado na página 26.

CRYOGENICS, S. RDK-415D2 4K Cryocooler Series. 2024. Disponível em: <https://www.shicryogenics.com/product/rdk-415d2-4k-cryocooler-series/>. Acesso em: 09 de março de 2024. Citado na página 25.

CRYOSRV. Sumitomo CSW-71D compressor. 2020. Disponível em: <https://www.cryosrv.com/product/sumitomo-csw-71d-compressor/>. Acesso em: 09 de março de 2024. Citado na página 25.

CRYOTRONICS, L. LakeShore model 224 cryogenic temperature monitor. 2023. Disponível em: <https://www.lakeshore.com/products/categories/overview/temperature-products/cryogenic-temperature-monitors/model-224-temperature-monitor/>. Acesso em: 09 de março de 2024. Citado na página 25.

CRYOTRONICS, L. S. *User's Manual. Model 224 Temperature Monitor*. [S.l.], 2015. 80-94 p. Disponível em: <https://www.lakeshore.com/docs/default-source/product-downloads/224_manual334ea6b3c2fd45b9929c71eaaea78171.pdf?sfvrsn>. Acesso em: 12 de março de 2024. Citado 2 vezes. Páginas 31 e 32.

DOCKER. Docker compose. 2024. Disponível em: <https://docs.docker.com/compose/>. Acesso em: 04 de abril de 2024. Citado na página 23.

DOCKER. Docker overview. 2024. Disponível em: <https://docs.docker.com/get-started/overview/>. Acesso em: 05 de janeiro de 2024. Citado na página 23.

EPICS. asynDriver: Asynchronous Driver Support. 2018. Disponível em: <https://epics.anl.gov/modules/soft/asyn/R3-1/asynDriver.html>. Acesso em: 14 de março de 2024. Citado na página 20.

EPICS. Channel Access Protocol Specification. 2019. Disponível em: <https://docs.epics-controls.org/en/latest/internal/ca_protocol.html#_introduction>. Acesso em: 03 de janeiro de 2024. Citado 3 vezes. Páginas 17, 18 e 19.

EPICS. EPICS Access Security. 2019. Disponível em: <https://docs.epics-controls.org/en/latest/access-security/specifications.html>. Acesso em: 05 de março de 2024. Citado na página 20.

EPICS. Getting started with EPICS. 2019. Disponível em: <https://docs.epics-controls.org/en/latest/getting-started/EPICS_Intro.html>. Acesso em: 02 de janeiro de 2024. Citado na página 17.

EPICS. About EPICS. 2024. Disponível em: <https://epics-controls.org/about-epics/>. Acesso em: 05 de março de 2024. Citado na página 16.

EPICS. Epics Projects. 2024. Disponível em: <https://epics-controls.org/epics-users/projects/>. Acesso em: 01 de abril de 2024. Citado na página 16.

EPICS. Record Reference Documentation. s.d. Disponível em: <https://epics.anl.gov/base/R7-0/6-docs/RecordReference.html>. Acesso em: 15 de março de 2024. Citado 2 vezes. Páginas 33 e 34.

F5. NGINX - Overcome the Challenges of Modern App Delivery. s.d. Disponível em: <https://www.nginx.com>. Acesso em: 14 de março de 2024. Citado na página 23.

FOUNDATION, M. What is JavaScript? 2024. Disponível em: <https://developer.mozilla.org/en-US/docs/Web/JavaScript>. Acesso em: 02 de janeiro de 2024. Citado na página 21.

GALVEZ, J. et al. Preliminary Design of a Cryogen-Free 6.6 T Superconducting Wavelength Shifter. *IEEE Transactions on Applied Superconductivity*, PP, p. 1–5, 01 2023. Citado na página 12.

HUBBLESITE. The electromagnetic spectrum. 2022. Disponível em: <https://hubblesite.org/contents/articles/the-electromagnetic-spectrum>. Acesso em: 02 de janeiro de 2024. Citado na página 11.

KATE, H. T. Superconducting magnets: Quench propagation and protection. 2013. Disponível em: <https://indico.cern.ch/event/194284/contributions/1472819/attachments/281522/393603/TenKate_-_CAS_-Handout-Quench-Erice-2103.pdf>. Acesso em: 03 de janeiro de 2024. Citado 2 vezes. Páginas 13 e 27.

LABORATORY, O. R. N. PVWS - PV WebSocket. s.d. Disponível em: <https://github.com/ornl-epics/pvws>. Acesso em: 14 de março de 2024. Citado 2 vezes. Páginas 21 e 30.

LNLS. Basic Small Messages Protocol - BSMP. 2018. Disponível em: <https://github.com/lnls-sirius/libbsmp/blob/master/doc/protocol_v2-30_pt_BR.pdf>. Acesso em: 09 de março de 2024. Citado na página 26.

LNLS. Como funciona o Sirius? 2022. Disponível em: <https://lnls.cnpem.br/sirius/como-funciona-o-sirius/>. Acesso em: 04 de janeiro de 2024. Citado na página 11.

LNLS. A luz síncrotron e seus efeitos. 2023. Disponível em: <https://lnls.cnpem.br/sirius/a-luz-sincrotron-e-seus-beneficios/>. Acesso em: 02 de janeiro de 2024. Citado na página 11.

MATERIAL-UI. Material-UI: React components for faster and easier web development. 2024. Disponível em: <https://material-ui.com>. Acesso em: 20 de março de 2024. Citado na página 40.

META. React: the library for web and native user interfaces. 2024. Disponível em: <https://react.dev>. Acesso em: 02 de janeiro de 2024. Citado na página 21.

NIST. What is synchrotron radiation? 2021. Disponível em: <https://www.nist.gov/pml/sensor-science/what-synchrotron-radiation>. Acesso em: 03 de janeiro de 2024. Citado na página 11.

PLOTLY. Plotly.js: Open-source JavaScript charting library. 2024. Disponível em: <https://plotly.com/javascript/>. Acesso em: 20 de março de 2024. Citado na página 40.

PSI, P. S. I. PCASpy Documentation. 2017. Disponível em: <https://pcaspy.readthedocs.io/en/latest/index.html>. Acesso em: 15 de março de 2024. Citado na página 35.

PSI, P. S. I. EPICS StreamDevice. 2018. Disponível em: <https://paulscherrerinstitute.github.io/StreamDevice/>. Acesso em: 14 de março de 2024. Citado na página 20.

RIVERS, M. Driver Support for Modbus Protocol under EPICS. 2019. Disponível em: <https://cars9.uchicago.edu/software/epics/modbusDoc.html>. Acesso em: 15 de março de 2024. Citado na página 34.

ROUTER, R. React router: Declarative routing for react.js. 2024. Disponível em: <https://reactrouter.com>. Acesso em: 04 de abril de 2024. Citado na página 41.