UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
ESCOLA DE ENGENHARIA
TRABALHO DE CONCLUSÃO EM ENGENHARIA FÍSICA

Simulador da Interface Gráfica do Usuário para o Espectrógrafo Echarpe.

Autor: Eduardo Godoy da Silveira

Orientadores: Ana Leonor Chies Santiago Santos, Charles José Bonatto e
Orientador - LNA: Orlando Verducci Junior

Porto Alegre, 28 de agosto de 2024

Sumário

Agradecimentos

Gostaria de agradecer os gigantes aos quais estou nos ombros.

Especialmente meus pais, por sempre me instigarem no conhecimento, me incentivarem nos estudos e não medir esforços para garantir que eu tivesse o melhor ensino possível.

Ao restante dos meus familiares, em especial ao meu avô materno Alziro Neri Spindler, pelo constante apoio e por me mostrarem que a educação é para todos e indispensável.

Aos colaboradores do Laboratório Nacional de Astronomia, em especial ao Orlando, pelas incontáveis reuniões no meet e horas me ensinando LabView.

Aos meus orientadores, Charles e Ana, por me mostrarem o que é fazer ciência de verdade e por terem me dado a oportunidade de trabalhar com eles ao longo desses anos.

Aos meus amigos, especialmente os que fiz na universidade, Mário, Augusto, Francisco, Lianna, Andréia e Filipe, por tornarem a jornada mais agradável e as equações diferenciais menos desgastantes.

E a minha namorada Giovana Lucca, pelo suporte, carinho e motivação.

Resumo

O Echarpe é um espectrógrafo de alta resolução (R=50,000) que está sendo desenvolvido para o telescópio Perkin & Elmer do Observatório do Pico dos Dias. Os dois canais do Echarpe serão alimentados por fibras ópticas e fornecerão um espectro combinado no intervalo entre 390-900 nm. O instrumento será utilizado no estudo de diversos temas, como composição química e atividade cromosférica de estrelas do tipo solar, análogas solares, planetas terrestres, evolução química da Galáxia baseada em estrelas gigantes, determinação precisa de dimensões absolutas de componentes de sistemas binários eclipsantes, entre outros. O Echarpe poderá ser utilizado tanto presencialmente quanto via internet no modo remoto. Em ambos os casos o astrônomo realizará as observações utilizando uma interface gráfica que fornecerá todas as opções de observação e de operação do instrumento. Neste projeto serão realizados os estudos para o desenvolvimento de uma versão **desta interface gráfica, a ser testada e avaliada pelos usuários finais antes da sua validação no telescópio**. O projeto envolve a utilização científica do instrumento, a definição dos requisitos de operação, o sistema de controle do mesmo e o software de controle no qual será baseada a interface. A interface real e a simulada serão desenvolvidas no sistema Labview, e quando validadas pela equipe de desenvolvimento, será disponibilizada para testes pelos usuários.

Lista de Figuras

Lista de Tabelas

1 Introdução

Esta monografia apresenta um projeto detalhado sobre a elaboração de uma interface gráfica para um espectrógrafo que está sendo desenvolvido no Laboratório Nacional de Astrofísica (LNA), o ECHARPE. Espectrógrafos compõem alguns dos instrumentos mais importantes para o desenvolvimento da ciência como conhecemos hoje, conseguindo analisar a composição química de qualquer emissor (ou refletor) de luz.

No campo da astrofísica, os espectrógrafos desempenham um papel fundamental. Eles são os principais agentes da espectroscopia, o estudo da interação da luz com a matéria. Como demonstrado pelo físico escocês James Clerk Maxwell (1831-1879) em seu artigo Dynamical Theory of the Electromagnetic Field, uma carga elétrica em movimento no espaço gera um campo magnético. Portanto, a luz é o que chamamos de uma onda eletromagnética e sua interação com a matéria deixa marcas em seu espectro[1]. A partir dessas marcas é possível obter, direta ou indiretamente, propriedades físicas e químicas de objetos emissores de luz.

O espectro da luz é um objeto de estudo dos físicos desde o século XVII, quando Isaac Newton (1643-1727) demonstrou que a luz branca, ao passar por um prisma de luz, se dispersa em seus comprimentos de onda (Figura 1). Isso acontece porque a refração depende da frequência, portanto o ângulo de refração muda conforme o comprimento de onda também muda. Esse é um fenômeno que podemos observar no cotidiano, como quando os raios-do-sol atravessam gotas de chuvas suspensas no ar, formando o que conhecemos como arco-íris. Os diferentes comprimentos de onda são interpretados pelo nosso cérebro como cores e podemos observar então que a luz do sol (branca) se torna colorida, do vermelho (maior comprimento de onda) até o violeta (menor comprimento de onda).

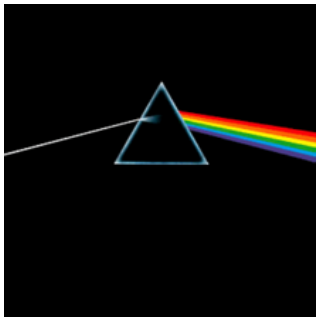

Figura 1: Capa do Álbum *The Dark Side of the Moon* (1973) que mostra a dispersão dos diferentes comprimentos de onda da luz

Fonte: `https://www.pinkfloyd.com/design/album_covers.php`

[1] "O espectro eletromagnético é a distribuição da intensidade da radiação eletromagnética com relação ao seu comprimento de onda ou frequência.- Kepler de Souza Oliveira Filho & Maria de Fátima Oliveira Saraiva. Disponível em: https://www.if.ufrgs.br/oei/cgu/espec/intro.htm

A partir dos estudos acerca do espectro, foram catalogadas por Joseph Ritter von Fraunhofer (1787-1826) 574 linhas escuras no espectro solar, nomeadas após linhas de Fraunhofer. Essas foram explicadas posteriormente pelo físico Gustav Robert Kirchhoff (1824-1887) e marcam a interação de um gás frio com um corpo opaco e quente. No caso do Sol, as linhas de Fraunhofer são geradas com a absorção das camadas mais externas (gás frio) com a emissão de espectro contínuo gerado no interior da estrela (corpo opaco e quente) como mostra a Figura 2.

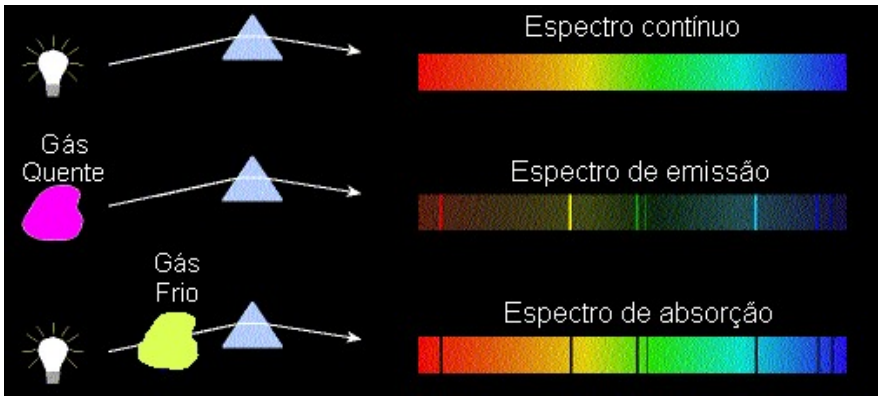

Figura 2: Esquemático da interação da luz com a matéria

Fonte: `http://astro.if.ufrgs.br/rad/espec/espec.htm`

No século XIX, o astrônomo William Huggins descobriu que essas absorções nos permitem inferir a composição química do corpo em questão e desde então diversas estrelas foram catalogadas e classificadas espectralmente. O estudo da composição química também pode se expandir para exoplanetas, onde variações na luz da estrela que o planeta orbita podem levar a conclusões sobre a composição atmosférica e a possibilidade de abrigar vida. A espectroscopia também proporcionou uma das maiores descobertas da astronomia o desvio para o vermelho (Redshift), estabelecendo Edwin Powell Hubble (1889 - 1953) como um dos maiores astrônomos da história. O citado astrônomo descobriu que o universo está em constante expansão devido ao desvio das linhas espectrais de galáxias distantes, em decorrência do efeito Doppler.

1.1 Espectrógrafos

Os espectrógrafos são os instrumentos utilizados para fazermos espectroscopia. Eles são usados em outras áreas além da astrofísica, mas seu funcionamento se mantém semelhante. Em espectrógrafos para astronomia, a captação de luz é feita pelo telescópio e é transmitida por meio de elementos ópticos ou fibras ópticas até sua dispersão. Geralmente se utilizam de uma rede de difração para o espalhamento da luz. Uma rede de difração funciona como uma grelha com várias ranhuras, onde a luz transmitida pelas várias fendas das ranhuras interage com as outras que também foram transmitidas com diferentes ângulos de incidência formando um espectro. Esses feixes são direcionados a um CCD (Charge-Coupled Device) que faz a leitura do espectro.

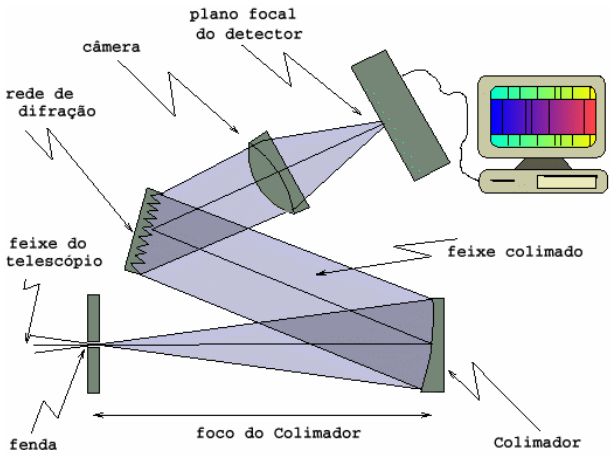

Figura 3: Esquemático do funcionamento do Espectrógrafo.

Fonte: `http://www.astro.iag.usp.br/~ronaldo/intrcosm/Glossario/Espectrografo.html`

Essa leitura é enviada a um computador que codifica os dados recebidos e os coloca a disposição para o astrônomo como dados digitais referentes às observações feitas

1.2 O ECHARPE

O ECHARPE (Espectrógrafo Echelle de Alta Resolução para o telescópio Perkin-Elmer) está sendo desenvolvido pelo Laboratório Nacional de Astrofísica para ser utilizado o telescópio de 1,60m do Observatório do Pico dos Dias, localizado em Brazópolis, Brasil. Trata-se de um espectrógrafo Echelle de alta resolução, R ~ 50.000, no intervalo entre 390 e 890nm em uma única exposição e abertura de 1,5 segundos de arco, o que significa que ele será capaz de resolver linhas de até 10pm. É um instrumento de dois canais, sendo o canal azul 390-610nm e o canal vermelho 580-890nm. Foi projetado para ser um espectrógrafo de bancada alimentado por fibra óptica.

O ECHARPE foi planejado para obter espectros estelares com R ~ 50000 e S/N adequados para a obtenção de parâmetros atmosféricos e composição química de objetos astronômicos. Uma relação S/R igual ou superior a 200 para magnitude V ~ 8 deve ser possível com tempos de exposição menores que uma hora, em boas condições de céu. A ampla cobertura permite a medição de centenas de linhas espectrais, abrangendo inúmeros elementos químicos e diferentes origens nucleossintéticas. Tais dados constituirão testes fundamentais para a evolução química da Galáxia em todos os seus contextos.

Adicionalmente, a cobertura proposta envolve os três indicadores cromosféricos clássicos de atividade magnética em estrelas do tipo F, G, K e M de anãs a gigantes. São elas: linhas H e K de Ca II no UV, Halpha no vermelho e trigêmeo do Ca II no infravermelho. Tais dados permitem um amplo diagnóstico da atividade nas estrelas, e as medições simultâneas permitem o estudo de várias classes de fenômenos transitórios, como as erupções, que afetam os diferentes indicadores de

maneiras distintas.

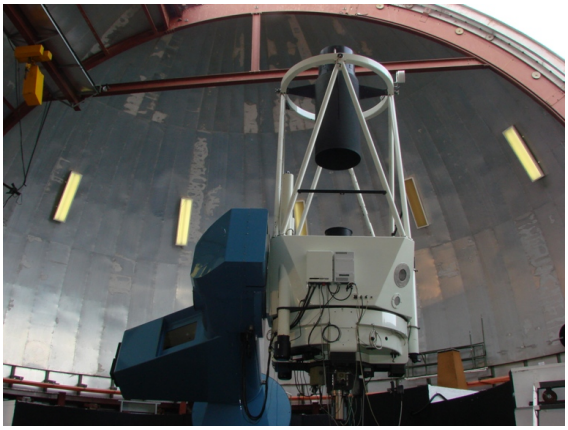

Figura 4: Telescópio Pelkin-Elmer.

Fonte: `https://www.gov.br/lna/pt-br/composicao-1/coast/obs/opd/telescopios/telescopios-do-opd`

1.3 Objetivo do projeto

A execução desse projeto visa implantar uma GUI (Graphical User Interface) para o funcionamento do espectrógrafo Echarpe que será instalado no telescópio 1,6m Perkin-Elmer do Observatório do Pico dos Dias. O Echarpe poderá ser utilizado tanto presencialmente quanto via internet no modo remoto. Em ambos os casos o astrônomo realizará as observações utilizando uma interface gráfica que fornecerá todas as opções de observação e de operação do instrumento.

A interface gráfica irá conter alguns dos comandos que o astrônomo poderá operar sobre o Echarpe e será testada e avaliada pelos usuários finais antes da sua validação no telescópio. O projeto envolve a utilização científica do instrumento, a definição dos requisitos de operação, o sistema de controle do mesmo e o software de controle no qual será baseada a interface. A interface real e a simulada serão desenvolvidas no sistema Labview, e quando validada pela equipe de desenvolvimento será disponibilizada para testes pelos usuários.

1.4 Descrição do parceiro do projeto

O presente projeto está sendo desenvolvido em colaboração com pesquisadores do Laboratório Nacional de Astrofísica, situado em Minas Gerais e vinculado ao Ministério da Ciência, Tecnologia e Inovação do Brasil. Fundado em 1985, o laboratório é uma instituição de renome que conta com quatro telescópios no Observatório Pico dos Dias, além coordenar a participação brasileira nos telescópios internacionais como o Gemini, SOAR e CFHT.

A professora Ana Chies Santos e o professor Charles Bonatto, orientadores desse projeto, desempenharam um papel fundamental na elaboração deste projeto, ao proporcionar a conexão com

a equipe do LNA. Através do Dr. Bruno Vaz Castilho de Souza, pesquisador do LNA, foi possível estabelecer contato e uma colaboração frutífera com os tecnologistas Dr. Orlando Verducci Junior e Francisco Rodrigues, essenciais no planejamento do projeto.

1.5 Escopo do projeto

A GUI de um instrumento, é apenas uma parte do complexo sistema de observação de um telescópio (figura 5), ela é a parte que interage diretamente com o usuário e permite que o astrônomo faça o processamento das imagens, reduções necessárias, além de configurar o instrumento e receber dados de telemetria para registro no cabeçalho dos arquivos de imagem.

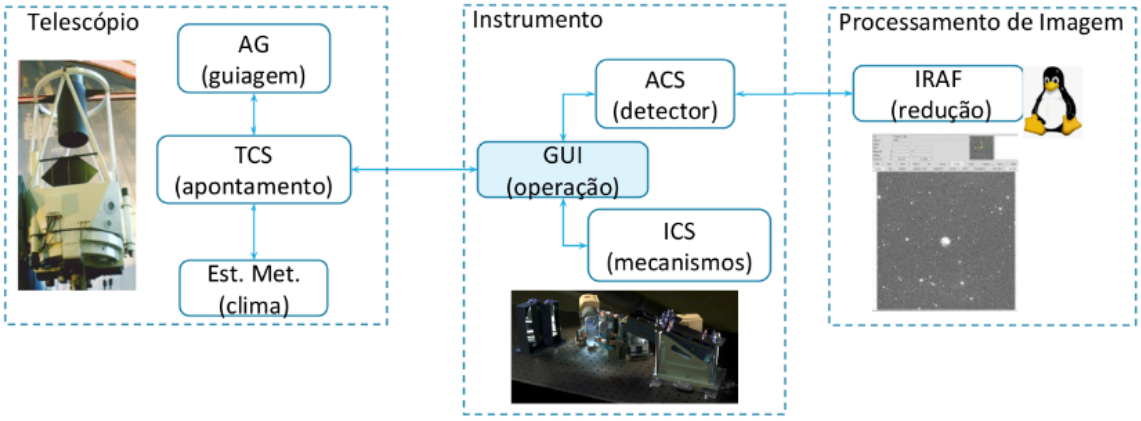

Figura 5: Esquemático do sistema de controle de um telescópio.

Fonte: Digrama fornecido pelos parceiros do projeto

Algumas dessas funcionalidades de um sistema de controle observacional operam sem a intervenção do usuário, como o fechamento da cúpula (em caso de precipitação) e os dispositivos de acompanhamento sideral e guiagem (que seguem o objeto escolhido para observação durante a noite e o mantêm estável no centro do campo do detetor). Já a GUI em questão, além de se comunicar com as outras aplicações do observatório, deverá disparar procedimentos específicos a partir da interação do usuário com o seu painel frontal.

As partes da Interface que contemplam esse projeto terão interação com o usuário, mas esse projeto não visa elaborar todos os métodos da GUI. O escopo desse projeto é desenvolver um embrião, uma parte do complexo sistema do Echarpe que se comunicará, em um primeiro momento, com dois servidores: uma estação meteorológica e o ICS (Instrument Control Server). O primeiro servidor irá fornecer as informações climáticas, que serão apresentadas em uma região destacada a ser desenvolvida no painel da GUI, já o segundo há de se comunicar com mecanismos relacionados ao espectrógrafo com o intuito de controlá-los, além de mostrar informações sobre seu estado atual.

Na secção 4.5 será possivel ver mais nitidamente as funcionalidades adicionais que terão que ser configuradas para que o ECHARPE funcione em sua totalidade.

1.6 Cronograma do Projeto

Com sugestão dos parceiros do LNA o cronograma do projeto foi elaborado com as seguintes etapas:

1. Elaborar o diagrama de classes da aplicação GUI (conhecer o instrumento);
2. Criar o projeto da GUI do ECHARPE no LabVIEW (EcharpeGui.lvproj);
3. No projeto LabVIEW, criar todas as classes definidas no diagrama do passo 1;
4. Configurar no LabVIEW os relacionamentos (de herança e de composição) entre as classes criadas no projeto – baseado no diagrama do passo 1;
5. Inserir em cada classe os primeiros atributos e métodos (vazios no início);
6. Criar, usando os métodos da classe EcharpeGui.lvclass, o módulo principal da aplicação (EcharpeGui.vi);
7. Esboçar o layout do Painel Frontal de EcharpeGui.vi e desenvolver o código dos eventos associados aos controles do painel;
8. Testes e correções;
9. Trabalho escrito;
10. Elaboração da apresentação;
11. Apresentação para a banca;

Estas foram dispostas no cronograma abaixo:

Tabela 1: Cronograma

Etapas	Outubro	Novembro	Dezembro	Janeiro	Fevereiro	Março	Abril	Maio	Junho	Julho
1										
2										
3										
4										
5										
6										
7										
8										
9										
10										
11										

2 Desenvolvimento de especificações

A GUI se comunicará com a estação meteorológica via protocolo TCP-IP (Cap. 3) utilizando métodos *Comm:Transmit* (para enviar comandos para ambos os servidores) e *Comm:Recive* (para receber as respostas dos servidores), obtendo informações como: temperatura, pressão atmosférica, velocidade do vento e umidade. Já com o ICS a aplicação utilizará o mesmo protocolo e mesmos métodos mas receberá dados diferentes como: o status das o status das lâmpadas halógenas e de Tório-Argônio (usadas para calibração), o status do espelho, tipo fold (que é acionado para inserir a luz das lâmpadas no caminho óptico do instrumento). Ainda, a GUI será capaz de enviar comandos atuando nos dispositivos do instrumento, acendendo e apagando lâmpadas, e movendo o prisma e o espelho de calibração.

A aplicação desenvolvida nesse projeto utiliza a linguagem gráfica LabVIEW (*Laboratory Virtual Instrument Engineering Workbench*) sendo uma linguagem orientada a objetos empregada no campo da automação industrial. Um projeto orientado a objetos, segundo (Vasconcelos & Verducci, 2011),

> "consiste em modelar e implementar o software em termos de um conjunto de elementos (objetos), que possuem características (atributos) e comportamentos (métodos) próprios da classe a que pertencem, e interagem entre si solicitando a execução de tarefas previamente definidas para o objeto referenciado, sendo desnecessário conhecer sua organização interna. A ideia principal é fazer com que o entendimento do software seja intuitivo e natural, pois as classes definidas no código do programa devem representar tipos de elementos do problema a ser tratado no mundo real."

A programação orientada a objetos (POO), ou Object-oriented programming (OOP), é um modelo de programação que visa organizar o código em torno de objetos, aproximando o programa entre o mundo real e virtual. Ela é um modelo de programação que se diferencia da tradicional programação de código estruturado. O código estruturado tem como objetivo criar um programa que execute uma sequência de tarefas de modo computacional. Ele tem algumas características, como o uso de sequências (comandos), condições (comandos condicionais como o if em python), repetições (loops), mas a principal característica é a possibilidade da utilização de variáveis globais, que poderão ser acessadas em qualquer rotina do programa. Enquanto isso, a programação orientada a objetos organiza dados e funções introduzindo o conceito de classes e objetos, onde a estruturação de dados ficaria confinada as classes e a execução da aplicação é baseada em troca de mensagem entre objetos.A programação OO é indicada para sistemas escaláveis (que permitem facilmente crescer em quantidade de componentes que os constituem) e modulares (que permitem realizar mudanças locais de tecnologia sem impactar o restante do código).

2.1 Características de POO

Os objetos têm esse nome, pois idealmente eles representam algo tangível, eles são uma instância de uma classe, representando uma entidade real ou conceitual. Já uma classe é o modelo para criação dos objetos. Ela é um conjunto de atributos (dados) e métodos (comportamentos) que funciona como um molde para a criação de um objeto concreto. A analogia mais clássica a se fazer é a de um carro, onde a classe carro em si não é algo concreto, mas quando falamos em Onix, Hb20 e Gol a imagem do objeto em si se torna tangível. Assim, nós podemos também imaginar atributos como marca, modelo e cor para no caso do objeto carro, tal qual como seus métodos como acelerar e frear.

Dos conceitos de classes e objetos se derivam características muito importantes para o paradigma da POO.

2.1.1 Encapsulamento

É um conceito para adicionar segurança e abstração à aplicação. Com ele é possível controlar o acesso de atributos e métodos encapsulando-os nas classes, impedindo o acesso direto e garantidno a abstração da implementação. Assim, como no exemplo do carro, podemos encapsular um método como acelerar, que não deve ser facilmente modificável para não haver alterações. Este método fica encapsulado no objeto e o chamamos de privado, de modo que, tudo observável de fora, é que o objeto (carro) executa o método (acelera), mas sem a visibilidade de como o método realiza aquilo. Isso também ocorre para atributos, como podemos usar o exemplo da velocidade do carro. Não fica explícito para as outras classes como o carro mede ou a calcula a velocidade, apenas o seu valor fica disponível.

Sendo assim, a programação orientada a objetos se diferencia do código estruturado, pois atributos encapsulados não podem ser utilizados como variáveis globais. Vale ressaltar que podem existir variáveis globais na POO, mas elas devem ser evitadas para se tirar proveito dos recursos da orientação a objeto. Em POO as variáveis globais são substituidas por atributos da classe que representa a aplicação. Assim, o acesso desses atributos e métodos dos objetos fica limitado a métodos específicos (readers e writers) mantendo a sua integridade.

2.1.2 Herança

Outro conceito importante na POO é o conceito de Herança. Esse conceito permite criar novas classes (chamadas classes derivadas ou subclasses) a partir de uma classe existente (chamada classe base ou superclasse). Essa técnica permite o compartilhamento de atributos e métodos entre essas classes. Ainda podemos ressaltar que a classe derivada pode adicionar atributos e métodos para tornar sua utilização mais específica. Podemos imaginar carros como o cenário perfeito para a analogia desse conceito. Onde temos um carro (classe) que tem diferentes modelos, onde cada modelo preserva alguns métodos e atributos, porém pode se modificar, como banco de couro, direção elétrica e ar condicionado digital, ou adicionar métodos novos, como controle de tração e teto solar.

2.1.3 Polimorfismo

Por último, mas não menos importante, o polimorfismo é um conceito onde diferentes objetos realizam o mesmo método de maneiras diferentes. Como seu nome já sugere, em grego, poli = muitas e morfismo = formas. Podemos usar o exemplo de tocar música em carros, onde em alguns pode ser feito via bluetooth e em outros utilizando CD. O método produz o mesmo resultado em ambos os casos, porém a maneira de interagir com o usuário e de executar seu processo é totalmente diferente. No LabVIEW, e na aplicação do ECHARPE, o polimorfismo se apresenta no momento de uma classe filha sobrescrever um método da sua classe mãe. O conceito de polimorfismo é mais abrangente do que isso, mas neste trabalho não foram exploradas outras formas de polimorfismo que não fossem métodos sobrescritos da classe mãe. Para exemplificar: imagine o método Prisma:move.vi com target como parâmetro e o mesmo método Prisma:move.vi com speed como parâmetro. No primeiro caso, o prisma move até a posição solicitada em target; no segundo caso, o prisma move continuamente com a velocidade especificada em speed. São duas formas de realizar o movimento do Prisma com um único método move.vi.

2.2 Descrição do contexto de uso

O Echarpe será utilizado tanto presencialmente, no observatório Pico dos Dias, quanto no modo remoto via internet, onde astrônomos do pais todo poderão obter espectros de alta resolução na faixa de 390-900 nm combinando os dados dos dois canais.

O espectrógrafo será utilizado para o estudo da Composição química e atividade cromosférica de estrelas do tipo solar, análogas solares e planetas terrestres, evolução química da Galáxia baseada em estrelas gigantes, determinação precisa de dimensões absolutas de componentes de sistemas binários eclipsantes, além de diversos outros corpos celestes.

Quanto a aplicação desenvolvida nesse projeto, ela será integrada com o restante da interface gráfica do ECHARPE, e irá informar ao usuário os dados coletados pela estação meteorológica e permitir a operação dos dispositivos do instrumento. As grandezas apresentadas no display serão declaradas como *strings* em um módulo que fara um *request* de STATUS e irá apresentar as informações como mostra um exemplo de painel na figura 6.

2.3 Avaliação comparativa

Como discutido em: Programação Orientada a Objetos em LabVIEW para uma Aplicação Distribuída de Controle de Posição de um Telescópio (Vasconcelos & Verducci, 2011), o método OOP que será utlizado neste projeto tem muitas vantagens sobre o modelo de código estruturado.

O modelo OOP conta com algumas técnicas que auxiliam a sua manutenção e expansão futura

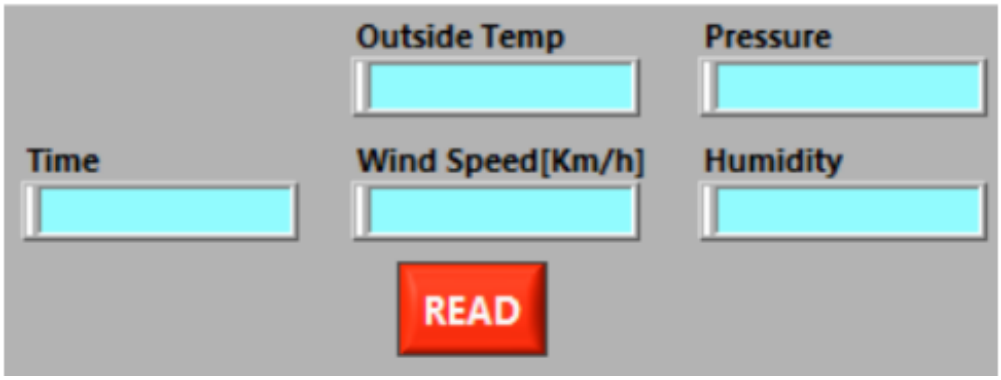

Figura 6: Exemplo de interface que mostra dados externos.

Fonte: Imagem fornecida pelos colaboradores.

como: Encapsulamento[2], herança[3] e polimorfismo[4](cap 4.2). O LabVIEW, por ser uma linguagem de programação gráfica, utiliza a criação de classes como blocos e as heranças podem ser vistas em forma de hierarquia entre os blocos. Essas heranças, interligam os blocos fazendo com que o código seja visualmente mais fácil de ser compreendido, além da facilidade de replicar o código em maior escala, fazendo com que a OOP seja uma escolha melhor para o desenvolvimento do projeto, em oposição ao método de código estruturado, embora este seja mais simples e rápido de implementado para aplicações menores.

[2]Técnica utilizada para dividir e isolar classes, protegendo seus atributos e métodos de acesso direto por outras classes.

[3]Técnica que permite criar classes (chamadas classes derivadas ou subclasses) a partir de uma classe existente (chamada classe base ou superclasse). Essa técnica permite o compartilhamento de atributos e métodos entre essas classes. Ainda podemos ressaltar que a classe derivada pode adicionar atributos e métodos para tornar sua utilização mais específica.

[4]Uma classe herdeira pode reescrever seus métodos e alterar os valores dos seus atributos para se adequar ao seu propósito.

3 Projeto Conceitual

Espectrógrafos no geral são instrumentos de muita sensibilidade e como comentado anteriormente, o ECHARPE é um espectrógrafo de alta resolução, com canais de fibra óptica com resolução de nanômetros. Por isso, é necessário um controle preciso e rápido das grandezas que possam alterar ou até danificar as medições do Espectrógrafo.

Com isso, foi necessário o desenvolvimento de um sistema de sensoriamento dessas grandezas externas, para que as condições nas quais o espectrógrafo é operado possam ser consideradas durante a espectroscopia, e para evitar eventuais problemas no dispositivo.

3.1 Conceito

Como mostra a figura 5, o as aplicações de controle do observatório são interligadas. Portanto, não faz sentido termos sensoriamentos independentes para cada instrumento. Assim o LNA conta com uma estação meteorológica própria (figura 7), que faz o sensoriamento para todas as aplicações ao mesmo tempo, compartilhando as informações via protocolo TCP-IP (Cap 4).

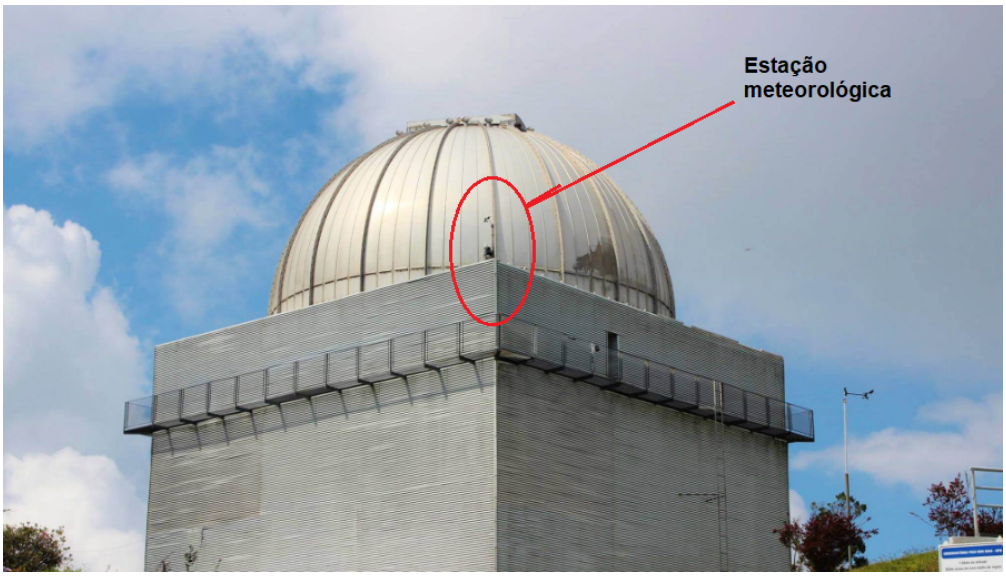

Figura 7: Estação meteorológica do Observatório Pico dos Dias.

Fonte: Imgem tirada pelo parceiro do projeto Orlando Verducci

Nem todas as informações medidas pela estação meteorológica são relevantes para o funcionamento do ECHARPE. Ele requisição para a estação e mostrará os dados de de: *OutsideTemp, Pressure, WindSpeed, Humidity* e as informará no display do espectrógrafo.

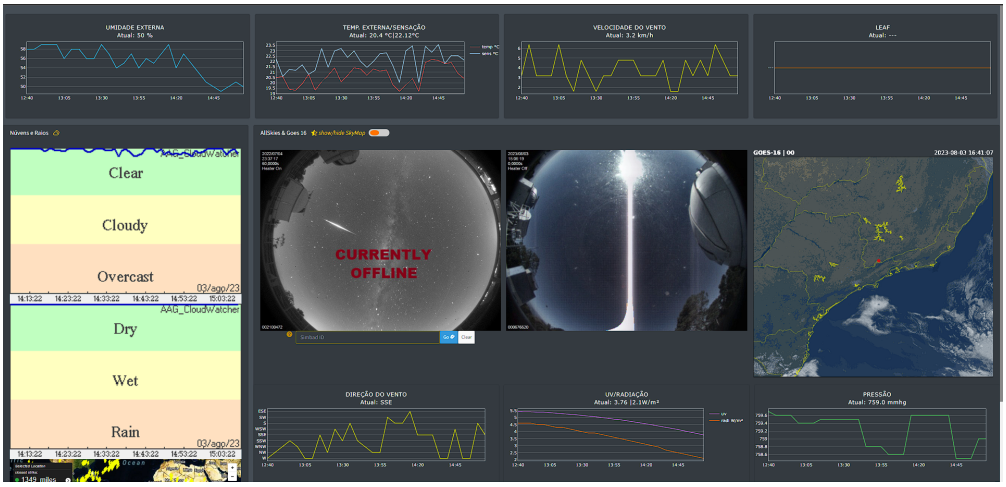

Figura 8: Interface da Estação meteorológica do LNA.

Fonte: Disponível em: http://200.131.64.237:8090/graficos/dashboard

3.2 Solução Proposta

A solução foi desenvolver uma aplicação utilizando OOP que integrará a estação meteorológica e o ICS com o ECHARPE, assim a aplicação fará um pedido de STATUS para estes dois servidores e mostrará as informações no painel frontal, além de enviar comandos para o ICS.

Como já discutido no capítulo 2 a escolha da utilização de programação orientada a objetos, por ser o método mais indicado para aplicação desse tipo, vai trazer benefícioss para ampliação e manutenção do projeto executado.

4 Projeto Detalhado

A aplicação do projeto seguirá o regime de orientação a objetos utilizando o Laboratory Virtual Instrument Engineering Workbench (LabVIEW) versão 8.20, que já vem sendo discutido ao longo desta monografia. Assim, a aplicação será desenvolvida com um conjunto de classes inter-relacionadas dentro do LabVIEW, para se comunicar com quaisquer servidores do observatório.

As aplicações da estação meteorológica e do ICS serão fornecidas nas versões simuladas pelo LNA. O escopo do projeto é a aplicação do ECHARPE que se comunica com estes servidores via TCP-IP.

4.1 Protocolo TCP-IP

Transmission Control Protocol/Internet Protocol, ou protocolo TCP/IP, é um conjunto de protocolos que possibilita a comunicação entre computadores, hoje em dia ele é a base da comunicação entre diferentes sistemas operacionais e dispositivos em todo o mundo. O Protocolo é a combinação fundamental de dois protocolos interligados, o Transmission Control Protocol (TCP) e o Internet Protocol (IP).

O protocolo TCP é o responsável de como os dados serão enviados. Ele divide os dados em pacotes e os sequencia, criando uma ordem lógica para que a informação não se embaralhe. Ainda, ele gerencia a conexão entre os dois dispositivos que será feita a troca de informação.

O protocolo IP é mais famoso devido ao termo endereço de IP. Esse protocolo define endereços únicos para cada dispositivo conectado à rede, permitindo a identificação e localização dos mesmos. Sendo assim, o protocolo IP é o conjunto de regras que definem como o endereçamento do IP sera feito para cada dispositivo na rede.

A operação do protocolo é feito em camadas, onde cada camada realiza uma função específica do sistema de comunicação como mostra a figura 6. As principais camadas são:

- **Camada de Aplicação:** É a camada que os usuários utilizam, como navegadores da web.É a camada que será elaborada nesse projeto.
- **Camada de Transporte:** É a camada onde opera o protocolo TCP, e é a responsável por tornar a comunicação confiável e coesa.
- **Camada de Rede:** É a camada onde opera o protocolo IP, gerenciando o endereço de IP dentre os dispositivos de mesma rede.
- **Camada de Enlace e Física:** Diza respeito da comunicação física dos dados entre os dispositivos em cabos como os de Ethernet ou conexões sem fio.

O LabVIEW fornece uma paleta de funções para implementação da comunicação via rede por TCP-IP, de modo que nesta aplicação da GUI do Echarpe não foi necessário desenvolver o código em baixo nível para estabelecer a conexão entre a GUI e os servidores.

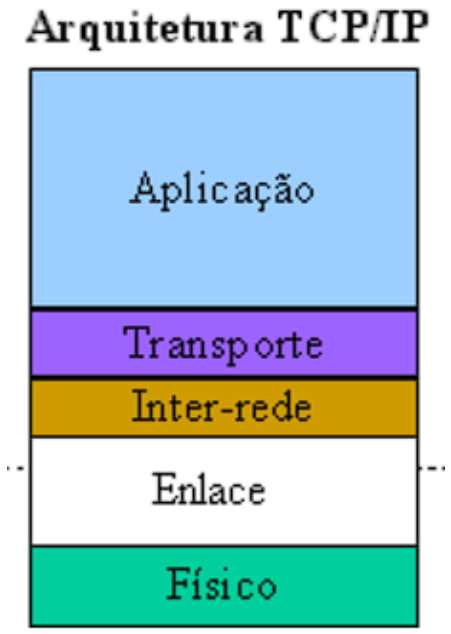

Figura 9: Estrutura da comunicação TCP-IP.

Fonte: *https* : *//www.gta.ufrj.br/grad/*03_1*/ip − security/paginas/introducao.html*

4.2 Diagramas de funcionamento

Por estarmos desenvolvendo uma aplicação utilizando programação orientada a objetos, o código é melhor representado em diagramas, com as classes definidas como retângulos, e seus respectivos métodos.

Os relacionamentos entre as classes aparecem no diagrama em formato de linhas com alguns elementos gráficos para indicar propriedades como herança. O diagrama do projeto pode ser encontrado na figura 10 e contempla todas as classes que estarão presentes.

As classes que possuem relação de herança são indicadas com setas em direção da subclasse para superclasse, ou seja, a seta indica de onde os métodos vem (classe mais especializada para mais geral). Aqui podemos pegar de exemplo as classes WeatherSensor.lvclass e TCPIP.lvclass, que herdam seus métodos das classes Device.lvclass e Comm.lvclass, respectivamente. Podemos perceber que as superclasses são mais generalizadas, como na Comm.lvclass, que tem métodos de comunicação para qualquer protocolo e na classe TCPIP.lvclass teremos configurado a estrutura do protocolo TCPIP já discutido anteriormente.

Outra relação importante é a de composição (*composition*), indicado pelos diamantes vazados no diagrama. A relação de composição permite que uma classe seja construída a partir de outras classes. Essa relação permite que a classe composta adquira indiretamente os atributos e métodos das classes que a compuseram. Isso fica claro com as classes EcharpeGui.lvclass e WeatherStation.lvclass onde

a classe da estação meteorológica não herda nenhum método da classe principal da GUI, com seu funcionamento independente dos outros métodos que serão agregados à EcharpeGui.lvclass. Além disso, a classe EcharpeGUI adquire a capacidade de se comunicar com a estação meteorológica ao ser composta pela classe WeatherStation, pois esta classe provê a comunicação com o servidor da estação meteorológica. Na verdade, o código da comunicação está contido na classe WeatherStation (filha de Server) devido a uma outra composição: desta com a classe TCPIP (filha da classe Comm), que contém o código para comunicação externa.

4.3 As classes

Foram desenvolvidos mais de 90 métodos para a GUI do Echarpe. A figura 10 mostra todas as classes, seus relacionamentos, bem como os mais relevantes métodos da aplicação dentre os 90 citados. A seguir consta a explicação do funcionamento de cada classe da GUI.

4.3.1 Comm.lvclass

É a classe onde são definidos os métodos base de comunicação entre as aplicações, o método Transmit (que é responsável pela transmissão de dados da aplicação), o método Receive (que é responsável pelo recebimento de dados pela nossa aplicação), e os métodos que criam e destroem o canal de comunicação, Init e End, respectivamente.

4.3.2 TCPIP.lvclass

A classe TCPIP é filha de Comm, portanto herda os métodos citados anteriormente. Essa classe foi criada para implementar especificamente a comunicação via protocolo TCP-IP. Sendo assim, os métodos base Transmit e Receive da classe Comm são sobrescritos pelos respectivos métodos da classe filha TCPIP, que contêm o código da comunicação via rede.

4.3.3 Server.lvclass

Classe que representa qualquer aplicação servidor, que fornece dados e serviços para a GUI do Echarpe. Nesta classe se define o método Handle que vai prover a comunicação e tratamento de mensagens trocadas entre o Echarpe e o servidor. Sendo um dos métodos mais importantes para o funcionamento da GUI, o método Handle é herdado pelos filhos de server onde é particularizado nas necessidades do servidor em questão.

4.3.4 StateMachine.lvclass

A classe StateMachine é a responsável por gerenciar a execução do tratamento de mensagens de cada dispositivo controlado pela GUI do instrumento por meio de sequenciamento de estados pré definidos. Cada estado é processado em um único ciclo de execução do loop principal e deve ter o controle de tempo máximo de permanência no mesmo estado. O estado inicial de cada processo definido na máquina de estados é carregado a partir de um gatilho externo (do usuário) e pode ser diferente em cada dispositivo, pois todo dispositivo tem sua própria máquina de estados. Esta classe

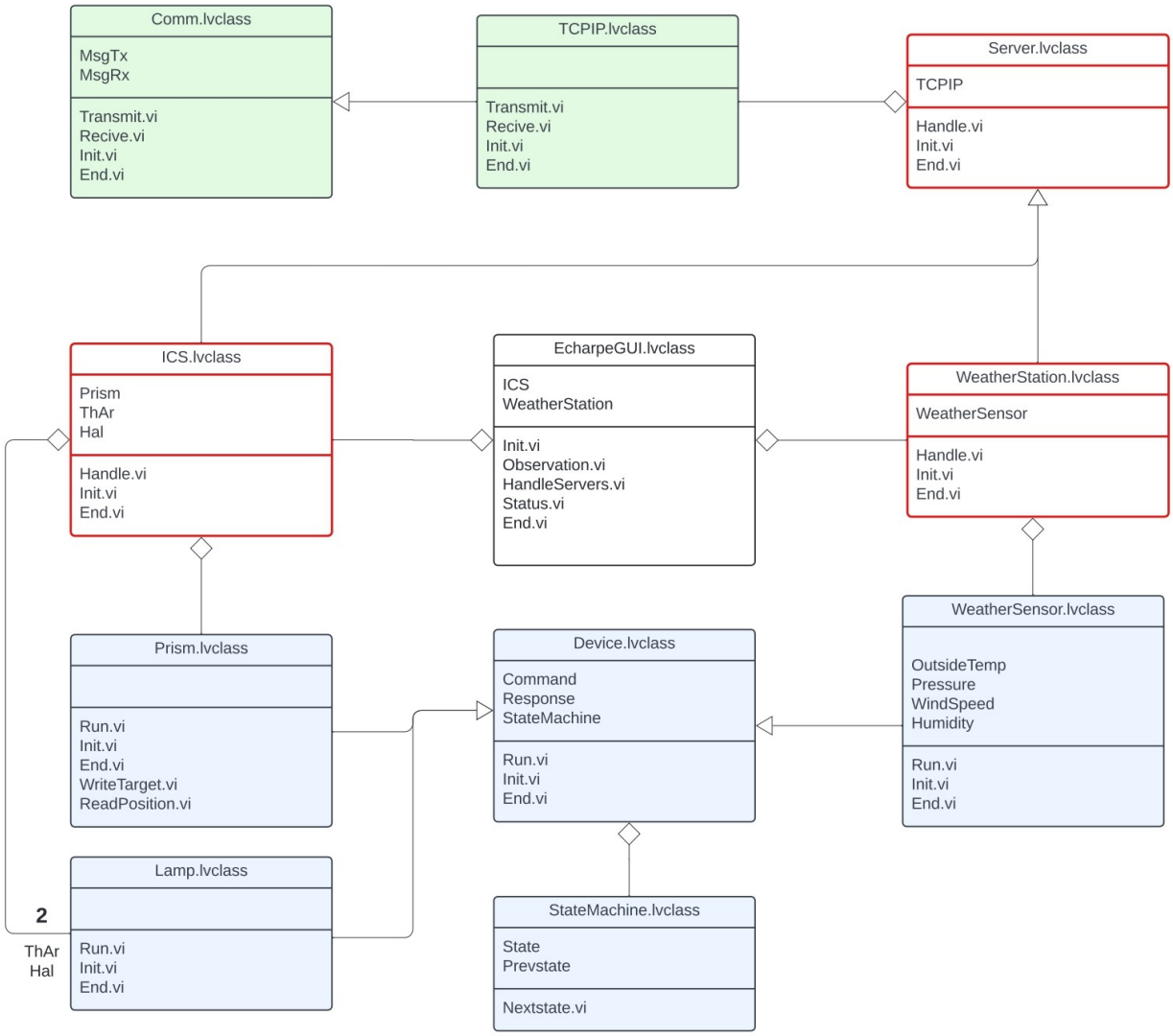

Figura 10: Diagrama de classes do projeto.

Fonte: Disponível em: *https : //lucid.app/lucidchart/ddc1cf58 − 725e − 473f − b294 − bd580c5ccad9/edit?viewport*$_{l}$*oc = −5892C − 4342C28892C13592CHWEp − vi − RS FOinvitationId = inv*$_{1}$*919a8a4 − 571b − 4eb5 − b3c3 − 8a4b6f2d93bd*

compõe a classe Device.lvclass.

4.3.5 Device.lvclass

É a classe mais generalista para representação dos dispositivos, portanto (como podemos ver na figura 10) a classe Device é a classe mãe da WeatherSensor, Prism, Lamp e qualquer outro

dispositivo que precise ser controlado ou monitorado. Esta classe provê o método Run.vi, que é sobrescrito pelas classes filhas, pois deve conter o código de geração dos comandos e tratamento das respostas do dispositivo específico, acessando os atributos Command e Response da classe mãe para enviar e receber mensagens do dispositivo físico.

4.3.6 WeatherStation.lvclass

A classe WeatherStation é a classe que representa o servidor da estação meteorológica. Sendo filha de Server, a classe funciona como meio de comunicação com a estação meteorológica, por meio do seu método Handle.vi, sobrescrito da classe mãe. Assim, o carregamento dos atributos desta classse, como a temperatura, acontece após o recebimento do status deste servidor.

4.3.7 ICS.lvclass

Assim como a classe citada anteriormente, a ICS é classe filha de Server, mas representa outro servidor, o Instrument Controler Server. É com esta classe que se realiza a comunicação para controlar três dispositivos do instrumento: o prisma e as duas lâmpadas. O método Handle.vi da classe ICS é um pouco mais complexo do que o da classe WeatherStation, pois o ICS além de responder sobre o estado dos dispositivos, recebe solicitações de ação por meio de comandos enviados pela GUI, como mover o prisma e acender ou apagar cada lâmpada.

4.3.8 WeatherSensor.lvclass

Sendo filha da classe Device, a WeatherSensor representa um dispositivo de WeatherStation, que é a telemetria das informações climáticas. É por meio desta classe que são tratados e separados os dados que vem da estação meteorológica. Os atributos dessa classe são encapsulados de modo que só os seus próprios métodos têm acesso aos dados e estes métodos são chamados apenas pela classe WeatherStation, que é a classe servidor que contém o dispositivo WeatherSensor.

4.3.9 Prism.lvclass

A classe Prism representa um prisma giratório que compensa a rotação da imagem do céu durante a observação. Assim como a classe WeatherSensor, também é filha de Device e herda seus métodos, mas, como já citado na descrição da classe ICS, classe que estabelece a comunicação com Prism, esta classe trata comandos que mudam o seu estado. O prisma físico é acionado durante uma observação ao longo da noite e periodicamente deve receber uma correção da sua posição com desclocamentos angulares de 0° até 359°). No modo de calibração este dispositivo fica parado em 0°. No modo engenharia, é possível posicionar o prisma com qualquer ângulo desejado.

4.3.10 Lamp.lvclass

A classe Lamp, tal qual a classe Prism, é filha de Device e é um dispositivo do ICS. Ela representa uma lâmpada de calibração do instrumento. Como há duas lâmpadas de calibração no Echarpe, há duas instâncias desta classe no código da GUI Echarpe. Ambas instâncias de Lamp possuem os mesmos atributos e os mesmos métodos, pois são representantes da mesma classe, mas

possuem controle e estado totalmente independentes. Assim, é possivel acionar uma lâmpada de cada vez no modo calibração e a leitura do estado das lâmpadas é individualizado. No modo de observação, as lâmpadas devem permanecer apagadas.

4.3.11 EcharpeGui.lvclass

É a classe principal da aplicação, pois contém a representação do instrumento completo e dos seus canais de comunicação com as outras aplicações do sistema de controle do observatório. É a partir dela, com a execução do módulo Main.vi, que todas as classes da aplicação são instanciadas e seus métodos acionados. O loop principal de Main.vi executa de modo cíclico três métodos da classe Echarpe: Observation.vi, que trata da execução dos modos de operação do instrumento; HandleServers.vi, que trata toda a comunicação de mensagens entre servidores; e por fim, Status.vi, que apresenta o status dos dispositivos ópticos do instrumento, além de informações da estação meteorológica.

4.4 Funcionamento da Aplicação

Os eventos principais da GUI acontecem a partir de sua classe principal, a EcharpeGui. Ela é composta pelas classes filhas de Server, i.e., WeatherStation e ICS, e, a partir destas, implementa os métodos de comunicação e tratamento de mensagens para controle e monitoramento do instrumento. O módulo principal da aplicação é chamado de Main.vi (Apêndice A), ele instancia a classe EcharpeGUI que contém o loop principal de execução, para iniciar a aplicação com o método Init. Init, como seu oposto End, estão fora da estrutura de *while*, e são executados apenas uma vez (no inicio e fim) da execução de Main. Dentro do while há uma estrutura de tratamento de eventos do usuário, com o evento timeout executando, a cada 100ms, os três métodos principais da classe EcharpeGUI (Observation.vi, HandleServers, Status.vi), como mostra o Apêndice A.

O evento de timeout é o padrão da aplicação, onde ela fica esperando o *clock* dar *timeout* para que ele seja executado repetidamente até que a aplicação seja encerrada. O tempo de timeout foi definido em 100ms e assim, a cada 100 milisegundos o ciclo se reinicia. Primeiramente a aplicação executa o Observation.vi,onde está implementado o controle de execução de uma observação, que compreende, entre outras ações, disparar e monitorar o movimento dos mecanismos para a configuração escolhida pelo usuário e iniciar e monitorar a aquisição de imagens nos dois canais para o completo gerenciamento da observação. Após, estabelece uma comunicação com as demais aplicações (com o método HandleServers (Apêndice B)).

O método Handleservers atende as solicitações originadas em Observation, dos diferentes servidores da aplicação, que no escopo desse projeto são dois, mas podendo ser ampliados devido a escalabilidade do código. Os servidores são acionados através de seu método SERVER.Handle, como no Anexo C, que é um exemplo de Handle da classe ICS. O método HandleICS utiliza um operador do LabVIEW de *Unbundle by name* para acessar os diferentes dispositivos do Instrument Control Server. O primeiro dispositivo da saída da função é o Prisma, com sua classe Prism.lvclass, que é executado a partir de seu método Run.vi, que podemos ver no Apêndice C em amarelo, já os dois abaixo são as lâmpadas de ThAr e Halógena, ambas instâncias da classe Lamp e executadas com seu método Run.vi em roxo. Após a execução de Run, o método HandleServers concatena os pedidos dos diferentes dispositivos em Write MsgTx.vi e envia usando Transmit.vi na parte superior direita.

Cada dispositivo tem seu respectivo com o método Run. Nos Anexos D e E, é possivel observar dois estados para o Run da classe Prism. Para o estado 1 (Apêndice D) podemos observar o caso *default*, onde o dispositivo carrega um pedido de STATUS, para que o ICS responda o ângulo no qual o prisma se encontra. Já no Anexo E nós temos o estado 10, que é ativado quando queremos apontar um alvo para o prisma girar, assim escrevemos o valor de *TARGET* o método HandleServers concatena com os demais pedidos para o Servidor ICS como comentado no parágrafo anterior. Assim, após a execução de HandleServers e o envio dos pedidos, o próximo método a ser executado em Main para o caso *Timeout* é o método Status.vi (Apêndice G) que também através de um *Unbundle by name* faz a leitura nos dispositivos. O resultado de cada um é colocado do lado de fora do case em uma variável para que seja apresentado no painel.

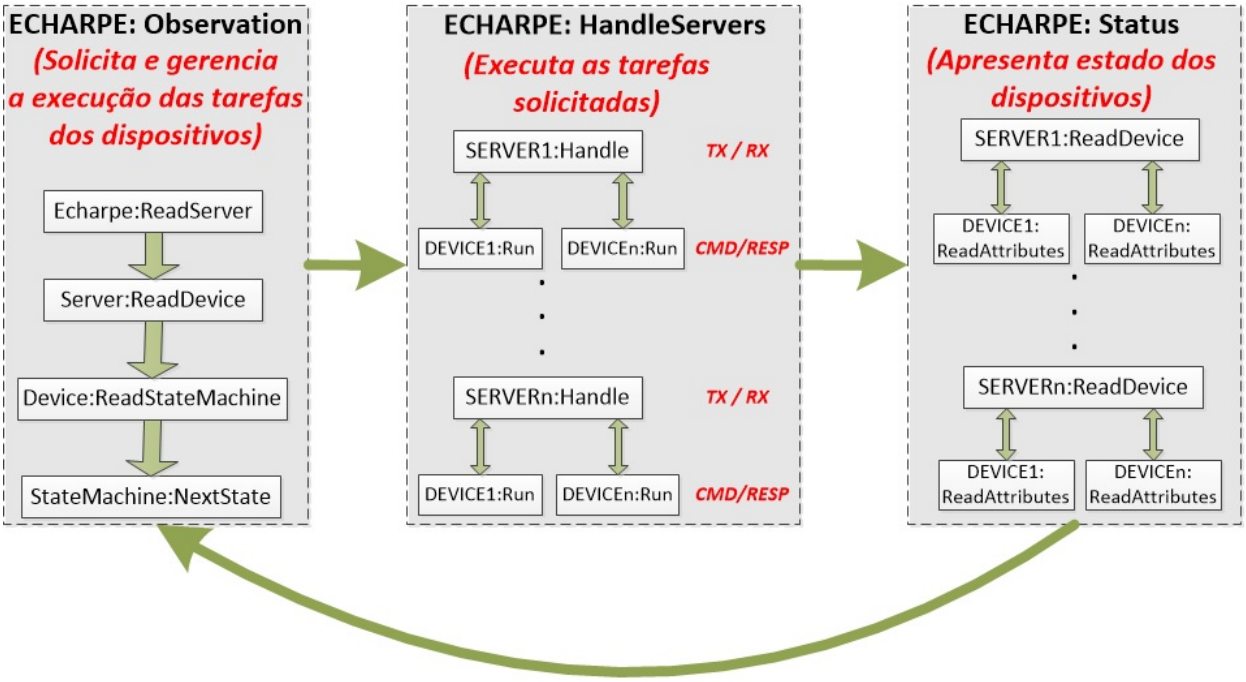

Figura 11: Blocos funcionais da aplicação.

Fonte: Disponível em: https://drive.google.com/file/d/17omhuNzMeTVG4f5MSnYgQBShw6$_1$TQ1X/view?usp = *drive$_l$ink*

A figura 11 é uma representação da aplicação, nela podemos perceber que o funcionamento da aplicação ocorre sequencialmente. Primeiro nós temos o bloco que gerencia as tarefas dos dispositivos, com o método Observation da classe Echarpe (ECHARPE.Observation). É nessa camada onde a aplicação conversa com o usuário por meio do painel e carrega nos dispositivos o próximo estado.

No segundo bloco temos a camada de comunicação que envia os comandos para os dispositivos, através de ECHARPE.HandleServers. Este método gerencia os servidores específicos e os aciona através de seu método SERVER.Handle, que por sua vez inicia seus respectivos dispositivos com o método DEVICE.Run carregados com o estado desejado no primeiro bloco.

Após os dispositivos serem acionados coletamos seus atributos. Por estarmos tratando de programação orientada a objetos, esse atributo não pode ser passado diretamente pra classe ECHARPE para ser mostrado no painel e sim precisa ser feito o caminho inverso para a sua apresentação. Atributo do dispositivo é lido pelo seu respectivo servidor, com SERVER.ReadDevice e então a classe Echarpe pode acessar seu valor, com a classe ECHARPE.Status, como mostra o terceiro bloco.

4.5 O painel principal

O painel da aplicação (figura 12) é o que podemos entender como o *Frontend* do método Main.vi. É nele onde o usuário poderá visualizar as informações coletadas pelos servidores de comunicação da GUI e configurar o instrumento para as aquisições de imagens de ciência e de calibração. Ele foi baseado na interface do polarímetro SPARC4 (Simultaneous Polarimeter And Rapid Camera in 4 bands), que também foi construído pelo LNA e tem grande influência na construção do Echarpe.

Na parte superior esquerda, podemos observar o nome do instrumento e o botão *Exit*, que encerra a aplicação, saindo do loop principal e executando o método End.vi. Logo abaixo temos os Leds de identificação dos status dos servidores. Como comentado anteriormente, este é um embrião do que será a GUI final do Echarpe, portanto, temos apenas dois leds acesos, que correspondem aos servidores do ICS e da Estação Meteorológica. Eles são acesos graças a um método chamado *Read Active* que informa se a comunicação com o servidor está ativa; a ativação da comunicação se dá sempre que há algum conteúdo na fila de comunicação do servidor, e a desativação da comunicação é detectada quando o servidor não responde as solicitações da GUI por um período maior do que o estabelecido para o servidor. Sua execução acontece em Status.vi (Apêndice G) logo após a função *Unbundle by name* e o resultado do método, que acende o led no painel frontal, é facilmente observado com a linhas verdes tracejadas que saem do método. Os demais servidores, como o TCSPD (Telescope Control System Pico dos Dias), os .dois canais de aquisição, o focalizador e o *Guider*, que faz a guiagem do telescópio durante a observação, ainda não estão operacionais. Os demais servidores do painel não estão implementados, mas a aplicação da GUI do Echarpe foi desenvolvida para ser facilmente expandida, adicionando novas classes específicas de Server na composição da classe EcharpeGUI e adicionando novas classes especificas de Device na composição de cada nova classe específica de Server, pois cada servidor tem pelo menos um dispositivo para ser controlado ou monitorado.

Mais abaixo temos dois campos onde o astrônomo preencherá e .adicionará, como keywords, ao *Header* da imagem com o botão *SET*. No final da coluna da esquerda nós temos os campos com várias informações, muitas não estão disponíveis como a ascensão reta e declinação do apontamento do telescópio (TCSPD) mas já podemos observar as informações vindas da Estação Meteorológica de testes executados localmente.

Na coluna da direita na parte superior temos as informações e controladores referentes ao Instrument Control Server. Os botões *Calib* e *Obs* configuram modos de operação para a GUI. O botão *Calib* indica que o usuário quer colocar a aplicação em modo de calibração, onde as lâmpadas devem ficar acesas e o prisma girado para a posição zero. Para isso o botão gera um evento que coloca a estrutura de eventos de Main.vi num estado diferente do *default*, como podemos ver no Apêndice H. Assim nós carregamos o estado 10 na na máquina de estados implementada no método Observation.vi da classe EcharpeGUI e esperamos até o *Timeout*. Após o *Timeout*, a estrutura de eventos de Main.vi volta para seu estado padrão(Apêndice A), mas agora com o estado 10 em EcharpeGUI: Observation.vi carregado. Então o funcionamento da GUI muda, o método Observation.vi (Apêndice I) sai do repouso, prepara o movimento do prisma carregando um 0 no

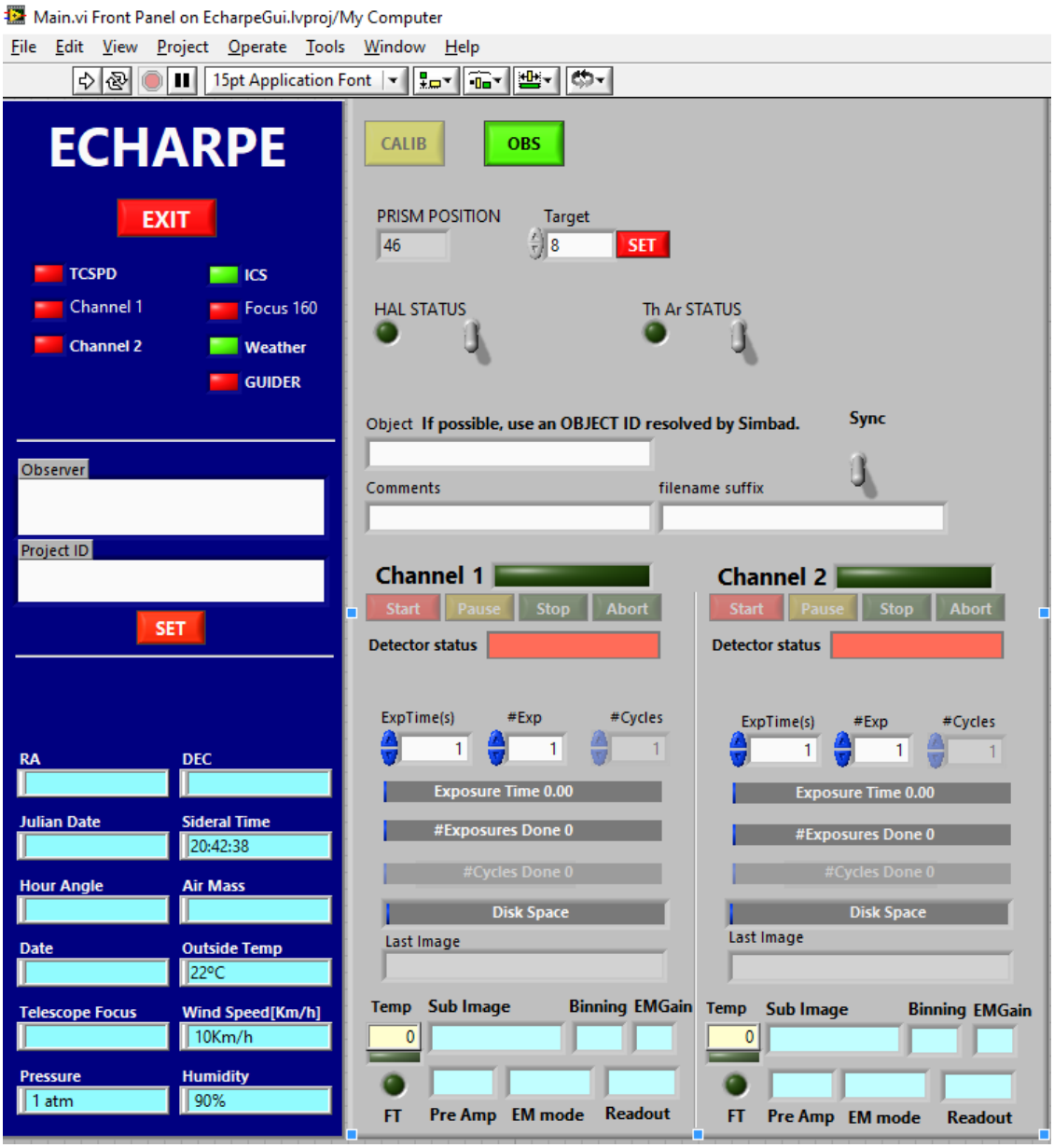

Figura 12: Painel frontal da aplicação (baseado intencionalmente na GUI da SPARC4 por INPE e LNA).

Fonte: Disponível em: https://drive.google.com/file/d/17omhuNzMeTVG4f5MSnYgQBShw6$_1$TQ1X/view?usp = *drive$_l$ink*

target do prisma com o método Write Target. O funcionamento dos dispositivos acionados por HandleServers é impactado pela ação executada em Echarpe:Observation.vi, como podemos ver no apêndice E. Onde o prisma em vez de fazer um pedido de status, concatena uma mensagem de *Prism + Target* para enviar ao ICS. No apêndice J podemos ver o que acontece com a classe Lamp

no estado 10 de Lamp:Run.vi, ela concatena com o nome da lâmpada (segundo método do *case*) com = *ON*, que é enviado e interpretado pelo ICS ligando a lâmpada. É possivel observar que em ambos exemplos de Run.vi no estado 10, após a sua execução, o estado 0 é carregado novamente, significando que na próxima iteração de Main.vi os dispositivos voltarão a fazer pedidos de Status recebendo o valor modificado pelo botão.

O botão Obs indica que o usuário deseja colocar a GUI em modo de observação. Assim, as lâmpadas devem ser desligadas e o Prisma entra em seu modo de observação. O processo é muito semelhante ao do parágrafo acima, que ao clicar no botão o método principal carrega oestado 20 em Echarpe:Observation.vi e desabilita o botão *calib*, como mostra o apêndice K. A diferença principal acontece no método Observation, que após o *Timeout* carrega o estado 20 nas lâmpadas (Lamp:Run.vi), desligando-as, e soma 1 no seu estado atual. Assim o método Observation vai para o estado 21 (Apêndice F), que diz como o prisma deve se comportar. No estado 21 podemos encontrar alógica de atualização do ângulo prisma, neste trabalho foi implementada uma lógica simples de incrementar o ângulo do prisma periodicamente, mas na versão final deverá ser inserida a equação do movimento do prisma para se obter a sua posição em função do apontamento do telescópio, essa lógica permite que o prisma fique girando de 0 até 359°, de 2 em 2°, simulando a compensação do moviment o movimento de rotação aparente dos corpos celestes.

Abaixo dos botões *Calib* e *Obs* estão os controles individuais dos dispositivos do ICS. Primeiro nós temos o prisma, onde na esquerda é possivel ver sua posição corrente em graus e na direita o campo para inserir a posição desejada e o botão para iniciar o movimento. Assim como nos casos anteriores, o botão *SET* muda o caso na estrutura de eventos de Main.vi (Apêndice N), carregando o valor desejado pelo usuário na variável em laranja Target Prism e levando o dispositivo para o estado 10, onde como já vimos anteriormente (Apêndice E), coloca na fila de comandos um pedido de movimentação do prisma.

As lâmpadas mais abaixo, são contoladas pelas chaves logo abaixo de seu nome com seu respectivo Led mostrando seu estado. O evento que elas geram também troca o caso de Main.vi (Apêndices L e M) onde nós temos o estado desejado sendo carregado. Pela chave se tratar de uma variável booleana, *ON* ou *OFF* não se está atribuindo um valor a lâmpada, mas sim, está apenas sendo trocado o estado do dispositivo. Para isso, utilizamos um seletor. Se .posição da chave estiver para cima, o seletor escolhe o estado 10 em Lamp:Run.vi para as lâmpadas, que como vimos no apêndice J, acende as lâmpadas, já se o seletor estiver para baixo, ele escolhe o estado 20 em Lamp:Run.vi, que apaga as lâmpadas.

As funcionalidades abaixo das lâmpadas não fazem parte do escopo desse projeto e portanto não foram configuradas. Ali ficam as funcionalidades referentes ao nome do arquivo de imagem gerado pelo sistema de aquisição da observação, com o nome do objeto observado e o tipo de arquivo no sufixo do nome. Por fim, nós temos os canais do espectrógrafo, que mostrarão as informações referentes as aquisições em cada um dos dois canais de fibra óptica, em seu respectivo comprimento de onda.

5 Conclusão

A construção de um espectrógrafo como o ECHARPE será um grande reforço para a astronomia observacional brasileira. A espectroscopia é um dos campos da astronomia com maior interesse pela comunidade científica brasileira e que traz muitas informações novas sobre o universo, que tanto interessa ao público geral. Ter um espectrógrafo feito por brasileiros, em território nacional, é uma grande prova da capacidade da capacidade da ciência brasileira, e mostra o grande podencial do desenvolvimento científico nacional.

Apesar de não estar completa, a GUI do ECHARPE se mostrou eficiente, coerente e confiável quando utilizada com os servidores de teste. Disponível para download em `https://github.com/EduGod0y/EcharpeGUI` é possivel testar a aplicação localmente, basta abrir o projeto e iniciar o arquivo Main.vi, assim como iniciar os servidores de testes com os métodos Main_WS.vi e Main_ICS.vi, nas pastas *Weather Station Server Folder LV 8_2* e *Instrument Control Server Folder* respectivamente.

A utilização de LabView para a construção da aplicação se mostrou fundamental, pois o seu código é desenvolvido na forma de diagramas de blocos e fluxo de dados, o que é familiar para cientistas e engenheiros. A forma intuitiva no gerenciamento dos servidores, das filas de comunicação e das classes em si é uma vantagem gigante do LabView em relação as outras linguagens. Além de um código gráfico de fácil entendimento, o Labview proporciona a utilização de orientação a objetos, que no contexto dessa aplicação, foi fundamental na replicação dos dispositivos, modularização do código e se mostrará ainda mais importante na continuação desse projeto, pois oferece a escalabilidade necessária para a sua implementação completa.

6 Referências

[1] DUARTE, Otto Carlos Muniz Bandeira. *IP Security*. Redes de Computadores 1 – 2003/1. Disponível em: https://www.gta.ufrj.br/grad/03_1/ip-security/. Acesso em: 05 de Fevereiro 2024.

[2] UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL. *Espectroscopia*. Disponível em: https://www.if.ufrgs.br/fis02001/aulas/aula_espec.htm#:~:text=Espectroscopia&text=A%20luz%20branca%2C%20ao%20passar,da%20luz%20se%20chama%20espectro.&text=O%20espectro%20nos%20dÃą%20grande,estrelas,%20principalmente%20temperaturas%20e%20raios. Acesso em: 10 de Novembro 2023.

[3] OLIVEIRA FILHO, Kepler de Souza; SARAIVA, Maria de Fátima Oliveira. *Espectroscopia*. Disponível em: http://astro.if.ufrgs.br/rad/espec/espec.htm. Modificado em: 23 ago. 2022. Acesso em: 05 maio 2024.

[4] UNIVERSIDADE DE SÃO PAULO. *Glossário de Espectrógrafo*. Disponível em: http://www.astro.iag.usp.br/~ronaldo/intrcosm/Glossario/Espectrografo.html. Acesso em: 14 de abril 2024.

[5] OBSERVATÓRIO NACIONAL. *Telescópios do OPD*. Disponível em: https://www.gov.br/lna/pt-br/composicao-1/coast/obs/opd/telescopios/telescopios-do-opd. Publicado em: 17 dez. 2020. Atualizado em: 16 fev. 2022. Acesso em: 14 abril 2024.

[6] HENRIQUE, João. *Programação Orientada a Objetos*. Disponível em: https://www.alura.com.br/artigos/poo-programacao-orientada-a-objetos. Atualizado em: 18 set. 2023. Acesso em: 10 maio 2024.

[7] CLEMENTE, Paulo. *Entendendo a Abstração em POO com Java*. Disponível em: https://blog.rocketseat.com.br/entendendo-a-abstracao-em-poo-com-java/. 31 jan. 2024. Acesso em: 13 maio 2024.

[8] VERDUCCI JUNIOR, Orlando. *Programação Orientada a Objetos em LabVIEW para uma Aplicação Distribuída de Controle de Posição de um Telescópio*. Universidade Federal de Itajubá, Curso de Especialização em Engenharia Web.

Apêndices

A Main

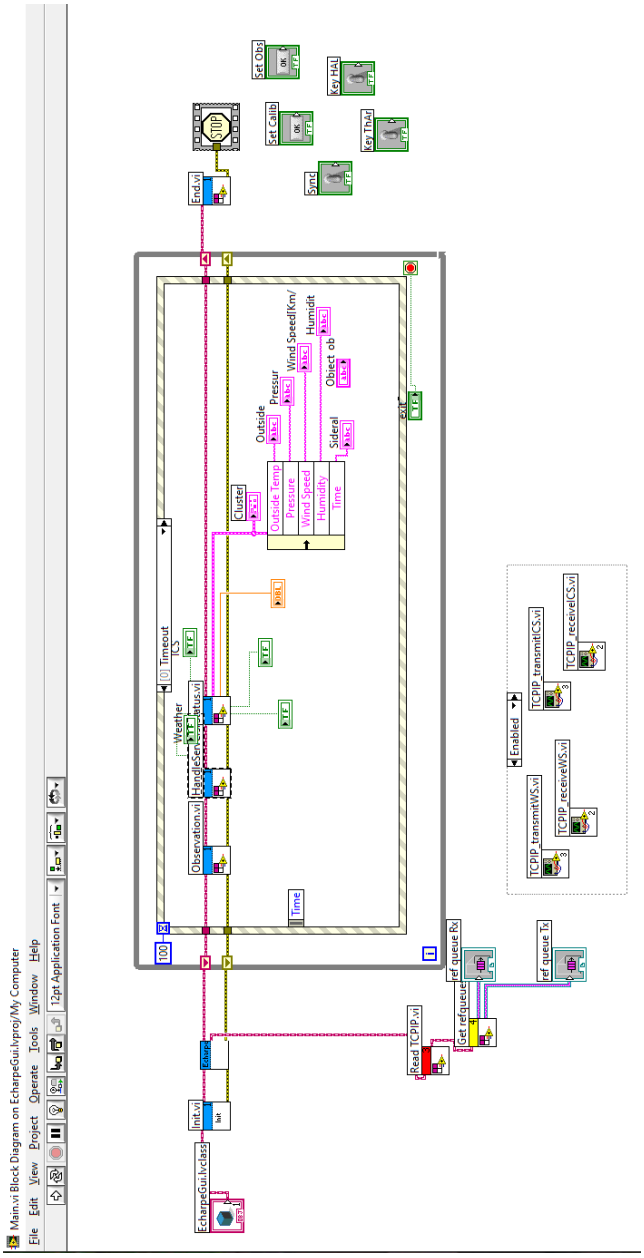

Figura 13: Estrutura do método Main.

B HandleServers

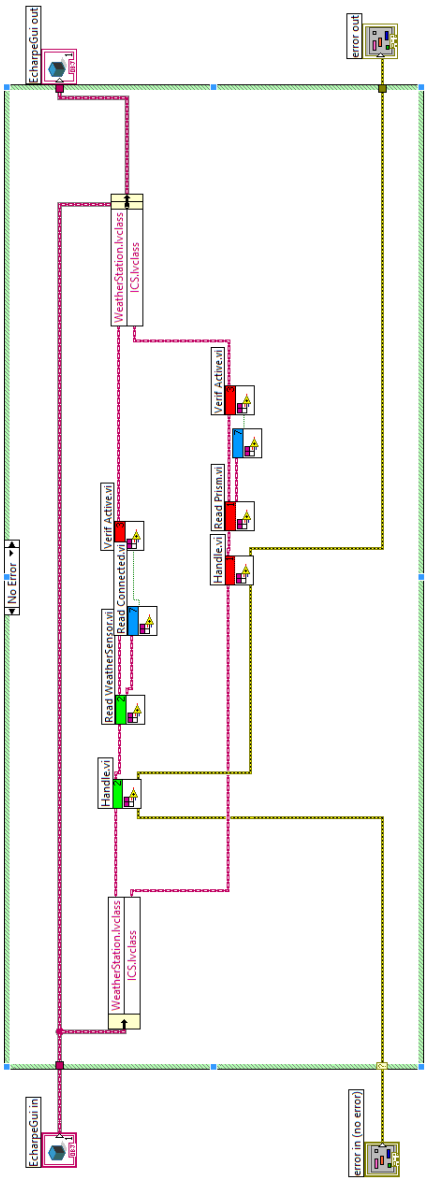

Figura 14: Estrutura do método HandleServers.

C HandleICS

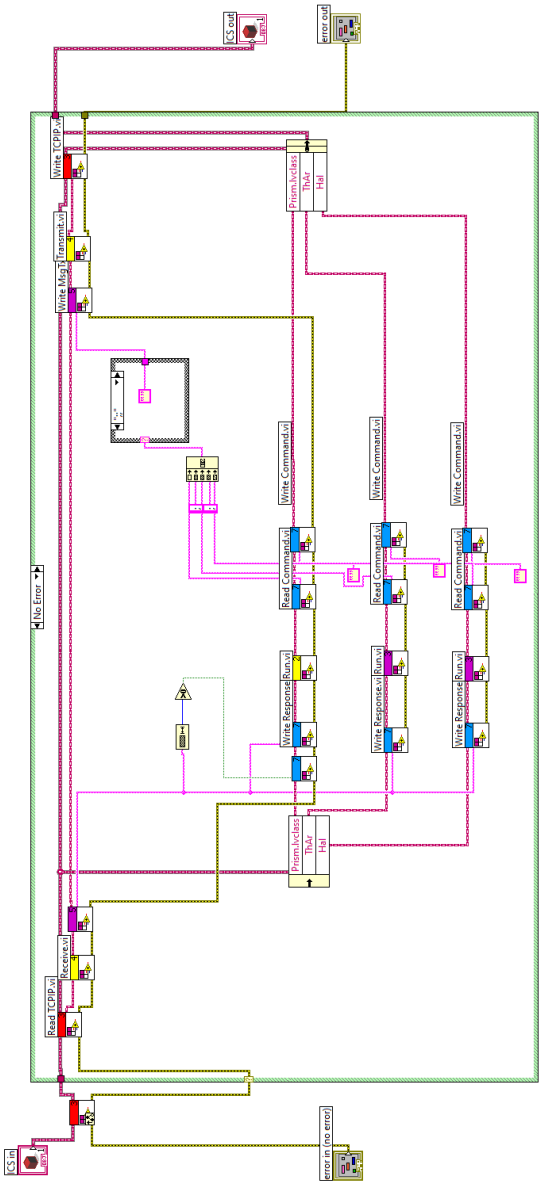

Figura 15: Estrutura do método Handle da classe ICS.

D RunPrism estado 1

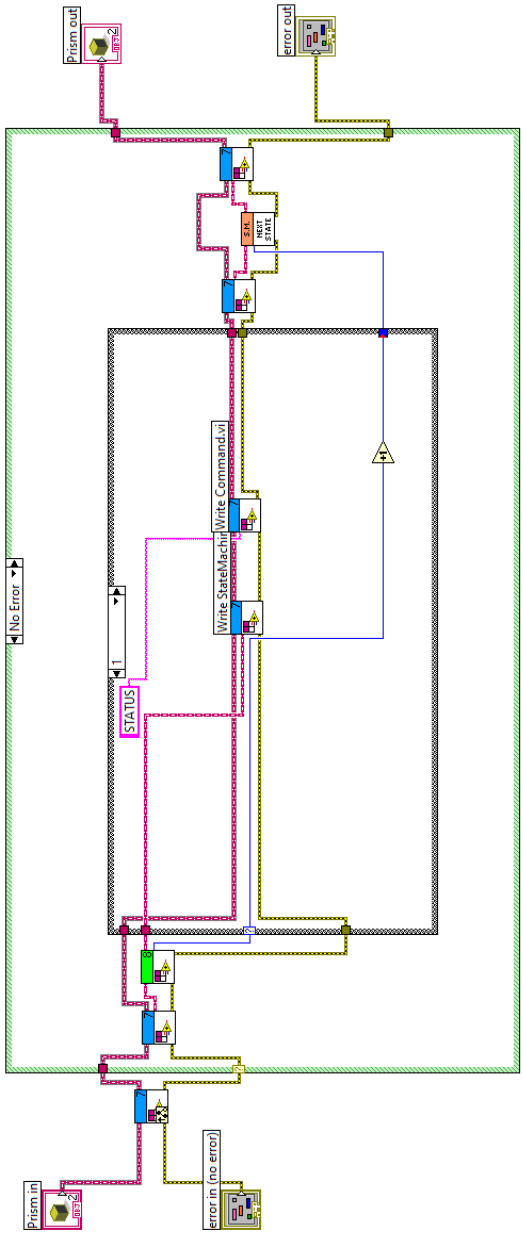

Figura 16: Estrutura do método Run da classe Prism no estado 1.

E RunPrism estado 10

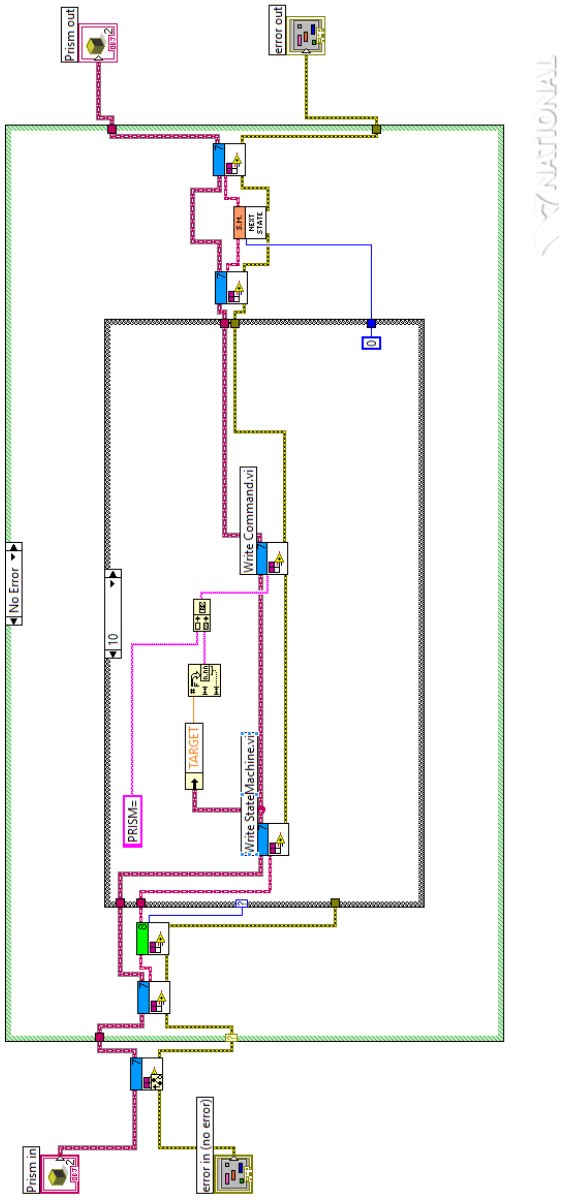

Figura 17: Estrutura do método Run da classe Prism no estado 10.

F Observation

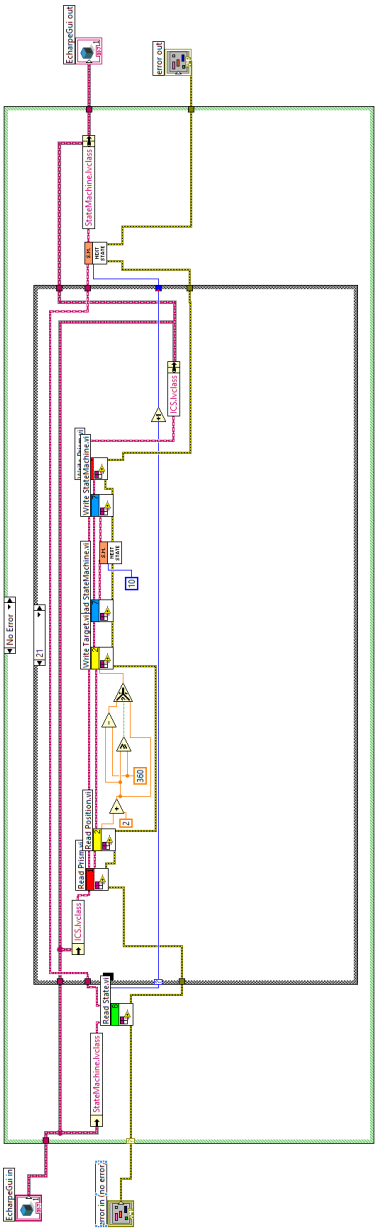

Figura 18: Estrutura do método Observation da classe EcharpeGUI.

G Status

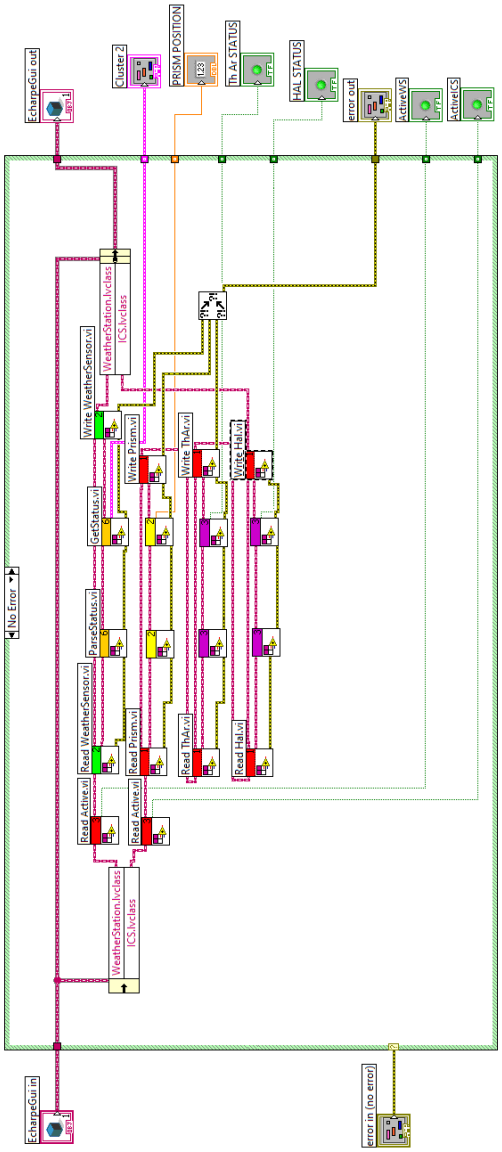

Figura 19: Estrutura do método Status.

H Main Set Calib

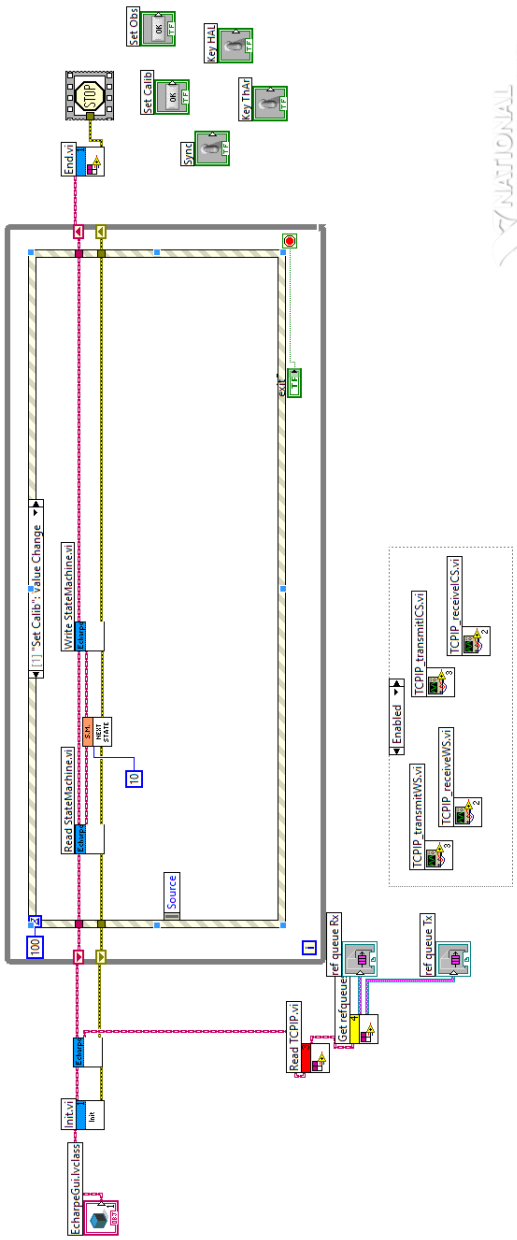

Figura 20: Estrutura do método Main ao clicar no botão Calib.

I Observation estado 10

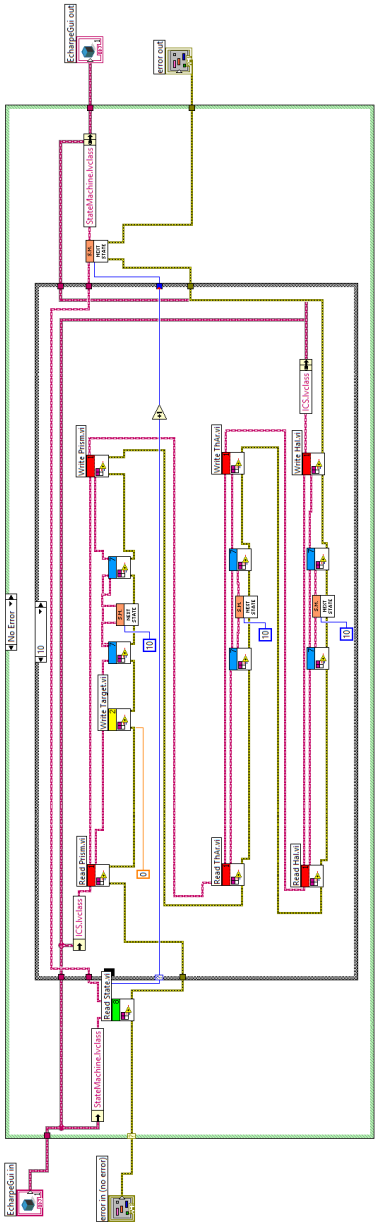

Figura 21: Estrutura do método Observation no estado 10.

J Lamp Run estado 10

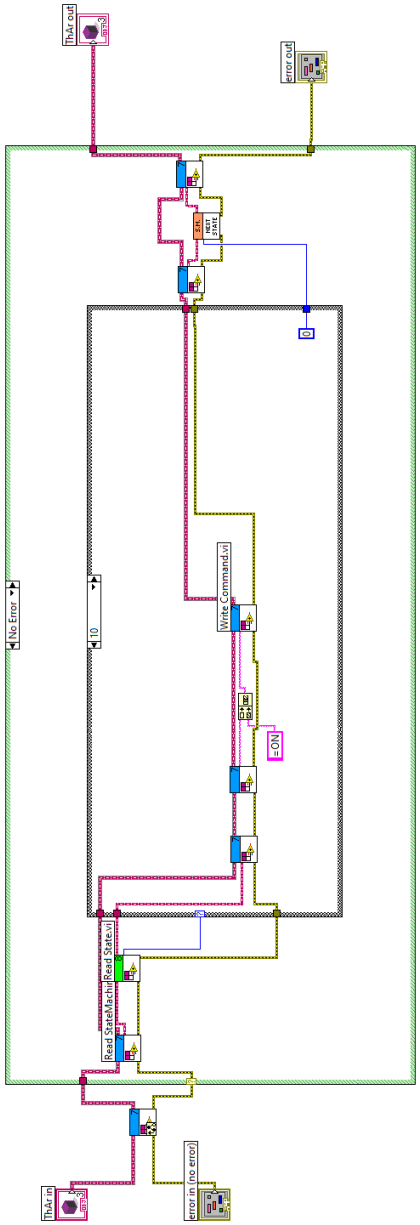

Figura 22: Estrutura do método Run da classe Lamp no estado 10.

K Main Set Obs

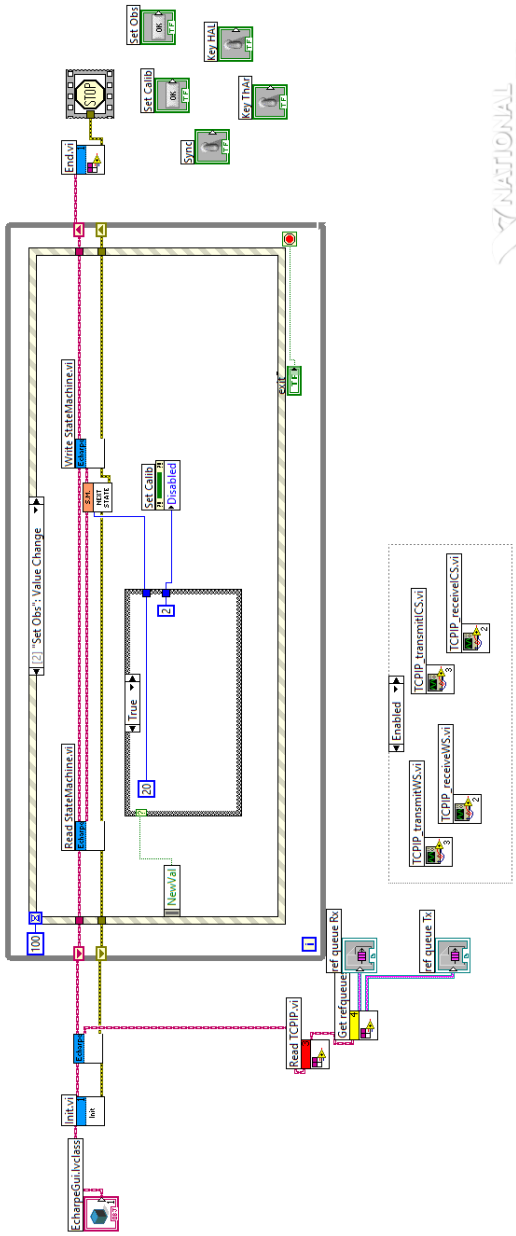

Figura 23: Estrutura do método Main ao clicar no botão Obs.

L Main Set Hal

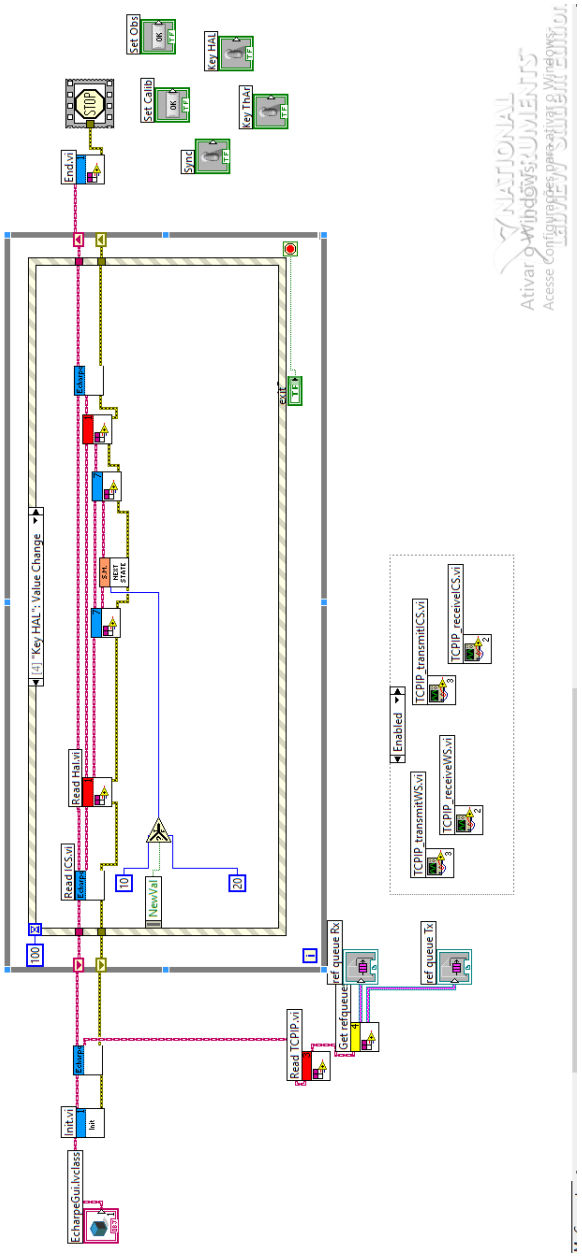

Figura 24: Estrutura do método Main ao clicar na chave que liga e desliga a lâmpada halógena.

M Main Set ThAr

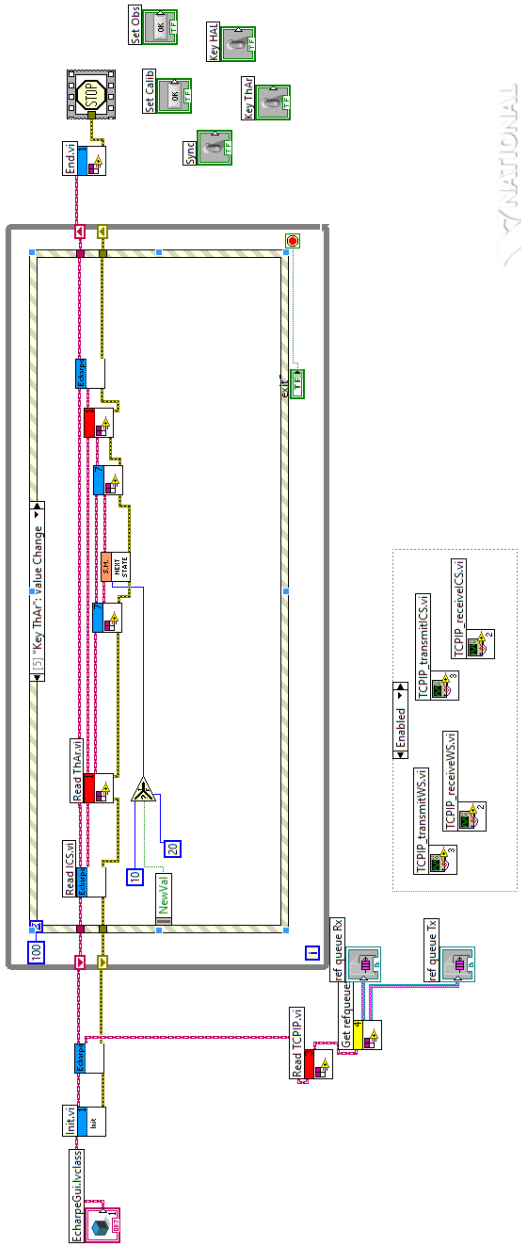

Figura 25: Estrutura do método Main ao clicar na chave que liga e desliga a lâmpada de ThAr.

N Main Set ThAr

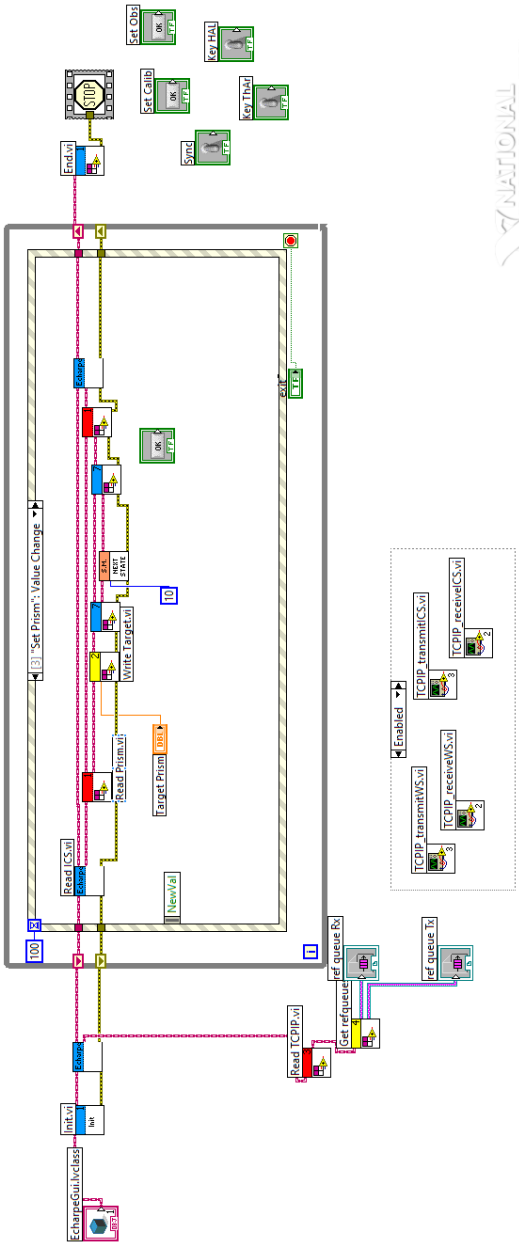

Figura 26: Estrutura do método Main ao clicar no botão que faz um pedido para girar o prisma.