Universidade Federal de Juiz de Fora
Faculdade de Engenharia
Curso de Engenharia Elétrica

Raphael Maghamez Mendes

SISTEMA AUTOMATIZADO DE ENSAIOS PARA AVALIAÇÃO DE PARÂMETROS ELÉTRICOS EM CONTROLADORES DE LED

Juiz de Fora
2016

Raphael Maghamez Mendes

Sistema automatizado de ensaios para avaliação de parâmetros elétricos em controladores de LED

Monografia submetida ao corpo docente do Curso de Engenharia Elétrica da Universidade Federal de Juiz de Fora, como parte dos requisitos necessários para a obtenção do título de bacharel em Engenharia Elétrica.

Orientador: Prof. Guilherme Márcio Soares, M. Eng

Juiz de Fora

2016

Raphael Maghamez Mendes

Sistema automatizado de ensaios para avaliação de parâmetros elétricos em controladores de LED

Monografia submetida ao corpo docente do Curso de Engenharia Elétrica da Universidade Federal de Juiz de Fora, como parte dos requisitos necessários para a obtenção do título de bacharel em Engenharia Elétrica.

Aprovada em 11 de março de 2016.

BANCA EXAMINADORA:

Prof. Guilherme Márcio Soares, M. Eng
Universidade Federal de Juiz de Fora, UFJF

Prof. Estêvão Coelho Teixeira, Dr. Eng
Universidade Federal de Juiz de Fora, UFJF

Alcindo Gandhi Barreto Almeida, M. Eng
Universidade Federal de Juiz de Fora, UFJF

Fernando José Nogueira, M. Eng
Universidade Federal de Juiz de Fora, UFJF

Aos meus pais, Ralph e Rita

AGRADECIMENTOS

Gostaria de agradecer a Deus por ter me iluminado e a todas as pessoas que contribuíram ao longo do caminho. Primeiramente, aos meus pais por toda a educação que me proporcionaram, pelo apoio incondicional e por sempre terem confiado em mim e no meu potencial. Além disso, a toda minha família que me apoiou de alguma forma.

Ao meu orientador que me conduziu com muita atenção e dedicação durante todos os momentos para o sucesso desse trabalho.

A minha namorada pela paciência diária e apoio nos momentos das dificuldades que encontrei pelo percurso.

Aos meus amigos da UFJF pela convivência e ótimos momentos vividos.

Ao Núcleo de Iluminação Moderna e a todos os seus integrantes, em especial ao Fabrício e o Ruan que me ajudaram nos primeiros passos e me deram suporte ao longo de todo trabalho.

É muito melhor lançar-se em busca de conquistas grandiosas, mesmo expondo-se ao fracasso, do que alinhar-se com os pobres de espírito, que nem gozam muito nem sofrem muito, porque vivem numa penumbra cinzenta, onde não conhecem nem vitória, nem derrota.

Theodore Roosevelt

RESUMO

Nos últimos anos, os temas relacionados a eficiência energética e sustentabilidade vêm ganhando destaque. Nesse contexto, a utilização de diodos emissores de luz tem aumentado significativamente em sistemas de iluminação. Esta grande atratividade dos sistemas de iluminação de estado sólido culminou no aumento da produção de luminárias de LED e de controladores de LED. Este aumento na quantidade de produtos disponíveis no mercado fez com que fosse necessária a criação de regulamentações no intuito de assegurar a qualidade de tais equipamentos. Dentre as principais normas brasileiras podem ser citadas a NBR 16026(ABNT, 2012b) e a NBR-IEC61347(ABNT, 2012c), que tratam dos requisitos de qualidade para o controlador de LED. Neste contexto, este trabalho propõe um dispositivo para realização de ensaios automatizados à luz da norma NBR 16026. Dentre as vantagens do sistema podem ser citadas: a reprodutibilidade dos testes, a redução do custo e tempo despendidos com os ensaios, bem como a confiabilidade dos resultados. O sistema é composto por um *hardware* que contém um sistema de aquisição de dados, responsáveis por adquirir as informações acerca das grandezas elétricas e uma placa composta por relés responsável por emular as diversas condições propostas na norma supracitada. Além disso, foi desenvolvido um software em C# para controlar e supervisionar o sistema. Foram realizados ensaios no *driver* da marca *INVENTRONICS*, modelo *EUC-075S045ST*, juntamente com uma luminária da marca *TECNOWATT*, modelo *Alpha*, a fim de validar o equipamento proposto.

Palavras-chave: Ensaios automatizados, automação, controlador de LED, NBR 16026, controle de qualidade

ABSTRACT

In recent years, issues related to energy efficiency and sustainability are gaining prominence. In this context, the use of LEDs has increased significantly in lighting systems. This great attractiveness of solid-state lighting systems resulted in increased production of LED luminaires and LED drivers. This increase in the quantity of products available on the market has made the creation of regulations necessary in order to ensure the quality of such equipments. Among the main Brazilian regulations may be cited NBR 16026 and NBR IEC61347, dealing with the quality requirements for LED drivers. In this context, this work proposes a device for performing automated tests in the light of NBR 16026. Among the advantages of the system can be cited: the reproducibility of tests, reducing the cost and time spent on testing and reliability of results. The system consists of hardware that contains a data acquisition system, responsible for acquiring information on the electric quantities and a board composed of relays responsible for emulating the various conditions proposed in the abovementioned standard. In addition, software in C# is designed to control and monitor the system. Tests were carried out in the INVENTRONICS brand driver, EUC-075S045ST model, along with a luminaire TECNOWATT brand, Alpha model in order to validate the proposed equipment.

Keywords: Automated tests, automation, LED drivers, NBR16026, quality control

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

ABNT Associação Brasileira de Normas Técnicas

ADC Analog-to-digital converter

CI Circuito integrado

CLP Controlador lógico programável

FP Fator de potência

GPIO General Purpose Input/Output

MCE Módulo de comutação de estados

MDGE Medidor digital de grandezas elétricas

NIMO Núcleo de Iluminação Moderna

SGBD Sistema gerenciador de banco de dados

UART Universal asynchronous receiver/transmitter

UFJF Universidade Federal de Juiz de Fora

UML Unified Modeling Language

USB Universal Serial Bus

SUMÁRIO

1 INTRODUÇÃO

A eficiência energética e o combate ao desperdício de energia são temas comuns em um mundo cada dia mais exigente em relação às questões ambientais e assuntos ligados ao desenvolvimento sustentável. A busca pela utilização mais eficiente da energia elétrica é uma tema recorrente no cenário mundial (NOGUEIRA et al., 2015).

Nos últimos anos tem sido visto um crescimento contínuo da utilização de LEDs como uma alternativa econômica e eficiente para iluminação artificial. De forma a atender a crescente demanda, a produção mundial de LEDs vem crescendo em ritmo acelerado. Para evitar a venda de produtos de má qualidade aos consumidores finais é importante a padronização dos produtos através de normas estabelecidas por órgãos normatizadores. É sabido que o bom funcionamento de um sistema de iluminação a base de LEDs não depende exclusivamente da qualidade dos diodos emissores de luz (LED), mas também de outros fatores, como o adequado projeto do controlador de LED[1]. As primeiras normas relacionadas a iluminação de estado sólido são recentes, entre elas pode-se citar a NBR 16026, responsável por definir os requisitos de desempenho dos controladores para módulos de LED. Entre os diversos ensaios exigidos por essa norma, pode-se encontrar ensaios referentes a avaliação de parâmetros elétricos. Esses ensaios são morosos e atualmente são feitos de forma manual por um operador. Além do fator tempo, existe o problema de reprodutibilidade, que é o erro associado a operadores em laboratórios distintos, ou em condições operativas diferentes (PILLET, 1994).

Diante desse contexto, é proposto o desenvolvimento de um sistema automatizado para avaliação de parâmetros elétricos em controladores de LED, de forma a aumentar a eficiência e a reprodutibilidade dos testes, assim como, reduzir custos e aumentar a confiabilidade dos ensaios. O sistema proposto é concebido apenas para *drivers* de malha fechada, visto que *drivers* de malha aberta representam uma parcela muito pequena disponível no mercado.

[1] O controlador de LED também é conhecido como circuito de acionamento eletrônico de LEDs ou *driver*.

O sistema proposto inclui um conjunto de *hardware* e um *software* computacional desenvolvido na linguagem C#, capaz de realizar dez ensaios de avaliação de parâmetros elétricos exigidos pela norma NBR 16026. Além disso, foram implementadas algumas funcionalidades extras como o armazenamento de dados relativos à *drivers* , luminárias e relatórios de resultados em um banco de dados através da integração do *software* computacional com um sistema gerenciador de banco de dados MySQL.

O segundo capítulo contém o histórico da automação mostrando suas origens até a utilização nos dias atuais. Serão apresentados também exemplos de sistemas automatizados de testes encontrados na literatura. O contexto da iluminação de vias públicas por LEDs será mostrado, assim como uma visão das normas atuais sobre *drivers* e LEDs, com especial abordagem da norma NBR 16026 a qual possui os ensaios realizados pelo sistema automatizado desenvolvido.

O terceiro capítulo é destinado à explicação da parte física que compõe o sistema, que inclui um medidor digital de grandezas elétricas e uma placa de comutação de estados com diversos relés. Serão apresentados as caraterísticas construtivas, assim como as funcionalidades de cada elemento que compõe o sistema.

O quarto capítulo destina-se à ambientação do usuário ao *software* computacional desenvolvido. Todos os módulos do *software* serão apresentados juntamente com as respectivas instruções necessárias para utiliza-los.

No início do quinto capítulo é feita uma abordagem teórica sobre linguagem orientada a objetos, banco de dados e linguagem unificada de modelagem (UML). Em seguida, será descrito como o *software* computacional foi desenvolvido.

No sexto capítulo serão apresentados os resultados obtidos dos ensaios realizados pelo sistema automatizado desenvolvido, comparando-os aos resultados obtidos por instrumentos comerciais após a realização dos ensaios de forma manual.

O sétimo capítulo realiza as considerações finais, conclusões e discute trabalhos futuros propostos.

2 REVISÃO BIBLIOGRÁFICA

Desde o advento dos relés eletromecânicos, a automação vem ganhando espaço em diversos setores da indústria, melhorando a eficiência dos processos e reduzindo custos. Hoje em dia é utilizada até mesmo em aplicações residenciais, proporcionando conforto a vida das pessoas. Esse capítulo aborda o histórico do surgimento da automação até a aplicação nos dias de hoje. Além disso, mostra o panorama atual da utilização de LEDs para aplicações de iluminação em vias públicas e as normas existentes que especificam os requisitos de qualidade dos produtos relacionados a iluminação de estado sólido, com enfoque na norma NBR 16026.

2.1 BREVE HISTÓRICO DA AUTOMAÇÃO

A automação pode ser definida como o conjunto de técnicas necessárias para tornar automática a execução de tarefas através de elementos eletromecânicos computáveis, substituindo o consumo de energia humana causado pelo esforço mental e físico. (SILVEIRA & LIMA, 2003)

A automação teve origem durante a revolução industrial no século XXIII, diante da necessidade de aumentar a produtividade e reduzir custos nas unidades fabris, em um momento que a mecanização começou a ser aplicada nos processos industriais. A mecanização consiste no emprego de máquinas para realizar alguma tarefa, a fim de substituir o esforço físico. Já a automação permitiu executar um trabalho através de máquinas controladas automaticamente. Nesse contexto, foram desenvolvidos os primeiros dispositivos simples e semiautomáticos, entre eles, os relés. Os relés são dispositivos eletromecânicos que foram muito aplicados durante o advento da automação. Porém, a sua inflexibilidade a mudanças no processo produtivo, o custo operacional elevado e o alto custo de desenvolvimento e manutenção, levou os gigantes painéis controladores com relés eletromecânicos e sua extensa fiação a serem substituídos por controladores lógico programáveis(PAREDE & GOMES, 2011) .

A automação ganhou destaque com a introdução dos *CLPs*(controladores lógico programáveis), a partir do século XX, em que diversos tipos de sistemas tiveram condições de se tornarem inteiramente automáticos. Os *CLPs* são controladores baseados em um microprocessador, que utilizam memórias programáveis a fim de armazenar instruções e implementar funções como lógica, aritmética, cronometragem e contagem para controlar máquinas e processos.(BOLTON, 2015)

Nos dias atuais a automação é empregada não apenas na indústria, mas também em residências ou em atividades comerciais. Entre os dispositivos mais utilizados estão os computadores e *CLPs*, além de outros dispositivos capazes de efetuar operações lógicas, como os microcontroladores.

2.2 TRABALHOS SIMILARES

Essa seção irá apresentar alguns sistemas automatizados de testes encontrados na literatura.

2.2.1 A COMPUTERIZED SYSTEM FOR TESTING BATTERIES IN FULL CONTROLLED ENVIRONMENT

A utilização de baterias recarregáveis pode ser atribuída a diversas aplicações tornando-se necessário o desenvolvimento de métodos mais rápidos de recarga das baterias. A fim de se testar a eficiência desses novos métodos foi proposto um sistema automático de testes em ambiente totalmente controlado. A bateria é submetida a testes em que diferentes métodos de carga e descarga são aplicados. Durante cada ensaio, sensores transduzem parâmetros como corrente, tensão, temperatura e pressão que são armazenados e processados pelo *software* computacional. (CARBALLO et al., 2000)

2.2.2 A TEST SYSTEM FOR CALIBRATING FLICKERMETERS

A introdução nas últimas décadas de um grande número de cargas não lineares tem causado uma deterioração da qualidade de energia, provocando por exemplo *flicker*, que é uma oscilação periódica do valor eficaz da tensão. O instrumento utilizado para medir o nível de *flicker* é chamado de flickerímetro. O artigo descreve o desenvolvimento de um novo sistema de calibração para testes em flickerímetros. O sistema é utilizado para medir todas as combinações de variações de amplitude e frequência necessárias para a calibração de flickerímetro. Dois geradores de onda controlados por computador são

utilizados para a produzir as condições de testes necessárias. O *software* computacional desenvolvido garante que os sinais modulados só serão medidos quando ocorrer uma modulação máxima ou mínima. (ARSENEAU; SUTHERLAND & ZELLE, 2002)

2.2.3 A MULTICHANNEL MEASUREMENT SYSTEM FOR AUTOMATIC TESTING OF ACOUSTIC CALIBRATORS AND ADJUSTMENT OF THEIR PARAMETERS

Um calibrador acústico é um instrumento gerador de sinal acústico estável de frequência conhecida. Os parâmetros de um sinal gerado deve ser estável sob diversas condições atmosféricas. Como outros dispositivos eletrônicos, os calibradores acústicos mudam seus parâmetros sob influência das condições atmosféricas. Diante desse contexto foi desenvolvido um sistema automático para testes em calibradores acústicos e ajuste de seus parâmetros.(PODGORSKI; DUMINOV & LEONIAK, 2007)

2.2.4 AN AUTOMATIC MEASUREMENT, TESTING AND DIAGNOSTIC SYSTEM FOR INDUCTION MOTORS

A pesquisa sobre falhas em rotores e estatores demanda medições em motores sob várias condições de cargas e fontes de alimentação. O sistema desenvolvido permite a realização de medições de diversas grandezas de forma totalmente automática (e.g., corrente e tensão no estator, velocidade do eixo, torque), além de possibilitar a personalização de ciclos de medição com condições de alimentação e carga programáveis. (SCHREIBER; FUCHS & JAKSCH, 2015)

2.2.5 SEMI-AUTOMATIC SYSTEM FOR TESTING DIELECTRIC PROPERTIES OF LOW-VOLTAGE BUSBAR

A fim de se verificar as propriedades dielétricas de barramentos de baixa tensão foi desenvolvido um sistema semi automático de verificação de testes. As propriedades dielétricas estão relacionadas às características como a corrente de fuga e a resistência de isolamento sob alta tensão .(XUEMEI & JIANGFENG, 2010)

O testador de tensão de *withstanding* e resistência de isolamento é utilizado para verificar as propriedades dielétricas, enquanto o testador de continuidade de terra é usado para medir a resistência de terra. Uma interface computacional foi desenvolvida na linguagem C# a fim de promover a comunicação com os instrumentos de testes.(XUEMEI & JIANGFENG, 2010)

2.3 ILUMINAÇÃO PÚBLICA EMPREGANDO LEDS

Os sistemas de iluminação artificial demandam cerca de 30% de toda energia elétrica gerada em todo mundo. Considerando essa grande fatia de consumo devido aos sistemas de iluminação artificial, é possível perceber um grande potencial para economia de energia. O uso dos diodos emissores de luz ou LEDs são uma alternativa para reduzir o consumo de energia.(POLONSKII & SEIDEL, 2008)

Os LEDs de potência vêm ganhando espaço nos últimos anos nas aplicações de iluminação em geral, com grande potencial para serem instalados inclusive em vias públicas. Em muitos países as luminárias tradicionais vêm sendo substituídas por luminárias que utilizam LEDs. Em 2008, foi inaugurada no Brasil a primeira instalação pública a utilizar LEDs. A cidade pioneira foi Nova Lima, em Minas Gerais, onde foi instalado um iluminação decorativa na Torre Alta Vila(NOGUEIRA, 2013).

As vantagens dos LEDs são muitas, além da economia de energia, pode-se citar:

- Elevada eficácia luminosa, podendo atingir 150lm/W (KRAMES et al., 2007)
- Longa vida útil, com capacidade de chegar a até 100.000 horas (LAUBSCH et al., 2010)
- Excelente resistência mecânica (NOGUEIRA et al., 2015).
- Possibilidade de controle da intensidade luminosa (NOGUEIRA et al., 2015).
- Capacidade de emissão de luz branca (NOGUEIRA et al., 2015).
- Possibilidade de implementação de um sistema de telegestão (NOGUEIRA et al., 2015).

Apesar do grande potencial dos LEDs, é preciso considerar que esses são uma tecnologia relativamente nova e ainda existe um longo caminho a ser percorrido para que sua aplicação em iluminação pública se torne uma realidade.(RODRIGUES, 2012).

A principal desvantagem da tecnologia LED está no seu alto custo de implementação. Isso significa que uma luminária LED possui um retorno financeiro longo, muitas vezes maior que a própria vida útil da luminária (SALES, 2011)

Outra desvantagem encontra-se na desconfiança com relação à durabilidade e confiabilidade dos sistemas que utilizam LEDs, visto que sua aplicação é recente e ainda não

há históricos que comprovem efetivamente a vida útil, manutenção do fluxo luminoso e durabilidade das luminárias LED ao longo do tempo (NOGUEIRA, 2013).

Um outro entrave para a expansão da utilização dos LEDs é a falta de uma legislação específica para sua fabricação, podendo levar os fabricantes a produzirem luminárias LEDs sem nenhum padrão ou mesmo qualidade duvidosa(NOGUEIRA, 2013). É importante ressaltar, que o bom funcionamento dos sistemas que utilizam LEDs não dependem apenas dos diodos emissores de luz, mas também necessitam que o circuito eletrônico de acionamento dos LEDs seja bem projetado. Caso contrário, o sistema não irá se mostrar eficiente e durável como se espera(NOGUEIRA et al., 2015). As primeiras normas lançadas no Brasil se referem aos circuitos de acionamento dos LEDs, conhecidos também como *drivers* ou controladores de LED. Entre elas, está a NBR 16026, que especifica os requisitos de desempenho que os *drivers* fabricados devem atender. O sistema automatizado desenvolvido por esse trabalho, baseia-se em alguns ensaios relacionados a parâmetros elétricos definidos por essa norma. Os ensaios que foram selecionados para serem realizados pelo sistema serão mostrados na seção 2.4

2.4 NORMAS

As normas brasileiras ligadas diretamente a iluminação com LEDs são recentes. A ABNT NBR 15129 e a ABNT IEC 60598-1 são normas aplicáveis a luminárias LEDs. A primeira especifica características e requisitos para luminárias utilizadas em iluminação pública (ABNT, 2012a). Já a segunda determina os requisitos para luminárias LEDs em geral(ABNT, 2010).

As normas referentes aos circuitos de acionamento de LEDs são a NBR 16026 e a NBR IEC 61347-2-13. A NBR 16026 - Requisitos de desempenho de dispositivo de controle eletrônico CC ou CA para módulos de LED (ABNT, 2012b), tem origem na tradução da norma IEC 62384 - *d.c. or a.c. supplied electronic control gear for LED modules - Performance*. Essa norma detalha ensaios para verificação do desempenho dos drivers de luminárias LED, além de recomendações em relação à classificação do tipo de controle do sistema de acionamento (driver sem controle, com controle de tensão, ou com controle de corrente) e outras características que os *drivers* devem conter(NOGUEIRA, 2013).

A NBR IEC 61347-2-13 - Requisitos particulares para dispositivos de controle eletrônico alimentados em CC ou CA para os módulos de LED (ABNT, 2012c) tem origem na tradução da norma IEC 61347-2-13 - *Lamp controlgear - Particular requirements*

for d.c. or a.c. supplied electronic control gear for LED modules - Safety. Essa norma apresenta requisitos de segurança e construção de drivers de luminárias LED além de alguns ensaios para análise dos circuitos de acionamento de LED(NOGUEIRA, 2013).

Alguns ensaios da norma NBR 16026 foram selecionados para serem realizados pelo sistema automatizado para avaliação de parâmetros elétricos em controladores LED. Os ensaios selecionados são divididos em três grupos: ensaios de características elétricas de funcionamento, ensaios operacionais para condições anormais e ensaios de durabilidade.

2.5 ENSAIOS DE CARACTERÍSTICAS ELÉTRICAS DE FUNCIONAMENTO

O primeiro ensaio para avaliação de características elétricas de funcionamento é o ensaio de máxima variação da tensão ou corrente de saída durante a partida. Nesse ensaio, a corrente – caso o *driver* tenha controle de corrente – ou a tensão – caso o *driver* tenha controle de tensão – deve ser avaliada na entrada do *driver* durante a partida. A avaliação deve ocorrer durante os 2s após a partida e não pode exceder 10% do seu valor nominal declarado pelo fabricante.

O ensaio para avaliação do fator de potência especifica que um driver com potência nominal maior ou igual a 25W devem ter alto fator de potência ($> 0{,}92$). Além disso, o *driver* não pode apresentar um fator de potência inferior ao valor indicado pelo fabricante com diferença maior que 0,05, quando o dispositivo de controle é operado em sua potência nominal. Isso quer dizer que, se o fator de potência especificado pelo fabricante é 0,98, o *driver* não pode operar com FP menor que 0,93.

O ensaio de máxima variação da corrente de entrada em regime de operação especifica que na tensão nominal de alimentação, a corrente de entrada do *driver* não pode exceder 10% do valor declarado pelo fabricante.

O ensaio de máxima variação de potência de entrada especifica que na tensão nominal de alimentação, a potência total de entrada do *driver* não deve exceder 10% do valor declarado pelo fabricante.

O ensaio de máxima variação da tensão de saída em regime de operação é realizado apenas em controladores com controle de tensão. A norma especifica que na tensão nominal de alimentação, a tensão de saída do *driver* não pode apresentar variação superior a $\pm 10\%$ do seu valor declarado pelo fabricante.

O ensaio de máxima variação da corrente de saída em regime de operação é realizado apenas em controladores com controle de corrente. A norma especifica que na tensão nominal de alimentação, a corrente de saída do *driver* não pode apresentar variação superior a ±10% do seu valor declarado pelo fabricante.

O último ensaio de avaliação de características elétricas de funcionamento é o ensaio de máxima variação da tensão ou corrente de saída, no qual o *driver* é submetido a tensões não nominais de entrada. Para controladores com controle de tensão, a grandeza de saída avaliada é a tensão, já para *drivers* com controle de corrente, a grandeza de saída avaliada é a corrente. Nesse ensaio, quando o controlador de LED é alimentado com tensão entre 92% e 106% da nominal, esse não pode sofrer variação superior a ± 10% na tensão ou corrente de saída, dependendo do tipo de controle.

2.6 ENSAIOS OPERACIONAIS PARA CONDIÇÕES ANORMAIS

Os ensaios operacionais para condições anormais submetem a saída do *driver* a duas condições de falta: curto-circuito ou circuito aberto. Em ambos os casos o dispositivo deverá funcionar normalmente após a extinção da falta, isto é, a tensão (dispositivos de tensão constante) ou corrente (dispositivos de corrente constante) não devem destoar de 10% de seu valor nominal.

No ensaio de curto-circuito, o *driver* quando alimentado na tensão nominal, não pode se danificar após ter suas saídas curto-circuitadas durante uma hora. No ensaio de circuito aberto, o *driver* não pode se danificar após ser alimentado com tensão nominal durante uma hora sem os módulos de LED inseridos.

2.7 ENSAIOS DE DURABILIDADE

A norma especifica dois tipos de ensaios de durabilidade: o ensaio de ciclo de choque de temperatura e o ensaio de comutação de tensão de alimentação.

Visto que o sistema proposto realiza apenas ensaios de avaliação de parâmetros elétricos, o ensaio de durabilidade selecionado é o ensaio de comutação de tensão de alimentação. Nesse ensaio, o *driver* deve ser ligado por 30 segundos e desligado por 30 segundos, enquanto é alimentado na tensão nominal. Este ciclo deve ser repetido 200 vezes sem os módulos de LED e 800 vezes com os módulos de LED. Após o ensaio, o *driver* deve operar normalmente durante 15 minutos com os módulos de LED inseridos.

2.8 CONCLUSÕES PARCIAIS

Esse capítulo apresentou um breve histórico da automação, mostrando como se deu seu surgimento assim como sua evolução e o emprego nos dias atuais. O desenvolvimento de sistemas automatizados mostram-se fundamentais diante da necessidade de aumentar a produtividade de processos e reduzir custos.

Sistemas para realização de testes de forma automatizada, desenvolvidos por outros pesquisadores, foram apresentados como exemplos da utilização da automação de testes.

O contexto da aplicação de LEDs para iluminação pública foi apresentado, mostrando que a tecnologia de LEDs é promissora. Dentre as principais vantagens oferecidas pela iluminação de estado sólido, pode-se citar: a economia de energia, a eficácia luminosa, a longa vida útil, a emissão de luz branca, entre outras.

Foi visto que para o funcionamento esperado dos LEDs é necessário que o circuito de acionamento dos diodos emissores de luz seja bem projetado e desenvolvido. Dessa forma, as normas de padronização da fabricação de controladores de LED são importantes. Foram citados as principais normas relativas a iluminação com LEDs, incluindo a NBR 16026, utilizada como referência para seleção dos ensaios realizados pelo sistema automatizado proposto por esse trabalho. Os ensaios selecionados dessa norma foram descritos no fim do capítulo.

3 HARDWARE

O objetivo desse projeto é criar um equipamento capaz de realizar ensaios para avaliação de parâmetros elétricos em controladores de LEDs de forma automatizada. Para ser possível a avaliação desses parâmetros, primeiramente é preciso obtê-los através de um instrumento de medição e aquisição de dados. O instrumento utilizado com essa finalidade foi o *Medidor digital de grandezas elétricas* (MDGE), desenvolvido em Coelho(2016). Portanto detalhes de implementação podem ser vistos na referência supracitada.

Para utilizar em conjunto com o MDGE foi desenvolvido uma placa com relés, chamado de *Módulo de comutação de estados*(MCE), de forma que fosse possível emular as diversas condições propostas, já que cada ensaio possui características e procedimentos peculiares.

O último elemento do sistema é o *software* computacional que será apresentado nos capítulos 4 e 5. O *software* é responsável por determinar os estados dos relés, configurar o instrumento de aquisição de dados e comandar os procedimentos necessários para a realização de cada ensaio.

Como pode-se perceber o sistema é composto por três elementos principais: o instrumento de aquisição de dados(MDGE), o MCE e o *software* computacional. Nesse capítulo serão apresentados o *Hardware* do sistema, ou seja, o MDGE e o MCE.

3.1 MEDIDOR DIGITAL DE GRANDEZAS ELÉTRICAS

O MDGE é composto por um *display* LCD, botões, sensores de corrente e tensão, além de um microcontrolador. Esses elementos são alocados em uma placa mãe. O diagrama de blocos do MDGE é apresentado na figura 1

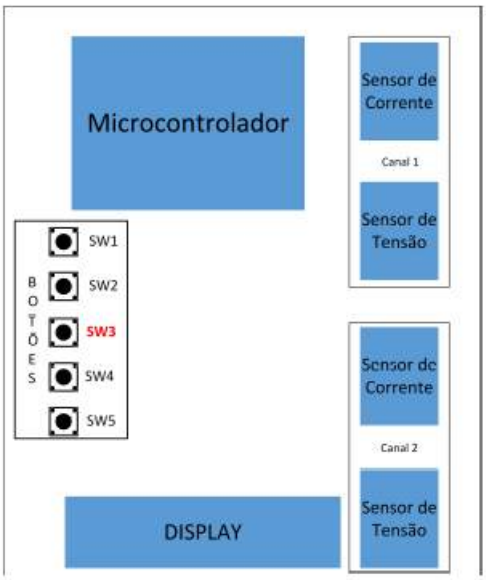


Figura 1: Diagrama de blocos do MDGE
(COELHO, 2016)

O *display LCD* informa a leitura das grandezas medidas e os botões são utilizados para configurar o próprio instrumento ou a visualização do *display LCD*. Porém, esses elementos só são úteis quando uma interface computacional não está ativa. Como os ensaios dependem diretamente da utilização de um *software* computacional, esses elementos não terão funcionalidade para o sistema automatizado desenvolvido.

3.2 MICROCONTROLADOR

O microcontrolador recebe os sinais analógicos oriundos dos sensores e os converte para sinais digitais através de um ADC(conversor analógico-digital) para em seguida realizar os cálculos necessários. O modelo *EK-LM4F120XL* empregado é fabricado pela *Texas Instruments* e pertence a família *Stellaris*. Possui um processador que opera a 80MHz e memórias *SRAM*, *flash* e *ROM*. Além disso, apresenta um conjunto de portas *GPIO* que funcionam como entrada e saída digital, podendo ser configurado para utilizar até 43 portas. As portas digitais operam com tensões entre 0 e 3,3V, suportando até um máximo de 5V sem danificar o microcontrolador(TEXAS INSTRUMENTS, 2013). A imagem do microcontrolador pode ser vista na figura 2.

Figura 2: Imagem do microcontrolador *EK-LM4F120XL*

O ADC de 12 bits proporciona uma boa resolução de 0,806mV, de acordo com a equação 3.1.

$$Resol._{mV} = \frac{3{,}3V - 0V}{2^{12}} = 0{,}806mV \tag{3.1}$$

O gráfico da figura 3 mostra o comportamento do conversor de acordo com a tensão na porta.

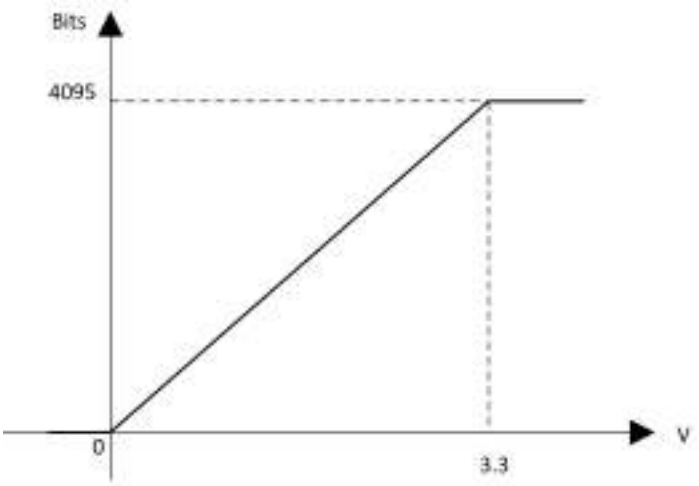


Figura 3: Comportamento do conversor de acordo com a tensão na porta (COELHO, 2016)

Depois que a informação é processada, ela é enviada para o *software* computacional, para permitir a visualização pelo usuário das medições efetuadas. A transmissão e recepção de dados entre o microcontrolador e a interface computacional é feita de forma serial, através de uma interface *UART*. Uma questão importante sobre a comunicação é a velocidade de troca de informações. Pode-se definir como *baudrate*, a taxa em que os bits são transmitidos pela interface *UART*. O transmissor e o receptor dos dados

devem estar configurados da mesma forma, para que seja possível a comunicação.

3.3 SENSOR DE TENSÃO

O sensor de tensão deve transduzir em uma escala de 0V a 3,3V um valor proporcional ao valor instantâneo de tensão. O diagrama correspondente ao sensor é mostrado na figura 4.

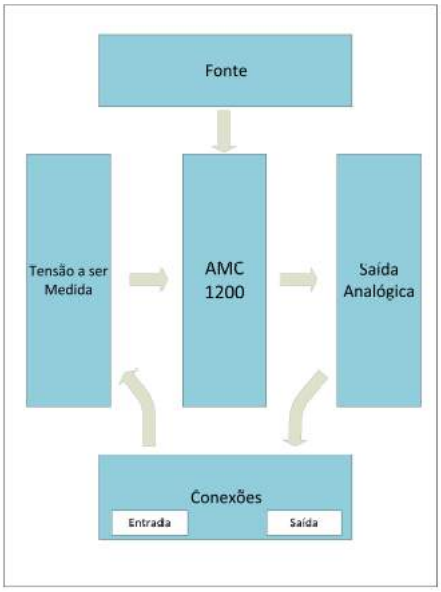


Figura 4: Diagrama de blocos do sensor de tensão
(COELHO, 2016)

A placa é dividida em duas partes: a parte à esquerda trabalha com potências elevadas que correspondem aos sinais que se desejam medir e outra a direita com potência menor onde se encontram os sinais que serão enviados ao microcontrolador. O circuito integrado presente no meio da placa é o amplificador AMC1200B, que faz a interface da tensão a ser medida pelo instrumento com o valor analógico que será enviado para o microcontrolador. Além de amplificar o sinal, proporciona isolamento galvânico.

O sensor possui 10 escalas de medição, com tamanhos de faixas de medição de 100V. A primeira representa uma faixa -50V a 50V até a décima -500V a 500V, acrescentando sempre 50V para o máximo e decrescendo 50V para o mínimo a cada aumento de escala. Para aumentar a escala o operador deve inserir *jumper's* na parte traseira do sensor. Cada adição de um *jumper* corresponde ao aumento de uma faixa.

Como o sinal de saída do CI que é enviado ao microcontrolador não abrange a faixa

de 0V a 3,3V, mas está sempre entre 0,29V e 2,29, existe perda de faixa de leitura do ADC, que ficará entre 360 e 2842 bits. Logo a resolução não será a da equação 3.1 e sim a da equação 3.2.

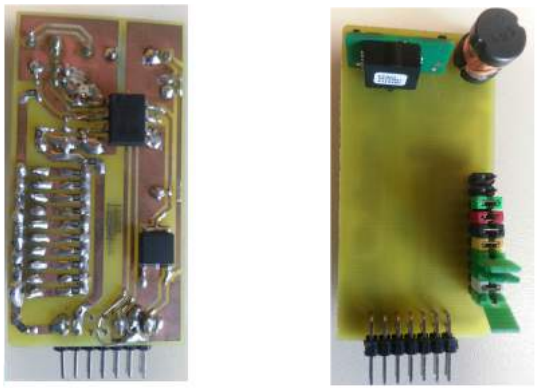
$$Resol._{mV} = \frac{FaixaTensao}{FaixaBits} \tag{3.2}$$

A resolução é calculada para cada faixa de medição, como pode ser visto na tabela 1.

Jumpers	Tamanho da faixa(V)	Resolução (Volts/Bit)
1	100	0,0403
2	200	0,0806
3	300	0,1209
4	400	0,1612
5	500	0,2015
6	600	0,2418
7	700	0,2821
8	800	0,3224
9	900	0,3627
10	1000	0,403

Tabela 1: Resumo das resoluções para o sensor de tensão (COELHO, 2016)

As imagens reais do sensor de tensão podem ser vistas na figura 5.



(a) Vista frontal (b) Vista traseira

Figura 5: Vistas do sensor de tensão (COELHO, 2016)

O sensor de corrente apresenta topologia similar ao sensor de tensão, porém existem pequenas diferenças que podem ser vistas com detalhes em Coelho(2016). O sensor de corrente realiza a medição de forma indireta, ou seja, através da tensão gerada pela corrente que passa por um resistor.

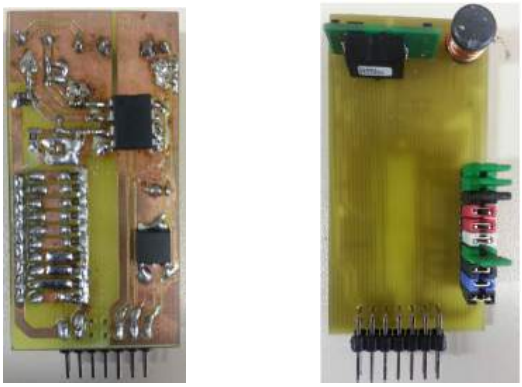
O sensor de corrente possui 10 escalas de medição, com tamanhos de faixas de medição de 500mA. A primeira representa uma faixa de -250mA a 250mA, até a décima de -2,50A a 2,50A, acrescentando sempre 250mA para o máximo e decrescendo 250mA para o mínimo a cada aumento de escala.

A tabela completa de resoluções de acordo com o número de *jumpers* pode ser vista na tabela 2.

Jumpers	Tamanho da faixa (mA)	Resolução(mA/Bit)
1	500	0,201
2	1000	0,403
3	1500	0,604
4	2000	0,805
5	2500	1,007
6	3000	1,209
7	3500	1,41
8	4000	1,611
9	4500	1,813
10	5000	2,014

Tabela 2: Resumo das resoluções para sensor de corrente (COELHO, 2016)

As imagens reais do sensor de corrente podem ser vistas na figura 92.



(a) Vista frontal

(b) Vista traseira

Figura 6: Vistas do sensor de corrente (COELHO, 2016)

O instrumento de aquisição de dados é dotado de 2 canais distintos para medição de grandezas elétricas, totalizando 4 sensores, sendo dois de tensão e dois de corrente. Dessa forma, é possível realizar medições simultâneas com duas cargas diferentes, ou mesmo fontes distintas.

Todos os dispositivos de *hardware* são alocados em um placa mãe o qual promove a integração entre todos. A figura 7 mostra a imagem de todos os hardwares que compõe o MDGE conectados.

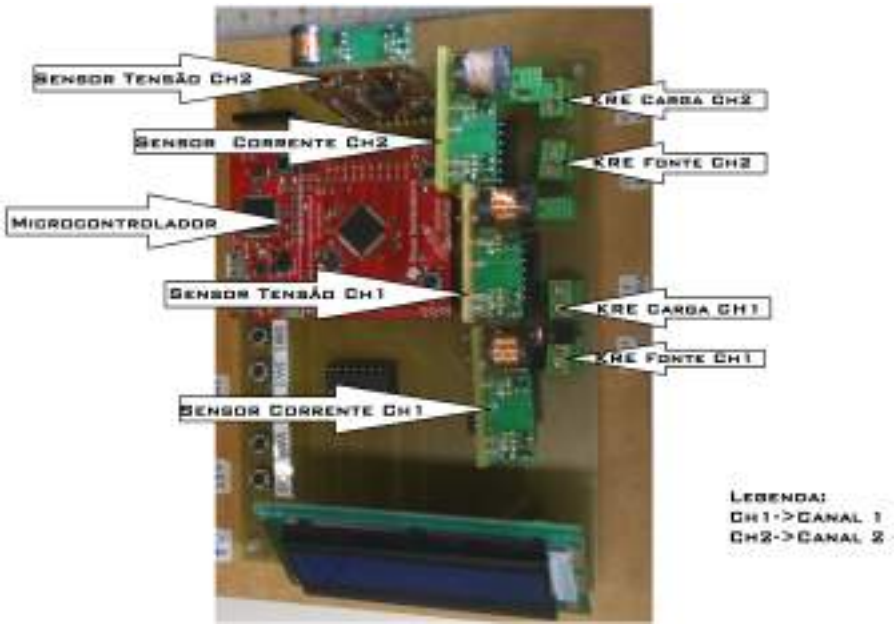


Figura 7: Foto do MDGE

A estratégia de roteamento da placa mãe posiciona os sensores de corrente em série com a carga e os de tensão em paralelo. Portanto, para utilizar um canal do instrumento, basta conectar a fonte a um conector KRE e a carga a outro conector KRE, como pode ser visto na figura 8.

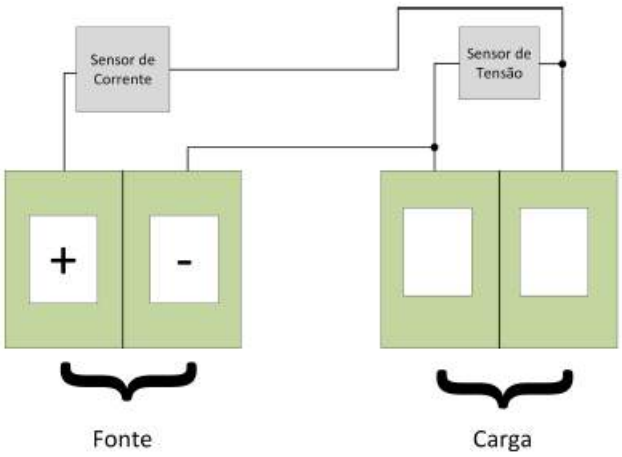
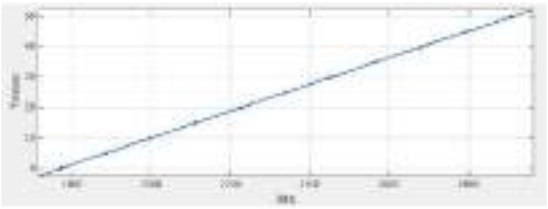


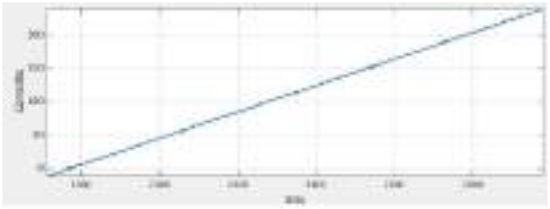
Figura 8: Esquemático representando as ligações dos sensores na placa mãe (COELHO, 2016)

Por fim, é necessário verificar a qualidade das medições realizadas a fim de constatar se os valores medidos são fidedignos. Com esse propósito, foram realizados ajustes nos sensores de forma que os valores medidos pelo equipamento se aproximem dos valores obtidos por um instrumento de referência, no caso um o *Wattímetro Yokogawa WT230*. Visto que os sensores possuem diversas escalas, o procedimento de ajuste deve ser feito para cada escala.

O processo parte do princípio do confrontamento entre o os valores medidos pelo *Wattímetro Yokogawa* e os valores convertidos pelo ADC do microcontrolador. Para cada ajuste, uma curva Tensão x *Bits* é traçada utilizando a técnica de interpolação polinomial. As curvas de ajuste para tensão e corrente com 1 *jumper* podem ser vista na figura 9. Todas as curvas de ajustes são encontradas na literatura indicada.



(a) Resultado da interpolação para o sensor de tensão com 1 jumper (V x Bits)



(b) Resultado da interpolação para o sensor de corrente com 1 jumper (mA x Bits)

Figura 9: Curvas de ajuste
(COELHO, 2016)

3.4 MÓDULO DE COMUTAÇÃO DE ESTADOS

Com o objetivo de emular as diversas condições peculiares necessárias para a realização de cada ensaio foi desenvolvido o *Módulo de comutação de estados*(MCE). Esse nome foi dado pois ele é basicamente um módulo de relés, os quais comutam seus estados, alterando as conexões elétricas do circuito. Dessa forma, é possível, por exemplo, alterar a tensão nominal de entrada utilizada, desenergizar todo o equipamento, curto-circuitar a saída do *driver*, entre outras funcionalidades que serão abordadas mais a frente. Além disso, a fim de atender o ensaio que utiliza tensões não nominais de entrada, foi concebida uma alternativa para o dispositivo, lançando mão do uso de dois transformadores que possuem relações de transformação de maneira a propiciar as tensões de 92% e 106% da nominal. O controle da comutação dos relés é feito através das portas *GPIO* do microcontrolador contido no MDGE.

A utilização em conjunto do MCE e do MDGE proporciona os requisitos de *hardware* necessários para o funcionamento do sistema. Uma visão geral do sistema, a partir de um diagrama de blocos, é apresentada na figura 10.

Figura 10: Diagrama de blocos do sistema completo

O diagrama do sistema completo é dividido em várias partes para facilitar o entendimento.

A figura 11 mostra o diagrama de blocos para o seletor de tensão nominal.

Figura 11: Diagrama de blocos para o seletor de tensão nominal

O seletor de tensão de entrada define a tensão nominal que será utilizada, 127V ou 220V.

A figura 12 mostra o diagrama de blocos para o *seletor de Transformador* e *Seletor Tensão Nominal ou não Nominal*.

Figura 12: Diagrama de blocos para o seletor de transformador e seletor de tensão nominal ou não nominal

A saída do seletor de tensão de entrada está conectada a ambos os transformadores. O *seletor de Transformador* definirá qual dos dois transformadores será utilizado, o correspondente a 92% ou 106% da tensão nominal. A saída desse seletor está conectada ao *Seletor Tensão Nominal ou não Nominal* que irá definir se a tensão no sistema será a nominal ou a não nominal oriunda da saída de um transformador.

A região do circuito em torno do canal 1 pode ser vista na figura 13.

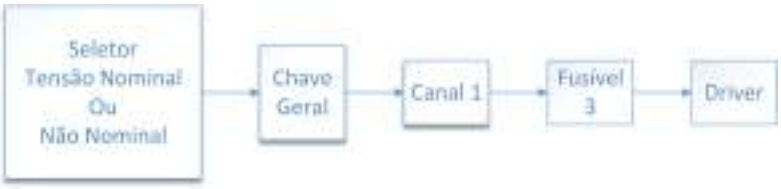


Figura 13: Diagrama de blocos da região do canal 1

Após o *Seletor Tensão Nominal ou não Nominal* encontra-se a chave geral permitindo que a alimentação do equipamento seja mantida ou interrompida. Em seguida, encontra-se conectado o canal 1 do MDGE, com o objetivo de possibilitar a coleta de parâmetros elétricos de entrada do *driver*, já que está alocado posteriormente ao controlador de LED.

A figura 14 mostra a região do circuito em torno do canal 2.

Figura 14: Diagrama de blocos da região do canal 2

A *chave curto* localiza-se a frente do *driver* e é responsável por curto-circuitar ou não a saída do controlador de LED. O canal 2 é alocado na saída do *driver*, de forma a obter os parâmetros elétricos de saída. A *chave circuito aberto* permite conectar o módulo de LEDs ou removê-los do circuito.

3.4.1 IMPLEMENTAÇÃO

O esquemático do módulo é mostrado na figura 15 .

Figura 15: Esquemático do MCE

Os conectores KRE implementam a conexão do MCE com componentes externos. A tabela 3 apresenta a função de cada conector KRE.

KRE	Conexão
1	Alimentação de 127V
2	Alimentação de 220V
3	Primário do transformador de 1:0,92
4	Primário do transformador de 1:1,06
5	Secundário do transformador de 1:1,06
6	Secundário do transformador de 1:0,92
7	Fonte do canal 1 do MDGE
8	Carga do canal 1 do MDGE
9	Entrada do controlador de LED
10	Saída do controlador de LED
11	Fonte do canal 1 do MDGE
12	Carga do canal 2 do MDGE
13	Módulo de LEDs

Tabela 3: Conectores KRE

A figura 16 ilustra as conexões da tabela 3.

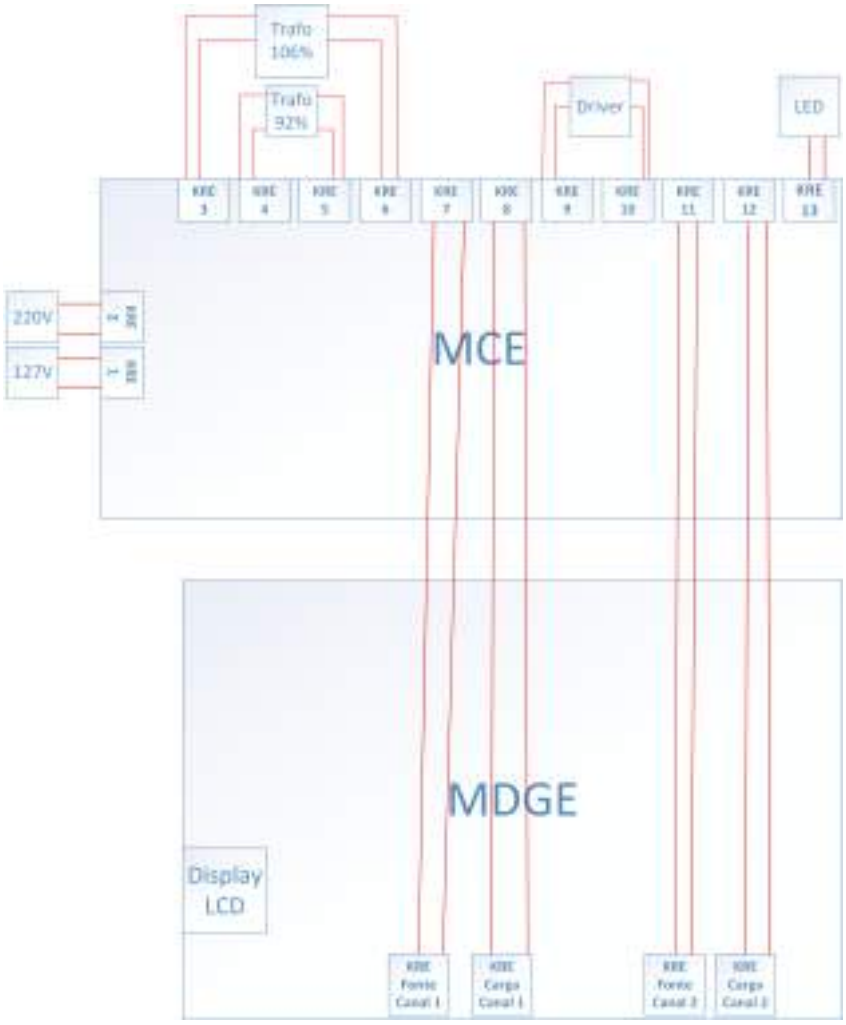


Figura 16: Conexões utilizando os conectores KRE

No intuito de implementar os seletores e as chaves do sistema foram utilizados relés.

Para entender o funcionamento de um seletor utilizando relés, é mostrado um exemplo na figura 17.

Tabela Verdade

Relé 1	Relé 2	Saída Relé 2
0	0	Entrada 3
0	1	Entrada 2
1	0	Entrada 3
1	1	Entrada 1

Figura 17: Seletor utilizando relés

De acordo com a figura 17, cada um dos relés pode ter seu contato(destacado

em vermelho) nas posições 0 ou 1. O sinal de saída do relé 2 depende da posição a qual se encontram os contatos dos relés. Dessa forma, é possível obter 4 configurações diferentes que são mostradas na tabela verdade. Nesse exemplo, ambos contatos se encontram na posição 0, portanto o sinal de saída do relé 2 é o sinal da entrada 3.

A figura 18 mostra um relé sendo aplicado como chave.

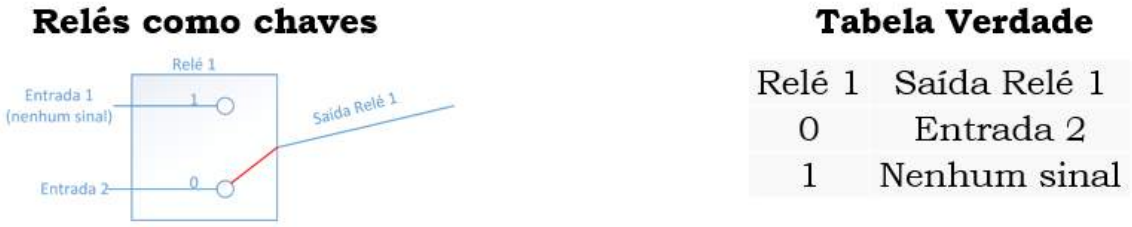


Figura 18: Relé sendo aplicado com uma chave

De acordo com a figura 18, novamente o contato do relé pode assumir apenas duas posições: 0 ou 1. Nesse exemplo, o contato se encontra na posição 0 e portanto o sinal na saída do relé é o sinal proveniente da entrada 2. Caso o contato estivesse na posição 1, o sinal de saída seria o sinal da entrada 1, porém essa entrada está desconectada do circuito e não recebe nenhum sinal. Dessa forma, o relé pode ser aplicado como uma chave.

A tabela 4 apresenta a aplicação de cada relé.

Relé	Aplicação
1	Seletor de tensão de entrada
2	Seletor de tensão de entrada
3	Seletor de tensão nominal ou não nominal
4	Seletor de tensão nominal ou não nominal
5	Chave geral
6	Seletor de transformador
7	Seletor de transformador
8	Chave de curto-circuito
9	Chave de circuito aberto

Tabela 4: Aplicação dos relés

Como pode-se notar, são utilizados dois relés para cada um dos seletores a seguir: *Seletor tensão de entrada*, *Seletor de tensão nominal ou não nominal*, *Seletor de trans-*

formador. Isso se deve ao fato que cada tensão é obtida através de dois fios, totalizando 4 fios possíveis, logo é necessário a utilização de 2 relés.

O conector *Header* tem a função de conectar o MCE ao microcontrolador, de forma que esse tenha condições de controlar os estados dos contatos dos relés. O *Header* possui 8 entradas que são conectadas aos pinos machos do microcontrolador. Entre os pinos utilizados pode-se citar o pino Vcc, com saída de 3,3V, que proporciona a alimentação dos circuitos de controle dos relés. O pino GND fornece o terra para os circuitos de controle. O restante dos pinos são configurados, no *firmware* do microcontrolador, para serem utilizados como portas GPIO e portanto serem capazes de determinar os estados dos contatos dos relés. Cada pino configurado como porta GPIO está relacionado ao controle de um ou mais relés, como pode ser visto na tabela 5. A figura 19 mostra a correspondência da conexão entre os pinos do microcontrolador e as entradas do conector *Header*.

Pino	Aplicação
Vcc	Alimentação dos circuitos de controle
C4	Controla os relés 1 e 2
A2	Controla os relés 3 e 4
E0	Controla o relé 6
F3	Controla o relé 5
E4	Controla o relé 7
A5	Controla o relé 8
GND	Terra dos circuitos de controle

Tabela 5: Aplicação associada a cada pino GPIO

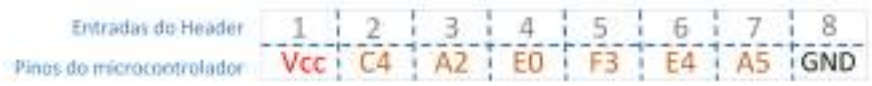


Figura 19: Conexão entre o *Header* e pinos do microcontrolador

Foi construído um cabo com fios de cores distintas e propriamente etiquetados de acordo com o pino a que deve ser ligado, a fim de facilitar a conexão entre as entradas do *Header* e os pinos do microcontrolador. A foto do cabo é mostrada na figura 20.

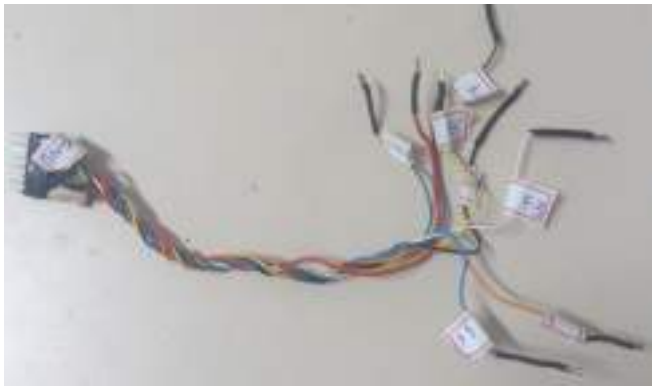


Figura 20: Foto do cabo que conecta as entradas do *Header* e os pinos do microcontrolador

Três fusíveis foram alocados no equipamento, no intuito de protegê-lo contra correntes elevadas devido a uma eventual falha que possa ocorrer.

Os fusíveis 1 e 2 protegem a entrada do equipamento. O fusível 1 está ligado em série a um fio oriundo do conector KRE 1, que corresponde a entrada da tensão de 127V. Já o fusível 2 está ligado em série a um fio oriundo do conector KRE 2, que corresponde a entrada da tensão de 220V.

O fusível 3 está ligado diretamente na entrada do *driver*. Esse fusível foi alocado para proteger todo o circuito que está anterior ao *driver*, caso ocorra alguma falha no circuito eletrônico do controlador de LED.

O conector *Jack* possibilita a conexão de uma fonte externa CC de 5V.

O circuito de controle é composto por dois resistores SMD 0805 125mW 1%, um isolador óptico 4N25(VISHAY SEMICONDUCTORS, 2010), um transistor BC337(FAIRCHILD SEMICONDUCTORS, 2015), um diodo 1N4148(NXP SEMICONDUCTORS , 2004) e um relé SRD-S-105D(SANYOU RELAYS, 2015).

O esquemático do circuito pode ser visto na figura 21.

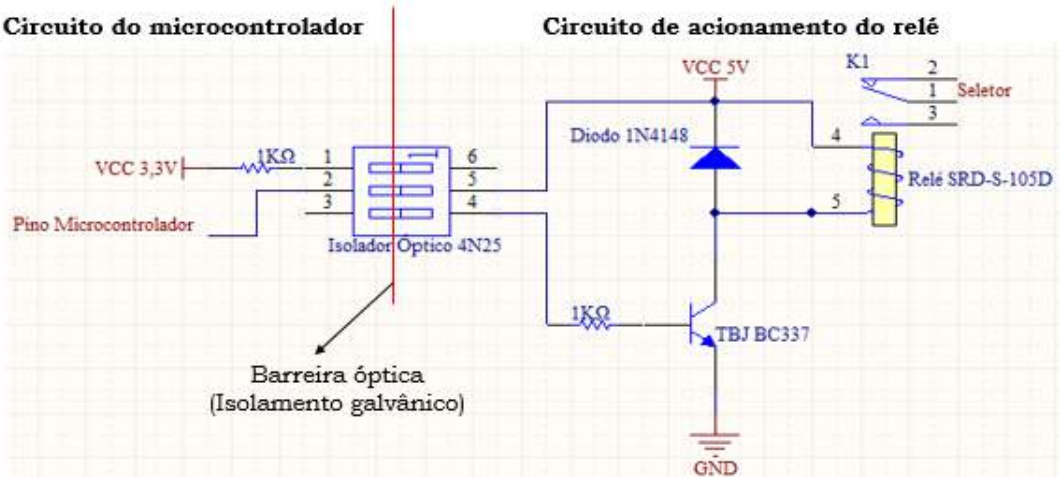


Figura 21: Esquemático do circuito de controle dos relés

O circuito de controle é dividido em dois circuitos diferentes. Um circuito é o do microcontrolador e o outro é o de acionamento do relé. O primeiro é alimentado pelo microcontrolador com tensão de 3,3V e o segundo por uma fonte externa de 5V.

O isolador óptico foi utilizado para isolar galvanicamente os dois circuitos, como forma de proteção ao microcontrolador e pelo fato dos circuitos trabalharem com tensões de alimentação diferentes, como explicado no parágrafo anterior.

O esquemático do isolador óptico 4N25 pode ser visto na figura 22.

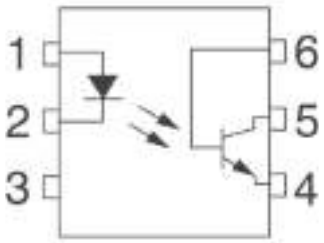


Figura 22: Esquemático do isolador óptico 4N25

O pino 1 do circuito integrado está conectado permanente aos 3,3V da placa do microcontrolador (pino Vcc) através de um resistor. O pino 2 do CI é conectado a um pino *GPIO* do microcontrolador, que apresenta dois estados digitais possíveis, alto (3,3V) e baixo (0V). Quando se encontra em estado alto não há diferença de potencial e não há corrente circulante. Já quando se encontra em estado baixo há passagem de corrente, provocando acendimento do LED interno e consequentemente fazendo com

que o fototransistor passe a conduzir corrente em seu circuito coletor-emissor, que irá energizar a bobina do relé, fazendo com que este dispositivo comute seu estado.

O diodo de alta velocidade foi empregado no projeto para evitar um surto de tensão causado pela bobina do relé quando a corrente é interrompida de maneira repentina pelo fototransistor. Este diodo também é chamado de diodo roda livre.

Os componentes necessários para a confecção da placa podem ser vistos na tabela 6.

Quantidade	Componente
1	Conector Jack DC J4, 3T
13	Borne Verde KF-128 KRE 2 conexões
3	Fusíveis
9	Relés 5VDC SRD-S-105D
9	Isoladores Ópticos 4N25
9	Transistores BC337
9	Diodos de Alta Velocidade 1N4148
18	Resistores SMD 1k 0805 125mW 1%
1	Placa de Fenolite cobreado para PCB

Tabela 6: Lista de componentes utilizados

A placa foi projetada com o auxílio do *software* Altium Designer 10 e dividida de uma forma que os circuitos de potência e o de controle ficassem separados, garantindo o isolamento galvânico entre eles. Esta estratégia é importante uma vez que os níveis de tensão de ambos os circuitos são muito diferentes.

As trilhas foram projetadas para 2 mm para o circuito de potência e 0,5mm para o circuito de controle. As dimensões finais da placa são de 16,1cm X 7,7cm.

O projeto final da PCB pode ser visto na figura 23.

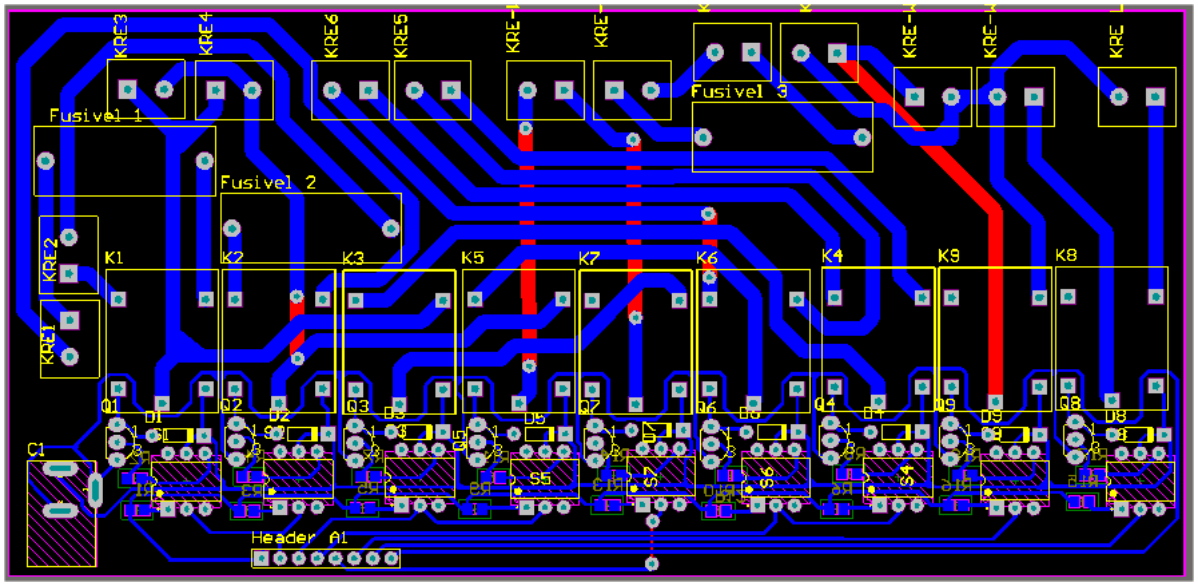


Figura 23: Projeto da PCB

A fresa Suda SD 2616 do laboratório NIMO foi responsável pela fabricação da placa. O *software* Altium é capaz de gerar um arquivo de imagem, que é reconhecido pelo sistema da fresa. Após fabricada, todos componentes foram soldados manualmente. O MCE pode ser visto na figura 24.

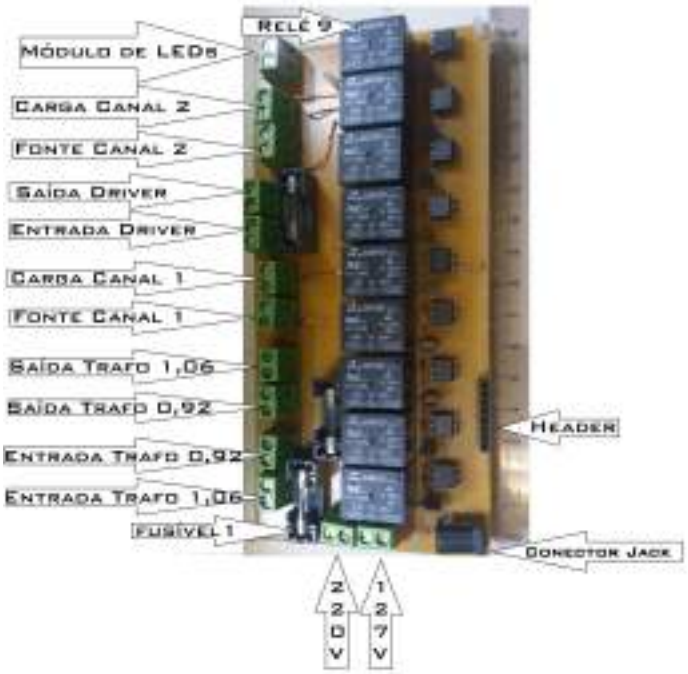


Figura 24: Foto do MCE

3.5 COMUNICAÇÃO

A comunicação entre o *software* computacional e o programa embarcado do microcontrolador é realizada por meio de uma conexão USB, usando um protocolo que segue o modelo mestre-escravo. Nesse sistema, o papel de mestre é desempenhado pelo *software* computacional e o escravo pelo programa embarcado do microcontrolador.

Nesse tipo de comunicação o mestre requisita informações do escravo, de forma que o escravo so irá se comunicar caso o mestre deseje. Uma vantagem dessa comunicação é um menor congestionamento no tráfego de dados (TANENBAUM, 2003).

A comunicação ocorre quando o *software* computacional envia um pacote de dados ao microcontrolador. Esse analisa o significado do pacote e realiza uma ação ou envia uma resposta ao *software* computacional.

Os pacotes básicos do protocolo de comunicação que são enviados para o microcontrolador podem ser vistos na tabela 7.

Nome do pacote	Pacote	Tamanho
JP1	I<0>	2 Bytes
JP2	H<0>	2 Bytes
SET	STR<00>	5 Bytes
CAN	Ch<0>	3 Bytes

Tabela 7: Pacotes básicos do protocolo

Os pacotes *JP1* e *JP2* são responsáveis por ordenar a configuração do número de *jumpers* do canal 1 e 2 do MDGE no programa embarcado, respectivamente. O caractere zero nesse pacote representa qualquer valor numérico de 0 a 9 correspondente ao número de *jumpers*. Se o valor enviado for zero, o microcontrolador irá interpretar como 10 *jumpers*.

O estado alto ou baixo das portas *GPIO* que acionam os relés é determinado pelo pacote de nome *SET*. De acordo com esse estado, o relé pode ser ou não acionado, como foi visto na seção 3.4.1. Nesse pacote o primeiro zero representa o número do pino do microcontrolador, de acordo com numeração apresentada na figura 25. O segundo zero representa o estado alto(1) ou baixo(0) que se deseja configurar a porta *GPIO*.

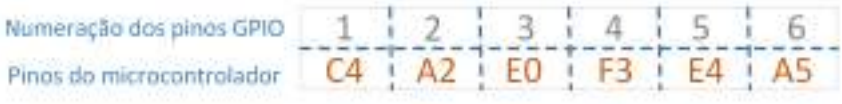


Figura 25: Numeração dos pinos GPIO

O pacote *SET* e suas aplicações são mostrados em detalhes na tabela 8.

Pacote	Função
STR10	Seleciona 127V
STR11	Seleciona 220V
STR20	Desconecta transformador
STR21	Conecta transformador
STR30	Seleciona transformador 1:1,06
STR31	Seleciona transformador 1:0,92
STR40	Abre chave geral do equipamento
STR41	Fecha chave geral do equipamento
STR50	Curto-circuita a saída do *driver*
STR51	Retira curto-circuito da saída do *driver*
STR60	Insere módulo de LEDs ao circuito
STR61	Remove módulo de LEDs do circuito

Tabela 8: Pacote *SET*

O *software* computacional requisita as medições coletados pelo MDGE através do pacote *CAN*. O caractere zero desse pacote é um valor numérico, que pode ser 1 ou 2, representando dessa forma o número do canal que está sendo solicitado uma resposta, ou seja, canal 1 ou canal 2.

O pacote recebido tem a característica mostrada na figura 26.

Figura 26: Pacote recebido como resposta ao pacote CAN
(COELHO, 2016)

No pacote CAN, o parâmetro inicial utilizado é o caractere ”a”ou ”b”, correspondendo ao canal 1 e 2, respectivamente. O restante dos dados são as grandezas elétricas

medidas e o número de *jumpers* configurado para cada sensor no momento da requisição de informações. Por fim, o parâmetro final é representado pelo caractere "f".

3.6 CONCLUSÕES PARCIAIS

Esse capítulo apresentou os *hardwares* necessários para o desenvolvimento do sistema. Foi possível compreender a integração entre o *Medidor digital de grandezas elétricas*(MDGE) e o *Módulo de comutação de estados*(MCE), além das funcionalidades de cada *hardware*.

O MDGE mostra-se fundamental durante os ensaios para a aquisição de dados referentes aos parâmetros elétricos na entrada e saída do controlador de LED. Para que seja possível essa coleta de dados na entrada e saída do *driver*, os dois canais distintos de medição do instrumento de aquisição de dados são conectados eletricamente no MCE, de forma que o primeiro canal é posicionado na entrada do *driver* e o segundo na saída do controlador de LED. Foi visto que cada sensor necessita de uma alteração manual do número de *jumpers* para mudar a escala do equipamento. A resolução de cada escala também foi apresentada, mostrando que a adição de *jumpers* implica no aumento da resolução.

Foi visto que os sensores dos canais transduzem o valor instantâneo de corrente ou tensão para um valor entre 0V e 3,3V, que é convertido para um sinal digital através do ADC do microcontrolador. Após o processamento dos dados, o microcontrolador envia para o *software* computacional de maneira serial o valor dos parâmetros elétricos medidos.

O MCE atua por meio de relés na alteração das conexões elétricas do circuito, possibilitando que as especifidades de cada ensaio seja atendida. Cada relé possui um circuito de controle próprio e é acionado no momento que um sinal digital baixo é setado na porta *GPIO* do microcontrolador, a qual é responsável pelo controle de um ou mais relés. O *software* computacional envia um protocolo para o microcontrolador definindo o estado atual das portas *GPIO* e consequentemente os estados dos contatos do relés. Dessa forma, o *software* computacional é capaz de gerenciar como estão as conexões elétricas da placa, possibilitado a emulação de todas as condições necessárias para a realização dos diferentes ensaios. Todos os pacotes de protocolos utilizados para a comunicação entre o programa embarcado do microcontrolador e o *software* computacional foram apresentados.

4 SOFTWARE

Esse capítulo tem por objetivo esclarecer os aspectos e características relacionadas a utilização do *software* desenvolvido. Cada módulo do *software* será mostrado em detalhes, de forma que seja possível entender a funcionalidade de cada um. Os detalhes acerca da implementação do *software* serão tratados no capítulo 5.

O *software* computacional, desenvolvido na linguagem C#, é responsável por gerenciar os diferentes tipos de ensaios, definindo os estados dos contatos dos relés, configurando o instrumento de medições digitais e requisitando dados acerca dos parâmetros elétricos medidos. Foram implementadas funcionalidades que utilizam um servidor de banco de dados, com o objetivo de armazenar permanentemente dados de *drivers*, luminárias e relatórios de ensaios realizados.

O *software* é dividido em cinco módulos:

1. Configuração da conexão com o equipamento.
2. Cadastro, edição e visualização de controladores de LED e luminárias.
3. Testes de funcionamento do equipamento.
4. Ensaios automatizados.
5. Banco de resultados de ensaios realizados.

4.1 TELA INICIAL

A tela inicial possui uma barra de menu que permite o usuário selecionar o módulo que deseja abrir. Para isso, o usuário deve clicar em *Início* e escolher um dos cinco módulos disponíveis. A figura 27 mostra a tela inicial do *software* e a barra de menu selecionada.

(a) Tela inicial do software

(b) Tela inicial com destaque a barra de menu

Figura 27: Tela inicial do *software*

4.2 CONFIGURAÇÃO DA CONEXÃO COM O EQUIPAMENTO

Esse módulo tem a função de configurar e conectar o equipamento ao computador. O *software* busca automaticamente as portas seriais disponíveis e o usuário deve selecionar em qual porta serial o equipamento está conectado. O usuário também deve selecionar qual a velocidade desejada de comunicação com o equipamento através da opção *Baud Rate*. Os parâmetros *databits*, paridade e *stopbits* já são previamente configurados como 8, nenhum e 1 e são iguais aos configurados no microcontrolador. Após selecionar a porta *COM* e o *Baud Rate*, basta clicar em *Conectar*. Caso não seja possível iniciar a conexão, uma janela será aberta informando o ocorrido. A figura 28 mostra a janela de configuração da conexão.

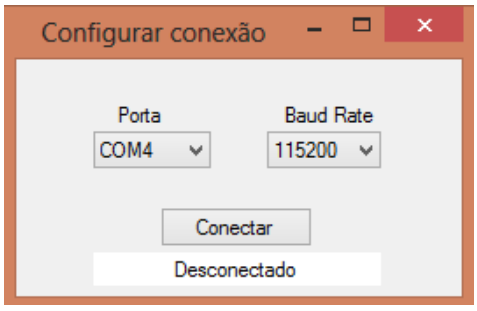


Figura 28: Módulo de configuração da conexão com o equipamento

4.3 CADASTRO, EDIÇÃO E BUSCA DE CONTROLADORES DE LED E LUMINÁRIAS

Esse módulo permite ao usuário cadastrar os controladores de LED e luminárias que serão utilizados nos ensaios, assim como editar ou buscar componentes já cadastrados.

A primeira aba permite o cadastro de novos controladores de LED e luminárias. Primeiramente, o usuário deve escolher se deseja cadastrar um *driver* ou uma luminária. No caso de *driver*, os dados que devem ser inseridos são: fabricante, código, eficiência(%), fator de potência, potência máxima(W), corrente de saída(A) e tensão de entrada(V). Para casos em que o *driver* aceite uma faixa de tensão de entrada, deve-se cadastrar a tensão em que serão realizados os ensaios. Além dos dados já mencionados que devem ser inseridos, o usuário ainda deve selecionar se o *driver* tem controle de tensão ou corrente.

Na seção imagem, pode-se selecionar uma imagem do *driver* para ser armazenada em conjunto com os dados. Caso o usuário deseje cadastrar uma luminária, basta selecionar a opção *Luminária*. Os dados que devem ser inseridos são: Fabricante, código, corrente(A) e tensão(V). Da mesma forma que o *driver*, também é possível selecionar uma imagem da luminária para ser armazenada. Após todos os dados serem inseridos, deve-se clicar em *Salvar*. Uma janela irá abrir informando se o *driver*/luminária foi cadastrado(a) com sucesso ou não no banco de dados. A figura 29 mostra um exemplo de cadastro de *driver*.

(a) Cadastrando Driver

(b) Janela aberta ao salvar *driver*

Figura 29: Aba de cadastro de *driver* e luminária

A segunda aba permite a edição de componentes já cadastrados. Para isso, o usuário primeiramente deve digitar o código do *driver* ou luminária que deseja editar e clicar em *Pesquisar* ou pressionar a tecla *Enter*. A outra possibilidade é buscar com

o campo do código vazio e serão exibidos todos os controladores de LED/luminárias já cadastrados. Caso o usuário tenha conhecimento de outros dados sobre o componente, mas não saiba o código especificamente, pode-se fazer uma pesquisa na aba *Buscar*, que ainda será explicada detalhadamente nos próximos parágrafos. Ao selecionar um componente encontrado, todos os dados referentes a ele serão mostrados nos campos de dados. Na seção *Imagem*, o usuário pode selecionar uma nova imagem ou excluir a existente no banco de dados. Ao fim da edição, basta clicar no botão *Salvar* e uma janela irá abrir informando o sucesso da operação. É possível também excluir um cadastro ao clicar no botão *Excluir*. A figura 30 mostra a edição dos dados de um *driver*.

(a) Editando Driver

(b) Janela aberta ao salvar edição dos dados do *driver*

Figura 30: Aba de edição de *driver* e luminária

A terceira aba permite ao usuário fazer uma consulta aos componentes já cadastrados. Basta escolher se está buscando um *driver* ou luminária e o tipo desejado de filtro para a pesquisa . Os filtros disponíveis para *driver* são: Fabricante, código, eficiência(%), fator de potência, potência máxima(W), corrente de saída(A), tensão de entrada(V), controle de tensão ou controle de corrente. Já para luminárias os filtros disponíveis são: Fabricante, código, corrente(A) e tensão(V).

Ainda é possível marcar a opção *Mostrar todas as colunas*, ou seja, dessa forma será disponibilizado para visualização todos os dados do *driver* ou luminária. A aba de busca, além de ser útil como uma forma de consulta aos componentes cadastrados, também pode ajudar o usuário a encontrar um *driver* ou luminária quando não se conhece exatamente seu código, mas sabe-se outros dados do componente. Dessa forma, caso o objetivo seja editar esse componente, basta encontrá-lo com a realização de uma busca e em seguida selecionar seu código clicando com o mouse sobre o mesmo. Por

fim, deve-se pressionar o botão *Copiar selecionado* e o código será automaticamente copiado para a aba de edição, onde serão exibidos seus dados e será possibilitada a edição. Um exemplo de busca de *driver* é visto na figura 31.

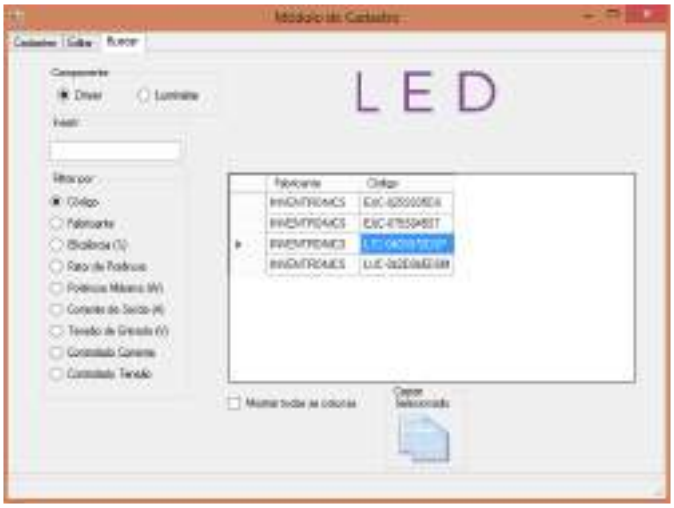


Figura 31: Aba de busca de *driver* e luminária

4.4 TESTES DE FUNCIONAMENTO DO EQUIPAMENTO

Esse módulo permite testar o correto funcionamento do equipamento e também visualizar as leituras do MDGE. Para realizar esses testes, é preciso que todos os equipamentos, incluindo o *driver* e luminária, estejam corretamente conectados, de acordo com o que foi mostrado no capítulo de *Hardware*.

Antes de iniciar algum teste, o usuário deve configurar o MCE, ou seja, definir qual condição que se deseja testar. Para isso, deve-se marcar os campos de seleção de acordo com certos critérios. A tensão nominal de entrada deve ser selecionada, 127V ou 220V.

Se o usuário optar por conectar um transformador, deve-se marcar a opção *Conectado*, e em seguida marcar a porcentagem referente ao transformador desejado. Caso deseje-se utilizar o transformador com relação de transformação 1:1,06, a opção 106% deve ser marcada. Senão, se o transformador desejado tiver relação de transformação 1:0,92, a opção 92% deve ser marcada.

Na linha referente ao *driver*, deve-se selecionar como será feita a conexão do controlador com o módulo de LEDs. Caso deseje-se conectar a saída do *driver* com o módulo de LEDs, a opção *Conectado LED* deve ser selecionada. Para colocar a saída do *driver* em circuito aberto ou curto-circuito, a opção *Aberto* ou *Curto* devem ser marcadas,

respectivamente.

Por fim, é preciso marcar a opção *Ligado* para fechar a chave geral do equipamento e testá-lo.

Para auxiliar o usuário na verificação do correto funcionamento do equipamento, é mostrada qual a tensão de entrada no *driver* esperada e o estado do módulo de LED (ligado ou desligado) esperado. No caso de curto-circuito, o *software* irá advertir o usuário com a palavra *curto-circuito* no canto superior direito da janela.

É disponibilizado também uma visualização do MDGE, de forma a permitir a verificação do correto funcionamento do equipamento. Para isso, basta clicar em *Visualizar Wattímetro*. Nessa visualização, estão disponibilizados todos os dados lidos pelo MDGE, tanto pelo canal 1 quanto pelo canal 2. Para que as medições sejam realizadas de maneira correta, o número de *jumpers* que devem ser inseridos nos sensores de tensão e corrente de ambos os canais devem estar de acordo com a escala apropriada como foi visto no capítulo 3. Ao configurar a escala deve-se atentar ao fato de que os valores das grandezas elétricas consideradas devem ser os de pico quando se tratarem de correntes e tensões alternadas. Neste equipamento, apenas o canal 1 do MDGE trabalhará com grandezas alternadas, uma vez que ele sempre estará conectado à entrada do controlador de LED.

Após inserir os *jumpers* nos sensores, é preciso informar ao microcontrolador a quantidade de *jumpers* inseridos, para que o programa embarcado possa selecionar a curva de ajuste adequada, como visto no capítulo 3. Para realizar essa configuração, é necessário selecionar o número de *jumpers* nas caixas *Jumpers Corrente* e *Jumpers Tensão*, para ambos canais. Os dados disponibilizados pelo MDGE são: tensão RMS, tensão média, corrente RMS, corrente média, potência, fator de potência e eficiência do *driver*. Através dessas leituras, é possível verificar se o equipamento está funcionando de acordo com o esperado para cada configuração do MCE. A figura 32 mostra a tela padrão de abertura do módulo de *Testes do Equipamento*, na qual a opção *chave geral* se encontra desmarcada. As aquisições apenas são feitas pelo MDGE e disponibilizadas para visualização quando a chave geral está fechada.

Figura 32: Módulo de testes de funcionamento do equipamento

4.5 ENSAIOS AUTOMATIZADOS

O módulo *Ensaios automatizados* permite o usuário realizar o principal objetivo desse trabalho, ou seja, executar os ensaios nos controladores de LED de forma automática. A primeira janela que é aberta é a de seleção do *driver* e da luminária. O usuário pode pesquisar por um código específico de *driver*/luminária, inserindo-o no campo de busca e clicando em *pesquisar*. A outra opção é buscar sem preencher o campo de texto e todos os códigos cadastrados serão disponibilizados para seleção. Em seguida, seleciona-se o *driver* que será testado e uma luminária compatível com esse *driver* e clica-se em *Próximo*. O *software* vai analisar a compatibilidade entre os componentes e caso sejam compatíveis a janela de configuração de *jumpers* do canal 1 será aberta. Caso contrário aparecerá uma janela de erro informando a incompatibilidade. A figura 33 mostra a janela que permite selecionar o controlador de LED e a luminária que serão ensaiados.

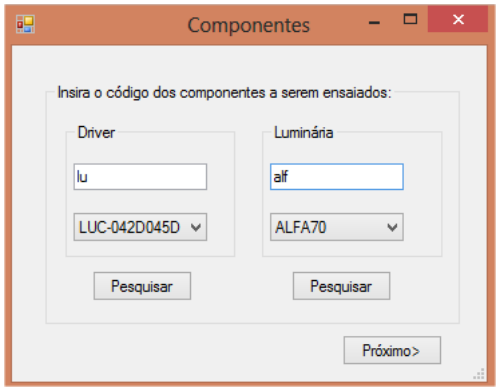


Figura 33: Janela de seleção dos componentes que serão utilizados

Na janela de configuração do equipamento, o usuário receberá as instruções de quantos *jumpers* devem ser inseridos nos sensores de corrente e tensão para iniciar os ensaios. Ao inserir o número indicado de *jumpers* em ambos os sensores do canal 1 basta clicar em *Próximo*. A janela seguinte se refere ao canal 2 e basta repetir o procedimento feito para o canal 1. Lembrando que é de vital importância essa configuração inicial do número de *jumpers*. Caso não seja realizada, há o risco de queimar o sensor de corrente ao ligar o módulo de LED, já que a corrente de entrada do *driver* apresenta um pico transitório durante a partida. A figura 34 mostra a janela de instrução para configuração dos sensores do MDGE.

Figura 34: Janela de configuração do equipamento

Após a configuração do número de *jumpers*, será aberta a janela para seleção de ensaios. Os ensaios disponíveis para o *driver* escolhido são habilitados para seleção, bastando o usuário marcar os testes desejados. A figura 35 mostra a tela que permite

selecionar os ensaios a serem realizados.

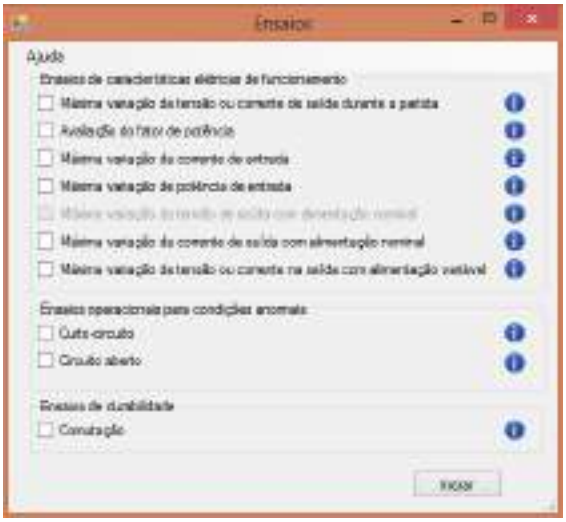


Figura 35: Janela de seleção dos ensaios

Caso o usuário deseje visualizar mais informações acerca de cada ensaio, basta deixar o mouse sobre a figura de informação a direita do nome de cada ensaio que será mostrado um resumo sobre o mesmo. Para um detalhamento completo acerca dos ensaios deve-se clicar em *Ajuda* e em seguida *Abrir documentação completa sobre os ensaios*. A figura 36 mostra a janela de documentação dos ensaios.

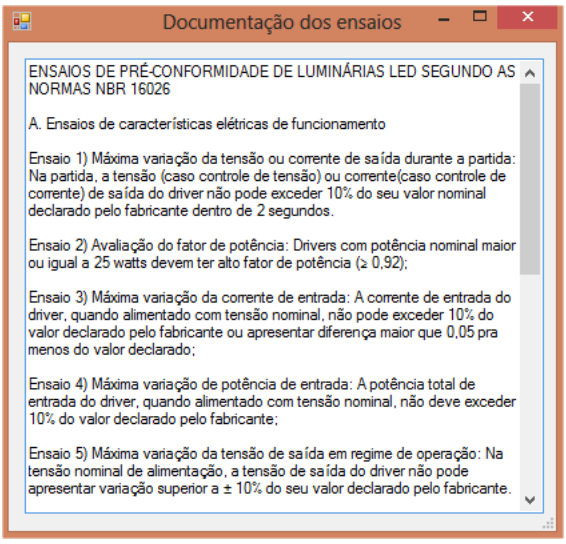


Figura 36: Documentação completa sobre os ensaios

Finalizado a seleção dos ensaios, basta clicar no botão *Iniciar* para começar a execução dos ensaios. Se a conexão do equipamento com o computador ainda não tiver sido realizada, irá abrir uma janela informando a necessidade da conexão e em seguida

a janela para configurar a conexão. Após conectar o equipamento, deve-se fechar a janela de configuração e clicar novamente em *Iniciar*.

A janela de execução será aberta e permitirá ao usuário verificar o andamento dos ensaios. Na parte superior da janela é informado o *driver* o qual está sendo submetido aos ensaios e também a luminária utilizada. A barra de progresso superior indica qual o ensaio está sendo realizado no momento e a barra de progresso inferior indica o andamento do atual ensaio. É possível realizar o cancelamento dos ensaios ao clicar no botão *Cancelar*. Dessa forma, o ensaio que estiver sendo executado no momento será cancelado e os próximos ensaios programados não serão executados. A figura 37 mostra a tela de execução de ensaios em um controlador de LED e luminária fictícios.

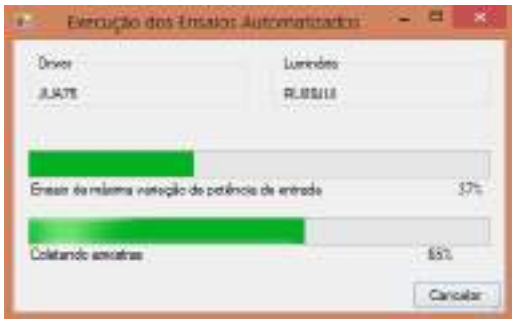


Figura 37: Janela de execução dos ensaios

Finalizados os ensaios, a última janela que será aberta é a *Resultados*. Todos os ensaios serão listados e do lado direito de cada um será mostrado o resultado, que pode ser: *aprovado*, *reprovado*, *cancelado* ou *não realizado*. A figura 38 mostra um exemplo da tela *Resultados*.

Figura 38: Janela de exibição dos ensaios realizados

Também estará disponível um relatório em arquivo de texto contendo todas as informações acerca de cada ensaio. O horário de início e fim dos ensaios, resultado de cada ensaio, amostras da grandeza elétrica alvo de cada ensaio, assim como a média dessas amostras. Esses dados obtidos em cada ensaio serão melhores explicados no capítulo 5.

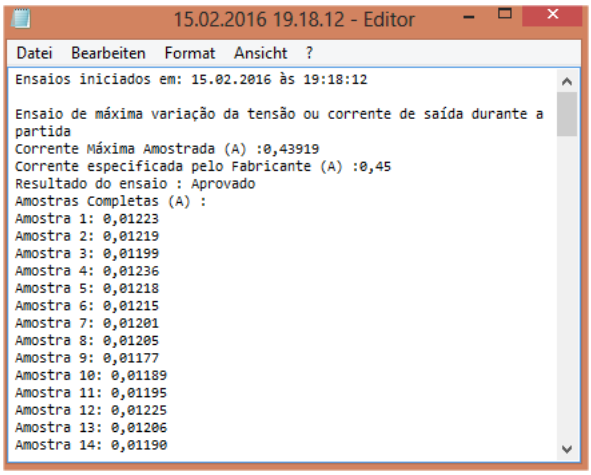


Figura 39: Relatório completo em formato texto sobre os resultados dos ensaios

4.6 BANCO DE RESULTADOS

O último módulo do *software* é o *Banco de resultados*, o qual irá utilizar um banco de dados para o armazenamento de resultados e relatórios de ensaios realizados. Para efetuar uma busca, deve-se digitar o código do *driver* e, caso seja encontrado registros no banco de dados, eles serão mostrados em uma tabela. As colunas da tabela informam os seguintes dados: hora de início e fim dos ensaios, código do *driver* e da luminária, o resultado de cada um dos ensaios e por fim o relatório completo em formato de texto acerca dos ensaios realizados. Para visualizar o relatório, basta selecionar uma célula da tabela e em seguida clicar em *Relatório txt*. No menu *Ajuda* é possível abrir a documentação completa dos ensaios para consultar a descrição de cada um, visto que na tabela os ensaios estão apenas numerados de 1 a 10, sem fornecer nenhuma informação complementar acerca do que se tratam os ensaios. A figura 40 mostra uma busca de *driver* no módulo *Banco de Resultados* .

Figura 40: Módulo *Banco de resultados*

4.7 CONCLUSÕES PARCIAIS

Esse capítulo apresentou todos os módulos do *software* possibilitando o entendimento para que o usuário possa cadastrar, editar e buscar um *driver* ou luminária, testar o equipamento, efetuar os ensaios automatizados, assim como verificar no banco de dados resultados e relatórios de ensaios já realizados.

Esse *software* é de fundamental importância para a automação dos ensaios, já que ele é necessário para estabelecer a comunicação entre o computador e o conjunto de *hardware*, comandando todos os procedimentos para a realização dos ensaios.

5 DESENVOLVIMENTO DO SOFTWARE COMPUTACIONAL

Esse capítulo tem o objetivo de demonstrar como foi desenvolvido o *software* computacional e quais foram as ferramentas necessárias para seu desenvolvimento. Inicialmente será feito uma revisão de conceitos teóricos utilizados e em seguida são apresentado os aspectos do desenvolvimento do *software*.

5.1 BANCO DE DADOS

Os bancos de dados estão cada dia mais presentes na vida das pessoas. Pode-se citar situações diárias de contato com banco de dados: depósitos e saques em bancos, busca de um livro na biblioteca informatizada da universidade, compra de um produto numa loja virtual na internet ou mesmo em um supermercado. Todos esses exemplos são atividades que requerem o acesso de um computador a um banco de dados. Dessa forma, os bancos de dados desempenham um papel importante em quase todas as áreas em que computadores são utilizados.

Um banco de dados nada mais é que uma coleção de dados que estão relacionados, os quais podem ser gravados e terem significado implícito (ELMASRI et al., 2005). As estruturas dos bancos podem ter complexidades variáveis, como por exemplo uma agenda telefônica pessoal, que possui apenas nome e número de telefone de conhecidos, contendo possivelmente até algumas centenas de registros. Por outro lado, em uma universidade a complexidade da estrutura do banco de dados é muito maior, contendo diversas categorias variadas e muitas delas relacionadas entre si, além de poder conter milhares de registros. Para definir um banco de dados é necessário especificar a estrutura de cada registro considerando o tipo de elemento a ser armazenado. No exemplo de uma universidade, o registro *Aluno* possui os dados como nome, data de nascimento, CPF, matrícula, turma, turno. Cada curso está relacionado a diversas disciplinas e cada disciplina por sua vez está relacionada a um código, ao número de créditos e ao departamento o qual oferece a disciplina. Como pode-se perceber, os dados estão relacionados e cada registro precisa ser armazenado através de um tipo de

dado específico. Por exemplo, o tipo de dado para o nome de um aluno deve ser uma cadeia de caracteres (*string*), já para o número de créditos de uma disciplina deve ser um inteiro (*integer*).

Um banco de dados pode ser gerado e gerenciado manualmente, mas atualmente a maioria dos bancos são automatizados. Dessa forma, são criados e mantidos por um conjunto de aplicativos próprios para realizarem essas tarefas e esse conjunto é chamado de sistema gerenciador de banco de dados (SGBD). Esse sistema permite ao usuário criar, manter, manipular e compartilhar bancos de dados(ELMASRI et al., 2005). Existem diversos SGBD disponíveis, entre eles pode-se citar o MySQL, MariaDB, Oracle, SQL-Server, entre outros.

O SGBD utilizado por esse trabalho foi o MySql, que representa um banco de dados do tipo relacional. Esse tipo de banco de dados representa os dados como uma simples coleção de linhas e colunas em tabelas bidimensionais, que se relacionam entre si, e dependendo desse relacionamento carregam dados de outras tabelas consigo como referência à tabela que se relaciona. Outros tipos de banco de dados podem ser citados: hierárquico, rede, orientado a objeto e objeto-relacional.

O MySQL foi desenvolvido pela companhia sueca MySQL AB e lançado em sua primeira versão em 1995 (DYER, 2008). Em 2008, a companhia sueca foi adquirida pela Microsystems Inc. pelo valor de 1 bilhão de dólares. Em 2010, a comissão europeia aprovou a compra da Microsystems Inc., e todos os seus produtos, incluindo o MySQL, pela companhia norte americana Oracle.

5.2 MYSQL

O sistema gerenciador de banco de dados MySQL é um dos mais populares quando o assunto é armazenamento dados. Estima-se que existam mais de 6 milhões de instalações do MySQL em todo mundo, atingindo a marca de 50 mil downloads por dia(TAHAGHOGHI & WILLIAMS, 2006). Muitas características contribuem para tal popularidade, dentre elas pode-se citar:

- Necessidade de *hardware* apenas modesto.
- Agilidade na recuperação de dados.
- Estabilidade e perfomance, inclusive para banco de dados extremamente extensos.

- Licença para uso próprio grátis, pagando apenas em casos que se distribua o MySQL associado a algum produto de forma comercial.
- Facilidade de interação direta a partir de programas escritos na maioria das linguagens de programação conhecida, como o C, PHP, Perl, Python, Ruby e as linguagens Microsoft .NET.

5.2.1 COMANDOS BÁSICOS

Essa seção apresentará os principais comandos do MySQL e como utilizá-los, assim como todos os bancos de dados criados e utilizados pelo sistema de ensaios automatizados.

Todos os registros de um banco de dados devem estar contidos em uma tabela e cada tabela pode conter uma ou mais colunas. Cada coluna armazena registros de um tipo de variável definida que pode estar contida nas seguintes categorias: tipos numéricos, tipos *string*, tipos espaciais ou tipos data/hora. Como exemplo de tipos de dados pode-se citar: *tinyint*, *bigint*, *char*, *varchar*, entre outros. Todos os tipos de dados numéricos podem ser vistos na tabela 9. Já a tabela 10 mostra os tipos *string* de dados existentes.

Nome do tipo	Significado
TINYINT	Inteiro muito pequeno
SMALLINT	Inteiro pequeno
MEDIUMINT	Inteiro médio
INT	Inteiro padrão
BIGINT	Inteiro grande
DECIMAL	Número em ponto fixo
FLOAT	Número em ponto flutuante com precisão simples
DOUBLE	Número em ponto flutuante com precisão dupla
BIT	Campo de bits

Tabela 9: Tipos numéricos de dados
(DUBOIS, 2009)

Nome do tipo	Significado
CHAR	Strig de comprimento fixo não binário
VARCHAR	Strig de comprimento variável não binário
BINARY	String binária de comprimento fixo
VARBINARY	String binária de comprimento variável
TINYBLOB	Objeto binário muito pequeno
BLOB	Pequeno BLOB [a]
MEDIUMBLOB	Médio BLOB
LONGBLOB	Longo BLOB
TINYTEXT	String não binária muito pequena

[a] Um BLOB (*objeto grande binário* significa uma coleção de dados binários armazenados como uma única entidade em um sistema de gerenciamento de banco de dados)

Tabela 10: Tipos *string* de dados(DUBOIS, 2009)

As tabelas dos tipos de dados espacias e data/hora não serão apresentados pois não foram utilizados nesse trabalho, porém elas podem ser encontradas em DUBOIS(2009).

Todos os comandos que serão apresentados podem ser inseridos no painel da aba *Query* e para executa-los basta clicar no botão com a imagem de um raio, como mostrado na figura 41.

Figura 41: Painel de inserção de comandos do MySQL Workbench

Nesse trabalho, apenas o banco de dados e as tabelas foram criadas utilizando o painel do programa *MySQL Workbench*, já que as ações de inserção, atualização, exclusão e recuperação de dados foram realizadas via interfaceamento com o *software* do sistema em C#.

Para entender os principais comandos é utilizado o exemplo de um banco de dados de uma livraria fictícia(DYER, 2008).

O primeiro passo é criar um banco de dados, utilizando o comando *CREATE*, como

pode ser visto na figura 42.

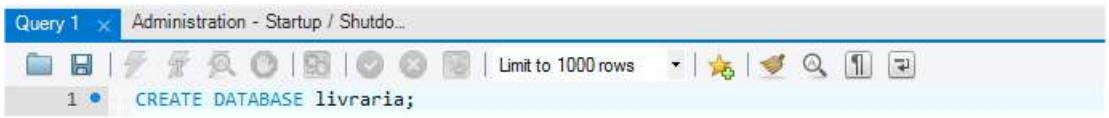


Figura 42: Criando um banco de dados

É preciso especificar com qual banco de dados se está trabalhando, utilizando o comando *USE*, como mostrado na figura 43.

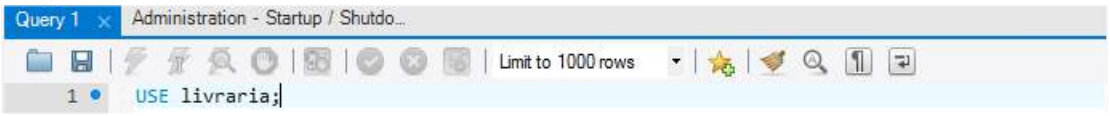


Figura 43: Selecionando um banco de dados

Ao criar uma tabela, deve-se especificar um nome para a própria, assim como o nome das diferentes colunas as quais vão armazenar dados. Lembrando que cada coluna deve ter um tipo de variável definido que é declarado no momento de criação da tabela após o nome da respectiva coluna. A figura 44 mostra a criação de uma tabela em um banco de dados.

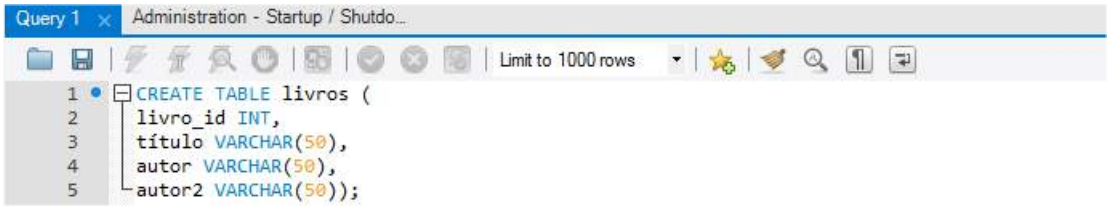


Figura 44: Criando uma tabela em um banco de dados

Pode-se perceber que no caso do tipo de variável *string* (VARCHAR) é preciso especificar também o tamanho da cadeia de caracteres. Nesse exemplo, o título de um livro foi limitado para conter até um máximo de 50 caracteres.

A tabela após ser criada pode ser alterada, para situações em que seja necessário adicionar ou excluir colunas, bem como editar o nome ou mesmo o tipo de dados que uma coluna armazena. Para isso deve-se usar o comando *ALTER* e os parâmetros *ADD*, *DROP*, *CHANGE*, de acordo com a figura 45.

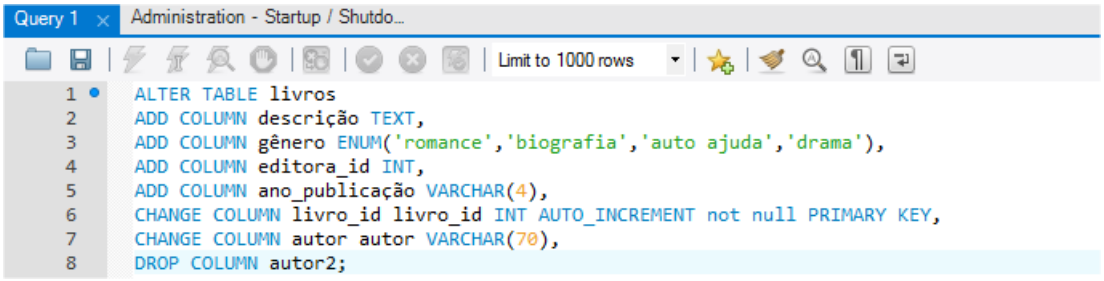


Figura 45: Alterando colunas de uma tabela

Analisando a figura 45, a coluna *autor2* foi excluída e novas colunas para descrição, gênero, editora e ano foram adicionadas. A coluna de gênero foi adicionada com o tipo de dado *ENUM*, que restringe os registros que serão adicionados a apenas algumas possibilidades especificadas.

A coluna *livro_id* permaneceu com o mesmo nome, porém seu tipo foi alterado, tornado-se do tipo inteiro e que tem auto incremento (*AUTO_INCREMENT*), ou seja, toda vez que um livro for adicionado, esse terá o *id* do último livro adicionado acrescido de um. Além disso, foi definido que a coluna *livro_id* é chave primária (*PRIMARY KEY*). A chave primária permite uma recuperação de dados muito mais rápida e, principalmente, permite identificar um determinado registro de forma única, permitindo relacioná-lo com outros registros de forma única. Além disso, uma tabela pode ter uma única chave primária, que pode ser formada por uma ou mais colunas (chaves primárias simples e compostas). No caso da livraria é coerente que cada livro seja registrado com um *id* diferente. O parâmetro *not null* não permite que o campo *livro_id* seja armazenado com um valor nulo.

A coluna *autor* permaneceu com o mesmo nome e tipo de variável, porém o tamanho de sua cadeia de caracteres foi alterado de cinquenta para setenta.

Para inserir dados na tabela utiliza-se o comando *INSERT* e os parâmetros *INTO VALUES*. É necessário especificar em qual tabela serão inseridos dados, as colunas envolvidas e os próprios dados, de acordo com a figura 46.

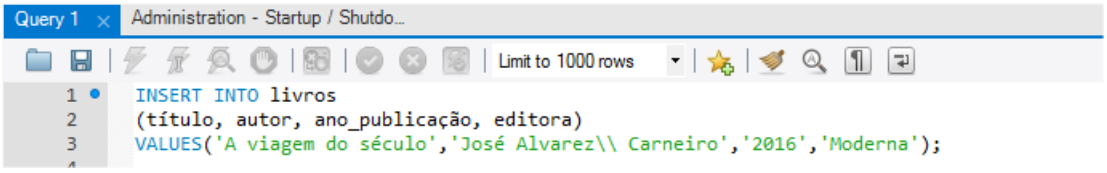


Figura 46: Inserido dados em uma tabela

Para atualizar dados já inseridos, utiliza-se o comando *UPDATE* e os parâmetros *SET*, *WHERE*. É necessário especificar qual a tabela, assim como a coluna e o valor que será atualizado, além da linha correspondente a esse registro através de uma condição, de acordo com a figura 47.

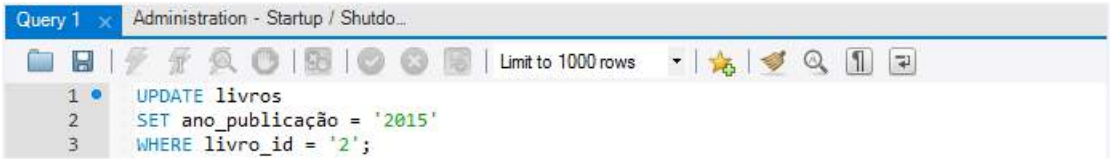


Figura 47: Atualizando dados em uma tabela

Como pode-se perceber a atualização foi feita na tabela *livros*, utilizando a condição *WHERE* para atualizar apenas o livro com *id* igual a dois. Já o ano de publicação foi alterado para 2015 através do parâmetro *SET*.

Para deletar um registro usa-se o comando *DELETE*, especificando o nome da tabela e também uma condição que determine exatamente o registro que será excluído, de acordo com a figura 48.

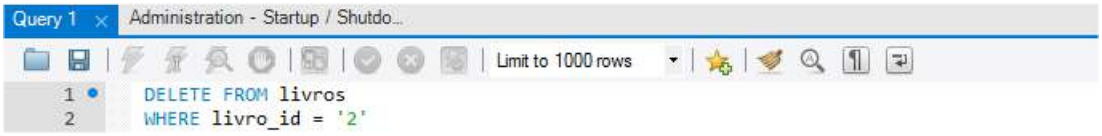


Figura 48: Excluindo dados de uma tabela

Nesse caso foi excluído o livro com *id* igual a dois, pertencente a tabela *livros*.

O sistema gerenciador de banco de dados MySQL permite realizar consultas a um banco de dados de forma bem simples, essas consultas são conhecidas como *query*. O comando *SELECT* e o parâmetro *FROM* são utilizados para execução dessa tarefa, de acordo com a figura 49.

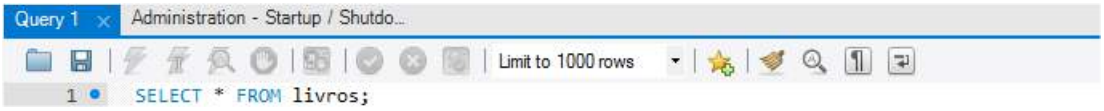


Figura 49: Consulta à um banco de dados

Utilizando o asterisco, todas as colunas e linhas da tabela *livros* serão selecionadas. Para selecionar apenas colunas específicas da tabela deve-se adicionar e especificar os

nomes de tais colunas após o comando *SELECT*, de acordo com a figura 50. Dessa forma, apenas as colunas *livro*, *título*, *descrição*, *autor* e todas as linhas correspondentes a essas colunas serão selecionadas.

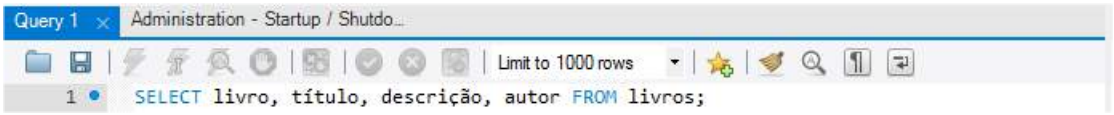


Figura 50: Seleção de colunas de uma tabela

Para selecionar uma ou mais linhas específicas deve-se adicionar novamente a cláusula condicional *WHERE*, de acordo com a figura 51. Neste caso as colunas *livro*, *título*, *descrição* e *autor* de todas as linhas cujo gênero seja biografia serão selecionadas.

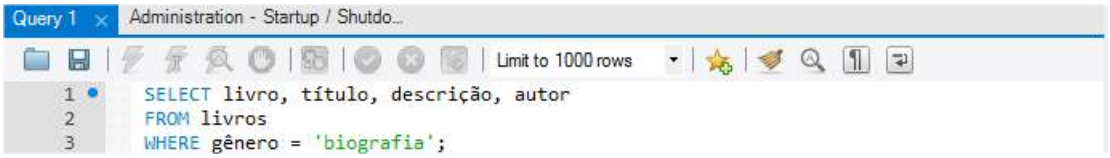


Figura 51: Seleção de linhas de uma tabela

Para selecionar registros iniciados ou terminados por uma sequência específica de caracteres, utiliza-se o parâmetro *LIKE*, de acordo com a figura 52.

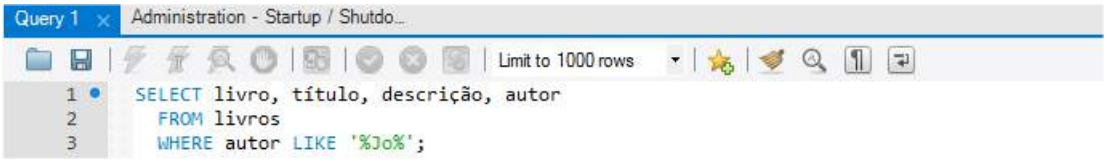


Figura 52: Seleção condicional de registros de uma tabela

Utilizando o comando da figura 52, todas as linhas que contenham a sequência de caractere *Jo* na coluna autor serão selecionadas, já que a palavra está cercada por dois símbolos de porcentagem. Caso houvesse apenas um símbolo de porcentagem posicionado no início da sequência, somente os registros terminados por *Jo* seriam selecionados. Analogamente, se houvesse apenas um símbolo de porcentagem posicionado no fim da sequência, somente os registros iniciados por *Jo* seriam selecionados.

Utilizando o comando *DESCRIBE* é possível visualizar os campos existentes nas tabelas, o tipo de variável correspondente a cada campo e ainda se o campo aceita valores nulos ou se é chave primária.

5.3 BANCO DE DADOS NIMO

Nesse trabalho foi criado inicialmente um banco de dados com o nome de *nimo* e nesse banco três tabelas, uma chamada *tabela_driver*, outra *tabela_luminaria* e uma última chamada *tabela_resultados*. A figura 53 mostra o diagrama relacional do banco de dados NIMO.

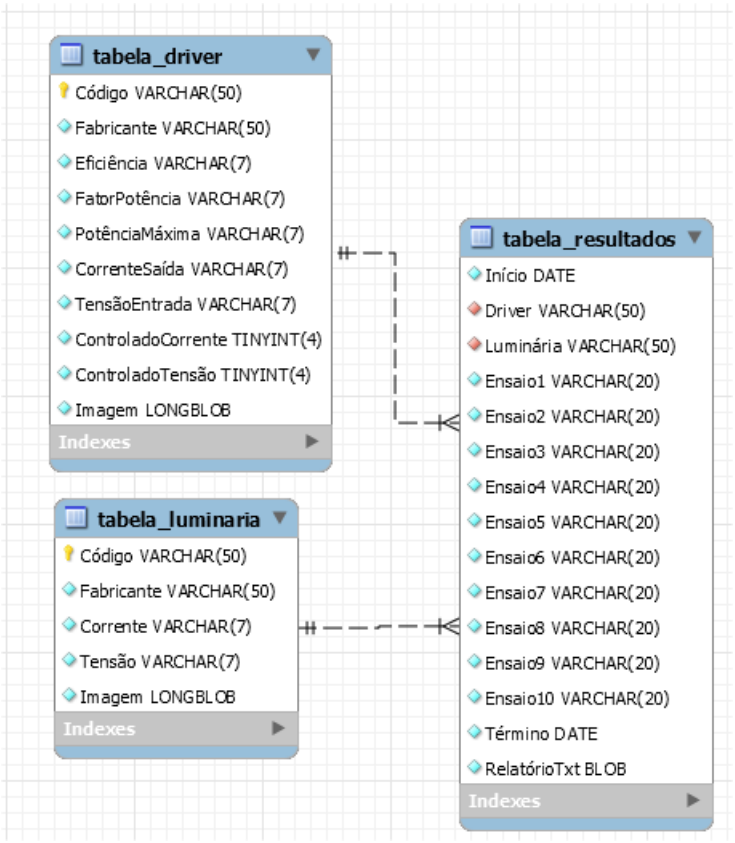


Figura 53: Diagrama relacional do banco de dados NIMO

A *tabela_driver* tem a função de armazenar os dados referentes ao cadastro de *drivers*. Os campos *Fabricante* e *Código* armazenam o nome do fabricante e código do driver, respectivamente. Os campos *Eficiência*, *FatorPotência*, *PotênciaMáxima*, *CorrenteSaída* e *TensãoEntrada* armazenam a eficiência (%), o fator de potência, a potência máxima do *driver* (W), a corrente de saída(A) e a tensão de entrada (V) do *driver*. Os campos *ControladoCorrente* e *ControladoTensão* armazenam um único *bit* representando se o *driver* tem controle de corrente ou de tensão, respectivamente. O campo *Imagem* armazena um arquivo binário que será convertido pelo *software* em C# para um arquivo de imagem, de forma que seja possível a visualização da foto do *driver*.

A *tabela_luminaria* tem a função de armazenar os dados referentes ao cadastro de luminárias. Os campos *Fabricante* e *Código* armazenam o nome do fabricante e o código da luminária, respectivamente. Os campos *Corrente* e *Tensão* armazenam a corrente, em Amper, e a tensão, em Volt, da luminária. O campo *Imagem* pode ser interpretado novamente da mesma forma que foi feito para o *driver*.

Como pode ser observado nas tabelas de *drivers* e luminárias todos os valores numéricos foram armazenados como uma cadeia de caracteres(string), por isso o *software* em C# irá converter para tipos numéricos, de forma que seja possível a realização de cálculos com esses valores. Os valores numéricos foram armazenados como *string* pois os mesmos são provenientes de campos de textos do *software* computacional.

A *tabela_resultados* tem a função de armazenar os resultados de ensaios realizados. Os campos *Início* e *Término* representam a data/hora do início e fim dos ensaios. O campo é *Início* é definido chave primária e não nulo, já que o início dos ensaios vai ter sempre obviamente uma única data/hora. Os campos *Driver* e *Luminária* armazenam o código dos componentes utilizados para os ensaios. Os campos de *Ensaio1* até *Ensaio10* armazenam apenas o resultados dos ensaios, ou seja, se foi aprovado, reprovado, cancelado ou não foi realizado. O campo *RelatórioTxt* armazena um arquivo binário que será convertido pelo software em C# para um arquivo texto, de forma que seja possível a recuperação e visualização do relatório completo dos ensaios.

O uso de banco de dados nesse trabalho foi importante para o armazenamento de dados referentes aos controladores de LED, luminárias e resultados de ensaios. Nessa seção foi mostrada, além de uma visão geral sobre banco de dados e MySQL, os principais comandos relacionados ao SGBD e que foram utilizados para a realização desse trabalho. Existem diversos outros comandos que não foram apresentados e podem ser encontrados na bibliografia indicada.

5.4 LINGUAGEM ORIENTADA A OBJETOS

Para o desenvolvimento do *software* computacional foi utilizada a linguagem C#. Essa linguagem teve origem no ano 2000, quando a companhia *Microsoft* anunciou seu lançamento. Suas raízes estão presentes na linguagem *C*, *C++* e *Java*. É uma linguagem muito poderosa que tem capacidades semelhantes a do *Java*, atendendo a maior parte das necessidades para o desenvolvimento de *softwares* em larga escala nos dias de hoje. (DEITEL, 2012)

O C# é uma linguagem orientada a objetos, de forma que, um conjunto de classes definem os objetos presentes no sistema. Cada classe possui atributos que podem ser interpretados como as características dos objetos provenientes dessa classe. Além disso, uma classe possui métodos que podem ser definidos como as habilidades e o comportamento dos objetos oriundos dessa classe. Pode-se perceber então, que uma classe define o modelo de um objeto, mas não representa um objeto em particular (MENDES, 2009) .

Para simplificar o entendimento desses conceitos, basta pensar num objeto televisão. Uma televisão possui características como cor, marca, modelo, tamanho, peso, número de portas USB e entradas HDMI, entre outras. Essas característica podem ser consideradas atributos de um objeto da classe televisão. Esse objeto pode ligar, desligar, mudar de canal, aumentar ou diminuir o volume, programar funções como *Sleep* ou mudar a entrada de vídeo. Todas essas ações podem ser consideradas como métodos de um objeto de uma determinada classe. As classes definem quais os atributos e métodos que um objeto pode assumir. Logo um objeto da classe televisão só pode conter os atributos e métodos que forem definidos para essa classe de objetos.

Novas classes podem ser criadas pelo desenvolvedor além de ser possível acessar uma larga coleção de classes pré-contruídas, chamada de *.NET Framework Class Library*. Essa biblioteca possibilita ao desenvolvedor criar *softwares* de maneira fácil e rápida. (DEITEL, 2012)

5.5 DIAGRAMA DE CASOS DE USO

O diagrama de casos de uso é um dos nove tipos de diagramas existentes na linguagem de modelagem unificada (UML). O UML é uma linguagem que define uma série de artefatos que facilita a tarefa de modelar e documentar os sistemas orientados a objetos que são desenvolvidos (GUEDES, 2011).

O diagrama de casos de uso busca apresentar uma visão externa e geral do sistema de forma a possibilitar que o usuário, inclusive pessoas leigas, possam compreender o comportamento do sistema. Através de uma linguagem simples, esse diagrama apresenta as funcionalidades do sistema sem demonstrar a forma que essas foram implementadas. É também de importante auxílio para a identificação e compreensão dos requisitos do sistema além de identificar quais os tipos de usuários que poderão interagir com o sistema e quais funções cada um poderá solicitar (GUEDES, 2011).

O diagrama de casos de uso é formado por atores e casos de uso. Os atores são identificados por "bonecos magros"e representam no diagrama qualquer elemento externo que esteja interagindo com o sistema. Uma breve descrição do ator localiza-se abaixo de seu símbolo. Já os casos de uso são identificados por elipses e representam as funcionalidades que podem ser solicitadas pelos atores que interagem com o sistema (GUEDES, 2011). A figura 54 mostra a simbologia dos atores e casos de uso que são utilizadas na construção dos diagramas.

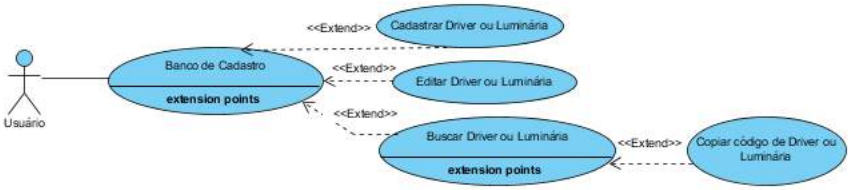
(a) Ator (b) Caso de uso

Figura 54: Simbologias utilizadas nos diagramas de caso de uso

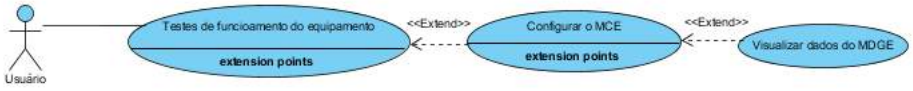
As associações podem representar a relação entre os atores e os casos de uso, entre os casos de uso ou entre os atores (AMBLER, 2005). As principais associações são chamadas de extensão, inclusão e multiplicidade. A extensão, representada por uma seta tracejada e sobre ela a palavra *extend*, aponta para um cenário opcional, ou seja, uma funcionalidade possível mas não obrigatória. A inclusão, representada por uma seta tracejada e sobre ela a palavra *include*, aponta para um cenário obrigatório, ou seja, considerando uma associação entre duas funcionalidades, a execução da primeira funcionalidade obriga a execução da segunda(GUEDES, 2011).

O diagrama de casos de uso do sistema é extenso, dessa forma foi dividido para cada módulo do *software* computacional de forma a facilitar a visualização, de acordo com a figura 55. As funcionalidades de cada módulo mostrado nos diagramas foram apresentadas no capítulo 4.

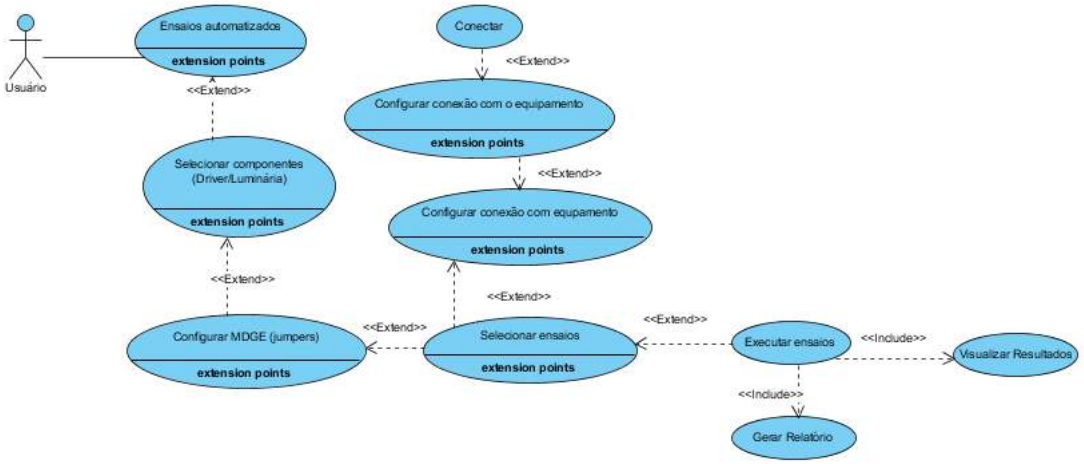
(a) Módulo de configuração da conexão com o equipamento



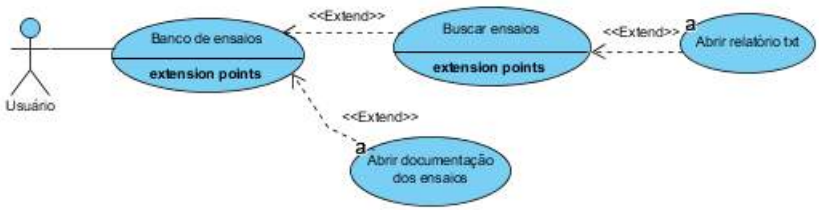
(b) Módulo de cadastro, edição e busca de componentes



(c) Módulo de testes do equipamento



(d) Módulo de ensaios automatizados



(e) Módulo de banco de resultados

Figura 55: Diagramas de caso de uso do sistema

5.6 DIAGRAMA DE CLASSES

Como já explicado na seção 5.4, a linguagem C# é orientada a objetos e implementada a partir de um conjunto de classes. Para facilitar o entendimento da interação entre as classes, é utilizado o diagrama de classes.

Um diagrama de classes representa uma apresentação das classes de um sistema, mostrando o relacionamento entre elas, bem como suas características (AMBLER, 2005). Em UML, a classe é representada por um retângulo com um ou mais compartimentos. O único compartimento obrigatório no retângulo é o superior, o qual indica o nome da classe. O compartimento central e o inferior são opcionais e representam os atributos e os métodos, respectivamente(WILLIAMS, 2004). A notação completa de um diagrama de classe UML pode ser visto na figura 56.

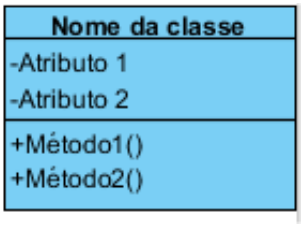


Figura 56: Representação UML para uma classe

A visibilidade do objeto na definição da classe é definida com um símbolo a esquerda do atributo de acordo com a tabela 11.

Tipo	Descrição
(-) privada	Apenas os objetos das classes podem acessar os atributos e métodos.
(#)protegida	Objetos de classes derivadas podem acessar os atributos e métodos.
(+)pública	Qualquer objeto pode acessar um atributo ou método da classe.
(˜)pacote	O atributo ou método é visível por qualquer objeto dentro do pacote.

Tabela 11: Tipos de visibilidade de objetos

Existem diversos tipos de relacionamentos entre as classes, entre eles pode-se citar: generalização, agregação, composição, associações, dependência, realização.

Os dois tipo de relacionamentos utilizados no desenvolvimento do *software* computacional foram a composição, a agregação e dependência.

O relacionamento de composição estabelece uma relação forte entre as classes, geralmente quando uma classe é instanciada dentro de outra. O símbolo que representa a composição em um diagrama de classes pode ser visto na figura 57.

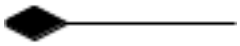


Figura 57: Símbolo de composição UML

O relacionamento de agregação estabelece uma relação mais fraca entre as classes, normalmente quando um objeto é passado como parâmetro construtor para outra classe. O simbolo para esse relacionamento pode ser visto na figura 58.

Figura 58: Símbolo de agregação UML

O relacionamento de dependência estabelece uma relação entre as classes em que geralmente uma classe apenas dispara um método de outra classe. O símbolo para esse relacionamento pode ser visto na figura 59.

Figura 59: Símbolo de dependência UML

5.7 DESCRIÇÃO DO SOFTWARE COMPUTACIONAL

Essa seção tem o objetivo de mostrar as novas classes que foram criadas durante o desenvolvimento do *software* computacional e o relacionamento entre elas. As classes pré-construídas da biblioteca *.NET Framework Class Library* que foram utilizadas está disponível na tabela 22 contida no apêndice A.

Foram criadas seis classes novas durante o desenvolvimento do *software*.

- Classe *Driver*

- Classe *Luminária*
- Classe *SerialWrite*
- Classe *SisDB*
- Classe *Ensaios Automatizados*
- Classe *Resultados*

A classe *Driver* armazena aos dados de um controlador de LED. O diagrama de classes pode ser visto na figura 60.

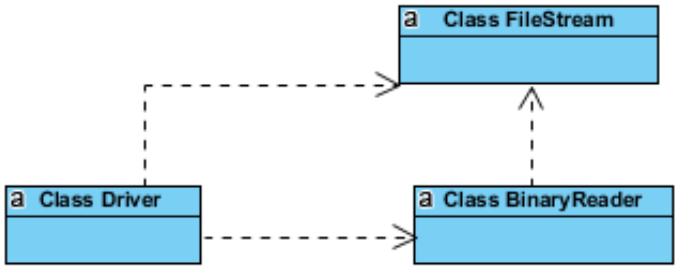


Figura 60: Diagrama de classes referente a classe *Driver*

A declaração da classe *Driver* pode ser vista na figura 61.

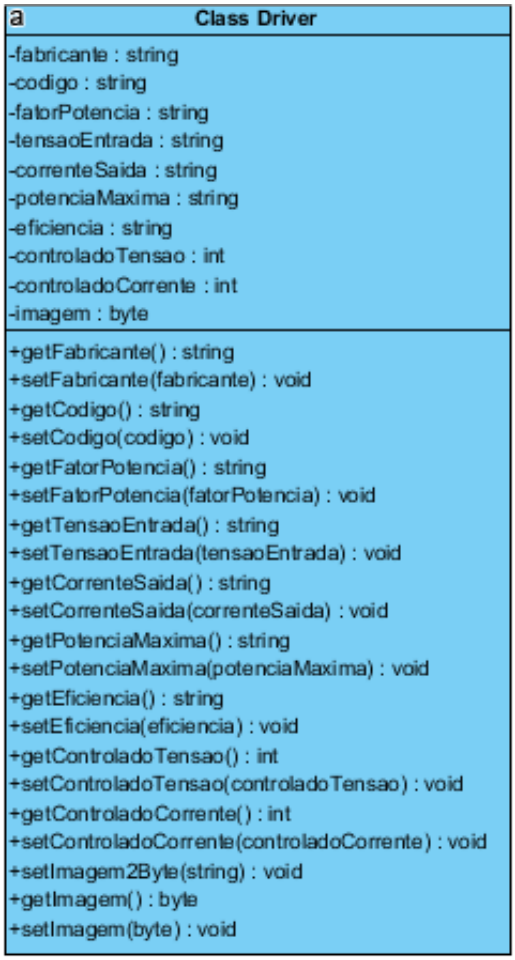


Figura 61: Atributos e métodos da classe *Driver*

A classe *Driver* tem os atributos referentes as características de um controlador de LED, ou seja, fabricante, código, fator de potência, tensão de entrada, corrente de saída, potência máxima, eficiência e tipo de controle (tensão ou corrente). Além disso, essa classe contém o atributo *imagem*, de forma a armazenar uma foto ou figura do *driver*.

A potência e corrente de entrada são obtidas indiretamente através dos dados fornecidos, de acordo com as equações 5.1 e 5.2. O cálculo é feito através de um método da classe *FormSelecionarComponentes* visto no diagrama de classes da figura 76.

$$P_{entrada} = \frac{V_{saida} \cdot I_{saida}}{eficiencia} \tag{5.1}$$

Substituindo $P_{entrada}$ na equação 5.2, obtém-se a corrente de entrada.

$$I_{entrada} = \frac{P_{entrada}}{V_{entrada} \cdot FP} \tag{5.2}$$

Os métodos referentes a classe *Driver* são em geral do tipo *get* e *set*, ou seja, são utilizados apenas para ler(*get*) e atribuir(*set*) valores aos atributos. Com exceção do método *setImagem2Byte*, que é utilizado apenas converter uma variável do tipo *Image* para *Byte*, já que não é possível armazenar no banco de dados uma variável do tipo *Image*. Nesse método cria-se um *stream*, através da classe *FileStream*, a partir da leitura do arquivo de imagem, e e em seguida é instanciado um objeto da classe *BinaryReader*, o qual possui o método *ReadBytes* que retorna a instância desejada do tipo *Byte*.

O diagrama de classes para a classe *Luminaria* é mostrado na figura 62

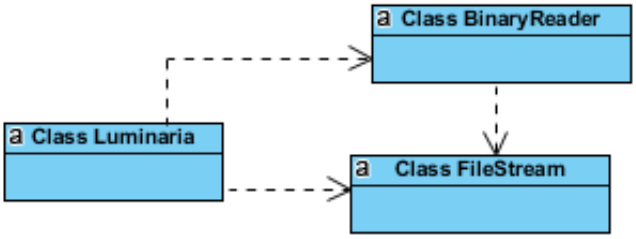


Figura 62: Diagrama de classes referente a classe *Luminaria*

A declaração da classe *Luminaria* pode ser vista na figura 63.

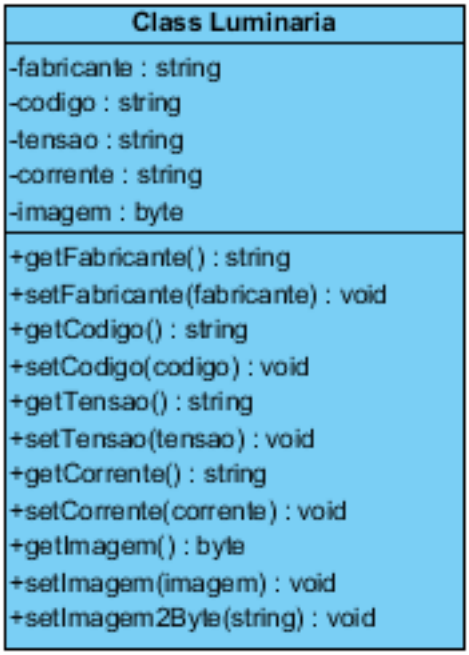


Figura 63: Atributos e métodos da classe *Luminaria*

A classe luminária tem os atributos referentes as características de uma luminária,

ou seja, fabricante, código, tensão e corrente. Além disso, essa classe contém o atributo *imagem*, de forma a armazenar uma foto da luminária. Com a mesma finalidade que a classe *driver*, a classe *luminária* possui um método *setImagem2Byte*.

A classe *SerialWrite* tem grande importância para a comunicação entre o *software* computacional e o programa embarcado do microcontrolador. O relacionamento dessa classe com outras pode ser visto na figura 64.

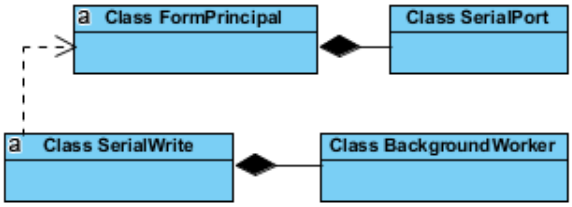


Figura 64: Diagrama de classes referente a classe *SerialWrite*

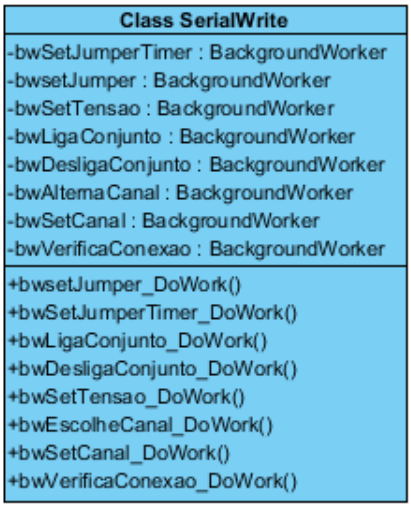


Figura 65: Atributos e métodos da classe *SerialWrite*

Os métodos dessa classe irão configurar não apenas configurações do MDGE, mas também o MCE. Os métodos da classe SerialWrite serão apresentados nos parágrafos seguintes.

O método *bwSetJumperTimer* verifica se houve alteração no número de *jumpers* configurado pelo usuário no módulo de *Testes do Equipamento*, já que nesse módulo o *software* computacional não conhece o *driver* e luminária que estão sendo utilizados. Acionado através de um objeto do tipo *Timer*, verifica o número de *jumpers* configurado no programa embarcado do microcontrolador do MDGE. Em caso de não igualdade com

o número definido pelo usuário, o *software* computacional irá configurar o programa embarcado do microcontrolador com o novo valor através do pacote *JP*, como visto no capítulo 3.

Os métodos a seguir têm a finalidade de configurar o MCE de maneira que o estado do sistema seja o desejado.

O método *bwsetJumper* configura o número de *jumpers* de ambos canais do MDGE no programa embarcado do microcontrolador antes de iniciar os ensaios, através do pacote *JP*. Como o número de *jumpers* é definido de acordo com o *driver* e a luminária selecionados, a configuração é feita automaticamente pelo *software*, não havendo a necessidade de uma configuração manual como no módulo *Testes do Equipamento*.

O método *bwSetTensao* verifica qual a tensão de entrada desejada do *driver* e envia um pacote *SET* referente a tensão nominal, e caso seja necessário também, um pacote referente ao transformador que será utilizado.

O método *bwLigaConjunto* tem como principal função realizar a energização do equipamento e configurar as conexões na entrada e na saída do *driver*. Primeiramente, é feita uma verificação se a ligação da saída do *driver* será um curto-circuito, um circuito aberto ou uma conexão com o módulo de LEDs e se o equipamento será ligado diretamente à fonte ou à algum transformador. Depois de verificado, pacotes *SET* adequados são enviados pela porta serial de acordo com as conexões desejadas. É interessante destacar, que apenas um segundo após a realização dessas conexões, será enviado o pacote *STR40* solicitando o fechamento da chave geral do equipamento, e consequentemente a energização de todo o conjunto.

O método *bwDesligaConjunto* realiza a desenergização do conjunto e a configuração *Default* de conexão do equipamento, que corresponde a alimentação do equipamento com tensão nominal, saída do *driver* conectada ao módulo de LEDs e chave geral aberta. Primeiramente a chave geral do equipamento é aberta, através de um pacote *STR41*. Em seguida, é verificado se algum transformador está conectado ao equipamento e qual o tipo de ligação é utilizada entre o *driver* e o módulo de LEDs. No caso de curto-circuito, aguarda-se 1s antes de removê-lo através de um pacote *STR51*. Para a saída do *driver* em circuito aberto, aguarda-se 30s antes do *driver* ser conectado ao módulo de LEDs pelo protocolo *STR60*. Caso algum transformador esteja conectado, será removido pelo protocolo *STR20*. Este tempo entre a retirada da falta e a energização do conjunto tem o objetivo de preservar a integridade do *driver* , uma vez que em circuito aberto a tensão na saída do *driver* tende a ser muito maior que a do módulo

de LEDs.

O método *bwAlternaCanal* envia alternadamente os pacotes *CAN* para requisição dos dados referentes ao canal 1 e para o canal 2. A execução desse método é sempre realizada através de um objeto da classe *Timer*. Esse método é utilizado no módulo *Testes Equipamento*, visto que é necessário a visualização simultânea das grandezas medidas de ambos os canais.

O método *bwSetCanal* envia o protocolo específico para a requisição dos dados referentes à apenas um canal, ou canal 1 ou canal 2.

O método *bwVerificaConexao* é disparado por um objeto da classe *Timer* com ciclos de 2s e tem o objetivo de verificar se o envio de dados através da porta serial está funcionando corretamente. Caso a comunicação falhe, a porta serial é fechada.

A classe *SisDB* é responsável pela interação do *software* computacional e do sistema gerenciador de banco de dados MySQL.

A figura 66 mostra os atributos e os métodos da classe *SisDB*.

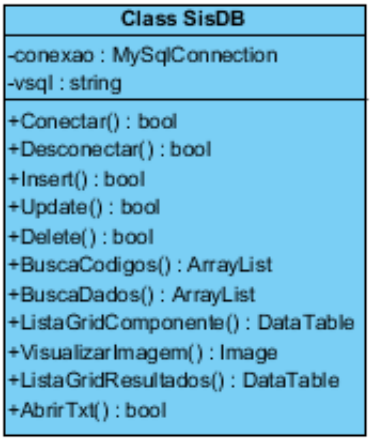


Figura 66: Atributos e métodos da classe *SisDB*

Essa classe possui dois atributos, chamados de *vsql* e *conexão*. O atributo *conexão*, pertencente a classe *MySqlConnection*, quando instanciado terá a função de conectar e desconectar o *software* computacional a um banco de dados armazenado em um servidor. O atributo *vsql* é utilizado para armazenar uma *string* correspondente a um comando do MySQL, que pode representar a inserção, exclusão, busca de dados, entre outros, como foi visto na seção 5.1. A classe contém diversos métodos de interação com o banco de dados e cada um representa uma ação diferente.

O diagrama de classes que mostra a relação da classe *SisDB* e *MySqlConnection*

pode ser visto na figura 66. Estas classes se relacionam nos métodos *conectar* e *desconectar*.

Figura 67: Diagrama das classes que se relacionam nos métodos *Conectar* e *Desconectar*

No método *conectar* é instanciado um objeto da classe *MySqlConnection* e é realizado a conexão entre o *software* computacional e o SGBD. No método *desconectar*, o estado da conexão é verificado e em seguida a conexão é encerrada caso esteja aberta.

O diagrama de classes complementar do módulo de Cadastro pode ser visto na figura 68.

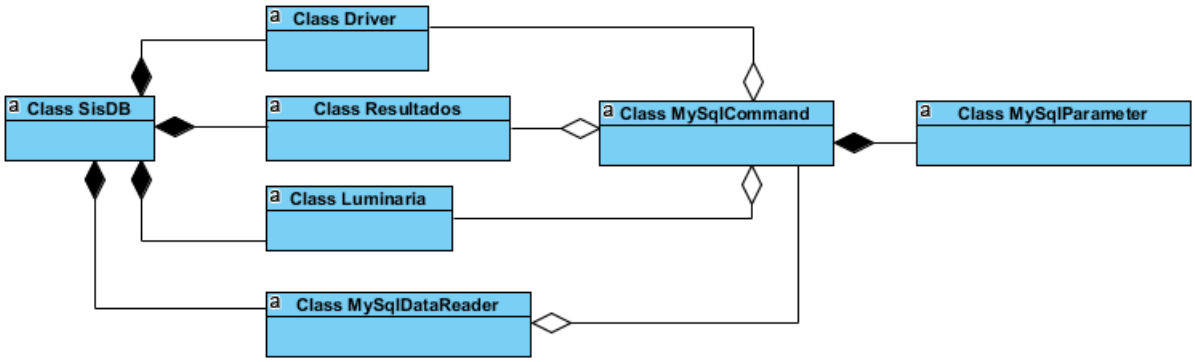


Figura 68: Diagrama de classes complementar do módulo de Cadastro

O método *Insert* tem como finalidade a inserção de dados de *drivers* e luminárias no SGBD. O método recebe como parâmetro um objeto da classe *object*, que pode representar qualquer tipo de objeto. No caso, serão passados como parâmetro um objeto da classe *Luminária*, *Driver* ou *Resultados*, de forma que o método vai reconhecer qual componente está sendo passado como parâmetro e consequentemente irá inserir os dados na tabela apropriada do banco de dados. Os atributos *vsql* e *conexão* são passados como parâmetro para a inicialização de um objeto da classe *MySqlCommand* que através do método *ExecuteReader* finaliza a inserção de dados no banco, retornando um objeto da classe *MySqlDataReader*. Se o objeto for da classe *Resultados*, esse possui um atributo referente ao caminho do arquivo *txt* gerado que corresponde ao relatório dos ensaios. Para que seja possível a inserção desse relatório no banco de dados, é necessário primeiramente converter os dados do arquivo de texto para um conjunto de dados do tipo *Byte*, já que não é permitido diretamente o armazenamento de um arquivo de

texto em um banco. Dessa forma, cria-se um *stream*, através da classe *FileStream*, a partir da leitura do arquivo de texto e em seguida um objeto da classe *BinaryReader*, o qual possui o método *ReadBytes* que retorna a instância desejada do tipo *Byte*.

O método *Update* tem como finalidade a atualização de dados de *drivers* e luminárias no banco de dados. Funciona da mesma forma que o método *Insert*, diferenciando-se pela string *vsql*, que corresponde ao comando MySQL correspondente para atualização de dados. A cláusula condicional utilizada para esse comando foi o atributo *codigo* da classe *Driver* ou *Luminaria*, de forma que o componente com o respectivo código seja atualizado.

O método *Delete* apaga registros do banco de dados. A string *vsql* correponde ao comando MySQL *Delete* para exclusão de dados. A cláusula condicional de busca utilizada é o código do *driver* ou luminária. É criado um objeto da classe *MySqlCommand* e utilizado o método *ExecuteNonQuery* dessa classe para a execução do comando MySQL.

O método *BuscaCodigos* é utilizado para preencher um objeto da classe *ComboBox*. Esse método retorna uma variável contendo os códigos de *drivers* ou luminárias encontrados no banco, ao receber como cláusula condicional uma *string* correspondente ao atributo *codigo* da classe *Driver* ou *Luminaria*. Esse método faz uma consulta ao banco buscando códigos que contenham no mínimo em alguma parte de sua cadeia de caracteres, a *string* passada como condição. Para exemplificar: se for passado como condição a string *EUC*, o método retornará todos os códigos que contenham pelo menos em alguma parte essa sequência de caracteres.

O método *BuscaDados* é utilizado para preencher objetos da classe *TextBox*. Esse método recebe como cláusula condicional, uma *string* correspondente ao atributo *codigo* da classe *Driver* ou *Luminaria* e retorna um vetor contendo todos os atributos da respectiva classe. Esse método, diferentemente do método *BuscaCodigos*, precisa receber uma *string* com o código exato correspondente a um único *driver* ou luminária contido no banco de dados, para que apenas os dados referentes a esse componente sejam retornados e preenchidos nos objetos da classe *TextBox*.

O método *ListaGridComponente* é utilizado para preencher uma tabela de dados. Esse método retorna um objeto da classe *DataTable* e recebe como parâmetros uma*string* correspondente ao componente que será buscado, *driver* ou luminária; uma *string* para ser a condição da busca, como no método *BuscaCodigos*; um valor do tipo inteiro correspondente a qual coluna da tabela deseja-se visualizar; um valor booleano

correspondendo a visualização ou não de todas as colunas da tabela. Nesse método um objeto da classe *MySqlCommand* é passado como parâmetro de uma instância da classe *MySqlDataAdapter*. Por fim, é utilizado o método *Fill* da classe *MySqlDataAdapter*, passando como parâmetro um objeto da classe *DataTable*, de forma a atualizar os dados da tabela com o resultado da busca no banco de dados.

O método *ListaGridResultados* é similar ao método *ListaGrid*. Porém, ao invés de acessar a tabela de *driver* ou luminária, esse acessa a tabela *Resultados*, referente aos resultados de ensaios já realizados. A cláusula condicional utilizada para essa consulta é o código do *driver* o qual deseja-se visualizar resultados.

O método *VisualizarImagem* tem o objetivo de recuperar uma imagem referente a um código específico de um *driver* ou luminária. Os parâmetros de entrada desse método são: uma *string* com o componente a ser processado (*driver* ou luminária), e outra com seu respectivo código. Além disso, o último parâmetro é um objeto do tipo *Image* contendo a imagem padrão, utilizada quando não há imagem disponível. É realizado o mesmo procedimento de execução de comando MySQL e criação de tabela do método *ListaGrid*. Em seguida, é verificado se a coluna correspondente a imagem do componente encontrado é nula. Caso não seja, é criado um *Stream* através da classe *MemoryStream* utilizando como parâmetro um objeto do tipo *byte*, correspondente ao que foi encontrado no banco de dados. Por fim, o *stream* é convertido através do método *FromStream* da classe *Image* para um objeto dessa mesma classe. É importante lembrar que esse procedimento de conversão de um objeto do tipo *byte* para um do tipo imagem (*Image*) é necessário, já que o banco de dados não é capaz de armazenar um dado do tipo imagem.

O método *AbrirTxt* tem o objetivo de gerar um arquivo com o formato *txt* referente aos ensaios realizados em um *driver* específico e iniciados em um determinado horário e dia, a partir do conjunto de dados do tipo *Byte* correspondente a esses ensaios. O procedimento de execução do comando MySQL e criação de tabela é o mesmo dos outros métodos já citados, utilizando como cláusula condicional a hora/data de ínicio dos ensaios, já que ela é única para cada batelada de ensaios. Encontra-se o conjunto de dados do tipo *Byte* na coluna da tabela correspondente aos resultados dos ensaios e somente a partir desse conjunto de dados, é utilizado o método *WriteAllBytes* da classe *File* para criar um arquivo do tipo *txt*. O arquivo é salvo com um nome padrão, correspondendo a data/hora do início da batelada de ensaios. Por fim, a classe *Process*, a partir do método *Start*, executa a abertura do arquivo de texto proporcionando ao usuário a visualização do relatório completo dos ensaios.

A classe *EnsaiosAutomatizados*, como já diz o nome, tem o papel de realizar os ensaios nos controladores de LED. Todos os procedimentos relacionados ao ensaios estão contidos nos métodos dessa classe. O diagrama de classes pode ser visto na figura 69.

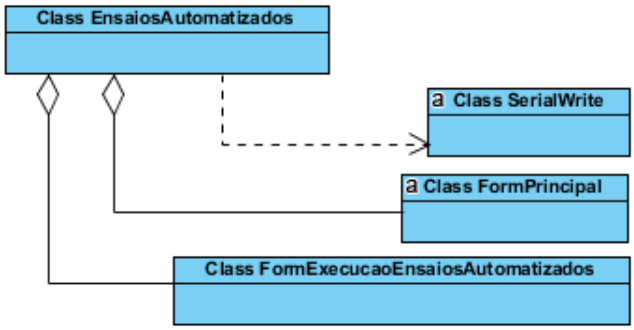


Figura 69: Diagrama de classes referente a classe *EnsaiosAutomatizados*

Os métodos e atributos da classe *EnsaiosAutomatizados* podem ser vistos na figura 70. Essa classe é invocada pela instância da classe *FormExecucaoEnsaiosAutomatizados*.

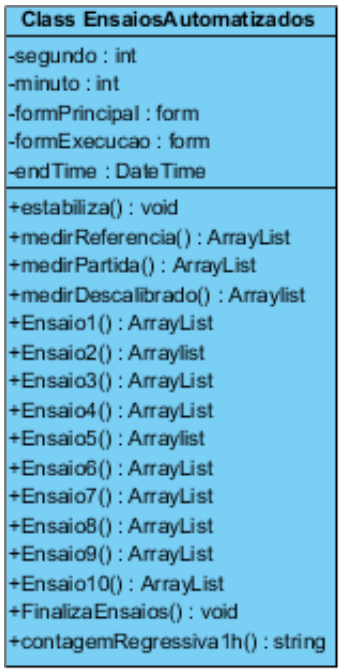


Figura 70: Atributos e métodos da classe *EnsaiosAutomatizados*

Os atributos *minuto* e *segundo* são utilizados apenas para armazenar o tempo restante de uma contagem regressiva realizada pelo método *contagemRegressiva1h*. Os atributos *formPrincipal* e *formExecucao* recebem os argumentos do construtor da classe

EnsaiosAutomatizados, de forma que são utilizados para que seja possível o acesso aos objetos do *Form* principal do *software* e do *Form* de execução dos ensaios. O atributo *endtime* da classe *DateTime* é utilizado para armazenar o horário de término dos ensaios.

O método *estabiliza* tem o objetivo de proporcionar tempo suficiente para que a corrente de entrada do *driver* se estabilize, visto que existe uma corrente de pico durante a partida. Para que isso seja possível, é feito um laço no qual cada iteração é interrompida, através do método *Sleep* da classe *Thread*, durante o intervalo pré-definido de 300ms, num total de 30 segundos ao fim de 100 iterações. A cada iteração a barra de progresso é atualizada, sendo acrescida em 1% seu valor.

O método *medirPartida* é utilizado apenas quando o ensaio de máxima corrente/tensão na partida é selecionado. Esse método coleta a cada 15ms uma amostra, totalizando durante os dois primeiros segundos da partida, 134 amostras.

O método *FinalizaEnsaios* retorna o atributo *endTime* com o horário correspondente ao término dos ensaios.

O método *ContagemRegressiva1h* implementa uma função responsável por contar regressivamente o tempo durante 1 hora e é utilizado dentro dos métodos *Ensaio8* e *Ensaio9*, referentes aos ensaios nos quais o driver tem sua saída, durante 1 hora, em curto-circuito ou em circuito aberto, respectivamente. A contagem regressiva é importante para informar ao usuário o tempo restante, através de uma legenda abaixo da barra de progresso.

Os métodos *Ensaio1* até *Ensaio10*, referem-se aos procedimentos para a realização dos ensaios. Em todos esses métodos , objetos da classe *ArrayList* foram utilizados para retornar os resultados dos ensaios ou para atualizar o *status* das barras de progresso. O *ArrayList* de resultados retornado por cada um desses métodos, conterá sempre o valor do parâmetro elétrico especificado pelo fabricante, a média das amostras coletadas, assim como um *array* contendo todas as amostras. Além disso, conterá uma string referente ao resultado que pode ser: *Aprovado*, *Reprovado*, *Cancelado* ou *Não Realizado*.

As descrição dos procedimentos para realização dos ensaios será feita nos próximos parágrafos.

Ensaio 1: Corresponde ao ensaio de máxima variação da tensão ou corrente de saída durante a partida. Nesse método é realizado apenas a média das amostras coletadas no método *MedirPartida* e essa é comparada ao valor especificado pelo fabricante. Caso a

média não ultrapasse 10% do valor especificado pelo fabricante, o ensaio é considerado aprovado, caso contrário será considerado reprovado.

Os ensaios 2, 3, 4, 5 e 6 são realizados em regime de operação e com tensão nominal na entrada. Em todos eles cada amostra do parâmetro elétrico em avaliação é coletado a cada 400 milisegundos, num total de 20 amostras e 8 segundos.

Ensaio 2: Corresponde ao ensaio de avaliação do fator de potência. É realizada a média das amostras coletadas e se essa for maior que 0,92 e não menor que 0,05 do valor indicado pelo fabricante, o ensaio é considerado aprovado, senão será considerado reprovado.

Ensaios 3 e 4: Correspondem aos ensaios de máxima variação da corrente de entrada e máxima variação da potência de entrada. Em cada ensaio, é realizada a média das amostras coletadas e se essa não apresentar variação maior que 10% em comparação com o valor especificado pelo fabricante, o ensaio é considerado aprovado, caso contrário será considerado reprovado.

Ensaios 5 e 6: Corresponde ao ensaios de máxima variação da tensão e máxima variação de corrente, ambos medidos na saída do *driver* em regime de operação. O ensaio de máxima variação da tensão de saída só é realizado para *driver* com controle de tensão. Da mesma forma o ensaio de máxima variação da corrente de saída só é realizado para *driver* com controle de corrente de saída. Se a média das amostras não apresentar variação maior que 10% em comparação com o valor especificado pelo fabricante, o ensaio é considerado aprovado, caso contrário será considerado reprovado.

Ensaio 7: Corresponde ao ensaio de máxima variação da tensão ou corrente de saída, quando o *driver* é submetido a tensões não nominais. Nesse ensaio a tensão de saída é analisada apenas para *driver* controlado por tensão. Já a corrente de saída é analisada apenas para *driver* com controle de corrente. Ele é dividido em duas partes, na primeira a tensão de entrada do *driver* é configurada pelo MCE para ser 106% da tensão nominal e na segunda é 92% da tensão nominal. Antes de iniciar os ensaios, a chave geral do equipamento é aberta para que a tensão de entrada seja modificada pelo MCE, através da comutação dos relés que conectam o transformador com relação de transformação 1:1,06. Após isso, a chave geral do equipamento é fechada e aguarda-se 30 segundos pela estabilização da corrente. Por fim, inicia-se a coleta das vinte amostras, como nos ensaios anteriores. Se a média das amostras não apresentar diferença maior que 10% em comparação com o valor especificado pelo fabricante, a primeira parte do ensaio é considerada aprovada, senão será considerada reprovada.

Em seguida, inicia-se a preparação para a segunda parte do ensaio. Novamente abre-se a chave geral do equipamento e a tensão de entrada do *driver* é alterada para 92% da tensão nominal, através da comutação dos relés que conectam o transformador com relação de transformação 1:0,92. Novamente são coletadas vinte amostras e a comparação com o valor especificado pelo fabricante é feita analogamente a primeira parte. O ensaio de máxima variação da tensão ou corrente na saída só é considerado aprovado se ambas as partes do ensaio forem aprovadas. Caso contrário o ensaio é considerado reprovado.

Ensaio 8: Corresponde ao ensaio de curto-circuito. Para esse ensaio a tensão na entrada do *driver* utilizada é a nominal. Inicialmente, a chave geral do equipamento é aberta e após um segundo o relé 8 comuta de estado, curto circuitando a saída do *driver*. Além disso, os relés anteriores ao canal 1 permitirão a alimentação do controlador de LED com tensão nominal. Após configurada as condições de pré-ensaio do MCE, a chave geral do equipamento é fechada e aguarda-se 30 segundos para estabilização da corrente. Vinte amostras da corrente de saída (caso controle de corrente) ou tensão de saída (caso controle de tensão) são coletadas. É feita a verificação da média desses parâmetros e se não apresentarem diferença maior que 10% em comparação com o valor especificado pelo fabricante o ensaio é considerado aprovado, senão é considerado reprovado.

Ensaio 9: Corresponde ao ensaio de circuito aberto. Todo o procedimento realizado no ensaio de curto circuito é repetido também para esse ensaio, com a diferença que a saída do *driver* será colocada em circuito aberto ao invés de em curto circuito. A verificação e análise das amostras é feita também da mesma forma que o ensaio de curto circuito.

Ensaio 10: Corresponde ao ensaio de comutação. Como já explicado no capítulo 2, o *driver* deve ser alimentado com tensão nominal por 30 segundos e desenergizado por 30 segundos, sendo que esse ciclo é repetido 200 vezes sem o módulo de LEDs conectado e 800 vezes com o módulo de LEDs conectado. Durante o total de mil ciclos, uma amostra de tensão e corrente é coletada por segundo para registrar no relatório o comportamento desses parâmetros durante todo o ensaio e principalmente verificar se o módulo de LEDs não está danificado. A verificação de corrente é principalmente importante porque caso o módulo de LED seja danificado, o *software* vai interromper o ensaio imediatamente e solicitar a troca do módulo de LEDs. Após o fim dos ciclos, a amostragem de corrente e tensão é feita a cada 500ms, durante 15 minutos. Se a média desses parâmetros não apresentarem variação maior que 10% em comparação com o

valor especificado pelo fabricante o ensaio é considerado aprovado. Caso contrário é considerado reprovado.

A declaração da classe *Resultados* pode ser vista na figura 71.

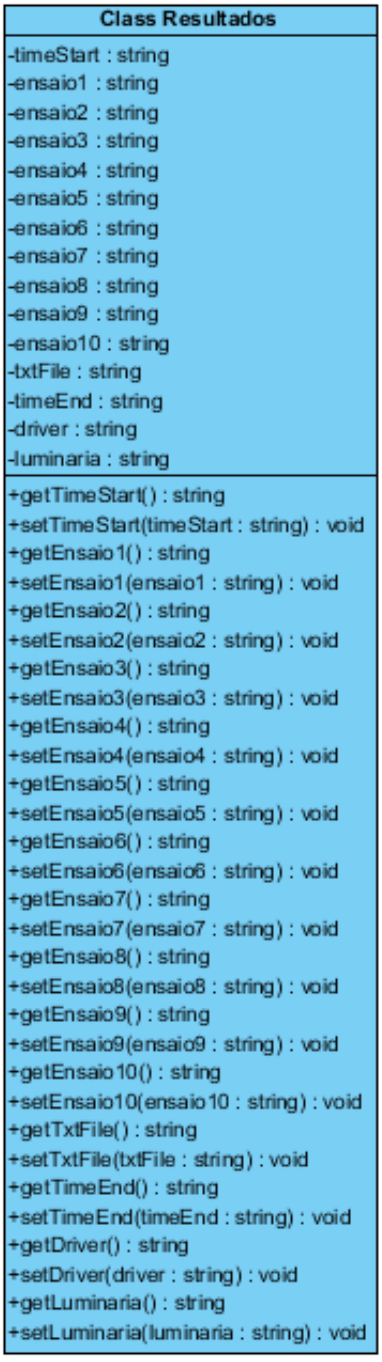


Figura 71: Atributos e métodos referentes a classe *Resultados*

A classe *Resultados* tem em todos seus atributos, propriedades do tipo *get* e *set*, ou seja, apenas leem ou setam valores.

O atributos *timeStart* e *timeEnd* armazenam a data/hora de início e término,

respectivamente, dos ensaios.

Os atributos referente aos ensaios - *ensaio1* até *ensaio 10* - armazenam apenas o resultados desses ensaios, ou seja, as seguintes cadeias de caracteres: *Aprovado*, *Reprovado*, *Cancelado* ou *Não Realizado*. O atributo *txtFile* armazena o caminho do arquivo *txt* contendo o relatório completo dos ensaios.

5.8 TELA INICIAL

A tela iniciado *software* computacional foi mostrada na figura 27. O relacionamento entre as classes utilizadas na tela inicial pode ser visto na figura 72.

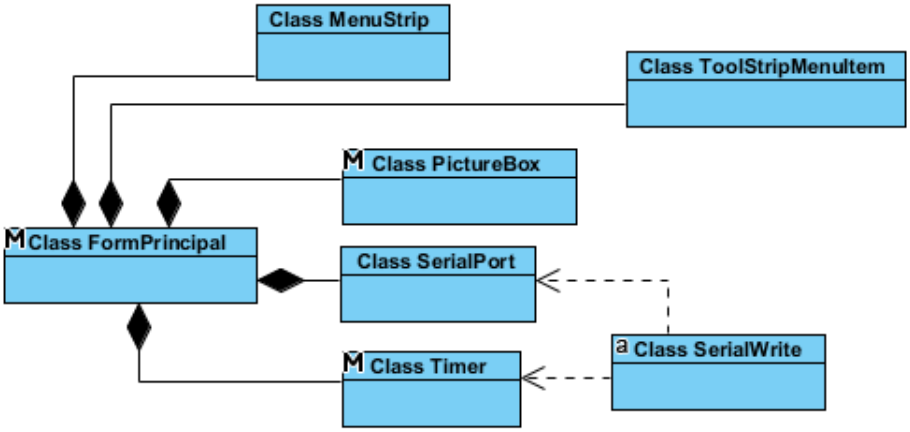


Figura 72: Diagrama de classes referente a *Tela inicial*

A classe *PictureBox* insere a imagem de fundo da tela principal, na qual pode-se ver o logotipo da UFJF e do NIMO. A classe *MenuStrip* fornece um sistema de menu para a janela. A classe *ToolStripMenuItem* cria itens para o sistema de menu, que representam cada módulo do *software* computacional. Cada item está relacionado a uma classe do tipo *Form*, já que quando o usuário clica em um item é criado um objeto dessa classe. A classe *Serial Port* estabelece conexão com a porta serial. A classe *Timer* dispara a execução de métodos da classe *SerialWrite*. A classe *SerialWrite* escreve na porta serial, como ja foi visto em detalhes na seção 5.7.

5.8.1 TELA DE CONFIGURAÇÃO

O relacionamento entre as classes utilizadas na tela de configuração pode ser visto na figura 73 e essa tela foi mostrada na figura 28.

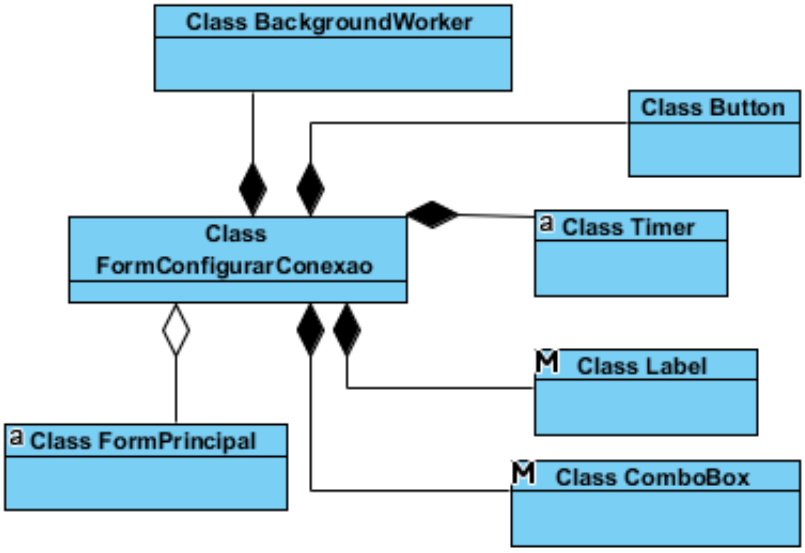


Figura 73: Diagrama de classes referente a tela de configuração

A classe *ComboBox* fornece campos para seleção da porta serial e também para seleção da velocidade de comunicação com o equipamento, o *Baud Rate*.

A classe *Button* verifica se o campo do objeto da classe *ComboBox*, referente a seleção da porta serial, não é nulo. Caso não seja, realiza a execução de um procedimento em uma *Thread* separada através da classe *BackgroundWorker*. Esse procedimento configura os relés do módulo na configuração padrão e através da classe *SerialPort* realiza a abertura da porta serial ou seu fechamento.

A classe *Timer* dispara uma função a cada 1s que busca e lista as portas seriais em uso no computador. Dessa forma, o equipamento pode ser conectado à porta USB do computador a qualquer momento que o *software* vai identificar a utilização da porta serial e mostra-la na *ComboBox*.

A classe *Label* mostra uma legenda de acordo com o estado da conexão atual, se conectado ou desconectado, além de mostrar o estado esperado do módulo de LEDs, se ligado ou desligado, e a tensão de entrada no *driver* .

5.8.2 TELA DO MÓDULO DE CADASTRO

O módulo *Banco de cadastro* está subdivido em três abas: *Cadastrar*, *Editar* e *Buscar*.

A aba de cadastro, mostrada na figura 29 tem simplesmente a finalidade de cadastrar *drivers* e luminárias. O diagrama de classes para essa tela pode ser visto na figura

74.

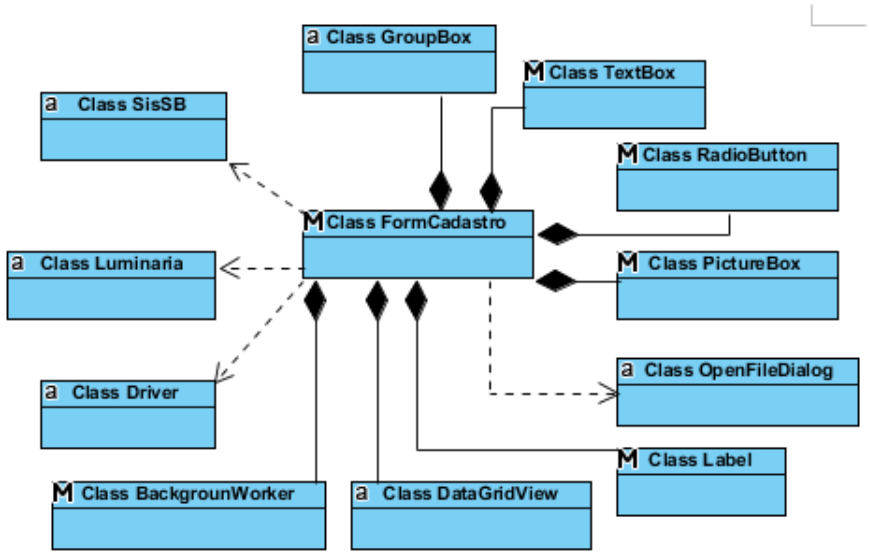


Figura 74: Diagrama de classes referente a tela de cadastro

A classe *TextBox* fornece as caixas de texto para inserção dos dados referentes ao *driver* ou luminária que será cadastrado.

A classe *RadioButton* instanciada permite a seleção do componente a ser cadastrado e também o tipo de controle do *driver*.

A classe *Label* é utilizada para criação das legendas dos botões e caixas de texto.

A classe *PictureBox* armazena as imagens dos botões e simula a funcionalidade de um botão. Ao serem clicados iniciam um processo em uma *Thread* separada a partir do evento *DoWork* da classe *BackgroundWorker*. O tipo de operação a ser executado depende do *botão* que será clicado e ao que está relacionado. Além disso, armazena a imagem de *loading* do tipo *gif*, que será visível apenas nos momentos em que uma operação estiver sendo realizada através do método *Visible*.

Ao clicar no botão *Salvar*, a operação de *BackgroundWorker* para esse botão é iniciada. As classes *SisSB*, *Driver* ou *Luminaria* são instanciadas. Se o componente a ser cadastrado é um *driver*, os dados referentes aos campos de textos da classe *TextBox* serão atribuídos aos atributos do objeto da classe *Driver*. Caso seja uma luminária, serão atribuidos ao objeto da classe *Luminaria*. Em seguida, os dados do componente são inseridos no banco de dados através do objeto da classe *SisDB* pelo método *Insert*. A classe *MessageBox* é utilizada ao fim da operação de cadastro no banco de dados para informar sobre o sucesso ou não da operação.

Ao clicar no botão *Selecionar*, a operação do *BackgroundWorker* para esse botão é iniciada. A classe *OpenFileDialog* é instanciada, e através do método *ShowDialog* realiza a abertura de uma caixa de diálogo possibilitando ao usuário a seleção da imagem do componente. O caminho da imagem selecionada será passado como parâmetro para o objeto da classe *PictureBox*, responsável por mostrar a imagem do componente.

A tela de edição, vista na figura 30, tem a finalidade de editar dados de *drivers* e luminárias já cadastrados ou mesmo excluir o componente do banco de dados.

A classe *Label* é utilizada para criação das legendas dos botões e caixas de texto.

A classe *RadioButton* instanciada permite a seleção do componente a ser cadastrado e também o tipo de controle do *driver*.

A classe *TextBox* fornece as caixas de texto com os dados do *driver* ou luminária selecionado pelo objeto da classe *ComboBox* e permite ao usuário editar os valores. O único dado que não pode ser editado é o código do componente, o qual tem sua caixa de texto bloqueada. Caso o código cadastrado esteja errado, o usuário deve excluir o componente e cadastrar novamente com o código correto.

Da mesma forma que na tela de cadastro, a classe *PictureBox* simula os botões.

O botão de apagar imagem carrega a imagem padrão(*Sem Imagem*) que pode ser acessada pela classe *Resources*.

O processo de salvar a edição funciona de forma similar ao de salvar da tela de cadastro, porém é aplicado o método *Update* da classe *SisDB*. É utilizado, como cláusula condicional de consulta ao banco de dados, a *string* encontrada na caixa de texto, referente ao código do componente. O procedimento ainda verifica se a imagem armazenada pelo objeto da classe *PictureBox* foi alterada para uma nova imagem do componente, e em caso afirmativo, irá converter a imagem para um dado do tipo *byte* e atribui-lo ao atributo *Imagem* da classe *Luminaria* ou *Driver*. Em caso negativo, se a imagem armazenada corresponder a imagem padrão, um valor nulo será atribuído ao atributo *Imagem*.

O evento de *BackgroundWorker* relacionado ao botão de excluir componente, efetua a exclusão no banco de dados do componente através de seu respectivo código.

A implementação do botão *Excluir* é feita passando o código selecionado no objeto de *ComboBox* como parâmetro pelo evento de *BackgroundWorker* e em seguida é inserido ao atributo *Codigo* da classe *Luminaria* ou *Driver*. Esse atributo é utilizado como parâmetro no método *Delete* da classe *SisDB*. Por fim, uma caixa de mensagem

é aberta indicando o sucesso ou não da operação, através da classe *MessageBox*.

A classe *Button* também é utilizada para criar o botão *Pesquisar* que irá buscar os códigos contendo a *string* inserida no objeto de campo de texto a esquerda do botão.

Ao ser clicado vai chamar um objeto de *BackgroundWorker*. No processo iniciado, a string inserida no campo de texto será atribuido ao atributo *Codigo* da classe *Driver* ou *Luminaria*. Através do método *BuscaCodigos* da classe *SisDB*, o objeto de *ComboBox* será atualizado com os códigos encontrados.

Os dados dos componentes mostrados nos campos de texto são atualizados sempre que houver uma mudança no índice do objeto de *ComboBox*, ou seja, quando ocorrer o evento *SelectedIndexChanged*. Esse evento irá chamar um objeto da classe *BackgroundWorker* que irá atualizar os campos de texto através do método *BuscaDados* da classe *SisDB*.

O módulo de cadastro também contempla uma tela de busca, cuja principal função é propiciar uma ferramenta para que o usuário possa buscar informações sobre luminárias e *drivers* apenas com algum parâmetro (e.g. fator de potência). Esta característica é útil para situações onde o código do equipamento é desconhecido a priori.

A classe *DataGridView* foi utilizada para criar a tabela onde serão disponibilizados os dados encontrados referentes a busca.

A classe *RadioButton* permite escolher o componente a ser buscado, assim como, o tipo de filtro desejado para a consulta no banco de dados. Para isso, é atribuído um valor do tipo inteiro, de um a nove para *driver* e um a quatro para luminária, cada um correspondendo a um tipo de filtro diferente. Os tipos de filtro existentes foram vistos no capítulo 4.

A classe *CheckBox* possibilita a visualização de todas as colunas da tabela de *Driver* ou *Luminária*, ou seja, todos os dados referentes aos componentes encontrados, através do objeto com nome *Mostrar todas as colunas*.

A classe *TextBox* fornece o campo de texto onde será inserido uma *string*, coerente ao tipo de filtro escolhido.

As três últimas classes citadas, *RadioButton*, *CheckBox* e *TextBox* estão relacionadas a um objeto da classe *BackgroundWorker*, que irá atualizar o objeto de *DataGridView*. O procedimento será iniciado pela classe *TextBox* através do evento *KeyUp*, que acontece após o usuário pressionar alguma tecla. Dessa forma, dispensa-se a necessidade de um botão para chamar o procedimento. Os objetos de *RadioButton* e

CheckBox também acionam esse procedimento ao ocorrer o evento chamado *Checked-Changed*, que acontece quando o estado de algum desses objetos é alterado, ou seja, quando esse é marcado ou desmarcado. No procedimento referido são instanciados objetos da classe *SisDB* e *DataGridView*. São passados como parâmetros do método *ListaGridComponente*: um valor inteiro correspondendo ao tipo de filtro selecionado, a *string* inserida no campo de texto, um valor booleano indicando se o *CheckBox* referente a visualização de todas as colunas está marcado e, por fim, o tipo componente que se deseja buscar.

A classe *PictureBox* simula o botão *Copiar Selecionado*. Esse botão é utilizado para visualizar na tela de edição, o componente respectivo a um código selecionado na tabela. Dessa forma, é possível visualizar seus dados e possivelmente fazer uma edição se o usuário desejar. O código selecionado na tabela é atribuído ao campo de texto de pesquisa da tela de edição pelo método *Text*, da classe *TextBox*. Em seguida, é chamado uma função, contendo um objeto de *Backgroundworker*, para realizar a atualização de todos os campos de texto relacionados aos dados do componente. Uma funcionalidade de cópia do código também é disponibilizada permitindo o usuário colar em qualquer ambiente o código selecionado. Para isso, foi utilizado o método *SetText* da classe *Clipboard*.

5.8.3 TELA DE TESTES DE FUNCIONAMENTO

A tela *Testes de Funcionamento* possibilita o usuário configurar o estado dos relés de acordo com o desejado e também disponibiliza a leitura do MDGE. Essa tela foi vista na figura 32 e seu diagrama de classes é mostrado na figura 75.

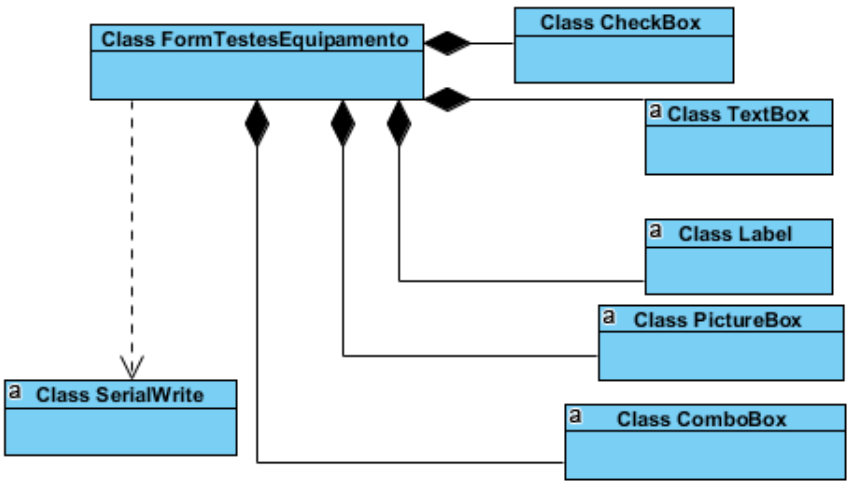


Figura 75: Diagrama de classes referente a tela *Testes de funcionamento*

As classes da tela *Testes de Funcionamento* tem os objetivos descritos nos próximos parágrafos.

A classe *CheckBox* fornece as caixas de seleção, as quais o usuário irá marcar para configurar o estado dos relés.

A classe *ComboBox* é utilizada para disponibilizar uma lista com o número de *jumpers* nos sensores de corrente e tensão dos canais do MDGE para que o usuário possa selecionar o número correspondente ao configurado manualmente nos sensores. Dessa forma, o *software* computacional irá configurar, no programa embarcado do microcontrolador, o número correto de *jumpers* que estão sendo utilizados.

A classe *TextBox* disponibiliza os objetos de caixa de texto, os quais recebem através da propriedade *Text* os valores correspondentes a leitura do MDGE.

A classe *Label* fornece a legenda para os objetos das classes *ComboBox*, *TextBox* e *CheckBox*.

A classe *PictureBox* fornece uma imagem que simula a funcionalidade de um botão. Através do evento *Click* expande ou reduz a janela para possibilitar a visualização das leituras disponibilizadas pelo MDGE.

A classe *SerialWrite* é invocada quando a caixa de seleção referente a *Chave Geral* é marcada ou desmarcada. Quando a caixa é marcada os métodos *bwSetTensao* e *bwLigaConjunto* dessa classe são chamados, configurando a tensão de entrada no *driver* e em seguida a saída do *driver* e energizar o equipamento através da chave geral. Quando é desmarcado, o método *bwDesligaConjunto* é acionado para desenergizar o equipamento.

5.8.4 TELA DE SELEÇÃO DOS COMPONENTES

A tela de seleção dos componentes, como visto na figura 33, possibilita a seleção dos componentes que serão utilizados nos ensaios automatizados. O diagrama de classes para essa tela pode ser visto na figura 76.

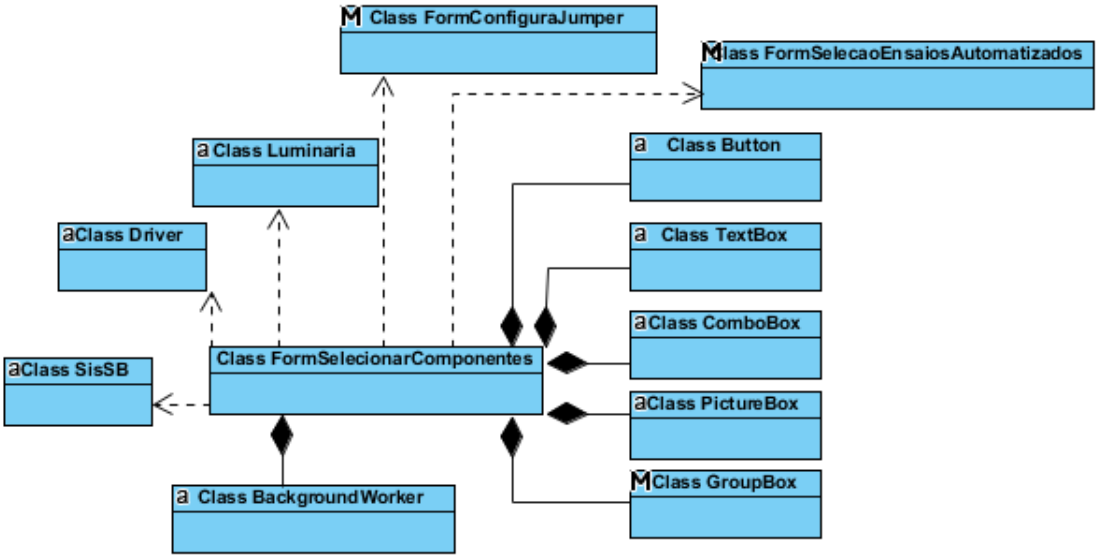


Figura 76: Diagrama de classes referente a tela de seleção de componentes

A classe *PictureBox* armazena apenas a imagem de *Loading*, já a classe *TextBox*, cria as caixas de texto para inserção dos códigos dos componentes que serão utilizados nos ensaios.

A classe *ComboBox* armazena os códigos dos componentes encontrados pela pesquisa e a classe *Button* é utilizada para a criação dos botões *Pesquisar* e *Próximo*.

O botão *Pesquisar* dispara a rotina de um objeto *BackgroundWorker*. A rotina busca no banco de dados o componente com o código inserido na caixa de texto e preenche com os resultados o *ComboBox* referente ao *driver* ou luminária. Para isso, é instanciado um objeto da classe *SisDB*, que através do método *BuscaCodigos* e utilizando como parâmetro os objetos *Luminaria* e *Driver*, retorna os códigos encontrados pela pesquisa. Essa rotina pode ser acionada também por um evento de *KeyUp* da classe *TextBox*, que verificará após usuário pressionar uma tecla, se essa corresponde a tecla *Enter*. Em caso afirmativo, a busca pode ser realizada apenas pressionando essa tecla quando o cursor estiver posicionado na caixa de texto.

O botão *Próximo* dispara uma outra rotina de um objeto *BackgroundWorker*. Essa rotina encontra os dados dos componentes já selecionados. Para sua implementação, instancia-se a classe *SisDB* que através do método *BuscaDados* e utilizando como parâmetro os objetos *Luminaria* e *Driver*, retorna os dados referentes a ambos componentes. A partir desses dados, é chamado uma função para calcular a corrente e potência de entrada do *driver*, já que esses dados dependem diretamente da luminária a qual está sendo acionada pelo *driver*. Após o cálculo, uma outra função é chamada para calcu-

lar o número de *jumpers* que deverão ser inseridos nos sensores do MDGE antes do início dos ensaios. Tendo como parâmetro todos os dados referentes aos componentes, utiliza-se as expressões 5.3 e 5.4 para definir o números de *jumpers* para o canal 1.

$$N_{jumpers} = ceil(\sqrt{2} \cdot \frac{V_{RMS}}{50}) \tag{5.3}$$

$$N_{jumpers} = ceil(\sqrt{2} \cdot \frac{I_{RMS}}{0{,}25}) \tag{5.4}$$

O cálculo para o canal 2 é feito de forma similar, porém, como esse canal está alocado na saída do controlador de LED a corrente circulante é contínua, logo as grandezas utilizadas para o cálculo são os valores médios de acordo com as equações 5.5 e 5.6.

$$N_{jumpers} = ceil(\frac{V_{Med}}{50}) \tag{5.5}$$

$$N_{jumpers} = ceil(\frac{I_{Med}}{0{,}25}) \tag{5.6}$$

Os números de *jumpers* calculados para os dois sensores de cada canal são truncados para o número inteiro mais próximo e maior. Por fim, a rotina chama a abertura da janela para configuração do MDGE, mais especificamente, a configuração manual do número de *jumpers* pelo operador do equipamento.

A classe *MessageBox* é utilizada para abrir uma janela de aviso no caso em que o *driver* selecionado não é compatível com a luminária. Além disso, também é utilizada para os casos em que o usuário deixou de selecionar um *driver* ou luminária.

5.8.5 TELA DE CONFIGURAÇÃO DO MDGE

O diagrama de classes que representa a tela de configuração do MDGE pode ser visto na figura 77. A imagem da tela foi mostrada na figura 34.

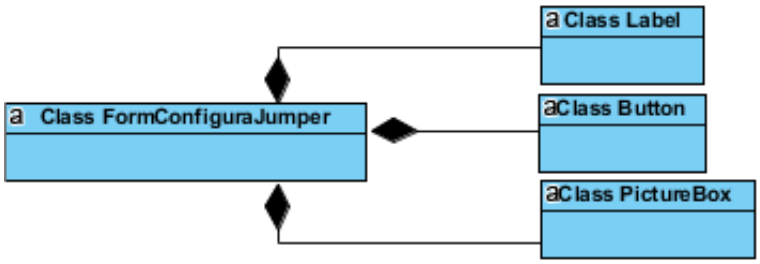


Figura 77: Diagrama de classes referente a tela de configuração do *MDGE*

A primeira janela representa a configuração de *jumpers* referente ao canal 1 do MDGE.

A classe *Label* representa as legendas das imagens e também o texto responsável por instruir o usuário a adicionar o número de *jumpers* necessários para o início dos ensaios. Esse número, calculado na janela anterior da classe *FormSelecaoEnsaios*, é recebido como parâmetro na inicialização da classe *FormConfiguraJumper*.

A classe *PictureBox* armazena as imagens da parte traseira dos sensores onde serão inseridos os *jumpers*.

A classe *Button* cria o botão que irá instruir a configuração do canal 2 do MDGE. Ao clicar no botão *Próximo*, apenas os objetos do tipo *Label* são afetados, de forma que a legenda será alterada de canal 1 para canal 2, assim como, o número de *jumpers* é modificado para os valores calculados do canal 2.

Após o fechamento da janela referente a configuração do número de *jumpers*, o *Form* da classe *FormSelecaoEnsaiosAutomatizados* será aberto, permitindo a seleção dos ensaios que serão realizados.

5.8.6 TELA DE SELEÇÃO DOS ENSAIOS AUTOMATIZADOS

A tela para seleção dos ensaios automatizados tem o diagrama de classes mostrado na figura 78, já a imagem dessa tela foi vista na figura 35.

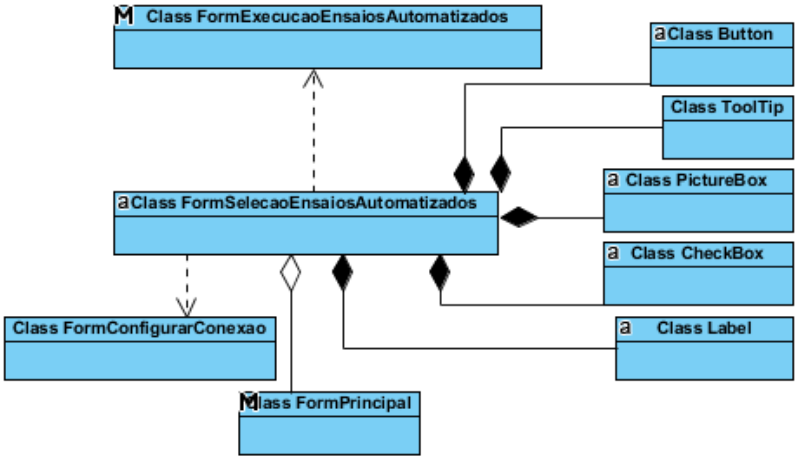


Figura 78: Diagrama de classes referente a tela de seleção dos ensaios

A classe *ToolTip* é utilizada para auxiliar o usuário que necessite de informações acerca dos ensaios. Ao posicionar o mouse sobre algum objeto da classe *PictureBox*, com o símbolo de informação, uma caixa com texto contendo a descrição dos ensaios é aberta. Outra alternativa para visualizar informações acerca dos ensaios é clicando no objeto da classe *ToolStripMenuItem*, que representa um item do sistema de menu. O único item existente tem o nome *Abrir Documentação dos Ensaios*, que ao ser clicado vai criar um *Form* da classe *Documentacao*, o qual permitirá a visulização da descrição de todos os ensaios dentro de um *TextBox*. Já a classe *CheckBox* cria objetos responsáveis por permitir a seleção dos ensaios automatizados.

Após a seleção dos ensaios, deve-se clicar no botão *Iniciar* da classe *Button* para começar os ensaios. Primeiramente é feita uma verificação se a porta serial está aberta através do método *IsOpen* da classe *SerialPort*. Caso seja retornado o valor booleano falso, significa que o usuário ainda não realizou a conexão do equipamento. Nesse caso, é instanciada a classe *FormConfigurarConexao* a fim de permitir que seja feita a abertura da porta serial. Caso o método *IsOpen* retorne o valor booleano verdadeiro, significa que a porta serial já se encontra aberta e o equipamento está conectado. Dessa forma, é instanciada a classe *FormExecucaoEnsaiosAutomatizados* que realizará os ensaios. São passados como parâmetro para o construtor da classe, os ensaios selecionados de acordo com os objetos do tipo *CheckBox* marcados.

5.8.7 TELA DE EXECUÇÃO DOS ENSAIOS

A tela de execução dos ensaios, referente ao objeto da classe *FormExecucaoEnsaiosAutomatizados*, tem o diagrama de classes da figura 79. Essa tela foi mostrada na

figura 37.

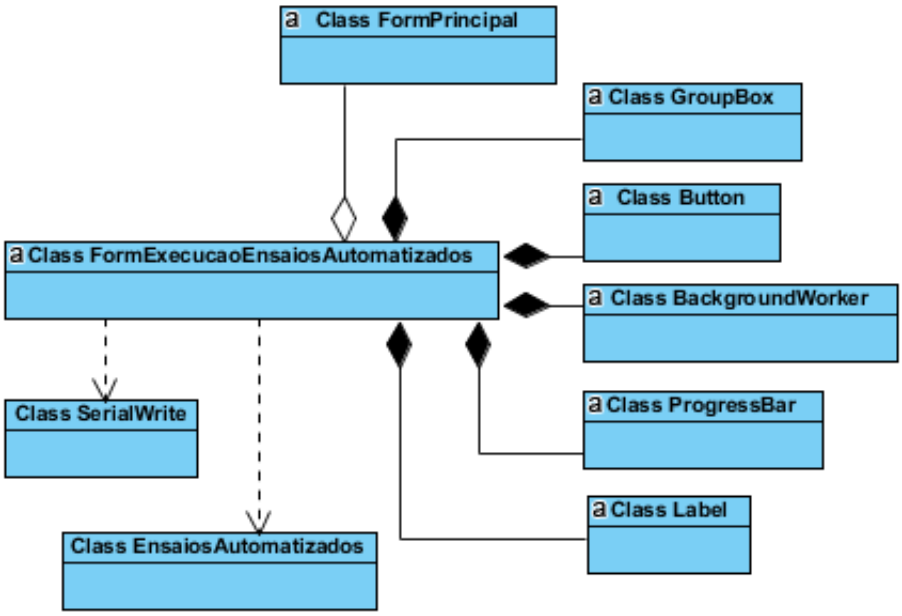


Figura 79: Diagrama de classes referente a tela de execução dos ensaios

Dentro da rotina de inicialização do objeto da classe *FormExecucaoEnsaiosAutomatizados*, é chamado um objeto de *BackgroundWorker* contendo as ações necessárias para a realização dos ensaios. Nessa rotina de plano de fundo são inicialmente instanciados objetos das classes *SerialWrite* e *EnsaiosAutomatizados*. Antes de fechar a chave geral do equipamento, são invocados os métodos da classe *SerialWrite* *bwsetJumper* e *bwSetTensao*. O primeiro vai configurar o número de *jumpers* de acordo com o que já foi calculado anteriormente e o segundo uma tensão nominal na entrada do *driver* de acordo com o que foi cadastrado para o respectivo *driver* no banco de dados, ou seja, 127V ou 220V. Em seguida, é chamado o método *bwLigaConjunto* a fim de energizar o módulo de LEDs e instantaneamente o método *medirPartida*, caso o primeiro ensaio tenha sido selecionado.

Antes de iniciar os ensaios em regime de operação é invocado o método *estabiliza* da classe *EnsaiosAutomatizados*, de forma que seja aguardada a estabilização da corrente de entrada do *driver*. Os ensaios em regime de operação selecionados, numerados de 2 a 10, serão realizados em sequência, através dos métodos respectivos a cada um. Os resultados de cada ensaio são retornados pelos mesmos métodos e ao fim de todos os ensaios, os resultados serão enviados como parâmetro para o método *FinalizaEnsaios*.

A classe *Button* fornece o botão para cancelar os ensaios. Esse botão ao ser cli-

cado irá interromper o ensaio atual e não iniciará mais nenhum programado. A verificação de solicitação de cancelamento é feita constantemente dentro da rotina de *BackgroundWorker*, através do atributo *CancellationPending*, que indica através de um valor booleano se um cancelamento foi requisitado.

A classe *ProgressBar* fornece as barras de progresso que indicam o andamento dos ensaios de uma forma gráfica. A barras são modificadas durante cada ensaio através do evento *ReportProgress* da classe *BackgroundWorker*.

A classe *Label* indica na parte superior da janela qual o *driver* e qual luminária estão sendo ensaiados. Além disso, fornece legendas para as barras de progresso, sendo possível verificar o andamento dos ensaios em forma de porcentagem, ou mesmo tempo, como em parte dos ensaios de curto-circuito e circuito aberto.

O fim da rotina de *BackgroundWorker* acontece quando é criado um objeto da classe *FormResultados* que disponibiliza para visualização os resultados dos ensaios.

5.8.8 TELA DE RESULTADOS

A tela de resultados, referente ao objeto da classe *FormResultados*, tem o diagrama de classes representado pela figura 80. Essa tela foi vista na figura 38.

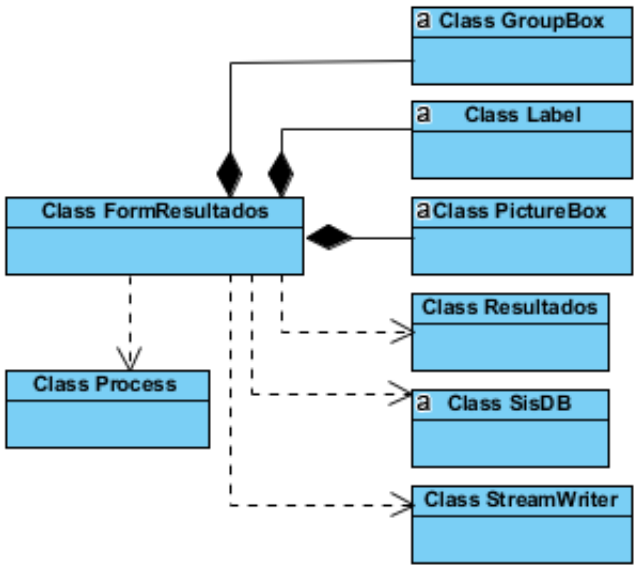


Figura 80: Diagrama de classes referente a tela de resultados

Os resultados dos ensaios são passados como parâmetro construtor da classe *FormResultados*. A tela de resultados possibilita verificar qual foi o resultado de cada ensaio, dentre os quatro tipos possíveis: *Aprovado*, *Reprovado*, *Cancelado* ou *Não Realizado*.

A classe *PictureBox* armazena as imagens com o texto: *Aprovado*, *Reprovado*, *Cancelado* ou *Não Realizado*. A imagem a ser mostrada é definida através dos atributos da classe *Resultados*, que contêm uma *string* para o resultado de cada ensaio. A classe *SisDB* também é instanciada com o objetivo de armazenar os resultados dos ensaios em *tabela_resultados* do banco de dados e para tanto foi utilizado o método *Insert*.

É gerado também um relatório completo acerca dos ensaios em formato *txt* utilizando o método *WriteLine* da classe *StreamWriter* para escrever em um arquivo externo de texto.

O botão *Abrir Relatório Txt* é também da classe *PictureBox*. Esse botão ao ser clicado, irá abrir o relatório gerado através do método *Start* da classe *Process*. É utilizado o caminho do arquivo como parâmetro do método.

5.8.9 TELA DO BANCO DE RESULTADOS

A tela *Banco de Resultados* permite ao usuário buscar resultados de ensaios já realizados anteriormente. O seu diagrama de classes pode ser visto na figura 81 e a tela foi mostrada na figura 40.

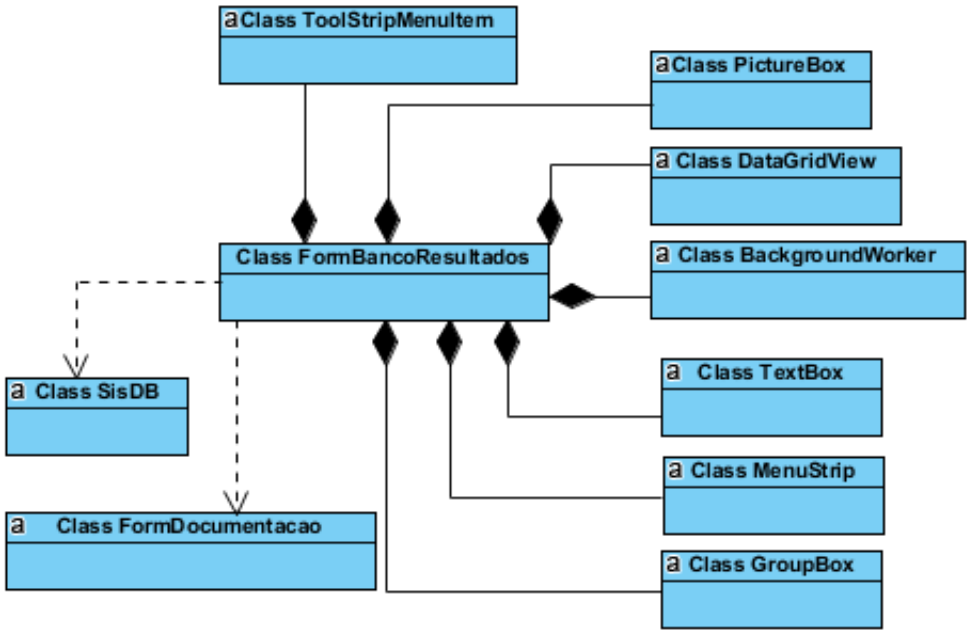


Figura 81: Diagrama de classes referente a tela do banco de Resultados

A classe *MenuStrip* e as outras relacionadas a ela foram implementadas da mesma forma que foi visto na tela de seleção dos ensaios automatizados e com a mesma finalidade, oferecer ajuda permitindo a visualização da documentação e descrição dos

ensaios.

A classe *TextBox* fornece o campo de texto onde é inserido o código do *driver*. Quando o campo de texto está selecionado e uma tecla é pressionada, ocorre um evento de *KeyUp* disparando a execução de um *BackgroundWorker*. Dentro dessa rotina é instanciada a classe *SisDB* e em seguida é utilizado o método *ListaGridResultados*. Esse método retorna uma tabela com os dados encontrados referentes ao código inserido no campo de texto e carrega essa tabela no atributo *DataSource* de um objeto da classe *DataGridView*.

A classe *PictureBox* proporciona a criação do botão *Abrir relatório Txt*, o qual tem a finalidade de abrir o relatório completo de uma batelada de ensaios selecionada. A implementação foi feita a partir da instanciação da classe *SisDB* e aplicação do método *AbrirTxt* utilizando como parâmetro a *string* referente a data e a hora da batelada de ensaios selecionada. É interessante notar que não é preciso selecionar exatamente a célula da tabela referente a data e hora. É possível selecionar qualquer coluna da linha referente a essa batelada de ensaios e clicar no botão para abrir o relatório, já que a própria rotina do botão lê a *string* contida na primeira coluna da tabela, correspondente a data e hora.

5.9 CONCLUSÕES PARCIAIS

Esse capítulo apresentou as características relacionadas ao código do *software* computacional e os aspectos referentes ao seu desenvolvimento. Primeiramente foram definidos conceitos básicos da teoria envolvida, como programação orientada a objetos, a linguagem C#, banco de dados e descrição de modelagem de sistemas através do UML.

Foi apresentado o diagrama de casos de uso do sistema de forma a descrever as principais funcionalidades do sistema e a interação dessas funcionalidades com o usuário.

Os atributos e métodos das classes desenvolvidas foram explicados em detalhes. Já as classes pré-construidas da biblioteca *.NET Framework Class Library* que foram utilizadas, podem ter suas descrições completas verificadas na referência bibliográfica indicada ou uma breve definição das mesmas podem ser encontradas no apêndice A. Para todas as telas do *software* computacional foram apresentados os diagramas de classes correspondente, permitindo um entendimento do relacionamento entre as classes.

6 RESULTADOS EXPERIMENTAIS

A fim de verificar o funcionamento do sistema proposto, foi realizada uma análise experimental no laboratório do Núcleo de Iluminação Moderna (NIMO) da Universidade Federal de Juiz de Fora(UFJF).

Além do conjunto de *hardware* relativo ao sistema automatizado, foram utilizados os seguintes equipamentos:

- 1 Fonte de alimentação *Tenma 72 6856 35*
- 1 Fonte CC 5V
- 1 Autotransformador ajustado com relação de transformação 1:0,92
- 1 Autotransformador ajustado com relação de transformação 1:1,06
- 1 *Driver INVENTRONICS* modelo *EUC-075S045ST*
- 1 Luminária *TECNOWATT* modelo *ALFA 70*
- 1 Notebook com o *software* computacional do sistema instalado

Os equipamentos alocados em uma bancados foram conectados ao *hardware* do sistema automatizado. A foto da bancada durante os ensaios pode ser vista na figura 82 e a foto do equipamento em destaque na figura 83.

Figura 82: Foto da bancada durante os ensaios

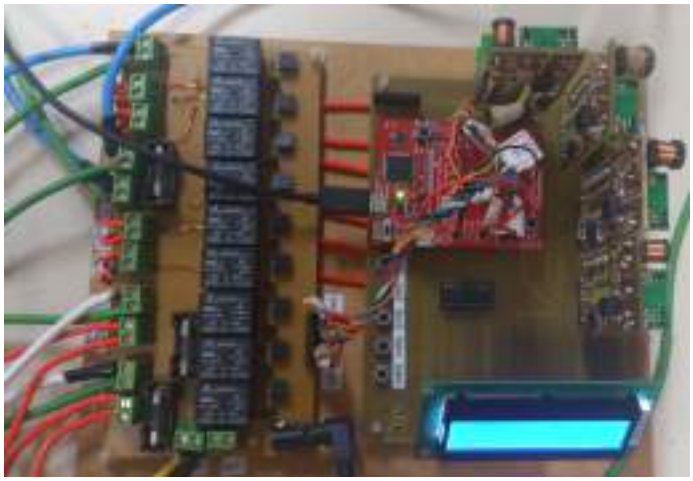


Figura 83: Foto do equipamento em funcionamento

Inicialmente o *driver* e a luminária utilizados foram cadastrados no banco de dados, através do módulo de cadastro do *software* computacional, como pode ser visto na figura 84. Após o cadastro torna-se possível a seleção dos componentes para a realização dos ensaios a qualquer momento, já que os dados relativos a esses componentes permanecerão salvos no banco de dados.

(a) Cadastro do *driver*

(b) Cadastro da luminária

Figura 84: Telas de cadastro *driver*/luminária

Em seguida, através do módulo *Ensaios Automatizados*, foi possível selecionar o *driver* e a luminária que serão utilizados, de acordo com a figura 85.

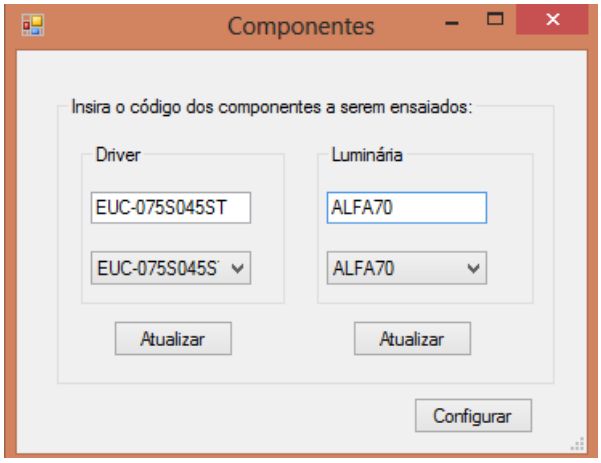


Figura 85: Imagem da seleção do driver e luminária utilizados

O próximo passo foi a configuração do MDGE. De acordo com o *software* foi necessário inserir 6 *jumpers* no sensor de corrente e 7 *jumpers* no sensor de tensão, ambos referentes ao canal 1. Já a configuração necessária para o canal 2 foi de 3 *jumpers* para o sensor de corrente e 5 para o sensor de tensão. As telas de configuração podem ser vistas na figura 86.

(a) Tela de configuração do canal 1

(b) Tela de configuração do canal 2

Figura 86: Telas de configuração dos canais

Devido a problemas na manufatura das placas do MDGE e do MCE, o *hardware* do sistema apresentou alguns problemas de funcionamento. Tais problemas ocorriam de forma intermitente durante o funcionamento do sistema e as principais causas prováveis estavam relacionadas à questões como mal contato em alguns conectores das placas e soldagem inapropriada de alguns elementos do circuito. Dessa forma não foi possível realizar todos os ensaios de uma única vez. Considerando que apenas ensaios

operacionais para condições anormais e o ensaio de durabilidade são os mais prolongados, os dez ensaios selecionados foram realizados em três momentos distintos, ou seja, primeiro foram realizados os ensaios de 1 ao 8, com exceção[1] do ensaio 5, em seguida o ensaio 9 e por fim o ensaio 10. A batelada inicial corresponde ao conjunto de ensaios de características elétricas de funcionamento além do ensaio de curto-circuito. Em seguida foi realizado o ensaio de circuito aberto e por fim o ensaio de comutação. A seleção dos ensaios através do *software* computacional pode ser visto na figura 87.

(a) 1ª seleção dos ensaios da batelada

(b) 2ª seleção dos ensaios

(c) 3ª Seleção dos ensaios

Figura 87: Telas de seleção para as 3 bateladas

As telas de execução dos ensaios podem ser vistas na figura 88.

(a) Tela de execução durante a batelada inicial de ensaios

(b) Tela de execução durante o ensaio de circuito aberto

(c) Tela de execução durante o ensaio de comutação

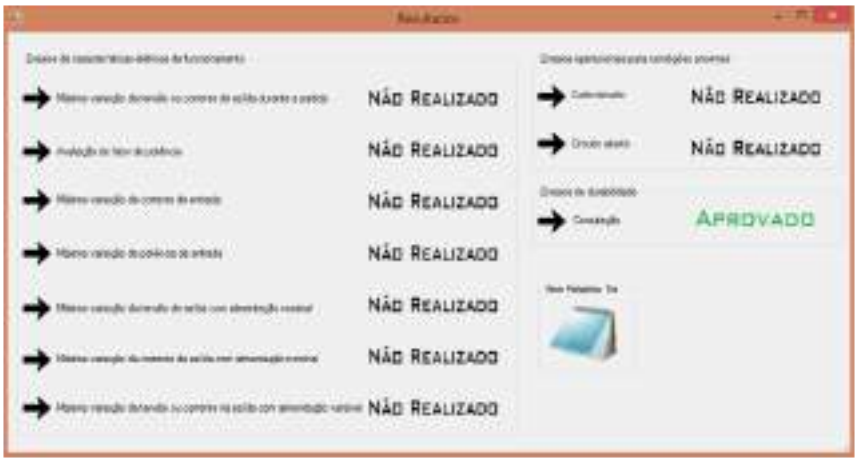
Figura 88: Telas de execução dos ensaio

Ao fim da execução dos ensaios foi aberta a tela correspondente aos resultados dos ensaios, de acordo com a figura 89.

[1]Como o *driver* que foi submetido aos ensaios tem controle de corrente, o ensaio de máxima variação da tensão de saída em regime de operação não foi realizado, visto que esse é feito apenas para *drivers* com controle de tensão

(a) Tela de resultados para a batelada inicial de ensaios

(b) Tela de resultados para o ensaio de circuito aberto



(c) Tela de resultados para o ensaio de comutação

Figura 89: Telas de resultado dos ensaios

6.1 BATELADA INICIAL DE ENSAIOS

A primeira batelada refere-se aos seguintes ensaios:

- Máxima variação da corrente de saída durante a partida.
- Máxima variação da corrente de entrada.
- Avaliação do fator de potência.
- Máxima variação de potência de entrada.
- Máxima variação da corrente de saída em regime de operação.
- Máxima variação da corrente na saída, sob tensões não nominais de entrada.
- Curto-circuito.

Com exceção do ensaio de curto circuito, todos os ensaios dessa batelada são de características elétricas de funcionamento e em conjunto tem duração total de 2min e 28s. Esse conjunto de ensaios foi repetido utilizando instrumentos comerciais, como o *Wattímetro Yokogawa WT230* e o osciloscópio *Tektronix DPO-3014*, a fim de validar os testes. Esses instrumentos podem ser vistos na figura 90.

(a) *Wattímetro Yokogawa WT230*

(b) Osciloscópio *Tektronix DPO-3014* (COELHO, 2016)

Figura 90: Instrumentos comerciais utilizados

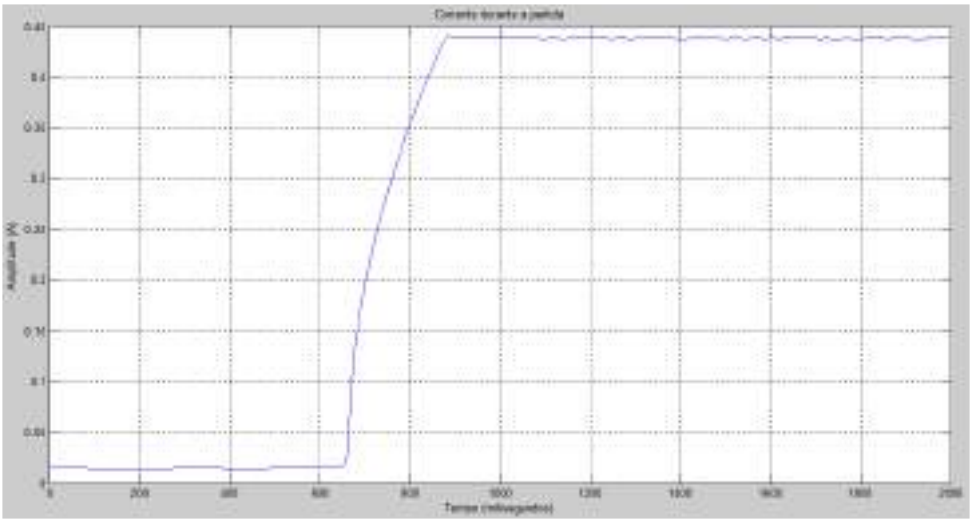
6.1.1 ENSAIO DE MÁXIMA VARIAÇÃO DA CORRENTE DE SAÍDA DURANTE A PARTIDA

Nesse ensaio a corrente de saída foi monitorada durante os dois primeiros segundos e foram coletadas 134 amostras, uma a cada 15ms. Uma frequência de amostragem maior não foi possível devido a limitação de *hardware* do MDGE. O *driver* foi aprovado nesse ensaio de acordo com o relatório gerado que pode ser visto na figura 91.

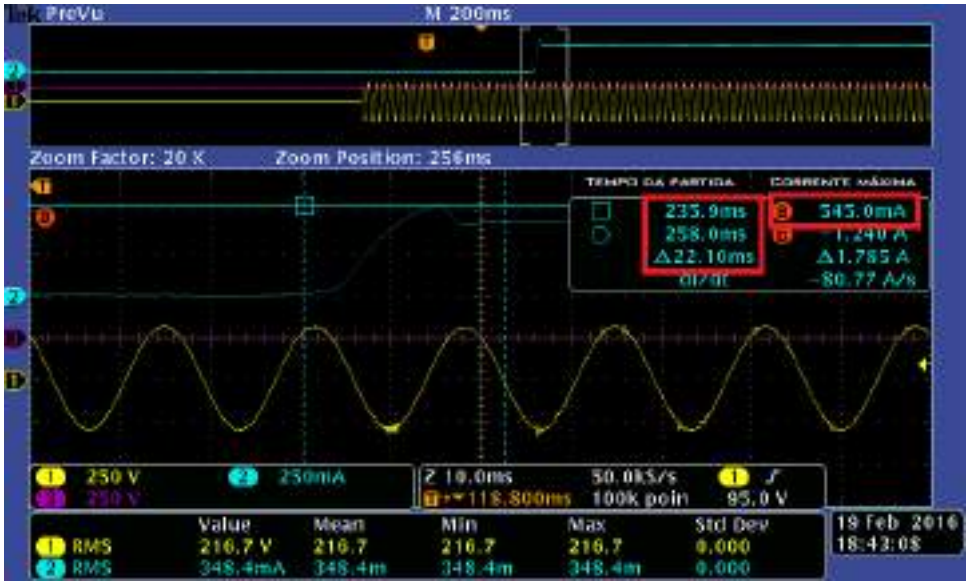
```
Ensaio de máxima variação da tensão ou corrente de saída durante a partida
Corrente Máxima Amostrada (A) :0,44081
Corrente especificada pelo Fabricante (A) :0,45
Resultado do ensaio : Aprovado
```

Figura 91: Relatório gerado para o ensaio de máxima corrente de partida

As amostras de corrente coletadas foram plotadas com o auxílio do *software* Matlab. Foi realizada também uma comparação da curva de corrente de partida utilizando o osciloscópio. As curvas podem ser vistas na figura 92.



(a) Curva de acordo com as aquisições realizadas pelo Wattímetro



(b) Curva de corrente gerada pelo osciloscópio

Figura 92: Gráficos de corrente durante a partida

Pode-se observar de acordo com as amostras coletadas pelo MDGE que a partida ocorreu entre 660 e 900 ms e a corrente máxima amostrada foi de 0,44081A. Já o osciloscópio registrou a partida durante 235,9 e 258ms, com uma corrente máxima de 0,55A. Essa diferença entre as correntes máximas obtida pelos dois equipamentos é justificada pela taxa muito superior de amostragem do osciloscópio, que é de 2,5 bilhões de amostras por segundo. O pico de corrente teve duração muito curta, porém

a norma não especifica nada em relação a tempo. Portanto, considerando apenas o maior valor absoluto medido e de acordo com o instrumento comercial, o *driver* foi reprovado nesse ensaio, visto que apresentou uma corrente 22% superior ao especificado pelo fabricante. A tabela 12 mostra o confrontamento entre os resultados obtidos pelos dois equipamentos.

	Sistema Automatizado	Osciloscópio
Tempo de partida (ms)	660 a 900	235,9 a 258
Pico de corrente amostrado (A)	0,44	0,55
Superior em relação ao fabricante	-2,2%	22%
Resultado do ensaio	Aprovado	Reprovado

Tabela 12: Comparação com os resultados obtidos pelo osciloscópio

6.1.2 ENSAIO DE AVALIAÇÃO DO FATOR DE POTÊNCIA

Nesse ensaio foi avaliado o fator de potência em regime de operação e o *driver* foi considerado aprovado. Foram coletadas 20 amostras do fator de potência, uma a cada 400ms. A média dessas amostras foi de 0,9399, apresentando uma diferença menor que 0,05 do fator de potência especificado pelo fabricante. A figura 93 mostra o relatório gerado do ensaio. A tabela 13 apresenta o confrontamento dos resultados com o *Wattímetro Yokogawa*.

```
Ensaio de avaliação do fator de potência
Fator de potência especificado pelo Fabricante :0,96
Fator de potência médio amostrado :0,9399
Resultado do ensaio : Aprovado
```

Figura 93: Relatório gerado para o ensaio de fator de potência

	Sistema Automatizado	Yokogawa
Fator potência medido	0,9399	0,9448
Variação em relação ao fabricante	2,09%	1,5%
Resultado do ensaio	Aprovado	Aprovado

Tabela 13: Comparação com os resultados obtidos pelo *Wattímetro Yokogawa*

6.1.3 ENSAIO DE MÁXIMA VARIAÇÃO DA CORRENTE DE ENTRADA

Nesse ensaio, a variação da corrente de entrada em regime de operação foi monitorada e permaneceu dentro da faixa esperada, com diferença menor que 10% da corrente especificada pelo fabricante. Foram coletadas 20 amostras da corrente de entrada, uma a cada 400ms. A média dessas amostras foi de 0,3369A, como pode ser visto na figura 94 que mostra o relatório gerado. O confrontamento com o resultado obtido pelo *Wattímetro Yokogawa* é mostrado na tabela 14.

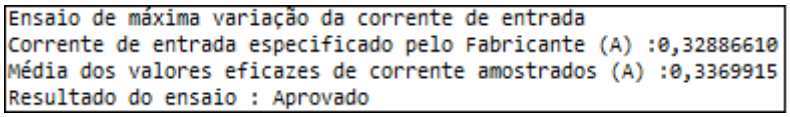


Figura 94: Relatório gerado para o ensaio de máxima variação da corrente de entrada

	Sistema Automatizado	*Yokogawa*
Valor eficaz da corrente de entrada medido (A)	0,3369	0,3329
Variação em relação ao fabricante	2,4%	1,2%
Resultado do ensaio	Aprovado	Aprovado

Tabela 14: Comparação com os resultados obtidos pelo *Wattímetro Yokogawa*

6.1.4 ENSAIO DE MÁXIMA VARIAÇÃO DE POTÊNCIA DE ENTRADA

O ensaio de máxima variação da potência de entrada em regime de operação obteve resultado satisfatório apresentando uma média de 69,42W de potência de entrada. Foram coletadas 20 amostras da potência de entrada, uma a cada 400ms. O relatório gerado para esse ensaio pode ser visto na figura 95. A tabela 15 mostra o confrontamento com o resultado obtido pelo *Wattímetro Yokogawa*.

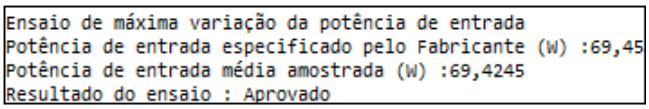


Figura 95: Relatório gerado para o ensaio de máxima variação da potência de entrada

	Sistema Automatizado	*Yokogawa*
Potência de entrada medida (W)	69,42	69,5
Variação em relação ao fabricante	0,04%	0,07%
Resultado do ensaio	Aprovado	Aprovado

Tabela 15: Comparação com os resultados obtidos pelo *Wattímetro Yokogawa*

6.1.5 ENSAIO MÁXIMA VARIAÇÃO DA CORRENTE DE SAÍDA EM REGIME DE OPERAÇÃO

Nesse ensaio a variação da corrente de saída em regime de operação foi avaliada. O *driver* foi considerado aprovado já que apresentou uma diferença menor que 10% para corrente de saída em comparação ao valor especificado pelo fabricante. Foram coletadas 20 amostras da corrente de saída como nos ensaios anteriores e a média dessas amostras foi de 0,4368A, como pode ser visto no relatório gerado da figura 96. A tabela 16 mostra o confrontamento com o resultado obtido pelo *Wattímetro Yokogawa*.

```
Ensaio de máxima variação da corrente de saída em regime de operação
Corrente de saída especificada pelo Fabricante (A) :0,45
Corrente de saída amostrada (A) :0,43687
Resultado do ensaio : Aprovado
```

Figura 96: Relatório gerado para o ensaio de máxima variação da corrente de saída

	Sistema Automatizado	*Yokogawa*
Corrente de entrada medida (A)	0,4368	0,4358
Variação em relação ao fabricante	2,9%	3,1%
Resultado do ensaio	Aprovado	Aprovado

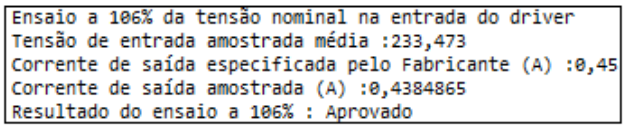
Tabela 16: Comparação com os resultados obtidos pelo *Wattímetro Yokogawa*

6.1.6 ENSAIO DE MÁXIMA VARIAÇÃO DA CORRENTE DE SAÍDA QUANDO O DRIVER É ALIMENTADO COM TENSÃO NÃO NOMINAL

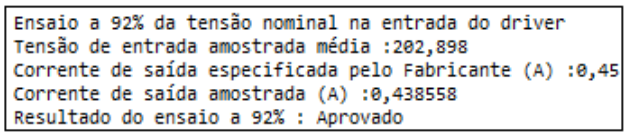
A corrente de saída foi monitorada quando o *driver* foi alimentado com tensão de 106% da nominal e em seguida com tensão de 92% da nominal. Para ambos os casos, 20 amostras da corrente de saída foram coletadas, sendo a amostragem feita da mesma forma que nos ensaios anteriores. O *driver* cumpriu os requisitos do ensaio e foi considerado aprovado já que para ambas as tensões de entrada a corrente de saída

não apresentou diferença maior que 10% da corrente especificada pelo fabricante. A média das amostras quando alimentado com tensão de 106% da nominal foi de 0,4373A e quando alimentado com tensão de 92% da nominal foi de 0,4369A. Os relatórios gerados pelo sistema automatizado para ambas tensões podem ser vistos na figura 97.

As tabelas 17 e 18 apresentam o confrontamento com os resultados obtidos pelo *Wattímetro Yokogawa*.



(a) Relatório parcial gerado quando o driver é alimentado com tensão correspondente a 106% da nominal



(b) Relatório parcial gerado quando o driver é alimentado com tensão correspondente a 92% da nominal

Figura 97: Relatórios gerados para o ensaio com alimentação não nominal

	Sistema Automatizado	*Yokogawa*
Tensão de entrada medida (V)	234,14	233,47
Corrente de saída medida (A)	0,4384	0,4378
Variação em relação ao fabricante	2,5%	2,7%
Resultado do ensaio	Aprovado	Aprovado

Tabela 17: Comparação de resultados para tensão de entrada a 106% da nominal

	Sistema Automatizado	*Yokogawa*
Tensão de entrada medida (V)	202,89	202,95
Corrente de saída medida (A)	0,4385	0,4382
Variação em relação ao fabricante	2,5%	2,6%
Resultado do ensaio	Aprovado	Aprovado

Tabela 18: Comparação de resultados para tensão de entrada a 92% da nominal

6.1.7 ENSAIO DE CURTO-CIRCUITO

Nesse ensaio, a saída do *driver* foi submetida a um curto-circuito e em seguida a corrente foi monitorada através de amostragem a cada 400ms, num total de 20 amostras. A média das amostras foi de 0,435A, enquanto a corrente especificada pelo fabricante é de 0,45A. Portanto, a corrente se encontra dentro da faixa aceitável e o ensaio foi considerado aprovado. O relatório gerado é mostrado na figura 98 e a tabela 19 apresenta a análise do resultado.

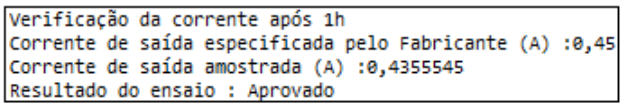


Figura 98: Relatório gerado para o ensaio de curto-circuito

	Sistema Automatizado
Corrente de saída medida (A)	0,4355
Variação em relação ao fabricante	3,2%
Resultado do ensaio	Aprovado

Tabela 19: Análise dos resultados do ensaio de curto-circuito

6.2 ENSAIO DE CIRCUITO ABERTO

No ensaio de circuito aberto, o módulo de LED foi removido e o *driver* foi alimentado sob tensão nominal durante uma hora. Após esse tempo, a corrente foi monitorada através de amostragem a cada 400ms, num total de 20 amostras. A média das amostras foi de 0,436A, enquanto a corrente especificada pelo fabricante é de 0,45A. O relatório gerado é mostrado na figura 99 e a tabela 20 apresenta a análise do resultado.

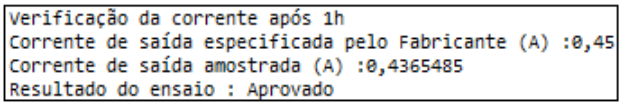


Figura 99: Relatório gerado para o ensaio de circuito aberto

	Sistema Automatizado
Corrente de saída medida (A)	0,436
Variação em relação ao fabricante	3,1%
Resultado do ensaio	Aprovado

Tabela 20: Análise dos resultados do ensaio de circuito aberto

6.3 ENSAIO DE COMUTAÇÃO

No ensaio de comutação o *driver* foi alimentado com tensão nominal por 30s e em seguida desenergizado por mais 30s. Esse ciclo é repetido por 200 vezes sem o módulo de LEDs inserido e em seguida, 800 vezes com o módulo de LEDs inserido. Devido aos problemas de *hardware*, esse ensaio foi o que mais apresentou dificuldades para realização por ser o mais prolongado, com duração de cerca de 17h. Durante o ensaio, o MDGE parava de responder ou a comunicação serial falhava. Dessa forma, para validar a ideia foram realizados 20 ciclos sem o módulo de LEDs e em seguida mais 20 ciclos com o módulo de LEDs inserido. Para esse reduzido número de ciclos o sistema se comportou adequadamente e o ensaio foi realizado como o esperado e com sucesso. Ao final dos ciclos, a corrente foi monitorada a cada 400ms, durante 15 min. A média da corrente observada foi de 0,444A, de acordo com o relatório gerado que pode ser visto na figura 100. A tabela 21 apresenta a análise do resultado.

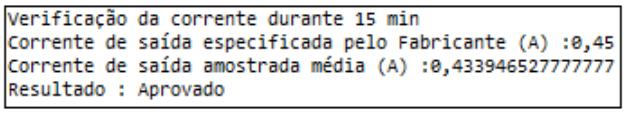


Figura 100: Relatório gerado para o ensaio de comutação

	Sistema Automatizado
Corrente de saída medida (A)	0,4335
Variação em relação ao fabricante	3,5%
Resultado do ensaio	Aprovado

Tabela 21: Análise dos resultados do ensaio de comutação

6.4 CONCLUSÕES PARCIAIS

Esse capítulo apresentou o funcionamento do sistema automatizado, através da realização de ensaios da norma NBR 16026 em um *driver* da marca *INVENTRONICS*.

Os ensaios propostos foram realizados com sucesso, com exceção do ensaio de comutação que não pôde ser realizado por completo, tendo sido feito parcialmente, devido a problemas de *hardware*. O *driver* ensaiado se comportou como especificado pela norma e foi considerado aprovado em todos os ensaios, de acordo com o MDGE, como pôde ser visto nos relatórios mostrados. Além disso, os resultados dos testes de características elétricas de funcionamento realizados pelo sistema automatizado foram comparados aos resultados obtidos por instrumentos comerciais a fim de validar o sistema proposto. Apenas o ensaio de máxima corrente de saída durante a partida apresentou resultado diferente entre os equipamentos, visto que o transiente da corrente é muito rápido. Dessa forma, o MDGE não foi capaz de capturar o mesmo valor de pico registrado pelo osciloscópio, já que esse possui uma taxa de amostragem muito superior ao do MDGE.

Diferentemente dos ensaios em regime de operação, que foi feito o confrontamento dos resultados com o *Wattímetro Yokogawa*, no ensaio de máxima corrente de partida, a comparação foi feita com o osciloscópio *Tektronix DPO-3014*, de forma a obter a curva de partida. Para os ensaios de curto-circuito, circuito aberto e comutação, não foram realizadas comparações com instrumentos comerciais, visto o prolongado tempo de execução desses ensaios.

7 CONCLUSÕES FINAIS

7.1 CONCLUSÕES

Este trabalho mostrou a concepção, conceituação e desenvolvimento de um sistema automatizado para testes de controladores de LED. O desenvolvimento de todas as etapas necessárias foi relatado ao longo dos capítulos desta monografia.

O capítulo 2 mostrou um pouco do histórico da automação, assim como alguns casos de aplicação da automação em ensaios de avaliação e validação de equipamentos encontrados na literatura. Foi apresentada a importância da normatização na fabricação de controladores de LED, visto que os *drivers* são fundamentais para garantir o funcionamento adequado e a longa vida útil de uma luminária de LEDs. As principais normas relacionadas a iluminação de estado sólido foram citadas, com maior enfoque na norma NBR 16026, que aborda os requisitos de desempenho do dispositivo de controle eletrônico CC ou CA para módulos de LED. Alguns ensaios exigidos por essa norma foram selecionados para serem realizados pelo sistema automatizado e os mesmos foram descritos.

O capítulo 3 mostrou os requisitos de *hardware* necessários para o desenvolvimento do sistema. Foi visto que o *hardware* do sistema é composto pelo medidor digital de grandezas elétricas(MDGE) e o módulo de comutação de estados(MCE). O primeiro possui sensores de corrente e tensão, além de um microcontrolador *Stellaris* responsável por converter os sinais analógicos enviados pelos sensores para parâmetros digitais através de um periférico ADC. Foi mostrada a necessidade do ajuste manual da escala a fim de melhorar a resolução do equipamento, assim como a correspondência entre número de *jumpers* nos sensores e resolução equivalente. Já o MCE, desenvolvido especialmente para o sistema automatizado proposto, atua por meio de relés de forma a alterar as conexões elétricas do circuitos e atender as especificidades de cada ensaio a ser realizado. Foi visto que cada relé possui um circuito de controle, o qual é acionado através de uma porta *GPIO* do microcontrolador. A porta é configurada pelo programa

embarcado no microcontrolador por meio de um protocolo específico, implementado por meio de uma comunicação serial entre o *software* computacional e o microcontrolador.

Depois de mostrada a parte física do equipamento no capítulo 3, foi apresentado no capítulo 4 o último elemento do sistema automatizado, o *software* computacional, responsável pela inteligência do sistema. Foram apresentados todos os módulos do *software* desenvolvido de forma a permitir o entendimento do usuário e possibilitar sua utilização. Foi visto que o *software* é composto por diferentes módulos, entre eles, o módulo que permite cadastrar, editar e buscar luminárias e *drives* para futuros ensaios. Há também o módulo para testar o funcionamento adequado do equipamento, permitindo ao usuário verificar se não há nenhum problema com o equipamento antes de iniciar os ensaios. Já o módulo *Ensaios Automatizados* possibilita a seleção dos ensaios a serem realizados, assim como a seleção do *driver* e luminária que serão utilizados. O módulo *Banco de Resultados* permite buscar resultados e relatórios de ensaios já realizados.

O capítulo 5 descreve os aspectos do desenvolvimento do *software*. Foi visto que o *software* foi desenvolvido na linguagem C# e implementada uma integração com o sistema gerenciador de banco de dados MySQL, a fim de armazenar dados de controladores de LED, luminárias e relatórios de ensaios realizados. Além disso, o *software* computacional envia comandos para o programa embarcado do microcontrolador, o qual realiza as tarefas solicitas controlando por exemplo o acionamento dos relés que definem as conexões elétricas do circuito do MCE, de forma a atender as especificidades de cada ensaio. O *software* computacional também é responsável por configurar algumas características relacionadas ao MDGE, como a escala de medição utilizada de forma que o *software* embarcado no microcontrolador possa ajustar a curva de calibração adequada. Além de todas essas funções, o *software* computacional solicita que o microcontrolador envie os parâmetros elétricos calculados relativos as medições realizadas para que sejam analisados e comparados com os valores fornecidos pelo fabricante. Foram revisados conceitos de linguagem orientada a objetos, banco de dados e descrição de *softwares* através da linguagem UML. Por fim, foi mostrado a descrição das classes criadas durante o desenvolvimento do *software* e seus relacionamentos com as classes da biblioteca *.NET Framework Class Library* para cada tela do *software* computacional.

O capítulo 6 mostrou os resultados experimentais alcançados através de ensaios realizados pelo sistema automatizado proposto. Os ensaios foram realizados com sucesso, com exceção do ensaio de comutação que que foi executado apenas parcialmente,

com um número reduzido de comutações, devido a problemas no *hardware* do sistema proposto. Já no ensaio de máxima variação da corrente durante a partida, o MDGE não teve o mesmo resultado de um instrumento comercial, devido a sua taxa inferior de amostragem, não sendo possível obter o pico real de corrente.

7.2 TRABALHOS FUTUROS

Como proposta para trabalhos futuros pode-se citar a necessidade da fabricação de uma nova placa, utilizando materiais de melhor qualidade (e.g., conectores mais robustos) e uma técnica de fabricação mais adequada. Além disso, uma melhoria no *software* computacional deve ser estudada no sentido de tratar possíveis problemas decorrentes de falhas no *hardware*.

Outro trabalho a ser realizado futuramente, é a integração em uma única placa do MDGE com o MCE, bem como o desenvolvimento de um único *software* computacional permitindo a utilização do MDGE em aplicações gerais de medições em cargas quaisquer, ou sendo aplicado em conjunto com o MCE, atuando como instrumento de medição para o sistema automatizado de ensaios para avaliação de parâmetros elétricos em controladores de LED.

Finalmente, um aumento da gama de ensaios do equipamento proposto deve ser avaliado, de modo que o sistema possa oferecer um diagnóstico ainda mais completo do controlador de LEDs.

REFERÊNCIAS

ABNT. *IEC 60598-1, Luminárias: Requisitos gerais e ensaios*. 2010.

ABNT. *NBR 15129, Luminárias para iluminação pública: Requisitos particulares*. 2012.

ABNT. *NBR 16026, Requisitos de desempenho de dispositivo de controle eletrônico CC ou CA para módulos de LED*. 2012.

ABNT. *NBR IEC 61347-2-13, Requisitos particulares para dispositivos de controle eletrônico alimentados em CC ou CA para os módulos de LED*. 2012.

AMBLER, S. W. *The Elements of UML (TM) 2.0 Style*. : Cambridge University Press, 2005.

ARSENEAU, R.; SUTHERLAND, M. E.; ZELLE, J. J. A test system for calibrating flickermeters. *Instrumentation and Measurement, IEEE Transactions on*, IEEE, v. 51, n. 4, p. 598–600, 2002.

BOLTON, W. *Programmable logic controllers*. : Newnes, 2015.

CARBALLO, C. et al. A computerized system for testing batteries in full controlled environment. In: IEEE. *Instrumentation and Measurement Technology Conference, 2000. IMTC 2000. Proceedings of the 17th IEEE*. 2000. v. 1, p. 395–399.

COELHO, F. de O. *Medidor digital de grandezas elétricas com capacidade de gerenciamento remoto*. Trabalho de conclusão de curso, 2016.

DEITEL, H. *Visual C# 2012 How to Program*. : Prentice Hall Press, 2012.

DUBOIS, P. *MySQL (Developer's Library)*. : Sams, 2009.

DYER, R. *MySQL in a Nutshell*. : "O'Reilly Media, Inc.", 2008.

ELMASRI, R. et al. Sistemas de banco de dados. Pearson Addison Wesley, 2005.

FAIRCHILD SEMICONDUCTORS. *BC337 BC338 - NPN Epitaxial Silicon Transistor - Fairchild*. : FAIRCHILD SEMICONDUCTORS, 2015. `https://www.fairchildsemi.com/datasheets/BC/BC337.pdf`.

GUEDES, G. T. Uml 2: uma abordagem prática. *São Paulo: Novatec*, 2011.

KRAMES, M. R. et al. Status and future of high-power light-emitting diodes for solid-state lighting. *Display Technology, Journal of*, IEEE, v. 3, n. 2, p. 160–175, 2007.

LAUBSCH, A. et al. High-power and high-efficiency ingan-based light emitters. *Electron Devices, IEEE Transactions on*, IEEE, v. 57, n. 1, p. 79–87, 2010.

MENDES, D. R. *Programação Java com ênfase em Orientação a Objetos.* : Novatec Editora, 2009.

NOGUEIRA, F. J. *Contribuições ao Uso de Diodos Emissores de Luz em Iluminação Pública*. Dissertação de Mestrado, 2013.

NOGUEIRA, F. J. et al. Aplicação dos diodos emissores de luz orientada a sistemas de iluminação pública. *CES Revista*, v. 27, n. 1, p. 31–49, 2015.

NXP SEMICONDUCTORS . *High-speed diodes.* : NXP SEMICONDUCTORS, 2004. `http://www.nxp.com/documents/data_sheet/1N4148_1N4448.pdf`.

PAREDE, I. M.; GOMES, L. E. L. *Eletrónica: automação industrial*. 2011.

PILLET, M. *Appliquer la maîtrise statistique des procédés (MSP/SPC).* : les Éd. d'Organisation, 1994.

PODGORSKI, A.; DUMINOV, S.; LEONIAK, R. A multichannel measurement system for automatic testing of acoustic calibrators and adjustment of their parameters. In: IEEE. *Instrumentation and Measurement Technology Conference Proceedings, 2007. IMTC 2007. IEEE*. 2007. p. 1–6.

POLONSKII, M.; SEIDEL, A. Reatores eletrônicos para iluminação fluorescente. *Editora Unijuí*, v. 1, 2008.

RODRIGUES, C. R. B. S. *Avaliação Experimental de Luminárias Empregando LEDs Orientadas a Iluminação Pública /*. Tese de Doutorado, 2012.

SALES, R. P. *LED, o Novo Paradigma da Iluminação Pública*. Dissertação de Mestrado, 2011.

SANYOU RELAYS. *Miniature Power Relay.* : SANYOU RELAYS, 2015. `http://www.sanyourelay.ca/public/products/pdf/SRD.pdf`.

SCHREIBER, R.; FUCHS, P.; JAKSCH, I. An automatic measurement, testing and diagnostic system for induction motors. In: IEEE. *Electronics, Control, Measurement, Signals and their Application to Mechatronics (ECMSM), 2015 IEEE International Workshop of*. 2015. p. 1–6.

SILVEIRA, L.; LIMA, W. Q. Um breve histórico conceitual da automação industrial e redes para automação industrial. *Redes para Automação Industrial. Universidade Federal do Rio Grande do Norte*, 2003.

TAHAGHOGHI, S. M.; WILLIAMS, H. E. *Learning MySQL.* : "O'Reilly Media, Inc.", 2006.

TANENBAUM, A. S. Computer networks, 4-th edition. *ed: Prentice Hall*, 2003.

TEXAS INSTRUMENTS. *Stellaris®LM4F120H5QR Microcontroller.* : Texas Instruments, 2013. `http://www.mouser.com/ds/2/405/lm4f120h5qr-124014.pdf`.

VISHAY SEMICONDUCTORS. *Optocoupler, Phototransistor Output, with Base Connection.* : VISHAY SEMICONDUCTORS, 2010. `http://www.vishay.com/docs/83725/4n25.pdf`.

WILLIAMS, L. An introduction to the unified modeling language. *Retrieved November*, v. 10, p. 2014, 2004.

XUEMEI, Y.; JIANGFENG, S. Semi-automatic system for testing dielectric properties of low-voltage busbar. In: IEEE. *Electrical and Control Engineering (ICECE), 2010 International Conference on*. 2010. p. 1876–1879.

APÊNDICE A – CLASSES DA BIBLIOTECA .NET FRAMEWORK CLASS LIBRARY

Nome da classe	Descrição
ArrayList	Implementa uma interface *IList* usando uma matriz cujo tamanho é aumentado dinamicamente conforme necessário.
BackgroundWorker	Executa uma operação em uma *thread* separada.
BinaryReader	Lê tipos de dados primitivos como valores binários em uma codificação específica. Através do método *ReadBytes*, lê o número de bytes especificado do *Stream* em uma matriz de *bytes* e avança a posição atual pelo número de *bytes*.
Button	Representa um controle de botão do Windows.
ComboBox	Representa um controle de *Combo Box* do Windows.
CheckBox	Representa um Windows *Check Box*.
DataGridView	Exibe dados em uma grade personalizável.
DataTable	Representa uma tabela de dados na Memória.
File	Fornece métodos estáticos para criar, copiar, excluir, mover e abrir um único arquivo e ajuda na criação de *FileStream* objetos.

FileStream	Fornece um *Stream* para um arquivo que podem ser acessados pelo índice.
Form	Representa uma janela ou caixa de diálogo que compõe uma interface para usuário do aplicativo.
GroupBox	Representa um controle que cria um contêiner que contém uma borda e um cabeçalho para o conteúdo de uma interface de usuário.
Image	Uma classe que fornece funcionalidade para o Bitmap. É utilizado um método de classe chamado Image.FromStream, para criar um *Image* a partir de um *stream*.
Label	Representa o controle para um *Text Label*.
List	Representa uma lista de objetos fortemente tipados que podem ser acessados pelo índice.
MemoryStream	Cria um *Stream*, um fluxo cujo armazenamento é baseado em memória.
MenuStrip	Fornece um sistema de menu para um formulário.
MySQLCommand	Representa uma instrução ou procedimento armazenado para ser executado em um banco de dados. Essa classe recebe como parâmetros ambos atributos já citados no parágrafo anterior, “vsql”e “conexão”.
MySqlConnection	Classe responsável pela conexão com o banco de dados. Recebe como parâmetro uma *string* que indica dados como o servidor, o nome do banco de dados e senha de acesso.
MySqlDataAdapter	Representa um conjunto de comandos de dados

	e uma conexão de banco de dados que são usadas para preencher a *DataSet* (conjunto de dados).
MySqlDataReader	Fornece uma maneira de ler um fluxo uni-direccional das linhas de uma base de dados MySQL Server.
MySqlException	Exceção gerada quando o servidor MySQL retorna um erro.
MessageBox	Exibe uma caixa de mensagem que pode conter texto, botões e símbolos que informam e instruem o usuário, como por exemplo, quando ocorre um erro durante a conexão com o banco de dados.
Object	Suporta todas as classes na hierarquia de classes do .NET Framework e fornece serviços de baixo nível para classes derivadas.
OpenFileDialog	Exibe uma caixa de diálogo padrão que solicita ao usuário para abrir um arquivo.
PictureBox	Representa uma caixa de imagem do Windows para possibilitar a exibição de uma imagem.
Process	Fornece acesso a processos locais e remotos e permite iniciar e parar os processos do sistema local.
ProgressBar	Indica o progresso de uma operação.
RadioButton	Permite que o usuário selecionar uma única opção de um conjunto de opções.
SerialPort	Representa um recurso de porta serial.
StatusStrip	Representa um controle de barra de status do Windows.
StreamWriter	Implementa uma *TextWriter* para escrever

	caracteres em um fluxo para uma codificação específica.
TabControl	Representa um controle que contém várias páginas que compartilham o mesmo espaço na tela.
TabPage	Representa uma única página em um *TabControl*.
TextBox	Representa um controle da caixa de texto do Windows.
TextBox	Representa um controle de caixa de texto do Windows.
Thread	Cria e controla um *thread*, define sua prioridade e obtém seu status. Essa classe foi utilizada para interromper a execução de alguns processos durante um tempo específico, através do método *Thread.Sleep*.
Timer	Gera um evento após um intervalo definido servindo com uma opção para gerar eventos recorrentes.
ToolStripMenuItem	Representa uma opção selecionável exibida em um *MenuStrip* ou *ContextMenuStrip*.
ToolTip	Representa uma janela pop-up pequena retangular que exibe uma descrição breve da finalidade de um controle quando o usuário posiciona o ponteiro do mouse no controle.

Tabela 22: Tabela de classes pré-construidas