
**Desenvolvimento de um Projeto de
Hardware-In-the-Loop (HIL) Integrado ao
Software PSP-UFU para Análise de Proteção de
Sistemas Elétricos de Potência Utilizando Relés**

Otávios Sousa Siqueira



UNIVERSIDADE FEDERAL DE UBERLÂNDIA
FACULDADE DE ENGENHARIA ELÉTRICA
PROGRAMA DE GRADUAÇÃO EM ENGENHARIA ELÉTRICA

Uberlândia
2025

Otávios Sousa Siqueira

**Desenvolvimento de um Projeto de
Hardware-In-the-Loop (HIL) Integrado ao
Software PSP-UFU para Análise de Proteção de
Sistemas Elétricos de Potência Utilizando Relés**

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

Área de concentração: Engenharia Elétrica

Orientador: Geraldo Caixeta Guimarães

Coorientador: Thales Lima Oliveira

Uberlândia

2025

*Este trabalho é dedicado a todos que me auxiliaram nessa jornada e me permitiram
chegar onde estou.*

Agradecimentos

Agradeço primeiramente a Deus Jeová, pela força, saúde e sabedoria concedidas ao longo desta caminhada.

À minha família, meu pai Luiz, minha mãe Enete e meu irmão Gustavo, pelo amor incondicional, apoio constante e por sempre acreditarem em mim, mesmo nos momentos mais desafiadores. Sem o suporte de vocês, esta conquista não seria possível.

Aos Professores, Doutor Thales Lima Oliveira e Doutor Geraldo Caixeta Guimarães, pela orientação precisa, paciência e incentivo durante todas as etapas deste trabalho. A dedicação de vocês foi fundamental para o desenvolvimento deste projeto.

Aos meus colegas de turma, Alexandre, Aline, Cássio, Eduardo, Felipe, João Dias, João Cardoso, José, Lisandra, Pedro e Heitor, pela colaboração, incentivo e estudos. Em especial, aos colegas e amigos Mateus Flausino Conceição e Victor Borges Felicíssimo, pela parceria, troca de conhecimentos, discussões, partidas de videogame e pela amizade construída ao longo da graduação. A colaboração de vocês foi essencial para a superação de desafios e para o crescimento mútuo, não sei se conseguiria me formar sem vocês.

A todos os professores, colegas e profissionais da Universidade Federal de Uberlândia que, direta ou indiretamente, contribuíram para a minha formação acadêmica e pessoal, deixo também meu sincero agradecimento.

Muito obrigado a todos!

*“Aqui no entanto nós não olhamos para trás por muito tempo,
Nós continuamos seguindo em frente, abrindo novas portas e fazendo coisas novas,
Porque somos curiosos ... e a curiosidade continua nos conduzindo por novos caminhos.
Siga em frente.”
(Walt Disney)*

Resumo

Este Trabalho de Conclusão de Curso (TCC) propõe o desenvolvimento de um sistema *Hardware-In-the-Loop* (HIL) integrado à Plataforma de Sistemas de Potência da Universidade Federal de Uberlândia (PSP-UFU) para fins de simulação e análise de proteção de sistemas elétricos de potência. A proposta surgiu no contexto de um projeto de Iniciação Científica com o objetivo de utilizar relés de proteção reais acoplados a um ambiente virtual de simulação, permitindo análises práticas e seguras. Para isso, adotou-se o software PSP-UFU como simulador, a placa Arduino UNO como interface de comunicação e a linguagem Python para transferência e leitura de dados entre software e hardware. A partir da exportação de dados do PSP-UFU em formato *Comma Separated Values* (CSV), foi implementado um sistema de envio e recepção serial de dados com o Arduino, possibilitando o teste e a visualização do funcionamento dos relés. Complementarmente, foi desenvolvido um conversor eletrônico com modulação *Sinusoidal Pulse Width Modulation* (SPWM) para simular sinais reais de corrente alternada, os quais atuam diretamente sobre os relés de proteção. Os resultados demonstraram a viabilidade do sistema proposto como ferramenta didática e de pesquisa, oferecendo um ambiente controlado, de baixo custo e com potencial de replicação para estudos em proteção de sistemas elétricos.

Palavras-chave: Sistemas Elétricos de Potência. Relés de Proteção. Simulação HIL. PSP-UFU. Arduino. SPWM.

Abstract

This Final Year Project presents the development of a Hardware-In-the-Loop (HIL) system integrated with the PSP-UFU software for simulation and analysis of power system protection. Originating from a Scientific Initiation project, the objective was to connect real protection relays to a virtual simulation environment, enabling practical and safe testing scenarios. PSP-UFU was used as the simulator, the Arduino UNO board served as the communication interface, and the Python programming language was employed to manage data transfer between software and hardware. By exporting data from PSP-UFU in CSV format, a system for serial data transmission and reception was implemented using Arduino, allowing the evaluation and visualization of relay operation. Additionally, an electronic converter with Sinusoidal Pulse Width Modulation (SPWM) modulation was designed to simulate alternating current signals, which directly trigger the protection relays. The results confirmed the feasibility of the proposed system as a low-cost, replicable, educational and research tool, offering a safe and controlled environment for studies in power system protection.

Keywords: Power Systems. Protection Relays. HIL Simulation. PSP-UFU. Arduino. SPWM.

Lista de ilustrações

Figura 1 – Master Kit Arduino Black Board da Robocore	14
Figura 2 – Interface Gráfica do PSP-UFU	24
Figura 3 – Modelo de um relé de proteção	25
Figura 4 – Placa Arduino modelo UNO	27
Figura 5 – Resposta enviada ao Visual Studio Code.	33
Figura 6 – Projeto do Inversor.	37
Figura 7 – Gráfico da Simulação de Tensão e Corrente para <i>duty cycle</i> de 90%. . .	37
Figura 8 – Gráfico da Simulação de Tensão e Corrente para <i>duty cycle</i> de 50%. . .	38
Figura 9 – Gráfico da Simulação de Tensão e Corrente para <i>duty cycle</i> de 10%. . .	38
Figura 10 – Desenho esquemático do Conversor SPWM	39
Figura 11 – Desenho <i>Printed Circuit Board</i> (PCB) do Conversor SPWM	40
Figura 12 – Desenho PCB do Conversor SPWM	40

Lista de siglas

A/D Analógico-Digital

AVR *Automatic Voltage Regulator*

CA Corrente Alternada

CAD *Computer Aided Design*

CBM *Condition Based Maintenance*

CC Corrente Contínua

CPU *Central Processing Unit*

CSV *Comma Separated Values*

DSP *Digital Signal Processor*

EEPROM *Electrically-Erasable Programmable Read-Only Memory*

EMI *Electromagnetic Interference*

FOSS *Free and Open Source Software*

FAPEMIG Fundação de Amparo à Pesquisa do Estado de Minas Gerais

FEELT Faculdade de Engenharia Elétrica

GIS *Gas-Insulated Substation*

GPS *Global Position System*

GUI *Graphical User Interface*

HIL *Hardware-In-the-Loop*

I2C *Inter-Integrated Circuit*

IC Iniciação Científica

IDE *Integrated Development Environment*

IEC *International Electrotechnical Commission*

IEDs *Intelligent Electronic Devices*

IOT Internet das Coisas

LCD *Liquid Crystal Display*

PCB *Printed Circuit Board*

PSP-UFU Plataforma de Sistemas de Potência da Universidade Federal de Uberlândia

PWM *Pulse Width Modulation*

RAM *Random-Access Memory*

ROM *Read-Only Memory*

SCADA *Supervisory Control and Data Acquisition*

SEP *Sistemas Elétricos de Potência*

SPI *Serial Peripheral Interface*

SPWM *Sinusoidal Pulse Width Modulation*

SRAM *Static Random-Access Memory*

TC *Transformadores de corrente*

TCC *Trabalho de Conclusão de Curso*

TP *Transformadores de potencial*

UART *Universal Asynchronous Receiver/Transmitter*

UFU *Universidade Federal de Uberlândia*

VSC *Visual Studio Code*

Sumário

1	INTRODUÇÃO	13
1.1	Objetivos	14
1.1.1	Objetivos Gerais	14
1.1.2	Objetivos específicos	14
1.2	Metologia do trabalho	15
1.3	Estrutura do trabalho	15
2	FUNDAMENTAÇÃO TEÓRICA	17
2.1	Sistemas Elétricos de Potência	17
2.1.1	Simulador de Sistema Elétrico de Potência	18
2.1.2	Subestações Elétricas e seus Componentes	18
2.2	Relés de Proteção e sua Aplicabilidade	19
2.3	Simulação HIL em Ambientes Acadêmicos	20
2.4	Microcontroladores e a Plataforma Arduino UNO	20
2.5	Conversores Eletrônicos Baseados em SPWM	21
3	METODOLOGIA	23
3.1	Introdução ao PSP-UFU	23
3.2	Relés de Proteção.	25
3.3	Interface.	26
3.3.1	Linguagem Python.	27
3.3.2	Arduino IDE.	27
4	EXPERIMENTOS E ANÁLISE DOS RESULTADOS	29
4.1	Projeto Conversor	33
4.2	Modelagem do Conversor	39
5	CONCLUSÃO	41
5.1	Principais Contribuições ao Trabalho	42

5.2	Trabalhos Futuros	42
	REFERÊNCIAS	43

Introdução

Partindo primordialmente como um projeto de Iniciação Científica (IC) no ano de 2023, sob orientação do Professor Doutor Thales Lima Oliveira, teve-se como projeto inicial o desenvolvimento de um sistema *Hardware-In-the-Loop* HIL com o objetivo de simular sistemas elétricos, integrando relés de proteção reais e comunicação serial. Para alcançar esse propósito, escolheu-se a Plataforma de Sistemas de Potência da Universidade Federal de Uberlândia PSP-UFU e o uso do Arduino UNO como interface entre o *software* e o *hardware*, no caso, o relé de proteção. O plano original previa o uso do protocolo de comunicação de dados denominado *Modbus* por meio das bibliotecas *Modbusino* (TRIPAULT, 2013) e *Modbus-Arduino* (BARBOSA, 2012). No entanto, durante o desenvolvimento, o protocolo adotado mostrou-se demasiado complexo, especialmente em relação à comunicação entre o Arduino e o computador, ocasionando obstáculos que demandaram ajustes.

Para cumprir os objetivos propostos, optou-se por utilizar a linguagem *Python* com o ambiente de desenvolvimento Visual Studio Code (VSC), juntamente com a biblioteca *pySerial*. Essa abordagem permitiu exportar dados do PSP-UFU para um arquivo CSV, que então era lido pelo programa em *Python* e transmitido para o Arduino via comunicação serial, simplificando o fluxo de dados.

Adicionalmente, o plano inicial previa a validação dos dados por meio de um *Liquid Crystal Display (LCD)* acoplado ao Arduino. Contudo, para viabilizar a replicação acadêmica do projeto e reduzir custos, decidimos exibir os valores diretamente no VSC, o que facilitou a visualização e análise dos resultados. Além disso, foi posteriormente projetado um conversor SPWM o qual permite que os dados adquiridos pelo Arduino sejam convertidos em um sinal de corrente perceptível ao relé de proteção.

Vale ressaltar que, para as fases iniciais do projeto, foi necessário adquirir uma placa Arduino UNO e realizar um estudo aprofundado de suas funcionalidades, de tal modo que o momento inicial sumariamente consistiu em um estudo dirigido. Para isso, foi obtido o Master Kit Arduino Black Board da Robocore (RoboCore, n.d.), que oferece um conteúdo abrangente com 48 tópicos de estudo, incluindo lógica de programação e

conceitos de Internet das Coisas (IOT).

Figura 1 – Master Kit Arduino Black Board da Robocore



Fonte: (RoboCore, n.d.)

1.1 Objetivos

1.1.1 Objetivos Gerais

O objetivo desse trabalho, que surgiu como uma IC, é o desenvolvimento de um sistema HIL para proteger sistemas elétricos de potência, utilizando ferramentas *open-source* como o simulador PSP-UFU. A plataforma permitiu testar ajustes de relés reais em simulações, servindo como ferramenta didática e de pesquisa para alunos e profissionais do setor elétrico.

1.1.2 Objetivos específicos

- ❑ Simulação da proteção de sistemas elétricos, através do *software* PSP-UFU integrado a um relé de proteção físico;

- ❑ Desenvolvimento de um código em *python* para aquisição de dados, tais como tensão, corrente e frequência advindas do PSP-UFU;
- ❑ Desenvolvimento de um código para Arduino, que permita-o adquirir e enviar dados entre o relé de proteção e o computador;
- ❑ Desenvolvimento de um conversor SPWM para integração do relé de proteção.

1.2 Metodologia do trabalho

Para desenvolvimento do trabalho, a metodologia adotada baseou-se inicialmente em um levantamento bibliográfico sobre sistemas HIL, relés de proteção e comunicação entre software e hardware, com foco na integração entre o simulador PSP-UFU e dispositivos físicos. Em seguida, foi realizado o estudo das funcionalidades do PSP-UFU, com ênfase nos módulos de fluxo de potência, análise de curto-circuito, estabilidade e interface gráfica.

A interface entre o PSP-UFU e o relé de proteção foi implementada utilizando a placa Arduino UNO, programada via *Arduino Integrated Development Environment* (IDE) em linguagem C/C++, e conectada ao ambiente de simulação por meio da linguagem Python, executada no VSC. A biblioteca `pySerial` foi utilizada para estabelecer a comunicação serial entre o Arduino e o computador. Os dados extraídos do PSP-UFU, como tensão, corrente e frequência, foram exportados em arquivos CSV e processados em Python, sendo posteriormente transmitidos ao Arduino.

Além disso, desenvolveu-se um conversor SPWM, projetado para transformar os sinais digitais em sinais de corrente alternada perceptíveis ao relé de proteção. O circuito foi simulado no software *LTSpice* e modelado no *KiCad* para a geração de sua placa de circuito impresso PCB.

Essa abordagem possibilitou a construção de um ambiente de simulação híbrido, com validação em tempo real do comportamento de relés físicos, contribuindo para aplicações didáticas e experimentais na área de proteção de sistemas elétricos.

1.3 Estrutura do trabalho

No capítulo 2 é descrita a fundamentação teórica deste trabalho, salientando o que é um Sistema Elétrico de Potência (SEP), o simulador do mesmo, no caso adotado, é o PSP-UFU. Além disso, são discutidos os principais componentes do SEP, o que são relés de proteção e sua aplicabilidade, bem como o que são sistemas HIL e o que são microcontroladores, especialmente, o Arduino.

No capítulo 3 é discutida a metodologia do trabalho, subdividindo em introdução ao PSP-UFU, o relé de proteção, a linguagem Python e o Arduino IDE.

No capítulo 4 são apresentados os resultados do protótipo simulado no *software* LTSpice e modelado no KiCad para a geração de sua placa de circuito impresso PCB.

Finalmente, no capítulo 5, as conclusões finais do TCC e sugestões de trabalhos futuros.

Fundamentação Teórica

Este trabalho trata do desenvolvimento de um sistema HIL aplicado a sistemas elétricos de potência, utilizando a integração entre o simulador PSP-UFU e um relé de proteção real, por meio de um microcontrolador Arduino UNO. Dessa forma, torna-se necessária uma contextualização dos sistemas elétricos de potência, seus equipamentos principais e o próprio simulador deste. Além disso, de forma complementar, faz-se necessário apresentar conceitos sobre microcontroladores e conversores baseados em SPWM.

2.1 Sistemas Elétricos de Potência

Os SEP compreendem o conjunto de instalações e equipamentos responsáveis pela geração, transmissão, distribuição e utilização da energia elétrica. Como destaca (BICHELS, 2018), a crescente complexidade desses sistemas exige metodologias robustas de análise e controle, garantindo o fornecimento contínuo, seguro e eficiente de energia. Com o passar do tempo, os SEP evoluíram de redes isoladas para sistemas interligados em escala nacional e internacional, promovendo maior confiabilidade e aproveitamento energético (Operador Nacional do Sistema Elétrico, 2017). Mello (MELLO, 2006) observa que a automação e digitalização são tendências consolidadas, que contribuem significativamente para a supervisão, controle e confiabilidade dos SEP.

Os SEP se organizam em quatro subsistemas principais:

- ❑ **Geração:** produção de energia elétrica a partir de fontes como hidrelétrica, térmica, solar e eólica;
- ❑ **Transmissão:** transporte de energia em alta tensão a longas distâncias;
- ❑ **Distribuição:** redução da tensão e entrega ao consumidor final;
- ❑ **Carga:** consumo de energia por setores residencial, industrial e comercial.

A operação dos SEP requer análises fundamentais como fluxo de potência, estudos de curto-circuito, estabilidade transitória e permanente, compensação reativa e análise de contingência (BICHELS, 2018).

A modernização dos SEP incorpora o conceito de *smart grids*, que integra *Intelligent Electronic Devices (IEDs)*, medidores inteligentes, redes de comunicação e sistemas *Supervisory Control and Data Acquisition (SCADA)*, possibilitando maior eficiência energética e resposta à demanda. A conectividade é impulsionada por tecnologias como IOT e redes 5G. A cibersegurança, nesse contexto, torna-se essencial para proteger os sistemas contra ameaças externas.

2.1.1 Simulador de Sistema Elétrico de Potência

O PSP-UFU é um software livre, multiplataforma e multilíngue, projetado para simulação, análise e visualização de sistemas elétricos. Possui interface gráfica amigável e ferramentas *Computer Aided Design (CAD)*, facilitando a construção de redes elétricas e sistemas de controle por meio de elementos visuais.

Entre suas principais funcionalidades, destacam-se:

- ❑ **Fluxo de potência:** análise do sistema em regime permanente;
- ❑ **Curto-circuito:** cálculo das correntes de curto-circuitos para diferentes tipos de faltas ou falhas;
- ❑ **Análise harmônica:** estudo da presença de harmônicos, relevante para a qualidade da energia;
- ❑ **Estabilidade transitória e dinâmica:** simulação da resposta do sistema frente a distúrbios.

Os elementos como barramentos, geradores, transformadores, linhas, cargas, capacitores, reatores e motores podem ser inseridos com parâmetros elétricos detalhados. Isso permite a modelagem precisa do sistema e torna o PSP-UFU uma ferramenta acessível e eficaz tanto para fins acadêmicos quanto profissionais.

2.1.2 Subestações Elétricas e seus Componentes

As subestações são instalações essenciais no SEP, responsáveis pela transformação dos níveis de tensão, manobra e proteção dos circuitos. Elas podem ser classificadas por função (transformadoras, de manobra, seccionadoras) e por estrutura (ao tempo, abrigadas ou blindadas – *Gas-Insulated Substation (GIS)*).

Seus componentes principais incluem:

- ❑ **Transformadores de potência;**

- ❑ Transformadores de corrente (TC) e Transformadores de potencial (TP);
- ❑ Disjuntores;
- ❑ Para-raios;
- ❑ Chaves seccionadoras;
- ❑ Barramentos.

Com os avanços tecnológicos, dispositivos como os IEDs tornaram-se fundamentais para a proteção e automação de subestações. Utilizando protocolos como *International Electrotechnical Commission (IEC) 61850*, esses equipamentos proporcionam comunicação eficiente e interoperabilidade. Sistemas SCADA também permitem supervisão remota, coleta de dados e automação de comandos (MELLO, 2006).

A tendência atual é a adoção de arquiteturas com barramentos de processo e de estação, que descentralizam a inteligência e facilitam a manutenção. Comunicação por fibra óptica e sincronização por *Global Position System (GPS)* também estão sendo amplamente adotadas. Além disso, a manutenção baseada em condição, *Condition Based Maintenance (CBM)*, permite prever falhas por meio de sensores e algoritmos, aumentando a disponibilidade e confiabilidade das instalações.

2.2 Relés de Proteção e sua Aplicabilidade

Os relés de proteção são dispositivos essenciais que monitoram continuamente as condições do sistema elétrico e atuam automaticamente em caso de anomalias, como sobre-corrente, subtensão, variações de frequência ou curto-circuito. Seu objetivo é garantir a seletividade, rapidez e confiabilidade no isolamento de falhas.

Inicialmente eletromecânicos, os relés evoluíram para versões digitais ou microprocessadas, que oferecem precisão, flexibilidade, registro de eventos e integração com sistemas SCADA (BLACKBURN, 2006). Eles operam com protocolos como Modbus, DNP3 e IEC 61850, sendo configuráveis via software.

Além da proteção de sobrecorrente, os relés também realizam funções diferenciais, utilizadas em transformadores, linhas e barras, detectando desequilíbrios entre correntes de entrada e saída.

Cotosck (COTOSCK, 2007) destaca a importância do uso de relés reais em ambiente acadêmico, favorecendo a compreensão prática e alinhando o ensino às demandas do mercado de trabalho. A integração com plataformas digitais amplia o potencial didático e profissional desses dispositivos.

2.3 Simulação HIL em Ambientes Acadêmicos

A simulação HIL integra modelos computacionais e dispositivos físicos reais em tempo real, permitindo testes seguros, repetitivos e eficientes. Essa abordagem tem-se mostrado altamente eficaz no ensino e na pesquisa em Engenharia Elétrica.

Com HIL, é possível simular falhas e analisar a resposta de relés de proteção, controladores e outros dispositivos. O uso do PSP-UFU em conjunto com o Arduino, como feito neste trabalho, permite simulações didáticas de baixo custo e grande valor educacional.

Chen et al. (CHEN; LI et al., 2013) defendem a eficácia do HIL no ensino de conceitos complexos. Glover e Sarma (GLOVER; SARMA, 1994) ressaltam a importância de experiências práticas na formação de engenheiros. Mello (MELLO, 2006) reforça que, em subestações digitais, o HIL pode ser usado para testar comunicação entre dispositivos, protocolos industriais e diagnósticos automatizados, fortalecendo a formação sistêmica do engenheiro eletricista.

Portanto, o uso de simulação HIL em ambientes acadêmicos amplia a aprendizagem e estimula o desenvolvimento de soluções inovadoras em automação, proteção e controle de sistemas elétricos de potência.

2.4 Microcontroladores e a Plataforma Arduino UNO

Microcontroladores são circuitos integrados capazes de realizar operações lógicas, aritméticas e de controle em sistemas embarcados. Eles integram em um único chip uma unidade central de processamento, *Central Processing Unit (CPU)*, memória (*Random-Acess Memory (RAM)* e *Read-Only Memory (ROM)* / Flash), temporizadores e interfaces de entrada/saída. Sua principal aplicação é no controle de sistemas automáticos em áreas como automação industrial, eletrodomésticos, instrumentação médica e sistemas embarcados veiculares.

Dentre as plataformas mais populares baseadas em microcontroladores está o **Arduino UNO**, amplamente utilizado em ambientes educacionais e de prototipagem. O Arduino UNO é baseado no microcontrolador **ATmega328P**, desenvolvido pela empresa Atmel (atualmente parte da Microchip Technology). Este microcontrolador possui as seguintes características:

- ❑ Arquitetura *Automatic Voltage Regulator (AVR)* de 8 bits;
- ❑ *Clock* de 16 MHz;
- ❑ 32 KB de memória flash, 2 KB de *Static Random-Acess Memory (SRAM)* e 1 KB de *Electrically-Erasable Programmable Read-Only Memory (EEPROM)*;
- ❑ 14 pinos digitais (6 com saída *Pulse Width Modulation (PWM)*);

- ❑ 6 entradas analógicas;
- ❑ Interfaces *Serial Peripheral Interface (SPI)*, *Inter-Integrated Circuit (I2C)* e *Universal Asynchronous Receiver/Transmitter (UART)* integradas.

A placa Arduino UNO possui um conversor USB-serial que facilita a comunicação com o computador para gravação e depuração dos códigos. Utiliza o *Arduino IDE*, ambiente de desenvolvimento simples e intuitivo, baseado em C/C++, o que contribui para sua adoção em projetos educacionais e de pesquisa.

Outro microcontrolador amplamente estudado e utilizado em aplicações didáticas e industriais é o **ATmega32**, também pertencente à família AVR da Atmel. Este microcontrolador compartilha muitas das características do ATmega328P, mas oferece:

- ❑ 32 KB de memória flash;
- ❑ 2 KB de SRAM;
- ❑ 1 KB de EEPROM;
- ❑ 32 pinos de I/O programáveis;
- ❑ Temporizadores de 8 e 16 bits;
- ❑ Conversor Analógico-Digital (A/D) de 10 bits com 8 canais.

O microcontrolador ATmega32 é comumente empregado em placas de desenvolvimento independentes ou em projetos que não exigem a padronização da plataforma Arduino. Ele é especialmente indicado para aplicações que demandam controle mais direto sobre os registradores e recursos do hardware, possibilitando a programação embarcada em baixo nível com maior flexibilidade e eficiência.

Sua utilização é amplamente documentada na literatura (MARGOLIS, 2011; MONK, 2016; Microchip Technology, 2021; Atmel Corporation, 2006), e seu custo acessível, aliado à facilidade de programação e à vasta comunidade de usuários, torna-o uma excelente opção para aplicações acadêmicas, protótipos e projetos de sistemas embarcados em geral.

2.5 Conversores Eletrônicos Baseados em SPWM

Os conversores eletrônicos são dispositivos fundamentais na conversão de energia elétrica em diferentes formas de tensão, corrente ou frequência, sendo amplamente utilizados em sistemas de acionamento, geração distribuída, sistemas fotovoltaicos e no controle de motores. Um dos métodos mais eficazes para controle de conversores é a Modulação por SPWM.

A técnica SPWM é aplicada principalmente em inversores, que convertem Corrente Contínua (CC) em Corrente Alternada (CA). Nessa modulação, o sinal senoidal de referência é comparado com uma portadora triangular de alta frequência. Os pulsos gerados comparam os níveis instantâneos dos sinais, resultando em uma forma de onda que, ao ser filtrada, se aproxima de uma senoide, permitindo controle preciso da amplitude, frequência e forma da tensão de saída (KAZMIERKOWSKI; KRISHNAN; BLAABJERG, 2002; RASHID, 2010; HOLMES; LIEW; MCGRATH, 2003).

Entre as principais vantagens da modulação SPWM destacam-se:

- ❑ Redução da distorção harmônica na forma de onda de saída;
- ❑ Facilidade de implementação digital, especialmente com microcontroladores e *Digital Signal Processor* (DSP);
- ❑ Controle eficiente da tensão RMS de saída;
- ❑ Compatibilidade com filtros de saída passivos, reduzindo perdas.

Conversores baseados em SPWM são empregados em diversas aplicações de engenharia elétrica, incluindo:

- ❑ Sistemas fotovoltaicos com inversores conectados à rede;
- ❑ Acionamento de motores de indução e motores *brushless*;
- ❑ Fontes de alimentação chaveadas e *no-breaks*;
- ❑ Sistemas de compensação ativa de energia reativa.

No contexto de sistemas elétricos modernos, a utilização de conversores com SPWM tem-se tornado indispensável pela sua eficiência, flexibilidade e compatibilidade com sistemas digitais. A integração desses conversores com microcontroladores, como o Arduino, permite a geração de sinais modulados com precisão, favorecendo aplicações acadêmicas e protótipos de baixo custo.

Além disso, o SPWM contribui para o controle de qualidade da energia elétrica, sendo uma técnica relevante para estudos envolvendo harmônicos, interferência eletromagnética, *Electromagnetic Interference (EMI)*, e análise de desempenho de sistemas de potência em ambientes laboratoriais e industriais.

Metodologia

A metodologia inclui levantamento bibliográfico sobre soluções HIL e linguagem Python, seguido de um estudo detalhado do PSP-UFU. Ferramentas para comunicação entre o simulador e microcontrolador (ATMega 32 presente no Arduino UNO) serão desenvolvidas em Python e C++, com uso de bibliotecas *open-source*.

3.1 Introdução ao PSP-UFU

Conforme descrito em (OLIVEIRA, 2024), o PSP-UFU é um *software* multiplataforma, multilíngue, livre e de código aberto (*Free and Open Source Software (FOSS)*), que oferece recursos avançados de *Graphical User Interface (GUI)* e ferramentas CAD para a realização de estudos em sistemas elétricos de potência.

O programa permite a construção de redes de transmissão elétrica e sistemas de controle por meio da inserção de elementos visuais. Para a visualização dos resultados, o *software* disponibiliza elementos de texto integrados à tela principal, bem como editores de tabelas e gráficos.

O principal objetivo do PSP-UFU é fornecer ferramentas eficientes de simulação para fins acadêmicos e de pesquisa, além de atender a aplicações industriais na área de sistemas elétricos de potência.

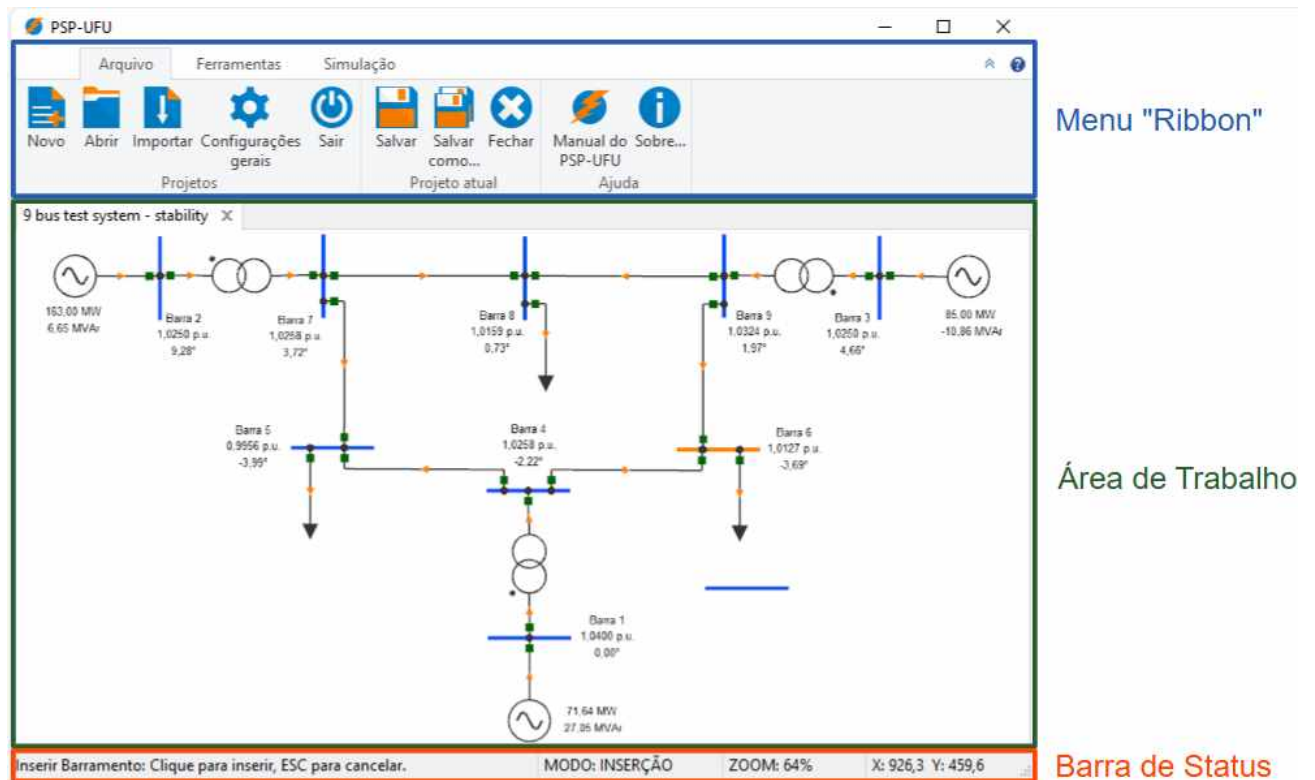
Entre suas funcionalidades, destacam-se:

- ❑ **Fluxo de potência:** cálculo do fluxo de carga em redes elétricas, permitindo a análise do comportamento do sistema em regime permanente.
- ❑ **Cálculo de curto-circuito:** avaliação das correntes de falta em diferentes tipos de curtos-circuitos, auxiliando no dimensionamento de dispositivos de proteção.
- ❑ **Análise de harmônicos:** estudo dos componentes harmônicos presentes no sistema, fundamental para garantir a qualidade da energia elétrica.

- ❑ **Estabilidade transitória e dinâmica:** simulação da resposta do sistema frente a perturbações, como variações de carga ou falhas, possibilitando a análise do comportamento dinâmico dos componentes.

A interface gráfica do PSP-UFU foi projetada para ser intuitiva e de fácil utilização, permitindo a construção de redes elétricas por meio de elementos gráficos. Os recursos CAD integrados viabilizam a inserção e edição de componentes como barramentos, geradores, transformadores, linhas de transmissão, cargas, capacitores, reatores, motores de indução, entre outros. Cada componente conta com formulários específicos para a inserção de dados elétricos e operacionais, o que facilita a modelagem precisa dos sistemas.

Figura 2 – Interface Gráfica do PSP-UFU



Fonte: (OLIVEIRA, 2024)

O PSP-UFU foi desenvolvido em linguagem C++, escolhida por sua eficiência e suporte à programação orientada a objetos. Para garantir a compatibilidade multiplataforma, utilizou-se o *framework wxWidgets*, que possibilita a criação de interfaces gráficas para diversos sistemas operacionais. Sua arquitetura modular facilita tanto a manutenção quanto a expansão das funcionalidades.

Esse software é amplamente adotado em instituições acadêmicas como ferramenta didática para o ensino de sistemas elétricos de potência, oferecendo uma plataforma prática para simulações e análises. Ademais, suas funcionalidades também atendem às demandas da indústria, permitindo a análise de redes elétricas reais e a simulação de diferentes cenários operacionais.

A Plataforma de Sistemas de Potência da Universidade Federal de Uberlândia (PSP-UFU) representa uma ferramenta poderosa e acessível para a análise e simulação de sistemas elétricos. Sua natureza de código aberto, aliada a uma interface gráfica robusta e funcionalidades avançadas, torna-a uma escolha ideal para aplicações tanto educacionais quanto profissionais no campo da engenharia elétrica.

3.2 Relés de Proteção.

Conforme descrito em (Token Engenharia, 2023), os relés de proteção são dispositivos indispensáveis na operação de sistemas elétricos, uma vez que realizam o monitoramento contínuo de parâmetros como corrente, tensão e frequência. Sua principal função é detectar condições operacionais anormais e, quando necessário, emitir comandos para isolar partes defeituosas do sistema, garantindo a integridade dos equipamentos e a confiabilidade do fornecimento de energia.

Figura 3 – Modelo de um relé de proteção



Fonte: (Token Engenharia, 2023)

Dentre as funções essenciais desses dispositivos, destacam-se:

- ❑ **Deteção de falhas:** Os relés são responsáveis por identificar eventos como sobrecorrente, sobretensão ou outros desvios significativos nos parâmetros elétricos. Quando essas ocorrências são detectadas, o sistema é imediatamente preparado para resposta.

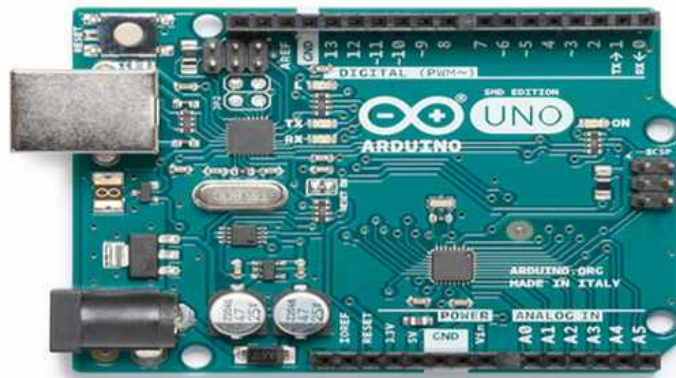
- ❑ **Atuação corretiva:** Após a identificação de uma falha, o relé envia sinais a dispositivos de manobra, como disjuntores, com o intuito de interromper a parte afetada da rede. Essa ação previne a propagação de danos e protege os demais componentes.
- ❑ **Classificação diversificada:** Há uma variedade de relés, cada um com uma aplicação específica. Entre os mais comuns, encontram-se os relés de sobrecorrente, de sobretensão, diferenciais e de proteção de distância.
- ❑ **Avanço tecnológico:** Originalmente eletromecânicos, os relés evoluíram significativamente. Atualmente, versões digitais e microprocessadas são preferidas devido à sua maior precisão, flexibilidade e recursos de automação.
- ❑ **Configuração personalizada:** Os relés modernos permitem ajustes finos de seus parâmetros operacionais, tornando-se adaptáveis a diferentes perfis de rede elétrica.
- ❑ **Integração com sistemas de automação:** Muitos modelos atuais oferecem recursos de comunicação com sistemas supervisórios, possibilitando o envio de informações em tempo real e uma integração eficaz com arquiteturas SCADA e de controle remoto.

Em síntese, os relés de proteção são fundamentais para a segurança e estabilidade das instalações elétricas, atuando como uma barreira preventiva contra falhas e contribuindo para a continuidade do serviço e a preservação dos ativos do sistema.

3.3 Interface.

Quanto à interface, aí tem-se o ponto no qual debruça-se o TCC, isto é, o mediador entre o *software* PSP-UFU e o *hardware* relé de proteção. Em questão, tal interface consiste na junção entre uma linguagem de programação, o Python, e o Arduino UNO.

Em síntese, o Python será responsável por adquirir os dados (tais como amplitude, frequência e *offset* de sinais de corrente e tensão) do PSP-UFU e transmiti-los ao Relé de Proteção, que por sua vez enviará sinais para atuação de disjuntores dentro do PSP-UFU. Por sua vez, o Arduino UNO será responsável por mediar os dados adquiridos tanto por parte do Python quanto do Relé, devidamente tratá-los e transmiti-los via comunicação Serial.

Figura 4 – Placa Arduino modelo UNO

Fonte: (Arduino, n.d.)

3.3.1 Linguagem Python.

Python é uma linguagem de programação de alto nível, amplamente utilizada por sua simplicidade e legibilidade. Foi criada por Guido van Rossum e lançada pela primeira vez em 1991. Python é uma linguagem de propósito geral, o que significa que pode ser utilizada para uma variedade de aplicações, desde desenvolvimento *web* até ciência de dados, aprendizado de máquina e automação de tarefas.

Algumas características que tornam Python popular incluem:

1. Sintaxe clara: Facilita a leitura e escrita de código.
2. Grande comunidade: Uma vasta quantidade de bibliotecas e frameworks disponíveis, como Django, Flask, NumPy, PySerial e pandas.
3. Multiplataforma: Funciona em diversos sistemas operacionais, como Windows, macOS e Linux.
4. Versatilidade: Suporta múltiplos paradigmas de programação, incluindo programação orientada a objetos, funcional e procedural.

3.3.2 Arduino IDE.

Conforme descrito em (Arduino, n.d.) (editado e traduzido):

"O Arduino é uma plataforma eletrônica de prototipagem de código aberto baseada em hardware e software fáceis de usar. As placas Arduino são capazes de ler entradas - luz em um sensor, um dedo em um botão ou uma mensagem do Twitter - e transformá-las em uma saída - ativando um motor, acendendo um LED, publicando algo online. Você pode dizer à sua placa o que fazer enviando um conjunto de instruções para o microcontrolador na placa. Para fazer isso, você usa a linguagem de

programação Arduino (baseada em Wiring) e o Software Arduino (IDE), baseado em Processing."

Por sua vez, Arduino IDE é um *software* usado para programar placas Arduino, amplamente utilizadas em projetos de *hardware*. O IDE fornece uma interface simples onde você pode escrever, compilar e carregar código (chamado de "*sketches*") nas placas Arduino.

Algumas características do Arduino IDE incluem:

- ❑ Código em C/C++: O ambiente usa uma versão simplificada de C/C++, permitindo que você escreva código de maneira fácil.
- ❑ Bibliotecas: Inclui uma série de bibliotecas que facilitam a programação e o controle de diversos componentes eletrônicos, como sensores, motores e *displays*.
- ❑ Compilação e *Upload*: O IDE compila o código e o carrega diretamente na placa Arduino via USB.
- ❑ Interface Simples: Possui uma interface intuitiva, com recursos como realce de sintaxe e verificação de erros.
- ❑ Compatibilidade: Suporta uma variedade de placas Arduino e até outras placas compatíveis.

Experimentos e Análise dos Resultados

No desenvolvimento deste TCC, obteve-se, por fim, uma junção entre dois códigos, sendo o primeiro em Python através da plataforma Visual Studio Code e o segundo no próprio Arduino IDE. Dito isto, segue o código em Python:

```
1  import serial
2  import time
3  import pandas as pd
4
5  # Lê os valores de um arquivo CSV
6  # Altere para o caminho correto do seu arquivo
7  # Lê o CSV sem cabeçalho
8  df = pd.read_csv('teste.csv', header=None)
9  # Converte a primeira linha em uma lista
10 float_values = df.iloc[0].tolist()
11 print(float_values)
12
13 # Configura a conexão serial
14 # Altere 'COM5' para a porta do seu Arduino
15 arduino = serial.Serial('COM5', 9600)
16 # Aguarda a conexão ser estabelecida
17 time.sleep(2)
18
19 # Envia as variáveis para o Arduino
20 for value in float_values:
21     # Envia o float como string seguida de nova linha
22     arduino.write(f"{value}\n".encode())
23     # Aguarda um pouco entre os envios
24     time.sleep(0.1)
25
26 # Aguarda a leitura dos dados enviados de volta
27 received_values = []
28 while len(received_values) < 3:
29     # Verifica se há dados disponíveis
30     if arduino.in_waiting > 0:
```

```
31     line = arduino.readline().decode('utf-8').strip() # Lê a linha recebida
32     received_values.append(line) # Adiciona a linha à lista
33
34 # Exibe os valores recebidos
35 for value in received_values:
36     print(value)
37
38 # Fecha a conexão
39 arduino.close()
```

Neste código vale ressaltar a importância de duas bibliotecas, a primeira é a Pandas, através dessa é possível acessar o arquivo CSV (nomeado teste.csv) produzido pelo PSP-UFU e devidamente convertê-lo em uma lista de valores *float* que são transmitidos para o Arduino. Outra biblioteca cerne deste código é a Pyserial, esta é fundamental para que haja a comunicação entre o computador (Python) e o Arduino UNO (Arduino IDE).

Isto posto, segue uma explicação detalhada do código Python:

1. Importação de Bibliotecas:

- ❑ O código começa importando as bibliotecas necessárias: `serial`, `time` e `pandas`. A biblioteca `serial` é utilizada para comunicação com o Arduino, `time` para gerenciar temporizações e `pandas` para manipulação de dados em formato CSV.

2. Leitura do Arquivo CSV:

- ❑ O arquivo teste.csv é lido utilizando a função `pd.read_csv`, que carrega os dados em um `DataFrame`. O parâmetro `header=None` indica que o arquivo não contém cabeçalhos.
- ❑ Os valores da primeira linha do CSV são convertidos em uma lista através de `df.iloc[0].tolist()`. Essa lista, chamada `float_values`, é impressa no console para verificação.

3. Configuração da Conexão Serial:

- ❑ A conexão com o Arduino é estabelecida através de `serial.Serial('COM5', 9600)`, onde `'COM5'` deve ser substituído pela porta correta utilizada pelo Arduino. A taxa de transmissão é configurada para 9600 bps.
- ❑ Um pequeno atraso de 2 segundos é introduzido com `time.sleep(2)` para garantir que a conexão esteja estabelecida antes de começar a enviar dados.

4. Envio dos Valores para o Arduino:

- ❑ Um loop percorre a lista `float_values` e envia cada valor para o Arduino. Cada valor é convertido em `string` e enviado seguido de uma nova linha (`\n`), utilizando `arduino.write()`.

- ❑ Um pequeno atraso de 0,1 segundos é adicionado entre os envios para evitar sobrecarga na comunicação.

5. Recepção de Dados do Arduino:

- ❑ O código aguarda a leitura dos dados enviados de volta pelo Arduino. Um loop é utilizado para coletar até três linhas de dados.
- ❑ Os dados recebidos são lidos utilizando `arduino.readline()` e decodificados de *bytes* para *string*. As linhas são armazenadas em uma lista chamada `received_values`.

6. Exibição dos Valores Recebidos:

- ❑ Após a coleta, os valores recebidos são impressos no console.

7. Fechamento da Conexão:

- ❑ Finalmente, a conexão serial com o Arduino é encerrada usando `arduino.close()`.

De modo análogo tem-se o respectivo código do Arduino IDE:

```
1  float freq;    // Variável para armazenar a frequência
2  float amp;     // Variável para armazenar a amplitude
3  float offset;  // Variável para armazenar o offset
4
5  void setup() {
6      Serial.begin(9600); // Inicia a comunicação serial a 9600 bps
7  }
8
9  void loop() {
10     // Verifica se há dados disponíveis
11     if (Serial.available() > 0) {
12         // Lê três floats da comunicação serial
13         freq = Serial.parseFloat(); // Lê o primeiro float (frequência)
14         while (Serial.read() != '\n'); // Aguarda a nova linha
15
16         amp = Serial.parseFloat(); // Lê o segundo float (amplitude)
17         while (Serial.read() != '\n'); // Aguarda a nova linha
18
19         offset = Serial.parseFloat(); // Lê o terceiro float (offset)
20         while (Serial.read() != '\n'); // Aguarda a nova linha
21
22         // Envia os valores recebidos de volta
23         Serial.print("Freq: "); Serial.println(freq);
24         Serial.print("Amp: "); Serial.println(amp);
25         Serial.print("Offset: "); Serial.println(offset);
26     }
27 }
```

1. Declaração de Variáveis:

O código começa com a declaração de três variáveis do tipo **float**:

- ❑ **freq**: utilizada para armazenar a frequência.
- ❑ **amp**: utilizada para armazenar a amplitude.
- ❑ **offset**: utilizada para armazenar o offset.

2. Função `setup()`:

A função `setup()` é executada uma vez quando o Arduino é inicializado. Nela, a comunicação serial é iniciada com `Serial.begin(9600)`, configurando a taxa de transmissão para 9600 bps. Isso permite que o Arduino se comunique com outros dispositivos, como um computador.

3. Função `loop()`:

A função `loop()` é chamada repetidamente enquanto o Arduino está em funcionamento. A primeira ação dentro dessa função é verificar se há dados disponíveis na comunicação serial com `Serial.available() > 0`. Se houver, o código prossegue para ler os valores.

4. Leitura dos Valores:

Frequência: O código usa `Serial.parseFloat()` para ler o primeiro valor **float**, que é armazenado na variável **freq**. Após a leitura, um loop aguarda até que um caractere de nova linha (`'\n'`) seja lido, garantindo que a leitura do próximo valor comece somente após a confirmação da nova linha. O mesmo processo é repetido para ler a **amp** (amplitude) e o **offset**, garantindo que cada valor seja lido e separado corretamente.

5. Envio dos Valores de retorno:

Depois que os três valores foram recebidos, o código envia esses valores de volta ao dispositivo que os enviou (como um computador) usando `Serial.print()` e `Serial.println()`. Cada valor é precedido por uma descrição (por exemplo, "Freq: "), facilitando a identificação no lado do computador.

Por fim, é possível ver pelo lado do computador a resposta afirmativa confirmando a validação dos dados:

Figura 5 – Resposta enviada ao Visual Studio Code.

```
PS C:\Users\otavio\OneDrive\Área de Trabalho\IC\PyDuino> &
C:/Users/otavio/AppData/Local/Programs/Python/Python310/python.exe
"c:/Users/otavio/OneDrive/Área de Trabalho/IC/PyDuino/projeto_final.py"
[54.54, 22.13, 85.8]
Freq: 54.54
Amp: 22.13
Offset: 85.80
```

Fonte: Autor

4.1 Projeto Conversor

Em complementação ao projeto, idealizou-se um conversor que será utilizado na saída do Arduino UNO após um prévio tratamento dos dados enviados para este pelo PSP-UFU via Python. No caso em questão, os dados podem ser utilizados para a realização de um SPWM, que sucintamente é um PWM senoidal modulando a largura dos pulsos em uma onda de controle para gerar uma forma de onda senoidal. Isso é feito comparando uma onda senoidal de referência com uma onda portadora em forma de triângulo ou dente de serra.

Para utilização do SPWM foi desenvolvido o seguinte código:

```
1  #include <avr/io.h>
2  #include <avr/interrupt.h>
3
4  // 60Hz:
5  // 40 = 2,4 kHz
6  // 170 = 10,2 kHz
7  // 340 = 20,4 kHz
8  #define SinDivisions (170) // subdivisões da onda senoidal.
9
10 static int microMHz = 16; // frequencia do Micro clock
11 static int freq = 60;    // frequencia senoidal
12 static long int period;  // período do PWM em ciclos do relógio
13 volatile unsigned int lookUp[SinDivisions];
14 static unsigned int lookUpOrig[SinDivisions];
15
16 static float a0In = 0.0;
17 static float a1In = 0.0;
18
19 static float amp = 1.0;
20
21 void setup() {
22     Serial.begin(9600);
```

```

23  double temp;
24
25  period = microMHz*1e6/(freq * 2)/SinDivisions;// período do PWM em ciclos do relógio
26  //period = microMHz*1e6/freq;
27  //Serial.println(period);
28
29  for(int i = 0; i < SinDivisions; i++){ // Gerando a tabela de consulta
30      temp = (0.5+0.5*sin(i*2*M_PI/(SinDivisions)))*period;
31      lookupOrig[i] = (int)(temp+0.5);
32      lookup[i] = lookupOrig[i];
33  }
34
35  // Inicialização dos registradores:
36  // TCCR1A, pg. 108, tabela 15-3
37  TCCR1A = 0b10100010;
38  /*10 OC1A = conectado e limpa na subida e define na descida. (não inversor)
39   *10 OC1B = conectado e limpa na subida e define na descida. (não inversor)
40   *00
41   *10 WGM1 1:0
42   */
43  TCCR1B = 0b00010001;
44  /*000
45   *10 WGM1 3:2
46   *001 sem pré-escala (clkI/O / 1 = 16MHz)
47   */
48
49  /* Obs: WGM1 = 1010 (modo 10 -> pg. 109, tabela 15-5)
50   * | Mode | Mode of Operation | TOP | Update of OCR1x at | TOV1 Flag Set on |
51   * | 10 | PWM, phase correct | ICR1 | TOP | BOTTOM |
52   *
53   * TOV1 = Flag de overflow do timer
54   */
55
56  TIMSK1 = 0b00000001;
57  /*0000000
58   1 Interrupção ocorre quando TOV1 é definida
59   */
60  ICR1 = period; // Interrupção ocorre quando atinge ICR1 (número de cliques de clock)
61  sei(); // Habilita interrupção
62  pinMode(10,OUTPUT);
63  pinMode(9,OUTPUT);
64  pinMode(13,OUTPUT);
65  }
66
67  void loop() {
68      // coloque aqui o código principal para entrar em loop:
69      //digitalWrite(13, trig);
70
71      float a0InNew = analogRead(A0);

```

```

72  float a1InNew = analogRead(A1);
73  int tol = 5;
74  if(abs(a0InNew - a0In) > tol) {
75      a0In = a0InNew;
76      amp = a0InNew / 1023.0;
77
78      Serial.print("Amp = ");
79      Serial.print(100 * amp);
80      Serial.println("%");
81
82      //ICR1 = (int)(period*dc+0.5);
83      for(int i = 0; i < SinDivisions; i++){
84          lookUp[i] = (int)((lookUpOrig[i] - 0.5*period)*amp + 0.5*period);
85      }
86  }
87  if(abs(a1InNew - a1In) > tol) {
88      a1In = a1InNew;
89      float dc = ((a1InNew / 1023.0)*0.5 + 0.75);
90      if(dc < 0.5) dc = 0.5;
91      Serial.print("FreQ = ");
92      Serial.print(freq * dc);
93      Serial.println(" Hz");
94
95      double temp = 0;
96      period = microMHz*1e6/(freq * dc * 2.0)/SinDivisions;// período do PWM em ciclos do relógio
97
98      for(int i = 0; i < SinDivisions/2; i++){ // Gerando a tabela de consulta
99          temp = (0.5+amp*0.5*sin(i*2*M_PI/(SinDivisions)))*period;
100         lookUpOrig[i] = (int)(temp+0.5);
101         lookUp[i] = lookUpOrig[i];
102     }
103     ICR1 = period;
104 }
105 }
106
107 ISR(TIMER1_OVF_vect){
108     static int num;
109     // change duty-cycle every period.
110     OCR1A = lookUp[num];
111     OCR1B = lookUp[SinDivisions - num - 1];
112
113
114     if(num == 0) {
115         digitalWrite(13, HIGH);
116     } else {
117         digitalWrite(13, LOW);
118     }
119
120     num++;

```

```
121     if(num >= SinDivisions){  
122         num = 0; // Reset num  
123     }  
124  
125 }
```

Mediante a tal tratamento é possível fazer com que o Arduino atue sobre o inversor controlando as chaves presentes neste. Assim, segue a lista de materiais do conversor idealizado (Chave/Driver/Componentes necessários):

- ❑ 4x Mosfets IRFP4668
- ❑ 2x Driver Mosfet UCC27714 High-Side/Low-Side
- ❑ 2x Capacitores de Bootstrap de $220\mu F$
- ❑ 2x Resistores de $5,0\Omega$
- ❑ 2x Diodos rápidos UF4007

Resistores/Capacitores para o correto funcionamento do conversor:

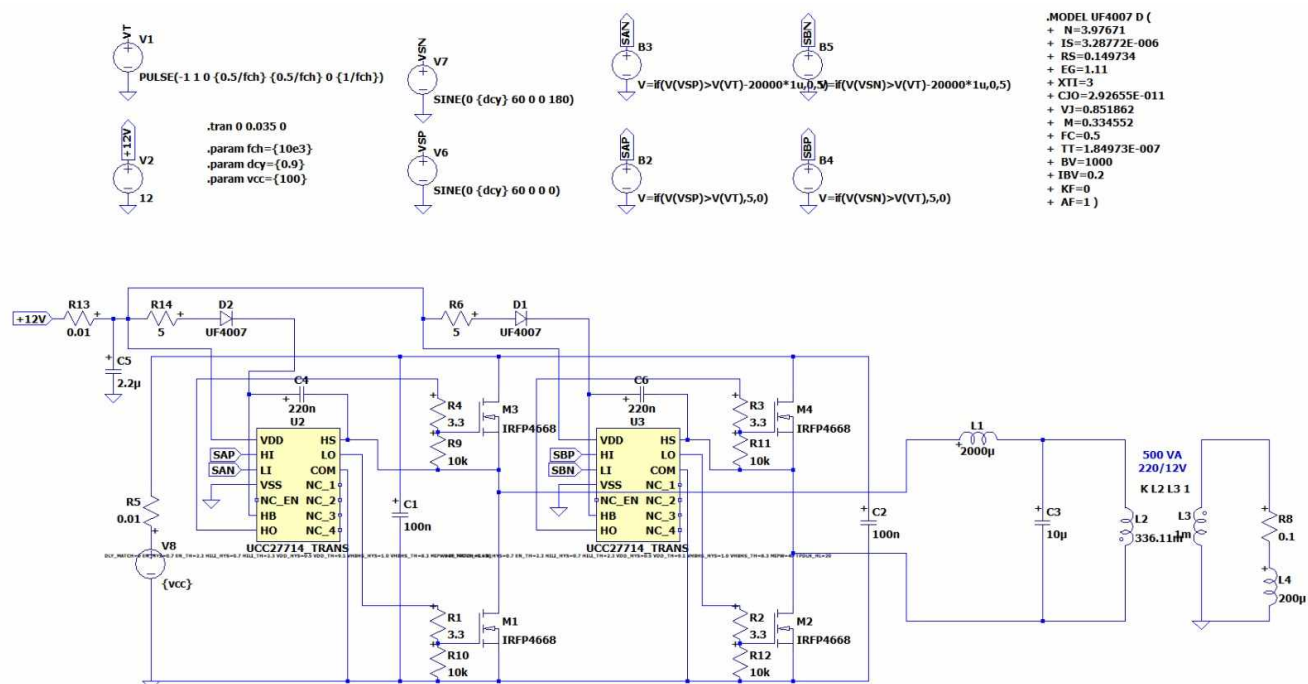
- ❑ 2x Capacitores de $100nF$
- ❑ 4x Resistores de $10,0k\Omega$
- ❑ 4x Resistores de $3,3\Omega$
- ❑ 1x Capacitor de $2,2\mu F$

Filtro:

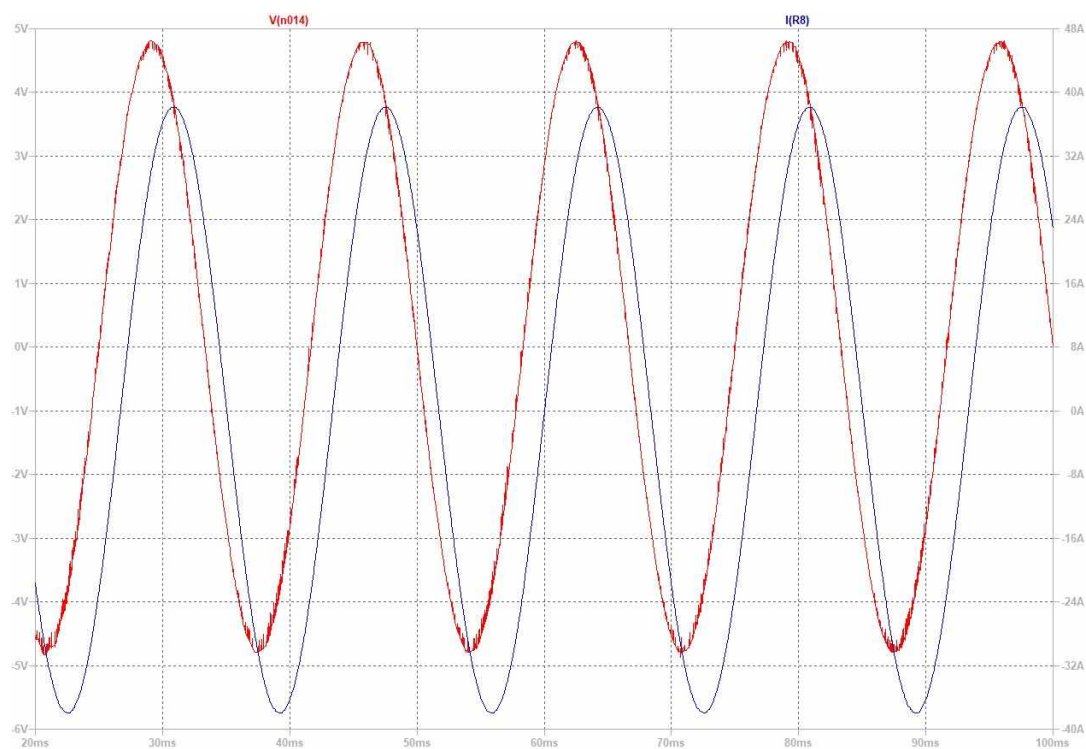
- ❑ 1x Indutor de $2000mH$
- ❑ 1x Capacitor de $10mF$ Transformador:
- ❑ 1x transformador 500VA 220/12V

Logo, para compreensão do projeto, segue uma simulação do conversor em três *duty cycle* (90%, 50% e 10%) ambas com uma frequência de chaveamento de 10kHz. Em sequência, tem-se as imagens das simulações realizadas no *software* LTspice:

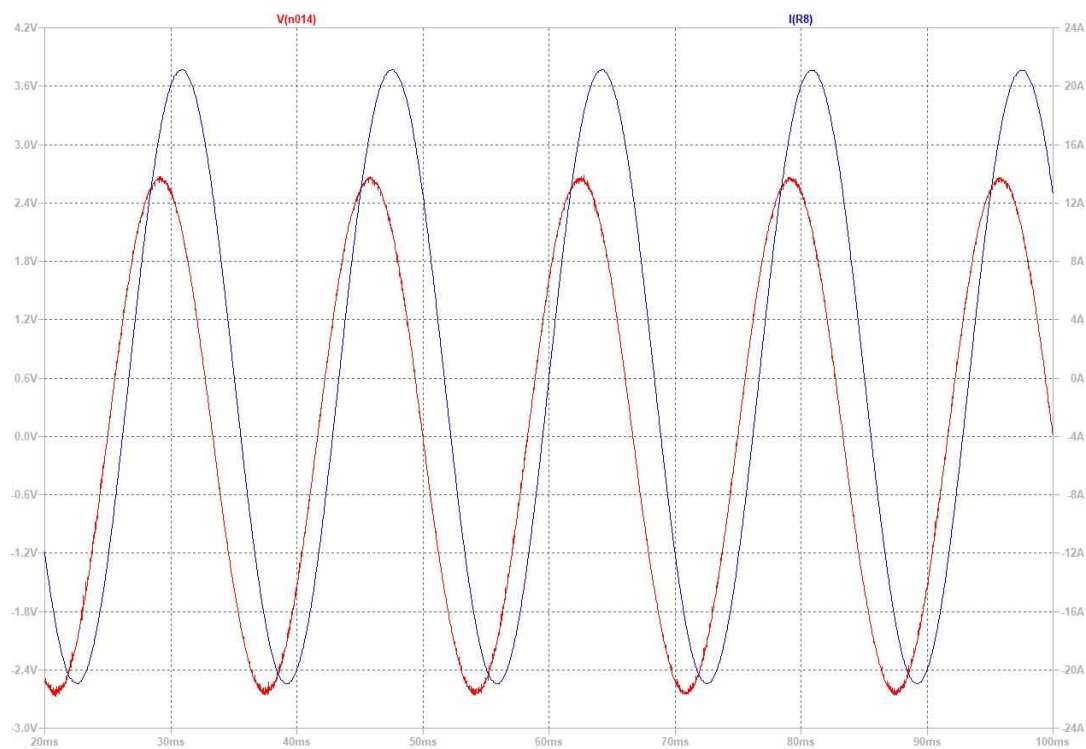
Figura 6 – Projeto do Inversor.



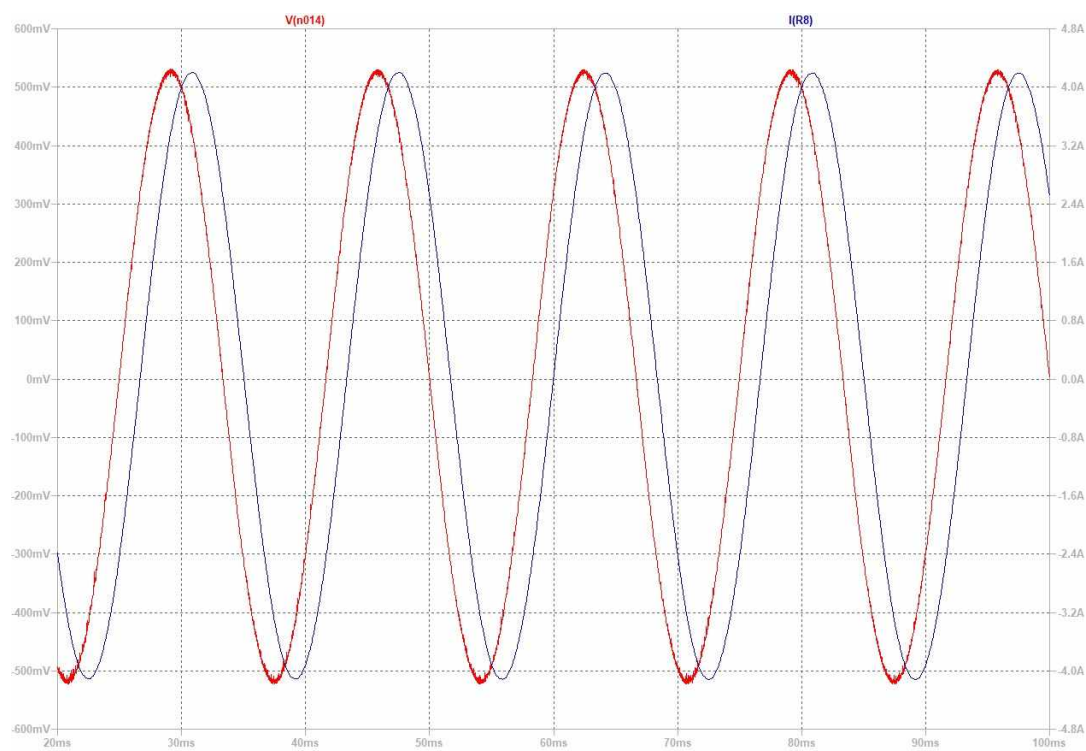
Fonte: Autor

Figura 7 – Grafico da Simulação de Tensão e Corrente para *duty cycle* de 90%.

Fonte: Autor

Figura 8 – Gráfico da Simulação de Tensão e Corrente para *duty cycle* de 50%.

Fonte: Autor

Figura 9 – Gráfico da Simulação de Tensão e Corrente para *duty cycle* de 10%.

Fonte: Autor

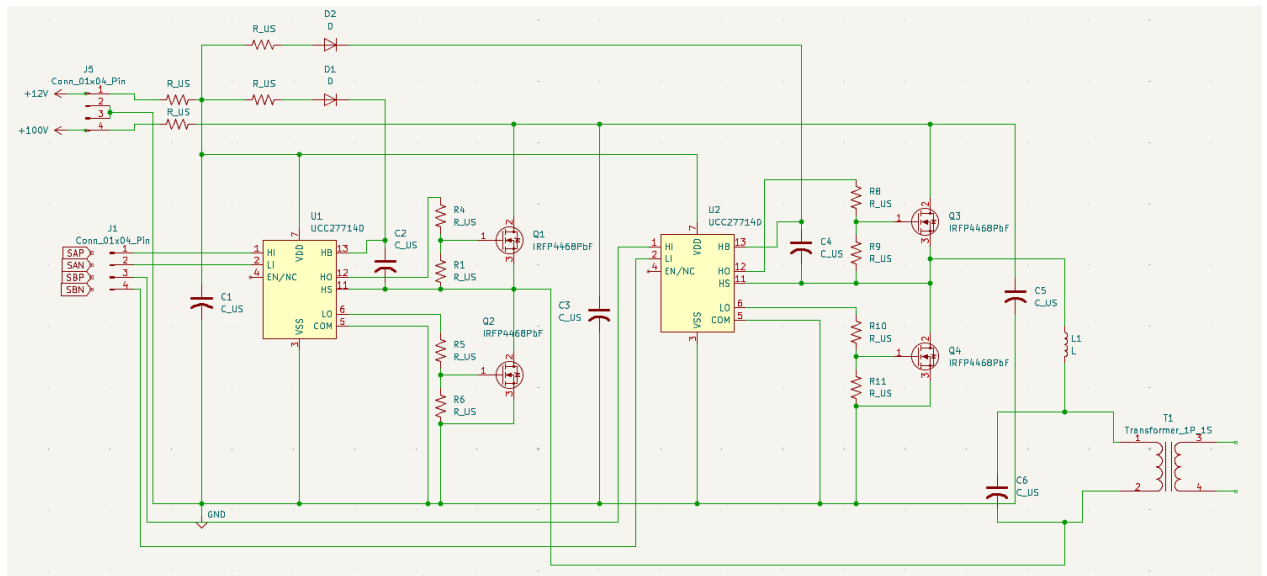
O objetivo desta simulação é que o conversor receba os dados de saída do Arduino e

os converta em um sinal de corrente alternada que simula um curto-circuito, resultando em uma corrente elevada. O ciclo de trabalho (*duty cycle*) determina a amplitude da corrente, conforme evidenciado nos gráficos apresentados. Além disso, é possível inserir qualquer valor de frequência no inversor, permitindo uma flexibilidade na configuração dos parâmetros de operação.

4.2 Modelagem do Conversor

Com o projeto final do conversor realizado e devidamente testado no *software* LTSpice, foi possível modelar o conversor através do *software* kiCad. Para tal, primeiramente é necessário que haja o desenvolvimento esquemático dos componentes de forma análoga à que foi realizada no LTSpice, porém sendo necessária a especificação de componentes comerciais.

Figura 10 – Desenho esquemático do Conversor SPWM

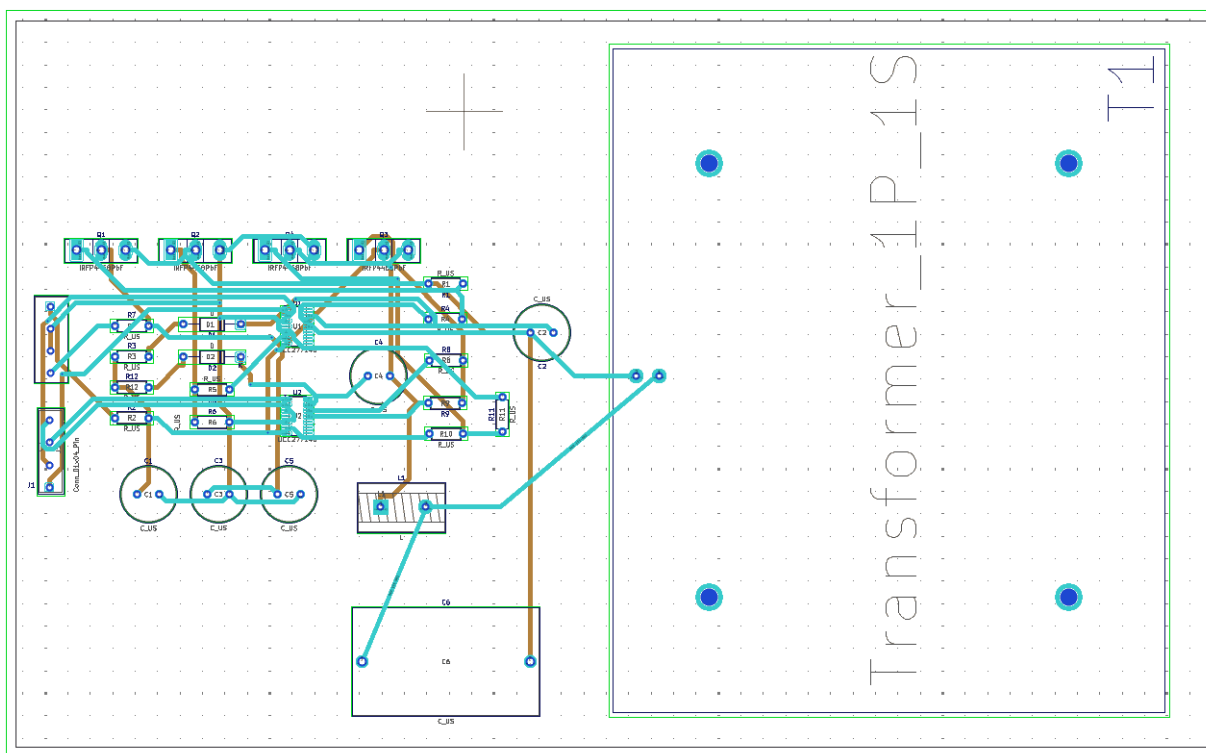


Fonte: Autor

Após a especificação do esquemático é possível a realização das trilhas do *PCB*. Vale ressaltar que, conforme observado em (PCB Brasil, 2025) e mediante testes simulados, tem-se que a corrente máxima da trilha é de 3,5 amperes de pico; portanto, a espessura mínima de uma trilha no circuito *PCB* é de 0,85 mm, de modo que projetou-se cada trilha com 1 mm de espessura por fator de segurança, conforme pode ser observado na figura 11 com marcações azuis (parte frontal da placa de circuito impresso) e marcações marrons (parte traseira da placa de circuito impresso).

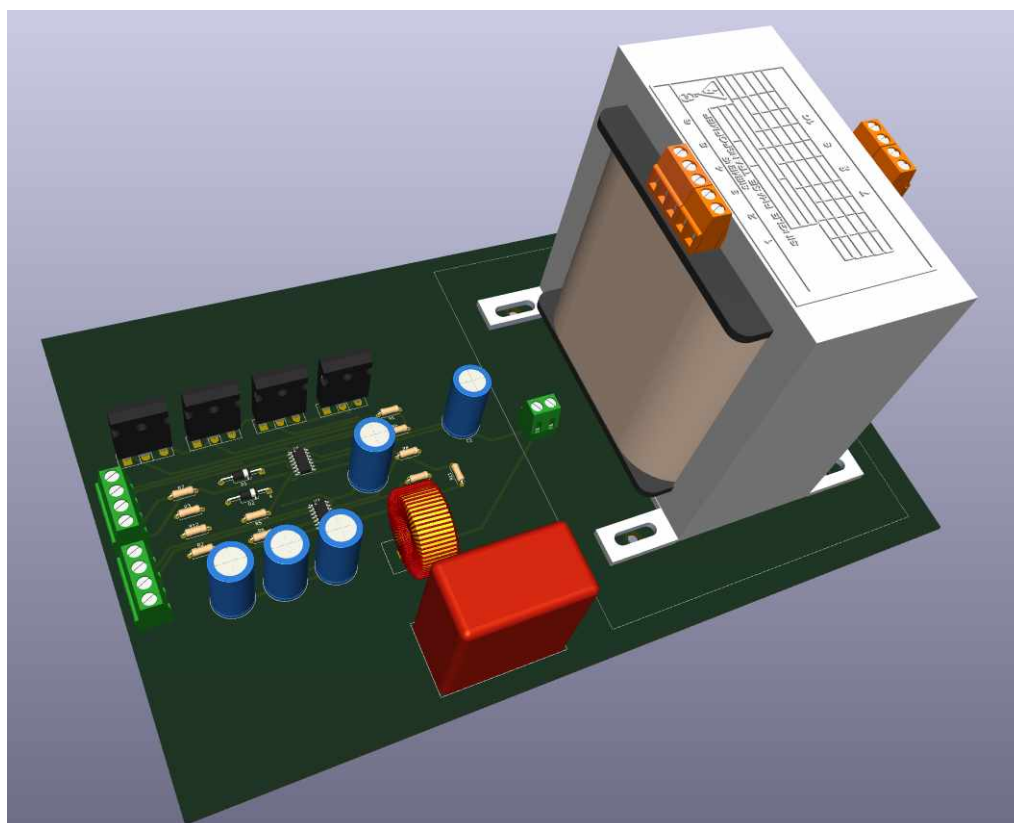
Com a realização destes circuitos, pode-se realizar a confecção do Conversor SPWM mediante a impressão do circuito. Isto posto, após a execução do arquivo de projeto em *PCB* pode-se realizar a modelagem em 3D do circuito para demonstração do conversor.

Figura 11 – Desenho PCB do Conversor SPWM



Fonte: Autor

Figura 12 – Desenho PCB do Conversor SPWM



Fonte: Autor

Conclusão

Os resultados da pesquisa demonstraram a eficácia do sistema HIL para simulações de relés, permitindo um ambiente seguro para testes sem a necessidade de uma rede elétrica real. Isso não apenas ampliou o conhecimento técnico dos participantes, mas também resultou na criação de uma ferramenta *open-source* amplamente utilizável por estudantes e profissionais da área elétrica.

O código desenvolvido foi disponibilizado no *GitHub* sob licença GPL, promovendo acesso livre e possibilidades de futuras adaptações. Esse projeto contribui para a formação de novos profissionais ao oferecer uma plataforma prática que conecta teoria e aplicação na proteção de sistemas elétricos.

O TCC se revelou um espaço valioso para o desenvolvimento de habilidades técnicas e práticas, além de gerar uma ferramenta que potencializa o aprendizado e a pesquisa em proteção de sistemas elétricos. A flexibilidade do sistema e sua natureza *open-source* oferecem um caminho promissor para estudos contínuos, beneficiando tanto a comunidade acadêmica quanto o setor elétrico.

Entre os principais aprendizados, destaca-se a compreensão mais profunda do microcontrolador AT Mega 32 por meio do Arduino IDE, bem como o desenvolvimento de habilidades em Python e o conhecimento do protocolo MODBUS, embora este último não tenha sido aplicado no produto final. Também é digno de nota o uso dos *softwares* LTSpice e Kicad na simulação e confecção de circuitos, tais como o conversor SPWM.

Em resumo, a pesquisa foi altamente proveitosa, promovendo aprendizado e o desenvolvimento de habilidades, além de criar uma ferramenta de baixo custo para o estudo de sistemas de proteção, especialmente relés. A possibilidade de conectar o relé diretamente ao Arduino, sem uma rede real, abre um leque de oportunidades para simulações variadas com o PSP-UFU.

5.1 Principais Contribuições ao Trabalho

A realização deste projeto foi primariamente possível devido à bolsa de fomento à pesquisa da Fundação de Amparo à Pesquisa do Estado de Minas Gerais (FAPEMIG), disponibilizada em caráter de IC o que permitiu, entre outros, a aquisição do kit didático Arduino Black Board da Robocore (RoboCore, n.d.). Além disso, deve-se ressaltar as plataformas *open-source* que permitiram o estudo, simulação e constituição do projeto, como o PSP-UFU, VSC, Arduino IDE, LTSpice e KiCad.

Também, de grande contribuição no projeto e pesquisa tem-se a Universidade Federal de Uberlândia (UFU), em especial a Faculdade de Engenharia Elétrica (FEELT), que permitiu ao longo dos 5 anos de graduação adquirir o conhecimento técnico, teórico e prático na formação como engenheiro eletricista. Tal aprendizado é resultado direto dos esforços próprios e dos docentes da instituição.

5.2 Trabalhos Futuros

Em futuros trabalhos é possível realizar a produção física do conversor SPWM e integrá-lo a um relé de proteção. Além disso, são necessárias futuras adaptações, para que haja o devido envio dos sinais captados do relé para a atuação dos disjuntores virtuais presentes no PSP-UFU.

Tais adaptações permitirão no futuro a alimentação em malha fechada do HIL, o que colaborará com o ensino em aplicações didáticas de baixo custo ou ainda comerciais para estudo e parametrização de relés. Neste caso, ainda é possível a amplificação do projeto, de modo a permitir o uso de mais relés simultaneamente.

Referências

Arduino. **Introduction to Arduino**. n.d. <<https://www.arduino.cc/en/Guide/Introduction/>>. Accessed: 2025-05-04.

Atmel Corporation. **ATmega32 8-bit AVR Microcontroller Datasheet**. 2006. <<https://www.microchip.com/wwwproducts/en/ATmega32>>. Acesso em: 6 maio 2025.

BARBOSA, A. S. **Modbus Arduino: A library that allows your Arduino to communicate via Modbus protocol**. 2012. Accessed: 2025-05-04. Disponível em: <<https://github.com/andresarmento/modbus-arduino>>.

BICHELS, A. **Sistemas Elétricos de Potência: Métodos de Análise e Solução**. Curitiba: Editora da Universidade Tecnológica Federal do Paraná, 2018. Disponível em: <<http://repositorio.utfpr.edu.br/jspui/handle/1/3182>>.

BLACKBURN, J. L. **Protective Relaying: Principles and Applications**. 3. ed. [S.l.]: CRC Press, 2006.

CHEN, C.; LI, J. et al. Hardware-in-the-loop simulation in power system engineering education. **IEEE Transactions on Power Systems**, v. 28, n. 4, p. 4170–4178, 2013.

COTOSCK, K. R. **Proteção de Sistemas Elétricos: Uma Abordagem Técnico-Pedagógica**. Tese (Dissertação (Mestrado)) — Universidade Federal de Minas Gerais, 2007.

GLOVER, J. D.; SARMA, M. S. **Power System Analysis and Design**. 2. ed. Boston: PWS Publishing Company, 1994.

HOLMES, D. G.; LIEW, T. A.; MCGRATH, B. P. Analytical and experimental study of a grid-connection technique using pwm inverter with sinusoidal current control. **IEEE Transactions on Power Electronics**, v. 18, n. 3, p. 658–664, May 2003.

KAZMIERKOWSKI, M. P.; KRISHNAN, R.; BLAABJERG, F. **Control in Power Electronics: Selected Problems**. [S.l.]: Academic Press, 2002.

MARGOLIS, M. **Arduino Cookbook**. 2. ed. [S.l.]: O'Reilly Media, 2011.

MELLO, N. F. B. de. **Automação Digital de Subestações de Energia Elétrica**. Tese (Dissertação (Mestrado)) — Universidade Federal do Rio de Janeiro, 2006.

Microchip Technology. **ATmega328P 8-bit AVR Microcontroller Datasheet**. 2021. <<https://ww1.microchip.com/downloads/en/DeviceDoc/ATmega328P-Data-Sheet-DS40002061B.pdf>>. Acesso em: 6 maio 2025.

MONK, S. **Programming Arduino: Getting Started with Sketches**. 2. ed. [S.l.]: McGraw-Hill Education, 2016.

OLIVEIRA, T. L. **Documentação da Plataforma de Sistemas de Potência - PSP-UFU**. 2024. Acessado em: 6 maio 2025. Disponível em: <<https://thales1330.github.io/PSP/pt/docs>>.

Operador Nacional do Sistema Elétrico. **Mapa do sistema elétrico de transmissão do Brasil**. 2017. <<http://www.ons.org.br>>. Acesso em: 6 maio 2025.

PCB Brasil. **Cálculo da Largura de Trilha - Circuito Impresso**. 2025. Acesso em: 6 maio 2025. Disponível em: <<https://pcbbrasil.com.br/calculo-trilha-pcb>>.

RASHID, M. H. **Power Electronics: Circuits, Devices and Applications**. 3. ed. [S.l.]: Pearson, 2010.

RoboCore. **Master Kit para Arduino com BlackBoard UNO R3**. n.d. <<https://www.robocore.net/kit-arduino/arduino-master-kit>>. Accessed: 2025-05-04.

Token Engenharia. **Relé de Proteção: O Que É, Quais as Funções e Como Configurar?** 2023. Acessado em: 6 maio 2025. Disponível em: <<https://tokenengenharia.com.br/rele-de-protecao-o-que-e-quais-as-funcoes-e-como-configurar/>>.

TRIPAULT, S. **Modbusino - A Modbus RTU implementation for Arduino**. 2013. <<https://github.com/stephane/modbusino>>. Accessed: 2025-05-04.