DOI: 10.48011/asba.v2i1.1486

Implementação de uma rotina de medição de pulsos de descargas parciais utilizando Python

Wendell Daniel Fernandes de Sousa * **Filipe Bulhões Froes** *
Helon David de Macedo Braz **
Euler Cássio Tavares de Macedo **
José Mauricio Ramos de Souza Neto **

* *Engenharia Elétrica, Universidade Federal da Paraíba, PB, (e-mail: {wendell.sousa, filipe.froes}@cear.ufpb.br).*
** *Departamento de Engenharia Elétrica, Centro de Energias Alternativas e Renováveis, Universidade Federal da Paraíba, PB, (e-mail: {helon, euler, mauricio}@cear.ufpb.br).*

Abstract: Programming is an essential tool in engineering. With new tools appearing very quickly to attend the market requirements, there's a constant necessity of innovation, even in programming education. One of the programming languages which has become very popular is Python, being one of the most used at the academy. With the objective of stimulating the study of this programming language in the group of research of partial discharges from Universidade Federal da Paraíba and utilizing as a motivation the necessity to optimize the experimental data collection of a partial discharges generator, developed previously, a measurement routine using Python was made. The challenge to the students was to automatize the pulse collection process through the oscilloscope, so that the acquisition process was made considering the autonomy of the acquisition and the interval between measurements. The measurement process would be incorporated as a tool to perform measurements of partial discharges to verify the statistic parameters of the partial discharges generator developed by the students.

Resumo:A programação é uma ferramenta essencial na engenharia. Com novas ferramentas surgindo rapidamente para atender as necessidades do mercado, há uma necessidade constante de inovação também no ensino de programação. Uma das linguagens de programação que tem ganhado cada vez mais popularidade é a linguagem Python, sendo uma das mais utilizadas no meio acadêmico. Com o objetivo de estimular o estudo dessa linguagem no grupo de pesquisa sobre descargas parciais (DP) da Universidade Federal da Paraíba e aproveitando a necessidade de otimizar a coleta de dados experimentais de uma célula geradora de descargas parciais desenvolvida previamente, foi feita uma rotina de medição em Python. O desafio lançado aos alunos da pesquisa foi automatizar o processo de coleta dos pulsos via osciloscópio, de modo que o processo de aquisição fosse realizado considerando tanto a autonomia das aquisições como o intervalo entre medidas. O processo de medição seria então incorporado pelo grupo como ferramenta para realizar medições de pulsos de DP a fim de verificar os parâmetros estatísticos da célula geradora de DP desenvolvida pelos estudantes.

Keywords: Partial discharges - Measurement - Automation - Python
Palavras-chaves: Descargas parciais - Medição - Automatização - Python

1. INTRODUÇÃO

O monitoramento de equipamentos elétricos de alta tensão é de suma importância para assegurar seu bom funcionamento e preservar sua vida útil, sendo parte crucial na manutenção preventiva. Nesse sentido, a medição de fenômenos eletromagnéticos que podem ocorrer nesses equipamentos, a depender do defeito presente, é uma ferramenta essencial. Em equipamentos como transformadores, motores, buchas e isoladores, pode haver a ocorrência do que chamamos Descargas Parciais.

De acordo com Kreuger (1989), a descarga parcial é uma descarga elétrica que ocorre em uma região do espaço sujeita a um elevado campo elétrico, cujo caminho condutor formado pela descarga não une os dois eletrodos de forma completa. Por meio de uma célula geradora de descargas parciais, podemos emular os mesmos tipos de descargas parciais que ocorrem em tais equipamentos, de forma segura e controlada, através de configurações de eletrodos que formam os chamados modelos de defeitos. Tais configurações têm suas limitações, dependendo do tempo de duração do experimento e do modelo de defeito utilizado, o mesmo pode acabar saturando ou se degradando, necessitando assim que o experimento seja

pausado para que o defeito possa passar por manutenção e, posteriormente, o experimento possa ser retomado.

No artigo de Froes et al. (2020), é descrito o desenvolvimento de uma célula geradora de descargas parciais que pode ser alimentada a partir de baixa tensão, se comparada aos níveis de transmissão e distribuição. Nele, foram realizados testes qualitativos, onde buscava-se verificar apenas a confirmação da incidência de DP na estrutura e, por isso, os experimentos feitos foram de curta duração (algumas dezenas de pulsos coletados). Os testes foram realizados de forma manual, tendo a necessidade de um operador para manusear o osciloscópio utilizadonos experimentos. Contudo, houve a necessidade de analisar os parâmetros estatísticos da célula desenvolvida e, com isso, a urgência por um método de aquisição de dados mais eficiente. A urgência pela otimização do método de medição se deu pela inviabilidade do método de medição manual, devido à necessidade de um universo amostral muito grande (centenas de amostras por experimento) para que fosse feita a análise dos parâmetros estatísticos, o que resultaria em tempos de experimento muito longos.

Nesse caso, o método de aquisiçao manual é inviabilizado não só por ser exaustivo para o operador, mas principalmente, prejudica todo o conjunto de dados e, por consequência, o rigor científico. Isso porque notou-se que as características das descargas, após um tempo longo de experimento, se modificavam significativamente com relação às descargas obtidas no início do experimento. Assim, conseguir minimizar o tempo entre as coletas sucessivas de pulsos, tornou-se ainda mais importante, visto que as condições do experimento mudam ao longo da duração do mesmo, prejudicando a consistência dos dados e aumentando a variabilidade das informações obtidas por meio dos pulsos.

Assim, automatizar o processo de aquisição de dados da célula geradora de DP não só facilita o processo de aquisição de dados, como também diminui o tempo do experimento e garante uma maior confiabilidade e consistência nos dados e, principalmente, contribui positivamente para o rigor científico, garantindo uma menor variação das condições sob as quais estão sendo realizadas as medições ao longo do experimento.

Dessa forma, aproveitando da ascensão da linguagem de programação Python e de seu vasto acervo de bibliotecas, foi implementada uma metodologia para automatizar o processo de medição de DP, considerando o uso de um osciloscópio para a aquisição dos dados, controlado via uma rotina desenvolvida em Python. Os objetivos da implementação foram criar a rotina de medição para ser utilizada como instrumento durante os experimentos com a célula geradora de DP estimular os alunos do grupo de pesquisa a desenvolverem suas habilidades em Python a partir de uma demanda real.

2. FUNDAMENTAÇÃO TEÓRICA

Neto (2014) afirma que "uma DP consiste no fluxo de elétrons e íons, que se deslocam (geralmente) através de uma pequena distância" e, com relação à duração de uma DP, acrescenta que, uma vez que a velocidade dos elétrons de um gás é muito maior do que a de íons, a medição da corrente de DP irá revelar um relativamente grande pulso de curta duração (causado pelos elétrons), podendo durar algumas unidades de nanosegundos, seguido de um pulso de duração muito mais longa, com magnitude inferior e de mesma polaridade (causada pelos íons), podendo durar algumas centenas de nanosegundos, de acordo com análises recentes.

Segundo Carminati et al. (1997), a carga aparente deslocada durante um pulso de descarga parcial é uma medida muito importante para o estudo desse fenômeno. Ela se relaciona diretamente com a falha do isolamento na qual ocorre tal evento, estando diretamente ligada às características físicas da mesma, como tamanho da cavidade, material do isolamento onde a cavidade foi formada e estado de degradação. Para isso, podemos utilizar diversos modelos de detecção de descargas parciais, que podem ser elétricos e não elétricos. De acordo com Macedo (2014), os principais métodos não-elétricos são os métodos acústico, óptico, químico e de radiofrequência. Por outro lado, o método elétrico consiste no acoplamento de um circuito de medição bastante sensível ao equipamento no qual as descargas parciais estão ocorrendo. Outra possibilidade de realizar a medição elétrica de descargas parciais é baseada na conexão não-invasiva de um sensor indutivo de alta frequência, HFCT (*High Frequency Current transformer*), e alta permeabilidade magnética ao cabo que liga o equipamento sob teste à malha de terra. Neste arranjo de medição, o HFCT é conectado de forma não invasiva ao cabo de conexão ao terra do equipamento sob teste. Neste caso, o nível de segurança da equipe responsável pela medição é maximizado, pois, caso haja a falha do equipamento sob teste, o sensor de corrente irá saturar e o mesmo não conduzirá alta tensão aos equipamentos de medição.

Ainda de acordo com Macedo (2014), para descrever as características de uma descarga parcial, podem-se utilizar três tipos de parâmetros relacionados ao tempo de observação do fenômeno, sendo esses os parâmetros básicos, os parâmetros deduzidos e os operadores estatísticos. Os parâmetros básicos são aqueles em que os sinais de descargas parciais são observados durante apenas um ciclo da tensão aplicada ao corpo de prova, dentre eles, a intensidade da descarga, a tensão inicial da descarga, também conhecida com tensão de incepção e a fase do pulso da descarga em relação à fase da tensão aplicada. Os parâmetros deduzidos descrevem a variação dos parâmetros básicos em função do tempo, como a soma das intensidades das DP, a quantidade de pulsos de DP, o valor médio da intensidade das DP, o valor máximo das DP. E os operadores estatísticos são operadores usados na avaliação estatística dos parâmetros deduzidos.

3. METODOLOGIA

3.1 Modelo de medição

O modelo de medição de descargas parciais escolhido para a aquisição dos pulsos foi o método não invasivo, utilizando um HFCT, configuração apresentada na norma IEC 60270 (2000), exemplificado na Figura 1. Tal configuração apresenta uma impedância de medição, composta pelo divisor de tensão e pelo capacitor, em paralelo com a célula gera-

dora de DP, com o HFCT enlaçando um dos cabos de alimentação da célula, orientado de forma a captar pulsos de polaridade positiva. Os pulsos captados pelo HFCT foram visualizados e salvos por meio do uso de um osciloscópio de 200 MHz com 4 GSamples/s, utilizado como estudo de caso para a implementação da metodologia. O cabo ligado ao HFCT foi conectado a um dos canais do osciloscópio enquanto a tensão de saída do divisor de tensão foi medida por meio de uma ponta de prova apropriada, conectada em outro canal do osciloscópio, afim de verificar a(s) fase(s) em que ocorriam os pulsos. Tal tensão é proporcional à tensão de alimentação em uma escala de 1 : 1000. Na Figura 2 é possível verificar o modelo e a foto da célula utilizada nos ensaios.

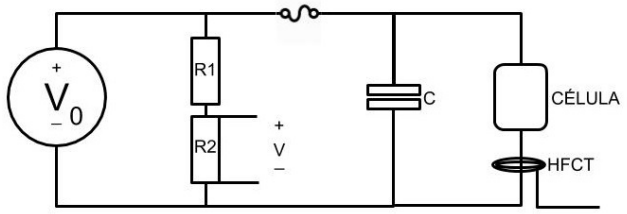
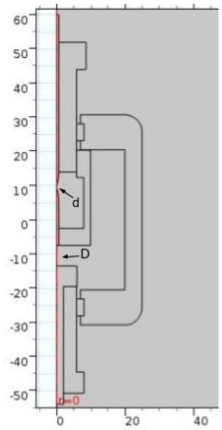


Figura 1. Modelo da estrutura de medição.



(a) Modelo 2D eixo-simétrico da estrutura configurada com o defeito tipo ponta-ponta flutuante.

(b) Foto da estrutura geradora de DP utilizada nos testes.

Figura 2. Célula geradora de DP utilizada.

3.2 Rotina de medição

Utilizando o modelo de medição explicitado, o mesmo que foi utilizado para os testes manuais, foi estabelecida a conexão do osciloscópio com o computador por meio da porta USB do equipamento marcada como *device*. Foi utilizado um osciloscópio da AGILENT TECHNOLOGIES™, do modelo MSO-X 3024A. Vale ressaltar que, a partir de 2014, esta empresa de testes e medição eletrônica passou a se chamar KEYSIGHT TECHNOLOGIES™. A interface de programação (API – *Application Programming Interface*) usada no instrumento é a *Virtual Instrument Software Architecture* (*VISA*), um protocolo de comunicação padrão na indústria de testes e medições.

A linguagem de programação utilizada para comunicação com o instrumento de medição foi a Python, a qual permite aplicar funções de uma biblioteca compartilhada chamada de *VISA* ou *PyVISA (.dll, .so, .dylib)* e usá-las em aplicações padrões.

O desafio então foi desenvolver um código em Python que, gerenciasse o osciloscópio, de modo que este detectasse um sinal de pulso gerado no dispositivo e armazenasse os dados referentes a ele diretamente no computador e em um *Pen Drive* (*Memória USB Flash Drive*), salvos em formato *.png* e *.csv*.

É utilizado no código desenvolvido um conjunto de bibliotecas necessárias para a implementação das funções. Desta forma, além da biblioteca *VISA* são usadas duas bibliotecas padrão do Python, *os* e *time*. Uma fornece funcionalidades dependentes do sistema operacional, enquanto a outra oferece funcionalidades referentes ao tempo, respectivamente.

Logo em seguida é definido o endereço do instrumento utilizado na sintaxe do protocolo *VISA*, pois é através deste endereço que ocorrerá a comunicação com o osciloscópio. Neste caso, foi definida uma variável *VISA_ADDRESS* para armazenar o endereço.

Definido o endereço de comunicação com o instrumento, deve-se especificar o endereço da biblioteca *VISA* no sistema operacional, e é feito isso no Python através da chamada da função *visa.ResourceManager()*, cujo argumento é o caminho no computador para a biblioteca. Em seguida, utiliza-se a função *open_resource()*, cujo argumento é o *VISA_ADDRESS* a fim de definir o instrumento para o nome do recurso que será usado ao longo de todo o código para execução das funcionalidades necessárias. Logo após, definiu-se uma quantidade de tempo absoluta, em milissegundos, para que o recurso fosse desbloqueado, caso este tempo fosse ultrapassado um erro seria retornado.

A fim de não haver falhas de comunicação com o instrumento devido a configurações anteriores realizadas e armazenadas no barramento, ou seja, nas linhas de transmissão das informações para o osciloscópio, é usada a função *.clear*. Após realizar todas as configurações descritas anteriormente, é necessário verificar se a comunicação com o instrumento está funcionando corretamente, então é usada a função *.query("*IDN?")* para que o nome do instrumento seja buscado e exibido na execução do código por meio da função *print()*. Observe que usa-se a função da biblioteca *VISA* na linguagem Python, porém o argumento da função é escrito no protocolo *VISA*. Vale ressaltar que as funções *.query* e *.write* são utilizadas para buscar informações e enviar comandos, respectivamente. O fluxograma das configurações iniciais do programa está representado na Figura 3.

Após a realização destas configurações iniciais básicas para a comunicação e preparação do instrumento, com o objetivo de receber os comandos enviados pelo código em construção, foram elaboradas três funções para realizar as operações de salvamento dos dados dos pulsos. De forma geral, a primeira função adquire os dados da imagem, a segunda define local de salvamento e o nome do arquivo, e a terceira armazena as formas de onda dos pulsos no *Pen Drive*.

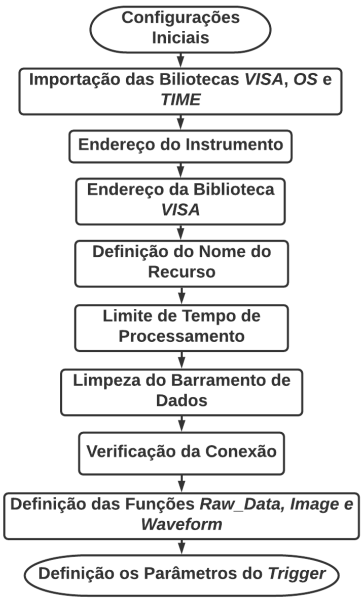


Figura 3. Configurações Iniciais.

A primeira função definida foi chamada de *Raw_Data*, posto que é responsável pela aquisição dos dados não tratados da imagem gerada do pulso no instrumento. Desta forma, definiu-se uma seção inicial do arquivo de imagem que contém o que é conhecido como cabeçalho. Para isto, criou-se uma lista de dados nomeada por *data_in*, que é o argumento da função. É no cabeçalho onde é possível encontrar informações das cores e demais dados da imagem. Para localizar a posição inicial do cabeçalho dentro da lista de valores criados, é preciso identificar o símbolo da cerquilha (#). Dado que se definiu a *data_in* no formato de uma *string* foi usado o comando *.find()* para localizar o símbolo.

Após identificar o início do cabeçalho do arquivo, insere-se uma estrutura condicional para alertar caso a posição inicial seja menor que zero. O tamanho do comprimento da imagem foi então definido após esta condição, a partir da posição onde se encontra o símbolo # acrescido de um (1), visto que a posição inicial é zero. Tendo então o tamanho do comprimento da imagem no formato inteiro, pode-se obter o tamanho total do arquivo da imagem definindo uma lista a partir da posição posterior à que se encontra o símbolo # até a posição equivalente ao tamanho do comprimento da imagem. Posteriormente, obtém-se o comprimento do cabeçalho por meio da soma da posição inicial (#) com o comprimento da imagem obtido anteriormente. Por fim, a função criada retorna a lista *data_in* com os dados flutuantes da imagem com base no comprimento calculado do cabeçalho e no tamanho do arquivo de imagem. Na Figura 4, está representado o fluxograma da função *Raw_Data*.

A segunda função criada, *Image*, é responsável por, efetivamente, salvar no computador a imagem capturada pelo instrumento. Ela recebe como argumento o parâmetro *i* tendo em vista que será usada numa estrutura de repetição. Para tanto, inicia-se definindo o caminho de salvamento do arquivo por meio da função *.chdir()* da biblioteca *os*, da qual também é usado o recurso *.getcwd()* para buscar

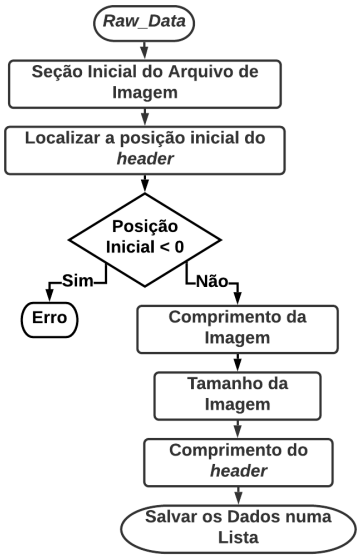


Figura 4. Função Raw_Data.

este local de salvamento e exibi-lo na tela por meio do comando *print()*.

Para evitar que avisos na tela, como a mensagem de salvamento dos dados, sejam salvos na captura da imagem usou-se a funcionalidade *.query(':SYSTEM:DSP ;*OPC?')*. O comando *:SYSTEM:DSP* remove a mensagem da tela e **OPC?* define o bit de operação de conclusão da tarefa. Também há a opção de salvar a imagem com o fundo padrão na cor preta, ou invertido, que seria na cor branca. O comando usado para manter a configuração padrão então foi o *.write(":HARDCOPY:INKSAVER OFF")*.

Realizadas estas configurações iniciais, por meio do comando *.write(":DISPlAY:DATA? PNG, COLOR")* foi dada a tarefa ao instrumento de capturar a imagem gerada pelo pulso em sua tela no formato *.png*. Em seguida, definiu-se uma variável para ler os dados desta imagem capturada por meio da função *.read_raw()*. Tendo então os dados lidos no Python, executa-se a função criada anteriormente, a *Raw_Data(data_in)*, para ter os dados da imagem capturada na forma não tratada, que no caso seria em blocos binários. A variável na qual foi armazenada estes códigos binários da imagem foi a *Image_Data*.

Neste ponto, define-se a variável que leva o nome do arquivo em função do diretório utilizado e do parâmetro *i*, a qual é chamada de *filename*, e também abre-se o arquivo especificado na *filename* na forma de escrita por meio do comando *.open(filename, "wb")*. Tendo então este arquivo aberto, gravam-se os dados obtidos da captura de tela do instrumento através do comando *.write(Image_Data)*. A função *Image* está representada na Figura 5.

A terceira função criada foi a *Waveform*, que numa tradução literal seria a forma de onda. Essa função então recebe os dados dessas informações capturadas pelo instrumento, a partir dos quais pode-se buscar padrões de comportamento de um tipo específico de DP.

A função também recebe como argumento o parâmetro *i* e é iniciada definindo e enviando os comandos sobre o tipo de arquivo a ser salvo (*CSV, ASCiixy* ou *BI-*

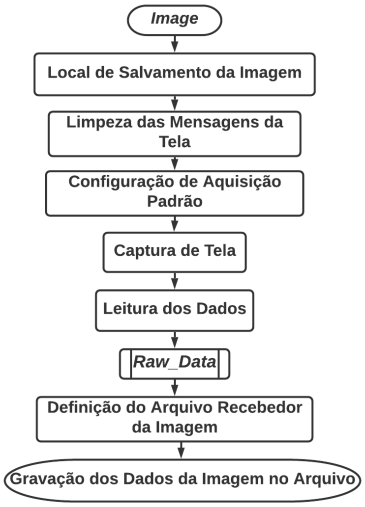


Figura 5. Função Image.

Nary), o nome do arquivo e seu local de armazenamento. Logo após, é necessário limpar todos os registros antes de emitir o comando para salvar os dados. Isso é necessário para que seja possível determinar adequadamente quando as tarefas foram concluídas, para isto é utilizado o comando *.query("*CLS;*OPC?")*. Por fim, o arquivo é salvo no *Pen Drive* por meio do comando *.write(':SAVE:WAVeform:STARt "' + str(location) + str(filename) +'"')*. A função *Waveform* está representada na Figura 6.

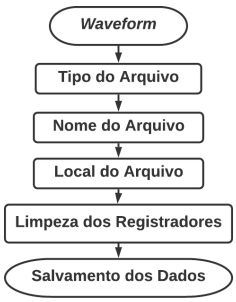


Figura 6. Função Waveform.

Após todas as configurações do aparelho serem realizadas e todas as funções necessárias serem criadas, pode-se enfim implementar uma estrutura de repetição que salvará a imagem reproduzida no instrumento e os dados do pulso que é gerado no modelo de defeito. Definiu-se então a estrutura de repetição, dado que a intenção é executar um bloco de código por um número de iterações a determinar. Dentro desta estrutura, foi criado um contador para que seja exibido na tela, ao longo da execução do código, em que iteração cada captura está sendo realizada.

Em seguida, são adicionados os comandos *.write(":STOP")* e *.write("*OPC?")* para que o instrumento pare qualquer execução em andamento e em seguida esteja preparado para capturar os dados medidos a partir do acionamento do *trigger*. Para isto, a funcionalidade *single* é habilitada por meio da instrução *.write(":SINGle")*. Vale ressaltar que a configuração do *trigger* pode ser realizada de forma manual ou também por programação, ficando então a critério do responsável pela aquisição destes dados. Após estas configurações no laço, usa-se a funcionalidade *time.sleep()* para que a execução seja interrompida por frações de segundos no laço a fim de não haver problemas na aquisição dos dados devido à alta velocidade de execução entre os comandos.

Neste ponto, é importante se obter a informação do acionamento do *trigger*, então cria-se uma variável que armazena a informação se o registrador de evento do *trigger* está ativo ou não. Desta forma, usa-se o comando *.query(":TER?")*, onde *TER* é uma sigla para *Trigger Event Register*. Este comando retorna 0, caso o *trigger* não for acionado, ou 1, caso seja acionado.

Vale destacar que o *TER* é limpo logo após cada leitura, ou seja, ele é lido e limpo automaticamente. Notou-se que em toda primeira iteração da estrutura de repetição criada, o *TER* retornava 0, provavelmente porque acabara de ser inicializado e por consequência limpo. Logo, usa-se o contador criado para inserir uma condição para o primeiro laço da execução a fim de que retorne ao início da estrutura e execute novamente o código para que o *TER* funcione adequadamente, ou seja, a condição consiste em saltar a primeira iteração.

Com base nesta variável interna, cria-se uma outra estrutura condicional dentro do laço, que, quando identifica que o *TER* é igual a 1 ,executa as funções criadas anteriormente, *Image* e *Waveform*. Caso o *TER* seja igual a 0 uma mensagem é exibida informando que nenhum dado foi adquirido e segue-se para o próximo laço. Por fim, envia-se um comando para limpar o barramento de dados (*.clear()*) e para encerrar o processamento após o número definido de laços (*.close()*). A representação da estrutura de repetição pode ser verificada na Figura 7.

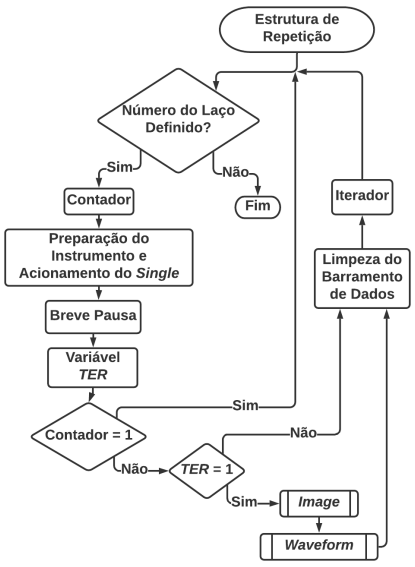


Figura 7. Estrutura de Repetição.

4. RESULTADOS E MEDIÇÕES

Uma vez criada a rotina, os testes foram refeitos utilizando o mesmo modelo de medição, porém, implementando a rotina de medição desenvolvida. Foram realizados três

testes com a célula geradora de DP, equipada com o defeito do tipo ponta-ponta flutuante, com a configuração que apresentou melhores resultados, tanto em amplitude quanto em estabilidade, durante os testes de validação da estrutura desenvolvida, de acordo com o artigo de Froes et al. (2020).

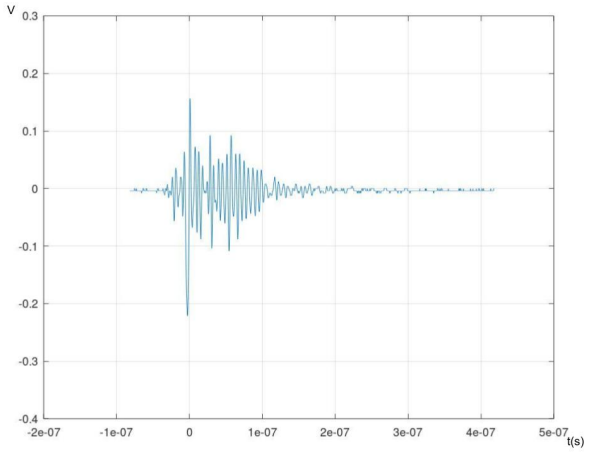
Dessa forma, foram feitos alguns testes preliminares para garantir que a rotina implementada estava funcionando de forma adequada e em seguida foi feita uma rodada de medições. O processo de aquisição durou aproximadamente dezesseis minutos e salvou, como o esperado, dados referentes a quinhentos pulsos de DP. Esse foi um salto significativo de eficiência quando comparado com o método manual, com o qual foi possível medir a mesma quantidade de pulsos em aproximadamente cinquenta minutos corridos, divididos em cinco baterias de dez minutos de duração, em média. Contudo, apenas quatrocentos e oitenta e três dos quinhentos pulsos puderam ser aproveitados, pois por alguma razão, provavelmente devido à falta de um *delay* entre funções, os arquivos salvos referentes a dezessete pulsos tinham valor nulo, como se tivessem sido salvos no momento anterior ao acionamento do *trigger*. É possível verificar a semelhança entre os pulsos de DP gerados pela mesma estrutura medidos com o método manual e com a rotina implementada, Figuras 8(a) e 8(b), respectivamente. É importante ressaltar que as medições referentes a cada método ocorreram em dias diferentes.

Vale ressaltar que não houve ocorrência de falsos positivos dentro do conjunto de amostras, ou seja, não houve sinais não nulos coletados que não fossem pulsos de padrão semelhante aos de DP, como foi observado posteriormente. Os pulsos foram analizados usando o método de comparação visual para verificar se de fato estavam dentro do padrão de DP do tipo interna, o que de fato ocorreu. Os pulsos obtidos foram tratados posteriormente por meio de outra rotina em Python para verificar as métricas convencionais referentes à configuração testada. Os resultados obtidos estão dispostos na Tabela 1 e na Figura 9.

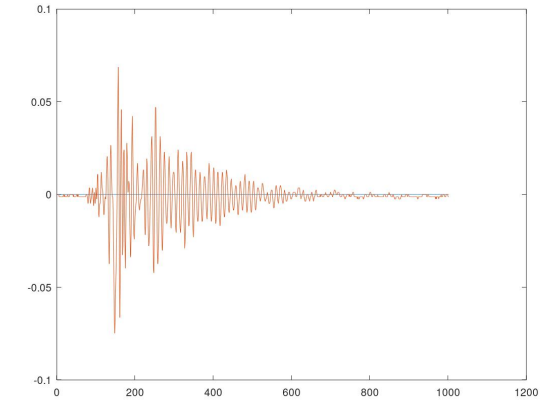
Tabela 1. Tabela de parâmetros calculados a partir dos pulsos coletados.

Energia acumulada	
Média	43,410 nJ
Desvio padrão	23,028 nJ
Valor mínimo	13,050 nJ
Valor máximo	242,792 nJ
Tempo de descarga	
Média	0,319 us
Desvio padrão	0,021 us
Valor mínimo	0,277 us
Valor máximo	0,380 us

Uma observação importante é que a rotina implementada é capaz de salvar tanto a forma de onda quanto a imagem da tela do osciloscópio ao mesmo tempo, enquanto se feito por um operador, isso acresceria ainda mais o tempo de aquisição de cada pulso. A redundância na coleta dos dados foi feita para facilitar a conferência dos mesmos, a fim de identificar uma eventual necessidade de ajuste de forma mais rápida. Contudo, se retirada a parte do código referente à aquisição das formas de onda tal como apresentadas na tela do osciloscópio, não é esperado uma



(a) Pulso de DP medido com o método manual.



(b) Pulso de DP medido com o método implementado.

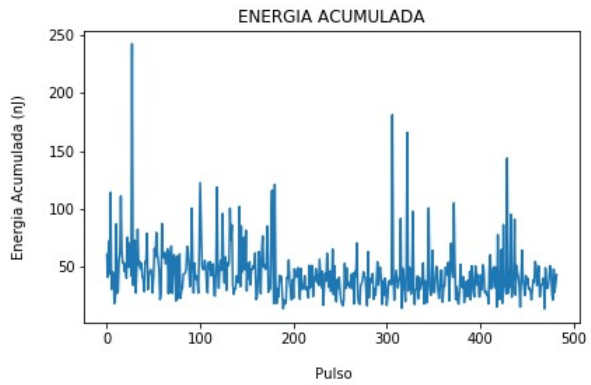
Figura 8. Comparação entre os pulsos medidos com diferentes métodos.

grande melhoria no desempenho geral da rotina como um todo. Isso porque o grande limitador de desempenho para essa rotina foi a sincronia entre o osciloscópio e o computador para que seja possível capturar os pulsos. De forma mais detalhada, é necessário ajustar o *delay* do *trigger* para maximizar a eficiência e minimizar as perdas de dados, como ocorreu durante os testes realizados em que alguns arquivos estavam nulos.

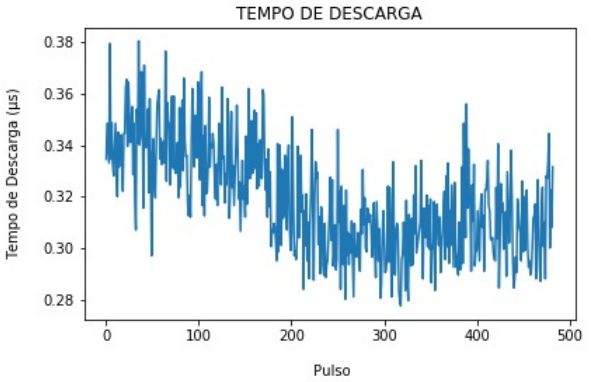
5. CONCLUSÕES

A metodologia proposta apresentou resultados satisfatórios, uma vez que o método de aquisição de dados implementado gerou resultados importantes para a pesquisa a respeito da célula geradora de DP. Tais resultados são importantes, pois viabilizam a coleta de centenas de amostras em minutos. Embora o próprio método implementado não esteja otimizado, o mesmo atende às necessidades da pesquisa. Além disso, o processo de desenvolvimento da ferramenta computacional serviu como uma aplicação prática da programação, dentro do curso de Engenharia Elétrica da Universidade Federal da Paraíba, para os participantes da pesquisa.

A prática ajudou os participantes a compreender conceitos gerais de programação e estimular o interesse com relação



(a) Gráfico da energia acumulada por pulso.



(b) Gráfico do tempo de descarga por pulso.

Figura 9. Gráficos de tempo e energia acumulada por pulso.

à programação, além de ter proporcionado a experiência de desenvolver um projeto que atendesse a uma demanda real. Desse modo, o presente trabalho contribuiu positivamente no processo de formação dos estudantes da graduação envolvidos na pesquisa, sendo uma amostra da ampla gama de ferramentas e aplicações possibilitadas pela programação.

AGRADECIMENTOS

Os autores agradecem ao PIBIC da Universidade Federal da Paraíba e ao CNPq por fomentar a pesquisa.

REFERÊNCIAS

E. Carminati and M Lazzaroni. (1997). A Contribution in Partial Discharge Detection. *IEEE Instrumentation and Measurement Technology Conference.* Ottawa, Canada.

E.C.T Macedo. (2014). Metodologia Para a Classificação de Descargas Parciais Utilizando Redes Neurais Artificiais. *Ph.D. thesis, Universidade Federal de Campina Grande.* Campina Grande-PB, Brazil.

F.B. Froes, H.D.M. Braz, E.C.T. Macedo and J.M.R.S. Neto. (2020). Desenvolvimento de uma célula aplicada à geração de descargas parciais a partir de uma fonte de baixa tensão. *VIII Simpósio Brasileiro de Sistemas Elétricos (Unpublished).* Santo André-SP, Brazil.

F.H. Kreuger. (1989). Partial discharge detection in High Voltage Equipment. *Butterworths.* London, England.

IEC 60270. (2000). High-Voltage Test Techniques - Partial Discharge Measurements. Geneve, Switzerland.

J.M.R.S Neto. (2014). Localização de Descargas Parciais Baseada em um Método Radiométrico - Prova do Princípio. *Ph.D. thesis, Universidade Federal de Campina Grande.* Campina Grande-PB, Brazil.

KEYSIGHT TECHNOLOGIES (2019). Keysight InfiniiVision 3000 X-Series Oscilloscopes Programmer's Guide. *Keysight Technologies, Inc.* Colorado Springs, CO, USA.

K.S. Oliveira Filho. (1998). Formatos de Imagens. Disponível em: http://astro.if.ufrgs.br/med/imagens/node28.htm. Acesso em: 14 de maio de 2020.

Python Software Foundation. (2020a). Os: Diversas interfaces de sistema operacional. Disponível em: https://docs.python.org/pt-br/3/library/os.htmlos-file-dirroject/PyVISA/. Acesso em: 06 de maio de 2020.

Python Software Foundation. (2020b). Time: Time access and conversions. Disponível em: https://docs.python.org/3/library/time.html. Acesso em: 06 de maio de 2020.

PyVISA. *(2020). PyVISA: Control your instruments with Python.* Disponível em: https://pyvisa.readthedocs.io. Acesso em: 06 de maio de 2020.