UNIVERSIDADE FEDERAL DE SANTA CATARINA
CENTRO TECNOLÓGICO
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA

Leonardo Acosta Rodrigues

Diagnóstico de Falha e Previsão de Vida Útil de Capacitores Eletrolíticos de Alumínio para Barramento de Conversores CA/CC/CA em Veículos Elétricos

Florianópolis
2023

Leonardo Acosta Rodrigues

Diagnóstico de Falha e Previsão de Vida Útil de Capacitores Eletrolíticos de Alumínio para Barramento de Conversores CA/CC/CA em Veículos Elétricos

Dissertação submetida ao Programa de Pós-Graduação em Engenharia Elétrica da Universidade Federal de Santa Catarina para a obtenção do título de Mestre em Engenharia Elétrica.
Orientador: Prof. Gierri Waltrich, Dr.

Florianópolis
2023

Ficha de identificação da obra elaborada pelo autor,
através do Programa de Geração Automática da Biblioteca Universitária da UFSC.

Rodrigues, Leonardo Acosta
Diagnóstico de Falha e Previsão de Vida Útil de Capacitores Eletrolíticos de Alumínio para Barramento de Conversores CA/CC/CA em Veículos Elétricos / Leonardo Acosta Rodrigues ; orientador, Gierri Waltrich, 2023.
195 p.

Dissertação (mestrado) - Universidade Federal de Santa Catarina, Centro Tecnológico, Programa de Pós-Graduação em Engenharia Elétrica, Florianópolis, 2023.

Inclui referências.

1. Engenharia Elétrica. 2. Veículos Elétricos. 3. Transformada de Fourier. 4. Mínimos Quadrados. 5. Vida Útil do Capacitor. I. Waltrich, Gierri . II. Universidade Federal de Santa Catarina. Programa de Pós-Graduação em Engenharia Elétrica. III. Título.

Leonardo Acosta Rodrigues

Diagnóstico de Falha e Previsão de Vida Útil de Capacitores Eletrolíticos de Alumínio para Barramento de Conversores CA/CC/CA em Veículos Elétricos

O presente trabalho em nível de mestrado foi avaliado e aprovado por banca examinadora composta pelos seguintes membros:

Prof. Joselito Anastácio Heerdt, Dr.
Universidade do Estado de Santa Catarina

Prof. Lenon Schmitz, Dr.
Universidade Federal de Santa Catarina

Certificamos que esta é a **versão original e final** do trabalho de conclusão que foi julgado adequado para obtenção do título de Mestre em Engenharia Elétrica.

Documento assinado digitalmente
Telles Brunelli Lazzarin
Data: 22/03/2023 16:54:10-0300
CPF: ***.400.719-**
Verifique as assinaturas em https://v.ufsc.br

Prof. Telles Brunelli Lazzarin, Dr.
Coordenador do Programa

Documento assinado digitalmente
Gierri Waltrich
Data: 10/03/2023 16:40:13-0300
CPF: ***.757.199-**
Verifique as assinaturas em https://v.ufsc.br

Prof. Gierri Waltrich, Dr.
Orientador

Florianópolis, 2023.

Este trabalho é dedicado à toda minha família e amigos
que contribuíram na minha formação acadêmica.

AGRADECIMENTOS

Primeiramente, agradeço a minha família, que sempre demonstraram apoio incondicional nas decisões nessa jornada. Especialmente, agradeço aos meus pais, que foram minha base e exemplo de perseverança e dedicação.

Agradeço imensamente ao meu orientador, professor Gierri, pela dedicação, paciência e disponibilidade incansável ao longo deste processo de elaboração da minha dissertação. Sua orientação foi fundamental para o sucesso deste trabalho e sua contribuição foi inestimável. Agradeço também por todos os conselhos valiosos que me ajudaram a evoluir como acadêmico e profissional.

Também agradeço aos professores do INEP, Roberto Francisco Coelho, André Luís Kirsten, Marcelo Lobo Heldwein, Denizar Cruz Martins, Telles Brunelli Lazzarin e Samir Ahmad Mussa cujos conhecimentos transmitidos nas disciplinas foram fundamentais para a construção de uma base sólida que me permitiu desenvolver essa dissertação. Agradeço também ao competente técnico Dr. Antônio Luiz Schalata Pacheco e ao secretário Diogo Duarte Luiz pela assistência na realização desta dissertação.

Ainda, estendo meus agradecimentos aos professores Joselito Anastácio Heerdt e Lenon Schmitz por se disporem a compor a banca de avaliação deste trabalho e contribuírem de maneira significativa para o seu desenvolvimento.

Por fim, agradeço aos colegas da turma 2020 que entraram comigo e seguiram determinados durante a jornada do mestrado mesmo com todas as dificuldades que passamos na passagem pela pandemia, causada pelo coronavírus.

RESUMO

Essa dissertação apresenta a análise, desenvolvimento, comparação teórica e validação via simulação e experimentação de dois métodos para estimação de vida útil para capacitores eletrolíticos utilizados em conversores CA/CC/CA para Veículos Elétricos (VEs). Visando desenvolver um sistema observador, que possa ser implementado sem a necessidade de modificações no sistema de modulação ou controle do conversor, diversos métodos foram revisados na literatura e comparados. Desses, dois trabalhos se destacaram com a possibilidade de não ser necessário modificação no sistema de modulação ou controle do conversor, sendo um utilizando uma derivação da Transformada Discreta de Fourier e outro uma variação recursiva do Método dos Mínimos Quadrados, que permitem a obtenção dos parâmetros de Resistência Série Equivalente (*Equivalent Series Resistance* - ESR) e capacitância (C), necessários para implementação dos algoritmos de predição do estado do componente. O processo de aplicação de ambos os métodos requer uma variação de tensão no capacitor de magnitude suficiente para ser detectada por circuitos integrados de precisão e processados por um Processador Digital de Sinal (*Digital Signal Processing* - DSP). Para tanto, uma modificação do circuito convencional do sistema de carregamento do VE é proposta, onde é sugerido um *bypass* que conecta o inversor do VE no conversor retificador, usando como entrada a rede elétrica para produzir a variação de tensão requerida no barramento, permitindo a análise dos parâmetros do capacitor. Antes de dar início ao funcionamento do sistema na totalidade, o capacitor é retirado do inversor e inserido em uma câmara fechada onde equações relacionando temperatura, ESR e C são obtidas com o auxílio de um medidor de impedância externo, visando construir o perfil atual do componente. Quando o sistema inversor/retificador está em operação, o DSP é conectado e os métodos são iniciados para se obter e estimar a vida útil do capacitor usando os parâmetros de ESR e C, considerando as equações baseadas em temperatura obtidas anteriormente. Após um longo período de análise e obtenção de dados, os parâmetros são inseridos em algoritmos de extrapolação cuja função é prever o momento de falha do componente, predizendo a duração da vida útil e o momento em que deve ser substituído. Por fim, simulações são apresentadas e um protótipo é montado na bancada do laboratório para comprovar e validar via resultados experimentais os métodos estudados.

Palavras-chave: Veículos Elétricos. Transformada de Fourier. Mínimos Quadrados. Vida Útil do Capacitor.

ABSTRACT

This dissertation presents the analysis, development, theoretical comparison and validation through simulation and experimentation of two methods for lifetime estimation of electrolytic capacitors used in AC/DC/AC converters for Electric Vehicles (EVs). With the objective of developing an observer system that can be implemented without the need for modifications to the converter's modulation or control system, several methods were reviewed in the literature and compared. Of these, two stood out as having the possibility of not requiring modification to the modulation or control system of the converter, one using a derivative of the Discrete Fourier Transform and the other a recursive variation of the Least Squares Method, which allow for the obtaining of the parameters of Equivalent Series Resistance (ESR) and capacitance (C), necessary for the implementation of component state prediction algorithms. The process of applying both methods requires a voltage variation on the capacitor of sufficient magnitude to be detected by precision integrated circuits and processed by a Digital Signal Processing (DSP). To this end, a modification of the conventional EV charging system circuit is proposed, where a bypass is suggested that connects the EV inverter to the rectifier converter, using the electric grid as input to produce the required voltage variation on the bus, allowing for the analysis of the capacitor parameters. Before the system as a whole starts to operate, the capacitor is removed from the inverter and inserted into a closed chamber where equations relating temperature, ESR and C are obtained with the help of an external impedance meter, with the aim of building the current profile of the component. When the inverter/rectifier system is in operation, the DSP is connected and the methods are started to obtain and estimate the lifetime of the capacitor using the ESR and C parameters, taking into consideration the temperature-based equations obtained previously. After a long period of analysis and data collection, the parameters are inserted into extrapolation algorithms that have the function of predicting the moment of component failure, predicting the duration of the lifetime and the moment it should be replaced. Finally, simulations are presented and a prototype is assembled on the laboratory bench to verify and validate the studied methods through experimental results.

Keywords: Electric Vehicles. Fourier Transform. Least Square Method. Capacitor Useful Life.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

Al-Caps	*Aluminum Electrolytic Capacitors*
BEV	*Battery Electric Vehicle*
BMS	*Battery Management Systems*
CCS	*Code Composer Studio*
DFT	*Discrete Fourier Transform*
DSP	*Digital Signal Processor*
DTMF	*Dual-Tone Multi-Frequency*
FCEV	*Hydrogen Fuel Cell Electric Vehicle*
FFT	*Fast Fourier Transform*
FIFO	*first in first out*
FT	*Fourier Transform*
HEV	*Hybrid Electric Vehicle*
IIR	*Infinite Impulse Response*
IrDA	*Infrared Data Association*
KERS	*Kinetic Energy Recovery System*
LS	*Least Square*
LTI	*Linear Invariante no Tempo*
MC	Monitoramento de Condição
MLC-Caps	*Multi-Layer Ceramic Capacitors*
MPPF-Caps	*Metallized Polypropylene Film Capacitors*
NiMH	Níquel-hidreto metálico
NRZ	*non-return-to-zero*
OBC	*On-Board Charger*
PFC	*Power Factor Correction*
PHEV	*Plug-In Hybrid Electric Vehicle*
P-HEV	*Parallel Hybrid Electric Vehicle*
RLS	*Recursive Least Square*
S-HEV	*Series Hybrid Electric Vehicle*
SP-HEV	*Series-Parallel Hybrid Electric Vehicle*
SPWM	*Sinusoidal Pulse Width Modulation*
UART	*Universal Asynchronous Receiver and Transmitter*
UPS	*Uninterruptible Power Supply*
VE	Veículo Elétrico
ZEBRA	Cloreto de sódio-níquel

LISTA DE SÍMBOLOS

C	Capacitância
R_p	Resistência de isolamento
R_{ESR}	ESR
L_{ESR}	ESL
$\hat{T}$	Vetor temperatura
C_0	Capacitância Inicial
ESR_0	ESR inicial
C_{in}	Capacitância estimada em uma temperatura específica
ESR_{in}	ESR estimada em uma temperatura específica
T_a	Temperatura lida por sensoriamento
T_0	Temperatura inicial
A_0	Constante obtida pelo método LS
A_1	Constante obtida pelo método LS
$ESR_{\text{DFT\|RLS}}$	ESR obtida pelo método DFT ou RLS
$C_{\text{DFT\|RLS}}$	C obtida pelo método DFT ou RLS
$ESR_{\text{corrigida}}$	ESR compensada por temperatura
$C_{\text{corrigida}}$	C compensada por temperatura
ESR_{ft}	Limite de ESR para detecção de falha
C_{ft}	Limite de C para detecção de falha
PHS_{ESR}	Situação de vida útil atual (*Present Health Status*) para a ESR
PHS_C	Situação de vida útil atual (*Present Health Status*) para a C
i_{ret}	Corrente de saída do retificador
i_{inv}	Corrente de entrada do inversor
i_{cap}	Corrente do capacitor
i_a	Corrente da fase A
i_b	Corrente da fase B
i_c	Corrente da fase C
v_{ag}	Tensão da fase A
v_{bg}	Tensão da fase B
v_{cg}	Tensão da fase C
v_{cap}	Tensão do capacitor
v_{S2}	Tensão do interruptor S_2
v_{S4}	Tensão do interruptor S_4
v_{S6}	Tensão do interruptor S_6
m_a	Índice de modulação da referência senoidal A
m_b	Índice de modulação da referência senoidal B
m_c	Índice de modulação da referência senoidal C

v_{tri}	Sinal triangular de referência
R_{eq}	Resistência equivalente conectada ao retificador trifásico para representar o inversor trifásico
C_1	Capacitância conectada ao retificador trifásico para testes com carga equivalente
C_2	Capacitância conectada ao retificador trifásico para testes com carga equivalente
C_3	Capacitância conectada ao retificador trifásico para testes com carga equivalente
C_{inv}	Capacitância conectada ao retificador trifásico para testes com inversor conectado
$v_{\text{cap},1}$	Tensão no capacitor C_1
$v_{\text{cap},2}$	Tensão no capacitor C_2
$v_{\text{cap},3}$	Tensão no capacitor C_3
$v_{\text{cap,inv}}$	Tensão no capacitor C_{inv}
k	Variável referente a harmônica desejada ($k = 0,1,2...N-1$)
X	Variável analisada no cálculo dos métodos envolvendo DFT
X_{re}	Parte real da harmônica analisada no cálculo da DFT
X_{im}	Parte imaginária da harmônica analisada no cálculo da DFT
n	Passos da iteração para o cálculo da DFT
N	Quantidade de pontos armazenados para o cálculo da DFT
f_s	Frequência de amostragem no cálculo da DFT
$f_{\text{s,min}}$	Frequência de amostragem mínima no cálculo da DFT
f_m	Frequência de interesse no cálculo da DFT
Q	Resto da divisão da ordem do harmônico desejada
D_0	Constante de simplificação do método Radix-4
D_1	Constante de simplificação do método Radix-4
D_2	Constante de simplificação do método Radix-4
D_3	Constante de simplificação do método Radix-4
D_4	Constante de simplificação do método Radix-4
W_N	*twiddle factor*
H_z	Função de transferência discreta
X_{amp}	Magnitude da componente harmônica obtida pelos métodos DFT
X_{pha}	Fase da componente harmônica obtida pelos métodos DFT
X_c	Impedância capacitiva
λ	Fator de esquecimento
K	Constante de Kalman
$\hat{\theta}$	Matriz que possui as constantes necessárias para encontrar ESR e C
P	Constante da covariância

ϕ	Matriz das funções de entrada e saída do sistema
Φ	Matriz das matrizes das funções de entrada e saída do sistema
ϵ	Erro a ser minimizado pelo método RLS
b_0	Constante relacionadas a ESR e C
b_1	Constante relacionadas a ESR e C
k_{re}	Constante da taxa de reação
A	Fator de frequência
E	Energia de ativação
R	Constante de gás
T	Temperatura absoluta
L	Vida útil remanescente do capacitor generalizada (em horas)
$L_{x,T}$	Vida útil remanescente do capacitor em relação a temperatura
$L_{x,i}$	Vida útil remanescente do capacitor em relação a corrente
L_x	Vida útil remanescente total do capacitor
L_o	Vida útil do capacitor nominal especificada no *datasheet*
k_B	Constante de Boltzmann
N_{av}	Constante de Avogrado
T_0	Temperatura de referência
T_x	Temperatura especificada
K_r	Fator determinado experimentalmente que depende do *ripple* de corrente
K_I	Fator que determina a influência do *ripple* de corrente na vida útil do capacitor
K_V	Fator que determina a influência da tensão na vida útil do capacitor
K_T	Fator que determina a influência da temperatura na vida útil do capacitor
$I_{\mathrm{rms,x}}$	Corrente RMS do capacitor em operação com harmônicos corrigidos para base do *datasheet*
$I_{\mathrm{rms,0}}$	Corrente RMS do capacitor no valor nominal
I_{rms}	Corrente RMS do capacitor
P	Potência dissipada pelo capacitor
β	Constante de radiação
A_r	Área de superfície do capacitor em cm^2
R_{th}	Resistência térmica do capacitor
U_x	Tensão nominal do capacitor utilizada para análise do fator de tensão
U_0	Tensão de operação do capacitor utilizada para análise do fator de tensão
n_v	Constante obtida experimentalmente para determinação do fator de tensão
E_0	Constante de simplificação do método Levenberg-Marquadt
E_1	Constante de simplificação do método Levenberg-Marquadt
E_2	Constante de simplificação do método Levenberg-Marquadt
E_3	Constante de simplificação do método Levenberg-Marquadt
ESR_{tx}	ESR a ser transmitida por comunicação UART

ESR_{rx}	ESR a ser recebida por comunicação UART
$conv_{\text{factor}}$	Fator de conversão para transmissão de dados para comunicação UART
$rmsWindow$	Janela móvel a ser aplicado o procedimento de RMS no *software* MATLAB
N_{null}	Quantidade de números vazios no vetor de janela móvel
x_{rms}	Variável genérica RMS

SUMÁRIO

1 INTRODUÇÃO GERAL

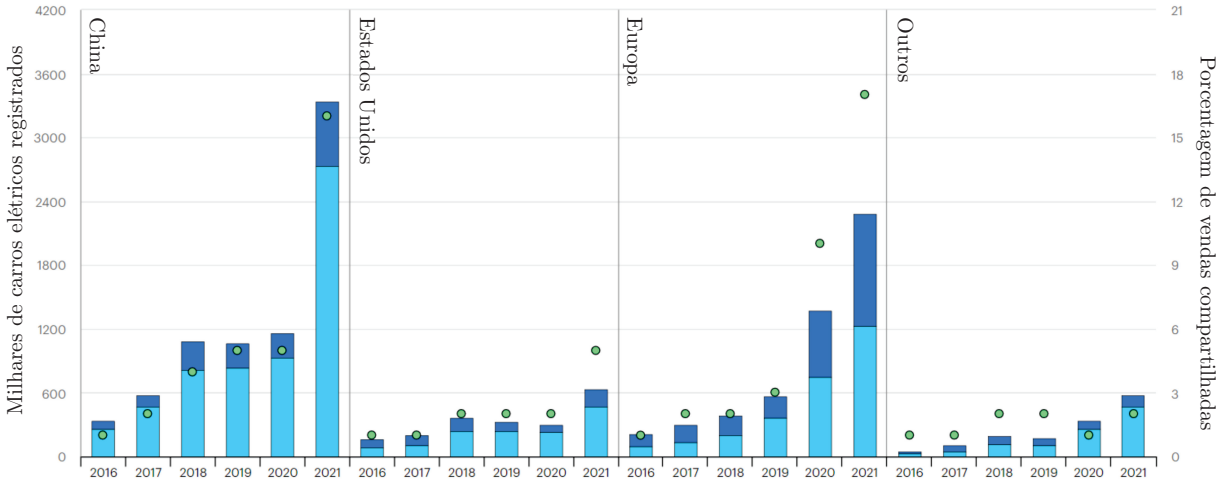
1.1 CONTEXTUALIZAÇÃO

O Veículo Elétrico (VE) é um dos principais focos de pesquisa quando considerados os danos ao meio ambiente que veículos baseados em petróleo causaram nas últimas décadas, fato atrelado principalmente ao excesso de emissão de CO_2 produzida por combustão interna com uso de diesel e gasolina. Além disso, diversos problemas como a possibilidade de escassez futura dos combustíveis fósseis (que por consequência gera também um aumento de preço inevitável segundo a Lei da Oferta e Demanda), globalização, aumento da poluição de ar, aumento populacional global e da quantidade de veículos circulando, forçaram a indústria automotiva a acelerar o passo no desenvolvimento de tecnologias envolvendo VE's (CHAN, 2002; SALMASI, 2007).

Dados mostram que as emissões de CO_2 reduziram de quase 2 bilhões de toneladas em 2021 para apenas 300 milhões em 2022, resultado da rápida recuperação global da crise econômica desencadeada pela pandemia. O aumento das emissões globais de CO_2 este ano seria muito maior (mais do que triplicaria para chegar perto de 1 bilhão de toneladas), se não fosse pelas grandes implantações de tecnologias de energia renovável e VE's em todo o mundo (IEA, 2022). Em 2021 a venda e registro de novos veículos elétricos no planeta bateu um recorde de quase 6,6 milhões (uma porcentagem de quase 9%), atingindo quase o dobro do que no ano anterior e contabilizando um total de quase 16,5 milhões de VE's circulando. A estimativa para 2030 é uma previsão de uma frota de mais de 300 milhões de VE's, contabilizando 60% de novas vendas (IEA, 2022).

A Figura 1 mostra a distribuição do comércio na China, Estados Unidos, Europa e outros nos anos de 2016 a 2021, onde em azul-escuro são dados referentes aos Veículos Elétricos Híbridos Plugin (*Plug-In Hybrid Electric Vehicle* - PHEV), em azul-claro aos Veículos Elétricos a Baterias (*Battery Electric Vehicle* - BEV) e em círculo verde a porcentagem de vendas. Perceba que a maioria do aumento da venda e registro de VE's se encontra na China, atingindo um número de 2,734 milhões só em BEV's e 600 mil PHEV's circulando no ano de 2021.

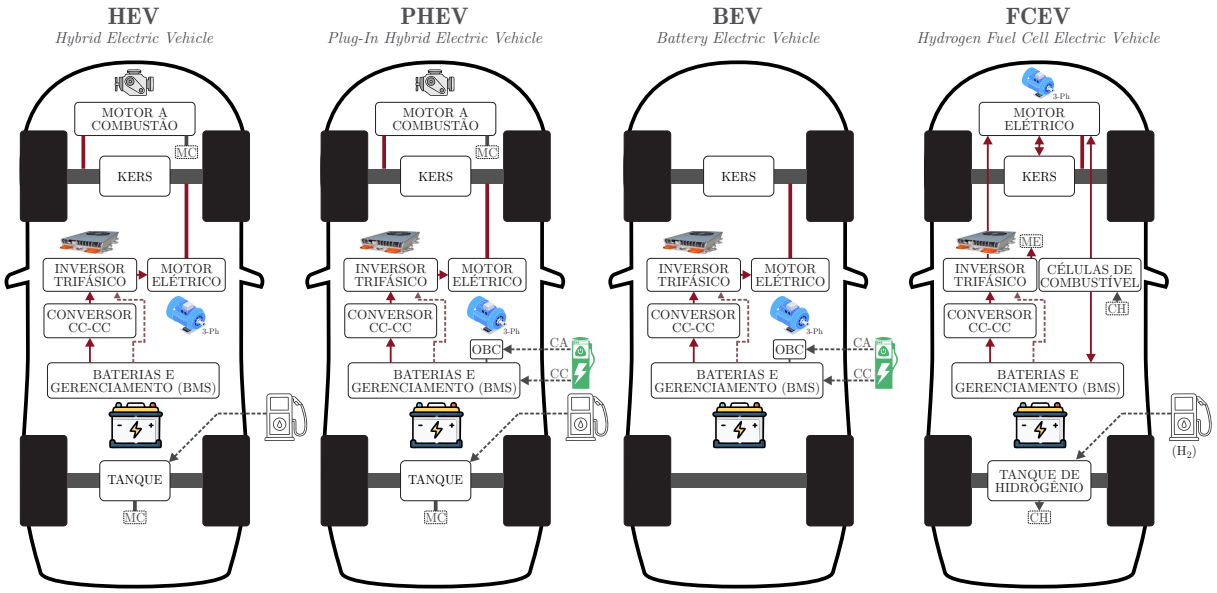
Figura 1 – Vendas e registros de VE's global.



Fonte: Adaptado de IEA (2022).

Essa preocupação com o futuro levou a diversas soluções e tipos de VE's. Dentre elas, expõe-se as 4 principais categorias: Veículo Elétrico Híbrido (*Hybrid Electric Vehicle* - HEV), PHEV, BEV e Veículo Elétrico a Células de Combustíveis (*Hydrogen Fuel Cell Electric Vehicle* - FCEV). A Figura 2 mostra o arranjo dos dispositivos no interior elétrico e mecânico dos tipos de veículos elétricos.

Figura 2 – Classificação de veículos elétricos.



Fonte: do Autor.

Onde, os componentes principais contidos nas topologias e ilustrados na figura são:

- Motor a combustão: tem seu princípio de energia baseado na reação de combustão advinda da mistura de ar e combustível, inflamado pela faísca de uma vela de ignição e produz força o suficiente para movimentar o veículo.
- Motor Elétrico: atualmente a maioria dos motores elétricos utilizados em VE's são de indução (assíncrono), que tem seu funcionamento baseado em CA, já que são os mais comuns, com melhores custo-benefício do mercado, possuindo um controle simples devido ao seu *design* e funcionamento.
- Freio Regenerativo (*Kinetic Energy Recovery System* - KERS): carros convencionais dissipam energia cinética em forma de calor durante a frenagem, nos carros elétricos parte dessa energia pode ser reaproveitada como eletricidade e ajuda a economizar combustível utilizando freios regenerativos.
- Inversor trifásico: dispositivo responsável por transformar a energia CC advinda do sistema de baterias e conversor CC/CC para CA, também controla o motor elétrico.
- Conversor CC-CC: tem caráter bidirecional, mantém a tensão vinda da bateria em um nível estável desejado e também controla a mesma no momento de carga.
- Baterias e Gerenciamento (*Battery Management Systems* - BMS): bloco que contém o sistema de baterias, geralmente dividido entre três baterias (bateria de alta tensão, 12 V e 24 V), incluindo claro os conversores CC/CC internos para regulação da tensão da bateria de alta tensão para as de tensão reduzidas, com objetivo de se utilizar nos sub-sistemas do veículo. O bloco BMS visa gerenciar a vida útil da bateria para longevidade e desempenho.
- *On-Board Charger* - OBC: bloco responsável por converter a energia AC da rede para CC (geralmente entre 400-800 V). De forma geral, quase sempre possuí um retificador, conversor PFC, sistemas de controle e conversor CC/CC isolado.
- Tanque/Tanque de Hidrogênio: Local de armazenamento dos combustíveis e gás de hidrogênio para o FCEV.

1.1.1 Veículo Elétrico Híbrido

O HEV utiliza tanto a bateria quanto a combustão interna para se mover, combinando as qualidades das duas tecnologias (motor a combustão e motor elétrico). Essa configuração permite, por exemplo, desligar o motor a combustão em momentos onde não se é necessário o uso da força total do veículo, alimentando os circuitos internos com a bateria. Ainda, os motores podem até mesmo atuar juntamente para produzir mais potência na aceleração.

Essa variação foi proposta principalmente para vencer os limites da tecnologia da época, que não era suficiente para que um veículo a bateria conseguisse superar em termos

de custos e o longo tempo de espera de recarga na estação (PANDAY; BANSAL, 2014). Os veículos híbridos podem ser separados em três tipos: Veículo Elétrico Híbrido em Série, Veículo Elétrico Híbrido em Paralelo e Veículo Elétrico Híbrido em Série-Paralelo.

O Veículo Elétrico Híbrido em Série (*Series Hybrid Electric Vehicle* - S-HEV) fornece energia durante a aceleração e recebe energia regenerativa durante a frenagem (ou seja, os capacitores são carregados em velocidade baixa e mantidos quase vazios em velocidade alta), reduzindo o estresse na bateria (KEBRIAEI; NIASAR; ASAEI, 2015). Além disso, nessa configuração o sistema de combustão interna apresenta um rendimento muito maior, pois não está conectado ao sistema de transmissão. Entretanto, como não há recarga externa da bateria que move o motor elétrico, a energia é transmitida através do gerador e do motor de tração, agregando volume e aumentando perdas de energia (SELVAKUMAR, 2021). Esse tipo de configuração é geralmente utilizada em veículos pesados como comerciais, militares, ônibus e até locomotivas, já que estas possuem mais espaço para o sistema de motor/geração (GAO; EHSANI; MILLER, 2005). Em resumo, no esquema dos veículos híbridos em série o motor a combustão nunca atua diretamente nas rodas, ele apenas aciona o gerador, que alimenta tanto a bateria quanto o motor elétrico.

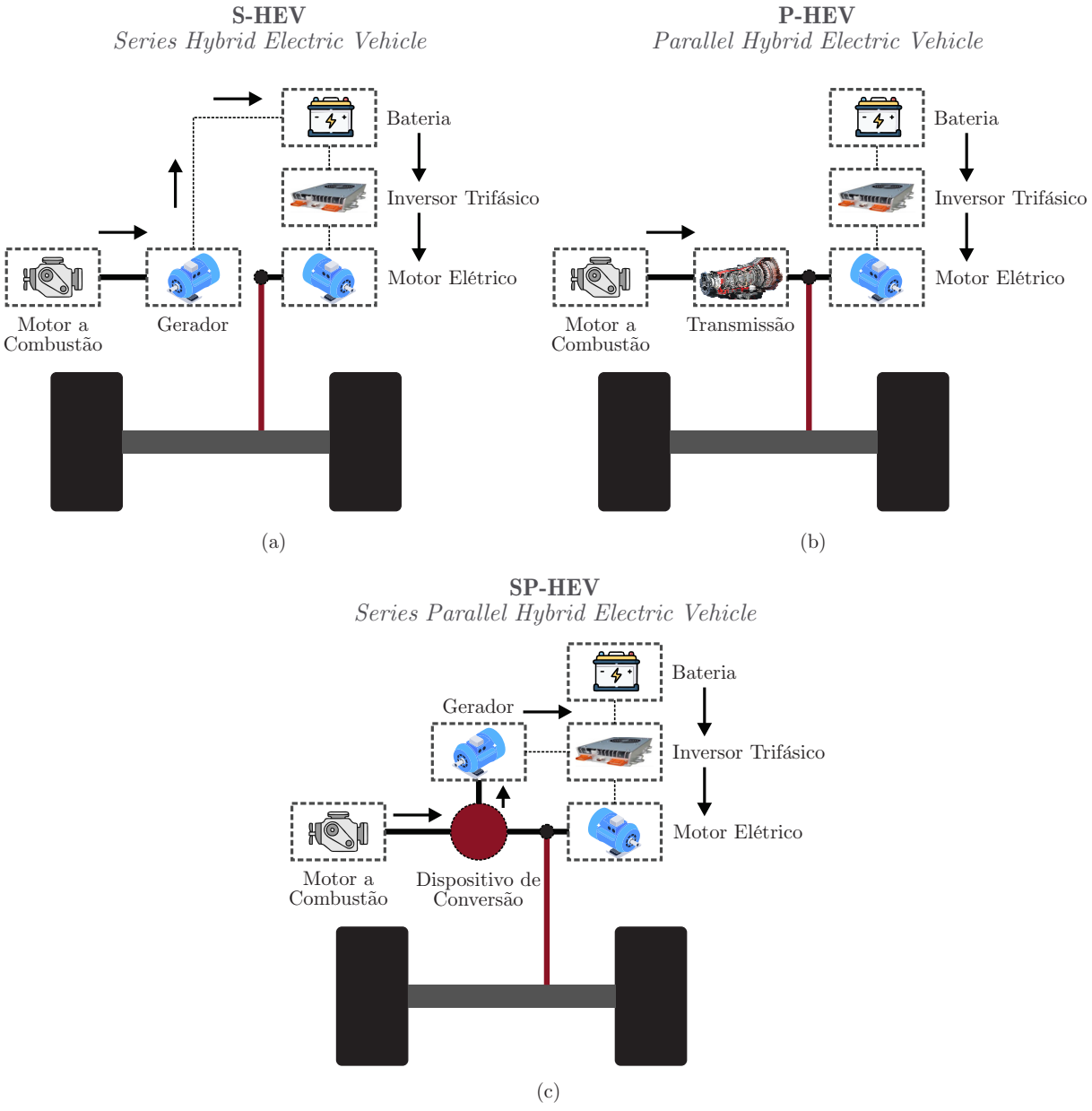
Já o Veículo Elétrico Híbrido em Paralelo (*Parallel Hybrid Electric Vehicle* - P-HEV) ambos motores podem fornecer seu torque para o movimento do veículo, contribuindo para se utilizar um motor menor e com menos perdas de energias. Entretanto, isso também torna a construção do veículo mais complexa, afetando reparo e manutenção (SELVAKUMAR, 2021). Nesse caso, a bateria também pode ser recarregada por frenagem regenerativa e durante movimento (quando a potência do motor a combustão é mais alta que a necessária para propulsão). Uma das desvantagens dessa configuração é que a bateria não pode ser carregada enquanto o carro não está se movendo, já que há acoplamento mecânico entre as rodas motrizes e o motor (sem embreagem) (KEBRIAEI; NIASAR; ASAEI, 2015). Devido a suas características compactas, a configuração híbrida paralela é usado em pequenos veículos, como carros de passageiros (GAO; EHSANI; MILLER, 2005).

Ainda, o Veículo Elétrico Híbrido em Série-Paralelo (*Series-Parallel Hybrid Electric Vehicle* - SP-HEV) é a configuração que une os dois tipos anteriores com um dispositivo de conversão, que leva nome de engrenagem planetária. Esse dispositivo tem como função realizar a conversão da velocidade do motor em torque, desacoplando a velocidade do motor da roda motriz. Quando a velocidade tem sentido contrário ao torque, o sistema opera como gerador, onde uma parte da energia é dividida para a força e outra para o gerador. Já quando a velocidade tem sentido positivo, o sistema opera como motor, entregando potência às rodas motrizes (GAO; EHSANI; MILLER, 2005). Esse tipo pode operar com uma ampla variedade de modos, por exemplo, em altas velocidades o motor assume o controle, enquanto em mais baixas opera como modo série, gerando energia para o inversor. Este sistema é mais caro do que um sistema paralelo puro, pois precisa de um gerador extra, um sistema mecânico que realize o desacoplamento e mais poder de

computação para controlar o sistema duplo (KEBRIAEI; NIASAR; ASAEI, 2015).

A Figura 3 ilustra a classificação de veículos elétricos híbridos citados.

Figura 3 – Classificação de veículos elétricos híbridos.



Fonte: do Autor.

Quando se trata do PHEV a única diferença é que o modelo paralelo possuí uma bateria maior. De qualquer forma, o ponto característico dessa configuração é a possibilidade de recarga da bateria através de fonte externa, as subdivisões (Serie, Paralelo e Serie-Paralelo) permanecem com a mesma lógica.

1.1.2 Veículo Elétrico a Bateria

Os Veículos Elétricos a Bateria são movidos exclusivamente por bateria e não há nenhum tipo de sistema a combustão internamente, onde o movimento depende apenas do motor elétrico. O sistema pode ser recarregado periodicamente em CA ou CC em um posto de recarga, mas também é recarregado com a utilização dos freios regenerativos (KERS). A distância máxima típica de um sistema desse tipo é geralmente em torno 100 a 150 km com carga máxima, mas modelos mais caros podem chegar a 300-350 km. Obviamente, por não possuírem o escapamento necessário para expulsar os gases advindos da utilização de um motor a combustão, esse tipo de veículo não produz nenhum tipo de gás poluente, e ainda que utilize energia não limpa de uma fonte tradicional na recarga da bateria, ainda se considera o modelo base para um futuro mais limpo e mais sustentável que aqueles que utilizam a combustão.

Atualmente as baterias mais utilizadas em VE's são as chumbo ácido, níquel-hidreto metálico (NiMH), lítio-íon e cloreto de sódio-níquel (ZEBRA). Cada uma tem suas particularidades e aplicações específicas, cujas características principais são mostradas na Tabela 3. Mais detalhes podem ser encontrados em Bhatt e El. Darieby (2018).

Tabela 3 – Características dos tipos de baterias utilizadas em VE.

Característica	**Chumbo ácido**	**NiMH**	**ZEBRA**	**Lítio-íon**
Energia Específica (Wh/kg)	30 a 45	30 a 80	90 a 100	90 a 220
Densidade de Energia (Wh/l)	60 a 75	140 a 300	160	280 a 400
Potência Específica (Wh/kg)	180	250 a 1000	150	600 a 3400
Ciclo de Vida	500 a 800	500 a 1000	1000	1000 a 8000
Descarga (%/mês)	2 a 4	20 a 30	0	2 a 5
Range de Temperatura (ºC)	-20 a 60	-20 a 60	270 a 350	-20 a 60
Custo Relativo ($/kWh)	150	500	270	700
Eficiência (%)	85	80	75 a 85	93

Fonte: Mezaroba (2021)

Por fim, a Tabela 4 mostra uma comparação entre as características principais dos veículos EVs, HEVs/PHEVs e FCVs citados até então.

Tabela 4 – Comparação das características principais dos EVs, HEVs/PHEVs e FCVs.

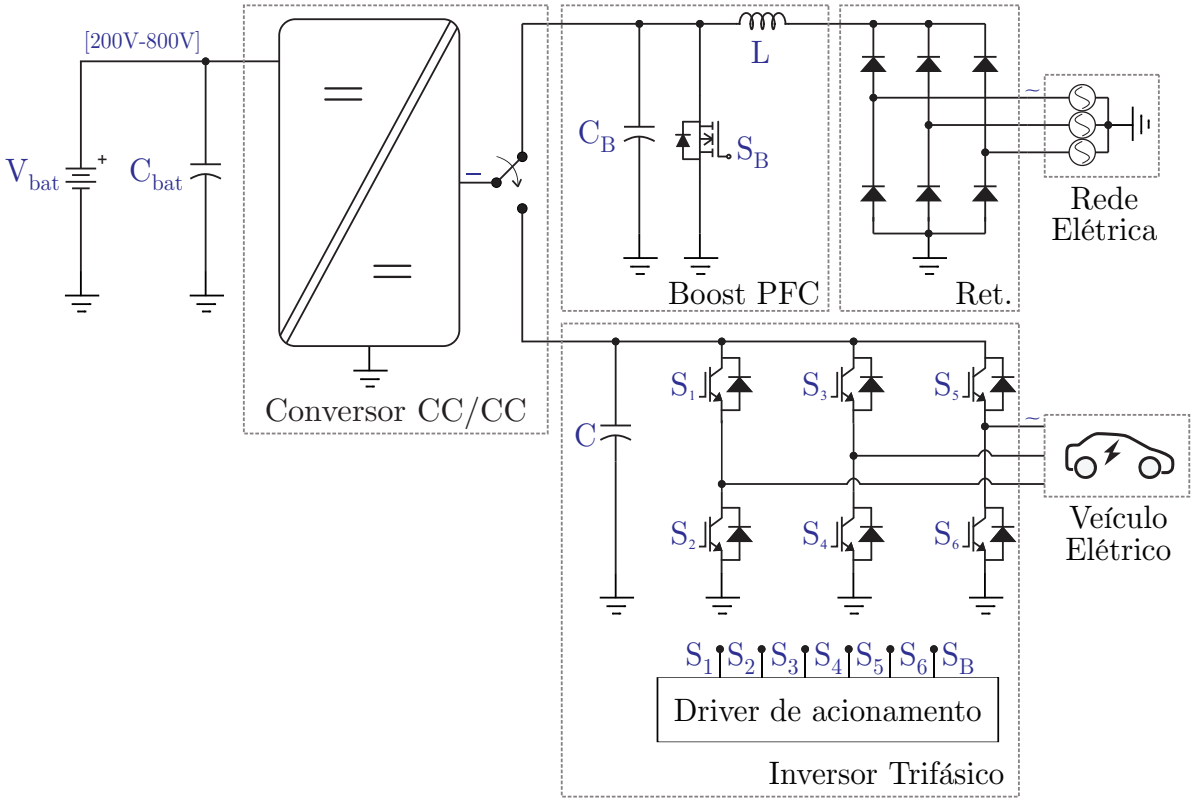
	EV	**HEV**	**FCV**
Propulsão	✓Motor elétrico	✓Motor elétrico ✓Mecânismo interno de combustão	✓Motor elétrico
Sistema de armazenamento	✓Bateria ✓Supercapacitor	✓Bateria ✓Supercapacitor ✓Fóssil ou combustíveis alternativos	✓Tanque de Hidrogênio ✓Bateria/Supercapacitor (necessário para potencializar a densidade de potência)
Características	✓Comercialmente disponível ✓Eficiência alta ✓Independe de combustível fóssil ✓Pouca distância efetiva ✓Alto custo inicial ✓Comercialmente disponível	✓Comercialmente disponível ✓Grande economia de combustível ✓Dependente de combustível fóssil ✓Grande distância efetiva ✓Maior custo que veículos a combustão	✓Zero emissão de poluentes ✓Eficiência alta ✓Independe de combustível fóssil ✓Alto custo ✓Comercialmente disponível, porém há escassez de postos de recarga disponíveis
Infraestrutura de fonte de energia	✓Instalações de carregamento da rede elétrica	✓Estação de gasolina ✓Posto de recarga	✓Hidrogênio
Maiores problemas	✓Tamanho da bateria e gerenciamento ✓Posto de recarga ✓Alto custo ✓Vida útil da bateria	✓Tamanho da bateria e gerenciamento ✓Controle, otimização e gerenciamento de múltiplas fontes de energia	✓Custo da célula, vida útil e confiabilidade ✓Produção de hidrogênio ✓Alto custo

Fonte: Adaptado de Kebriaei, Niasar e Asaei (2015).

1.1.3 Inversores Trifásicos e a Vida Útil dos Capacitores de Barramento

Nesse trabalho, as topologias de interesse são os modelos PHEV e BEV pela presença da conexão com a rede elétrica para carga da bateria. Perceba que nesses modelos um dos elementos-chave para a conversão de energia e controle do motor são os inversores de tensão trifásicos. Como exemplo, considera-se um VE em uma visualização do esquemático do circuito elétrico da Figura 2, onde em praticamente todos os casos é possível encontrar a seguinte configuração para recarregamento da bateria em alimentação CA: retificador trifásico - conversor *boost* para correção de fator de potência (PFC) - conversor CC/CC bidirecional para controle de carga da bateria - inversor trifásico (o qual é conectado ao veículo elétrico). A Figura 4 apresenta um modelo genérico desse tipo.

Figura 4 – Esquemático do circuito elétrico de um VE alimentado em CA e durante a conexão com a rede para carga da bateria principal.



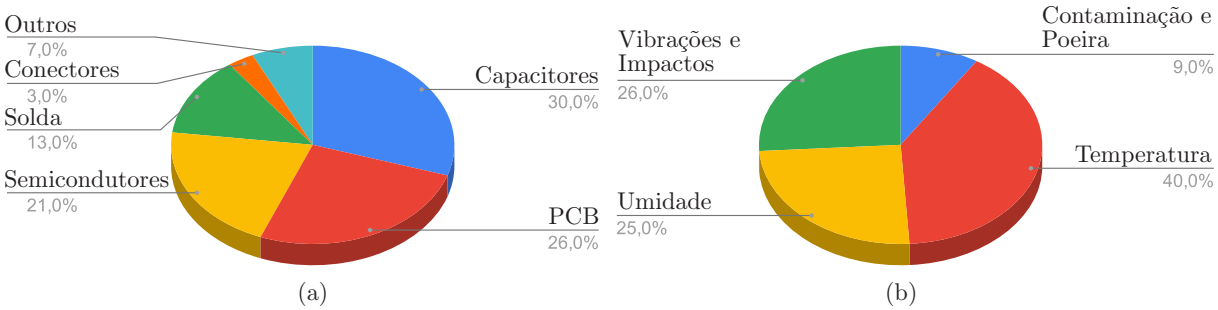
Fonte: do Autor.

Na Figura 4 no canto inferior direito, observa-se um modelo genérico de um inversor de tensão trifásico, responsável por converter a tensão CC advinda da bateria e do conversor CC/CC para a conexão com o carro elétrico alimentado em CA. A tensão do barramento desse dispositivo é mantida pelo capacitor (C), elemento que terá principal foco nessa dissertação e representa uma parte considerável na análise de vida útil total do conversor em questão. O barramento CC de um sistema de acionamento de um veículo elétrico normalmente inclui capacitores de filtro volumosos com grandes perdas e indutâncias parasitas. A faixa de operação da frequência de comutação típica de IGBTs de Silício (Si) em aplicações EV/HEV se encontra geralmente entre 5-10 kHz (STEWART et al., 2017).

Capacitores utilizados em barramentos CC são componentes fundamentais na maioria dos conversores eletrônicos, já que possuem a função de suprimir variações de tensão, absorver harmônicos e realizar o balanço de potência instantânea do sistema. Em algumas aplicações, eles também podem prover energia suficiente para transientes e operações anormais (ZHAO et al., 2020). Esses capacitores também apresentam a maioria das falhas de um conversor (como mostra a Figura 5) e podem até mesmo ser relacionados a predição da vida útil de um conversor, já que seu mau funcionamento ocasiona problemas no funcionamento do sistema na totalidade (VENET; DAMAND; GRCLLCT, 1993; VENET, P.

et al., 2002; LAHYANI et al., 1998; HARADA; KATSUKI; FUJIWARA, 1993; IMAM, A. M. et al., 2007; IMAM, A. et al., s.d.; GASPERI, 2005).

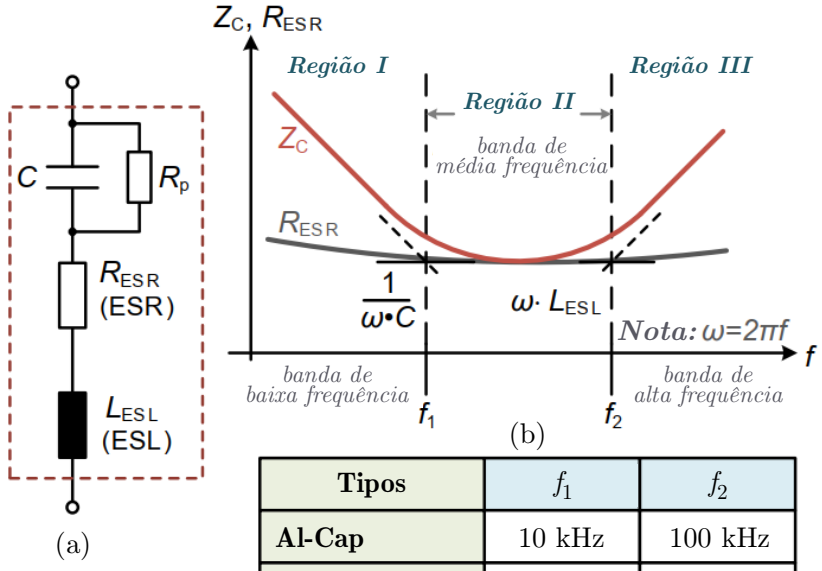
Figura 5 – Distribuição de ocorrência de falhas em inversores: (a) Distribuição de falhas em componentes, (b) Distribuição de falhas por motivação.



Fonte: Adaptado de Sundararajan et al. (2018) e Kim et al. (2020).

Dentre os diversos tipos de capacitores existentes, os mais utilizados em barramento de inversores são Capacitores Eletrolíticos de Alumínio (*Aluminum Electrolytic Capacitors* - Al-Caps), Capacitores de Filme de Polipropileno Metalizado (*Metallized Polypropylene Film Capacitors* - MPPF-Caps) e os Capacitores Cerâmicos Multicamadas (*Multi-Layer Ceramic Capacitors* - MLC-Caps). O tipo de capacitor que irá compor o barramento CC do inversor deve ser escolhido com base nas necessidades específicas de cada aplicação, onde variáveis como variação de tensão, temperatura, estresses mecânicos e elétricos são considerados (WANG, H.; BLAABJERG, 2014). O esquemático elétrico simplificado desses três tipos pode ser representado como na Figura 6, onde também é ilustrado o comportamento da impedância na frequência. Observa-se que a impedância (Z_C) de um capacitor pode ser aproximada por uma reta para frequências baixas ($f < f_1$), enquanto a Resistência Equivalente em Série (ESR) (R_{ESR}) se aproxima de uma exponencial. A Z_C tem uma influência significativa nas frequências baixas, enquanto que a R_{ESR} é dominante nas frequências médias. Já nas frequências altas, a Indutância Equivalente em Série (ESL) (L_{ESL}) é a mais dominante. Em eletrônica de potência para sistemas de potências médias e altas, a faixa de frequência de interesse geralmente é baixa, e por conta disso, a influência da indutância serie equivalente (L_{ESL}) pode ser desconsiderada.

Figura 6 – Circuito elétrico equivalente e característica de impedância de capacitores: (a) Circuito elétrico equivalente, (b) Característica de impedância na frequência, (c) Valores típicos de f_1, f_2 em 25ºC, onde os tipos de capacitores são Al-Cap, MPPF-Cap, e MLC-Cap são SLPX (470 µF/450 V), B32778-JX (480 µF/450 V) e KCM55WC71E107MH13 (100 µF/25 V) respectivamente (ZHAO et al., 2020).



(a) (b)

Tipos	f_1	f_2
Al-Cap	10 kHz	100 kHz
MPPF-Cap	88,5 kHz	99,4 kHz
MLC-Cap	225 kHz	300 kHz

(c)

Fonte: Adaptado de Zhao et al. (2020).

Onde:

- C: Capacitância;
- R_p: Resistência de isolamento;
- R_{ESR}: Resistência Série Equivalente (ESR);
- L_{ESL}: Indutância Série Equivalente (ESL);

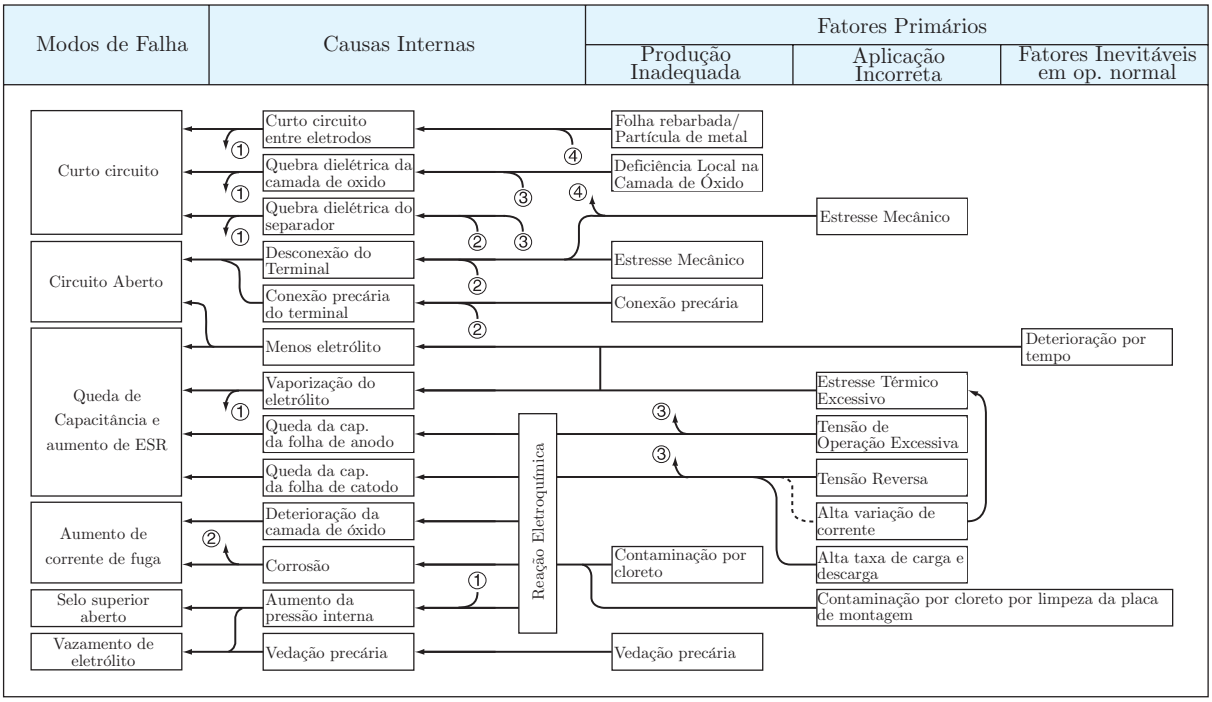
Um capacitor eletrolítico é composto por uma folha de ânodo, cátodo e uma placa separadora com eletrólito. O eletrólito contém um solvente e um soluto, como etilenoglicol e borato de amônio, respectivamente. Um aumento na temperatura interna acelera a evaporação do eletrólito do capacitor. O volume do eletrólito é reduzido, de modo que os túneis gravados não são totalmente preenchidos com o eletrólito. A redução da área de superfície efetiva, que atua como um eletrodo, diminui a capacitância (ABO-KHALIL, 2012). Em alta temperatura, o processo de evaporação é muito mais acelerado, então o tempo de vida é mais encurtado (PARLER, S., 2002). A vida útil nominal dos capacitores eletrolíticos é normalmente de 1.000–10.000 h, não sendo o suficiente para a maioria das aplicações (SHOYAMA; NAKA; NINOMIYA, 2004). Geralmente poucas informações sobre

os capacitores eletrolíticos são disponibilizadas pelos fabricantes, reforçando a necessidade do diagnóstico de deterioração do componente, precisando ser investigado para manutenção do conversor e previsão da vida útil dos mesmos (KIEFERNDORF; FORSTER; LIPO, 2002; AELOIZA et al., 2005). Em suma, Al-Caps podem atingir a maior densidade de energia e menor custo por Joule, entretanto, com relativamente altos valores de ESR, baixa ondulação de corrente e problemas de desgaste devido à evaporação do eletrólito (WANG, H.; BLAABJERG, 2014).

O aumento da ESR em Al-Caps é provocado pela evaporação da solução eletrolítica do capacitor por conta da passagem do tempo ou aceleração do processo por estar sujeito à alta temperatura. Esse fenômeno, além de aumentar a ondulação de tensão por conta da queda de tensão no componente, também dissipa calor e aumenta a temperatura do mesmo, afetando a operação e danificando o capacitor.

A Figura 7 mostra um resumo com modos de falha, causas internas e fatores primários como produção do componente inadequadamente pelo fabricante, uso em aplicações incorretas e fatores inevitáveis em operação normal.

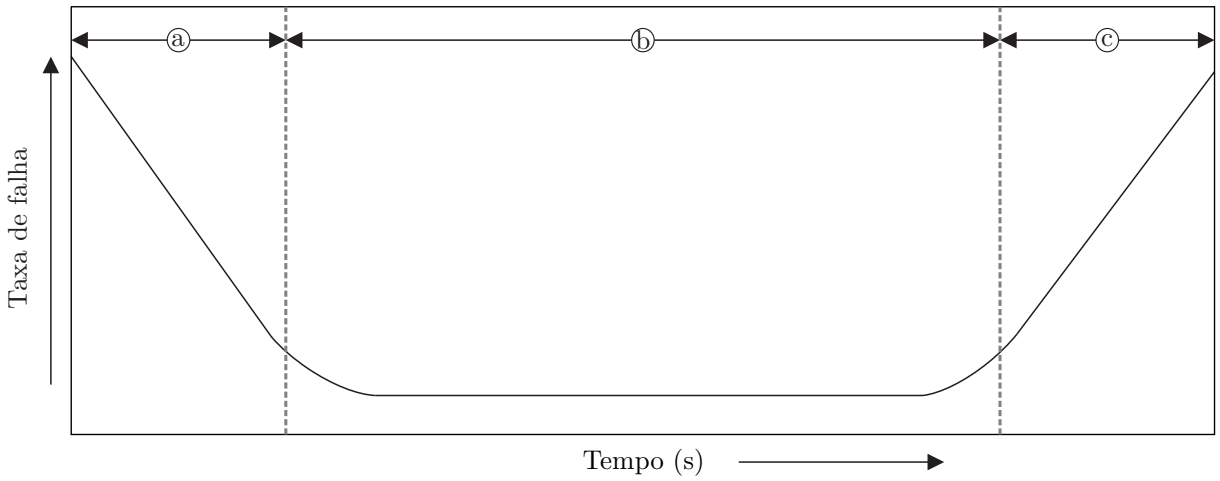
Figura 7 – Distribuição das motivações de falhas em capacitores eletrolíticos.



Fonte: Adaptado de Chemi-Con (2013).

Quando se trata da vida útil do capacitor eletrolítico em específico, a taxa de falhas em relação ao tempo apresenta um formato de banheira, como mostra a Figura 8.

Figura 8 – Probabilidade de falhas em capacitores eletrolíticos.



Fonte: do Autor.

As etapas de probabilidade de falha nos capacitores eletrolíticos podem ser divididas em três partes (SPANIK; FRIVALDSKY; KANOVSKY, 2014):

- (*a*) - Período de Falha Precoce: nesse intervalo as falhas acontecem geralmente por defeitos de fabricação e os componentes são removidos antes da distribuição.
- (*b*) - Período de Falha Aleatório: nesse intervalo a probabilidade de erro geralmente é baixa. Os capacitores eletrolíticos apresentam menos falhas nesse período que semicondutores e capacitores de tântalo.
- (*c*) - Período de Falha de Desgaste: por fim, nessa etapa o capacitor finalmente chega ao fim de sua vida útil, momento em que o eletrólito foi gradualmente evaporado e difundido, o que consequentemente leva a diminuição da capacitância.

Já os capacitores MPPF para uso em sistemas de potência e baixa frequência geralmente tem formato estrutural cilíndrico ou quadrado. Ambos possuem extremidades metalizadas que atuam como pontos de conexão para cabos externos. Esse tipo de capacitor usa filmes de polímeros muito finos, normalmente em torno de 8 µm de espessura, metalizados com alumínio ou liga metálica, com 20 nm ou menos de espessura. O filme metálico pode ter uma resistividade entre cerca de 2 Ω/metro e 10 Ω/metro. A espessura deste filme confere uma propriedade importante aos capacitores MPPF, a de autorrecuperação de falhas pontuais no dielétrico, que ocorre quando ocorre uma quebra parcial do dielétrico. A carga forma uma corrente através do canal de descarga, gerando efeito Joule, fazendo com que a camada de metal ao redor da ruptura evapore (BROWN, 2007). O núcleo do capacitor é feito de dois filmes de polipropileno de material isolante não polar; cada filme de polipropileno é coberto na superfície com uma camada de metal de espessura nanométrica por tecnologia de evaporação a vácuo, cada filme possui uma

camada de metalização muito fina em um lado com uma borda vazia no outro lado. Os filmes cobertos são co-enrolados na haste cilíndrica por enrolamento não indutivo. Um arco de plasma é usado para pulverizar um revestimento de zinco em cada extremidade do cilindro laminado. A metalização escapulada se conecta seletivamente à borda total de cada filme coberto. Os núcleos dos capacitores são conectados em paralelo para formar um MPPF por soldagem de fios de folha de cobre. Com exceção da parte central, o resto do invólucro é isolado por aspiração e enchimento de cola isolante (SUN, 2021).

O aumento da ESR ou redução da capacitância em um capacitor MPPF é geralmente provocada por reação eletroquímica devido a estresses sofridos durante a operação, que transforma lentamente o alumínio em alumina, enquanto o dielétrico permanece intacto. Esse processo é conhecido como corrosão e depende da temperatura aplicada, magnitude da tensão CA e frequência do sinal. Uma tensão CC aplicada com o mesmo valor de pico da tensão alternada não pode iniciar ou auxiliar na reação de corrosão (MAKDESSI et al., 2015).

A variação da ESR mostra que a diminuição da capacitância se deve principalmente ao aumento da espessura da interface eletrodo/dielétrico e não a uma perda de superfície. Com a avaliação da superfície do óxido, a distância entre a metalização e o dielétrico aumentará, causando a diminuição progressiva da capacitância. De fato, a oxidação de metalização iniciada pela incorporação de umidade na superfície do metal (alumínio) é um caso especial de transição de fase que envolve não apenas a mudança de propriedade física, mas também uma mudança química permanente (MAKDESSI et al., 2015).

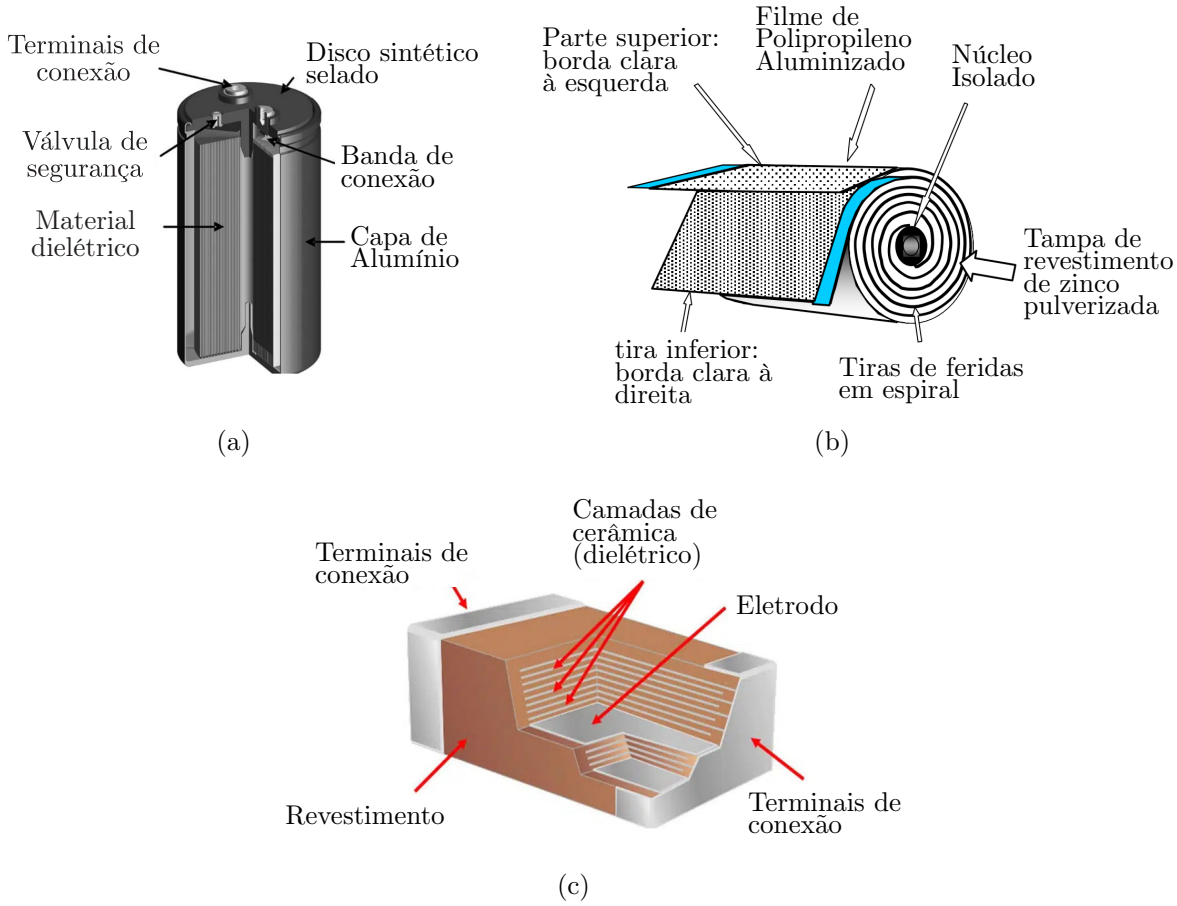
De forma geral, capacitores MPPF possuem características de autorrecuperação, alta confiabilidade, longa vida útil (pode chegar a mais de 10 anos (DANG; KWAK, 2020)), alta densidade de energia, e é amplamente utilizado em sistemas de energia pulsada, porém, sua ESR é geralmente muito baixa (em torno de 10^{-5} (SUN, 2021)), e a maioria dos algoritmos presentes na literatura não a consideram ou tem grande imprecisão em medi-la (a estimativa de vida útil é baseada apenas na estimação de capacitância).

Por fim, os capacitores MLC são os que usam o material de cerâmica como dielétrico. São formados basicamente pelo material dielétrico, eletrodo interno e externo. Esses capacitores geralmente tem capacitâncias extremamente baixas (1 nF e 1 μF (GUDAVALLI; DHAKAL, 2018)) e são compactos, pouco volume, elevada estabilidade e valores de ESR e ESL extremamente baixos, com faixa de tensão semelhante a de Al-Caps (JING; WANG, Q.; RAN, 2020; GUDAVALLI; DHAKAL, 2018). Em contrapartida, devido ao custo dos metais preciosos usados como eletrodo (tradicionalmente uma liga de 85% de prata/15% de paládio), os MLCs não têm sido tradicionalmente usados em circuitos de alta potência (STEWART et al., 2017).

O interior dos três tipos de capacitores mencionados são apresentados na Figura 9 enquanto a Figura 10 mostra o desempenho dos Al-Caps, MPPF-Caps e MLC-Caps.

Capacitores MPPF e MLC possuem ESRs muito baixas e geralmente tem suas

Figura 9 – Interior dos capacitores: (a) Al-Caps, (b) MPPF-Caps, (c) MLC-Caps.



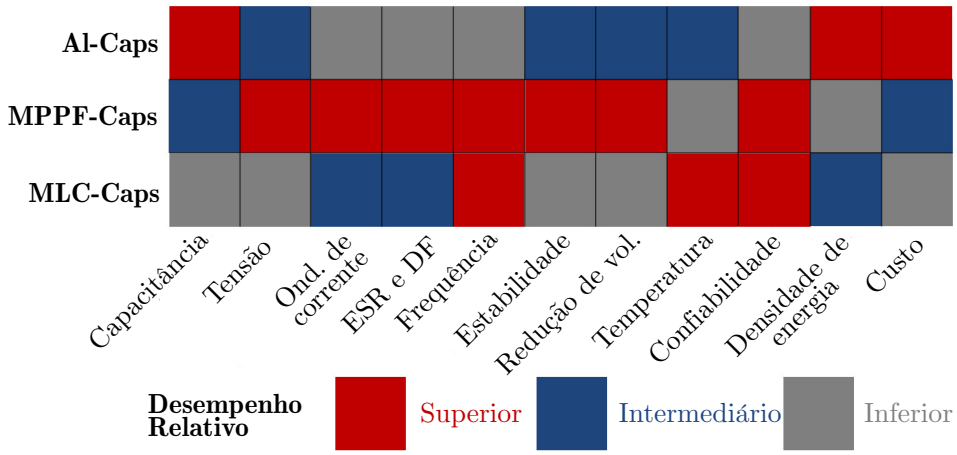
Fonte: Adaptado de Abdennadher et al. (2010), Brown (2007) e Gudavalli e Dhakal (2018).

vidas úteis longas e confiabilidade o suficiente para não ser necessário um monitoramento contínuo, porém a grande maioria da indústria de inversores de tensão ainda utilizam capacitores eletrolíticos por conta do baixo custo por capacitância e grande capacidade de armazenar energia por volume. Com base no exposto, muitos autores propõem diferentes métodos/algoritmos para determinar a ESR dos capacitores eletrolíticos (VENET; DAMAND; GRCLLCT, 1993; VENET, P. et al., 2002; LAHYANI et al., 1998; HARADA; KATSUKI; FUJIWARA, 1993; IMAM, A. M. et al., 2007; IMAM, A. et al., s.d.; GASPERI, 2005). Entretanto, muitos parâmetros, como sensoriamento adicional, e uma requisição computacional são necessárias, o que torna mais complexo o processo de instrumentação e monitoramento, além de tornar o sistema mais caro (ABDENNADHER et al., 2010).

Para capacitores eletrolíticos de alumínio, um critério do fim da vida útil amplamente aceito é quando a capacitância sofre uma redução de 20% ou quando sua ESR é dobrada. Para capacitores de filme, 2% a 5% já podem indicar o fim da vida útil do componente.

Sistemas modernos de fontes de alimentação ininterruptas (*Uninterruptible Power*

Figura 10 – Desempenho dos três tipos (Al-Caps, MPPF-Caps e MLC-Caps) de capacitores citados.



Fonte: Adaptado de Huai Wang e Blaabjerg (2014).

Supply - UPS) podem utilizar uma grande variedade de sensores, e com isso, pode-se utilizar poder computacional para aproveitar o sensoriamento já existente e realizar técnicas de predição rodando no *background* e em tempo real (ABDENNADHER et al., 2010). Uma revisão dos diversos métodos propostos é realizada por (SOLIMAN; WANG, H.; BLAABJERG, 2016) demonstrando uma comparação entre os algoritmos e uma estimação de erros de cada um é discutida em detalhes. Além disso, a utilidade para determinação da indicação da vida útil também é explorada. A evolução dos métodos de avaliação desde o ano de 1993 a 2016 são listadas. Na visão dos autores, os seguintes pontos são levantados:

1. Métodos baseados em *software* com redução ou poucos esforços adicionais de *hardware* são atraentes para aplicações da indústria que exigem alto desempenho e confiabilidade. As vantagens deste tipo de métodos estão divididos em duas partes:

   a) Pode ser aplicado para ambos os novos conversores de energia ou já existentes;

   b) É uma tendência, visto que o custo de controladores digitais e recursos de computação estão reduzindo.

2. Métodos com sensoriamento de corrente de baixo custo e indutivo podem superar muitas das deficiências de métodos baseados em sensores atuais.

3. Implementação integrada de monitoramento de condição, proteção e outras funções auxiliares para capacitores em aplicações.

Um artigo publicado em 2020 (ZHAO et al., 2020) também mostra uma revisão dos métodos existentes até o momento, atualizando a pesquisa com novas metodologias. Na visão dos autores, o objetivo principal de empregar um circuito de MC (Monitoramento

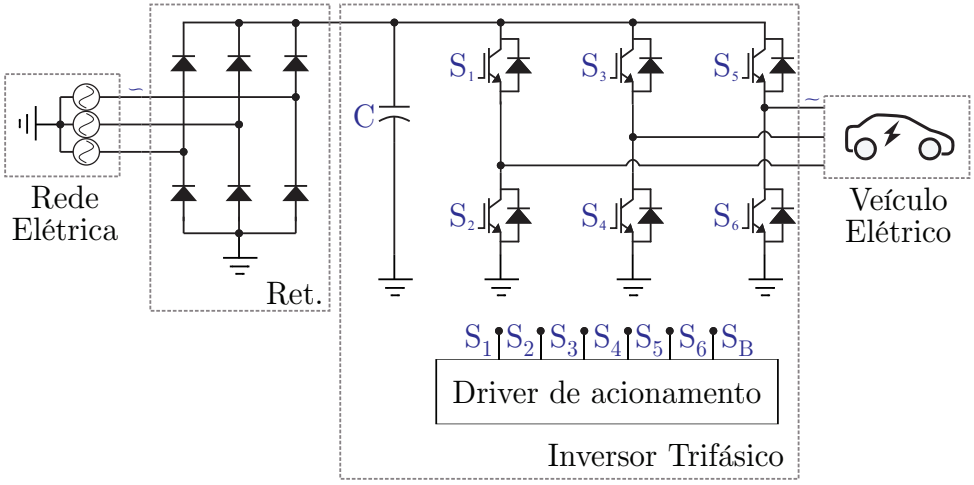
de Condição) em aplicações industriais é estimar com precisão os parâmetros do capacitor sem custo adicional e risco de confiabilidade, ou seja, sem circuitos de *hardware* e sensores adicionais.

Dentre os principais trabalhos analisados nos artigos de revisões citados, ressaltam-se as três principais características que estão sendo buscadas para alcançar o objetivo dessa dissertação:

- Sistema AC/CC/AC (onde há configuração rede elétrica - retificador - inversor);
- Métodos *online* (realizar medições enquanto o sistema funciona) e não-invasivos (sem possibilidade de alterar controle, modulação, injetar corrente ou tensão em nenhuma parte interna do circuito - este trabalho busca um sistema capaz de atuar apenas como um observador, sem alterar o *hardware* existente);
- Capacidade de estimar capacitância e ESR simultaneamente em capacitores.

Com base nessas premissas, propõe-se a modificação da estrutura padrão apresentada na Figura 4 criando um *bypass* entre a saída do retificador com a entrada do barramento CC do inversor trifásico enquanto o VE está parado e alimentado em CA, para estimativa da ESR, capacitância e vida útil do capacitor. A Figura 11 apresenta a modificação proposta, que se define como uma topologia padrão de um conversor CA/CC/CA.

Figura 11 – Conversor CA/CC/CA conectado a um VE.



Fonte: do Autor.

Na Tabela 5 é apresentado os principais trabalhos envolvendo conversores AC/CC/AC, suas principais características, comparações, vantagens e limitações.

Tabela 5 – Comparação dos principais artigos envolvendo conversores AC/DC/AC.

Referencias	**Resumo**	**Vantagens**	**Limitações**
Sundararajan et al. (2020)	Utiliza algoritmo de Goertzel para estimar Capacitância e ESR.	Ambos valores de ESR e Capacitância podem ser obtidos.	Sensor de corrente no capacitor.
Ahmad et al. (2018)	A segunda harmônica da rede elétrica é extraída para estimar a capacitância.	A técnica proposta é não-invasiva e não precisa de sensores adicionais.	Apenas válido para conexões monofásicas onde existe segunda harmônica.
Wechsler et al. (2012)	Oscilação de tensão e corrente no capacitor e durante operação normal são medidas e calculadas para estimar a capacitância de MPPF-Caps.	Não precisa de interferência externa (modificação no sistema de controle).	Modificações nos sensores e alta frequência de amostragem é necessária (realiza processamento dos dados em tempo real). Como o trabalho teve foco em MPPF-Caps, não teve preocupação em medir ESR.
Sang Bin Lee et al. (2011), Yu et al. (2012) e Kwang-Woon Lee et al. (2008)	(*)	Não necessita de *hardware* e sensores adicionais e o método de controle é simples.	Os métodos só são aplicados quando os motores são parados e precisa haver modificação e interferência invasiva no controle e modulação do sistema.
D.C. Lee et al. (2005), Xing-Si Pu et al. (2013) e Xingsi Pu et al. (2009)	Estima a capacitância e ESR injetando uma componente AC de 30 Hz no capacitor.	Não necessita de *hardware* e sensores adicionais.	Esta metodologia só pode ser utilizada sem carga e pode violar limites de distorção harmônica.
Nguyen e Dong-Choon Lee (2015)	Durante a operação regenerativa dá maquina AC, o motor é usado para gerar a componente CC no barramento para monitorar o capacitor.	Não precisa de *hardware* adicional e tem alta precisão.	Precisa que a máquina opere no modo regenerativo e adiciona a existência de grandes componentes subharmônicas. Também trabalha com injeção de corrente no sistema e modifica a modulação.
Li et al. (2022)	Um sinal de baixa frequência é injetado via controle para estimar a capacitância.	O método pode ser aplicado enquanto o sistema está em operação normal e a injeção do sinal não produz distorção harmônica significativa.	Também trabalha com injeção de corrente no sistema e modifica a modulação. Além disso, a ESR não pode ser calculada, pois o sinal injetado é de extrema baixa frequência, região onde a impedância capacitiva domina.
Meng e Zhang (2021)	(**)	Funciona com o circuito em operação normal. Não precisa de sensores adicionais.	(***)

Fonte: Adaptado de Li et al. (2022).

Onde:

* ESR e capacitância são obtidas através da oscilação de tensão e corrente do

capacitor, as medidas são realizadas modificando a modulação do inversor quando o motor está parado (isso facilita as medições, pois a temperatura do capacitor e do motor são próximos, e essa pode ser estimada com base na resistência do estator, reduzindo o número de sensores totais, além de manter um ambiente totalmente controlado com ruídos reduzidos).

** Este método é baseado na estratégia de controle intermitente ON-OFF, que fornece um circuito de descarga específico para o banco de capacitores do barramento CC. A corrente de descarga é a corrente de uma fase especificada, mensurável em um sistema típico. O estágio de descarga normal dos capacitores do barramento CC é modificado ativamente, e o processo transitório é usado para a estimativa online da capacitância do barramento CC.

*** Modifica o controle e modulação do sistema durante o período de descarga do capacitor com uma sequência ON-OFF com razão cíclica constante para medir a capacitância (convertendo a corrente não medida do capacitor para a corrente de fase). Não é possível medir a ESR com esse método.

Com base nos pontos levantados e os métodos já existentes na literatura, essa dissertação busca investigar, comparar e validar dois métodos que se destacaram na revisão bibliográfica, ambos com abordagens diferentes para diagnosticar e estimar a vida útil de capacitores eletrolíticos para aplicações em veículos elétricos com conversores CA/CC/CA, um utilizando Método dos Mínimos Quadrados Recursivo (*Recursive Least Square* - RLS) (adaptação de (YU et al., 2012)) e outro baseado na utilização da Transformada de Fourier Discreta (DFT - *Discrete Fourier transform*) (adaptação de (SUNDARARAJAN et al., 2020)).

1.2 PROPOSTA DO TRABALHO E OBJETIVOS

O trabalho tem como objetivo geral estudar, implementar e comparar duas técnicas para diagnóstico e previsão da vida útil de capacitores empregados em conversores CA/CC/CA utilizando a conexão proposta na Figura 11. A fim de proporcionar uma visão mais ampla do trabalho proposto, pode ser detalhado em objetivos específicos:

1.2.1 Objetivos específicos do trabalho

- Levantar os estudos já realizados na literatura, os problemas enfrentados e soluções apresentadas para previsão e diagnóstico da vida útil de capacitores.
- Propor melhorias nas técnicas existentes e implementações práticas para predição dos parâmetros que descrevem a vida útil de capacitores;
- Desenvolver os algoritmos de predição de forma otimizada em tempo real, com armazenamento na nuvem;

- Investigar as técnicas através do equacionamento completo e validar com resultados de simulações;
- Implementar a experimentação em bancada visando validar o sistema proposto na prática, utilizando a conexão proposta na Figura 11, sem alteração do circuito de potência original.

1.3 JUSTIFICATIVA

Como demonstrado na introdução geral, capacitores são os componentes que mais causam falhas nos inversores, esses são utilizados para realizar a interface entre o sistema de baterias e conversor CC/CC com o motor elétrico de um VE. Assim, como justificativa principal, essa dissertação busca métodos na literatura que se preocupam em estimar a vida útil do capacitor nesse caso em específico operando como um observador externo, prevendo o momento em que o componente irá falhar antes de causar maiores danos ao circuito.

1.4 ESTRUTURA DA DISSERTAÇÃO

A estrutura desta dissertação está dividida em cinco capítulos, incluindo esse introdutório que apresenta a contextualização e revisão bibliográfica do estudo em questão.

O capítulo 2 apresenta o desenvolvimento matemático das metodologias estudadas, desde a construção do vetor referente a relação de ESR e capacitância para diversas temperaturas, o desenvolvimento matemático dos métodos de estimativa e a extrapolação dos dados obtidos para obtenção do momento em que o capacitor irá falhar.

Realizada a análise matemática, o capítulo 3 apresenta os resultados de simulação ideais para ambos os métodos.

Já o capítulo 4 tem em vista validar as simulações e a teoria desenvolvida com resultados experimentais em bancada no laboratório.

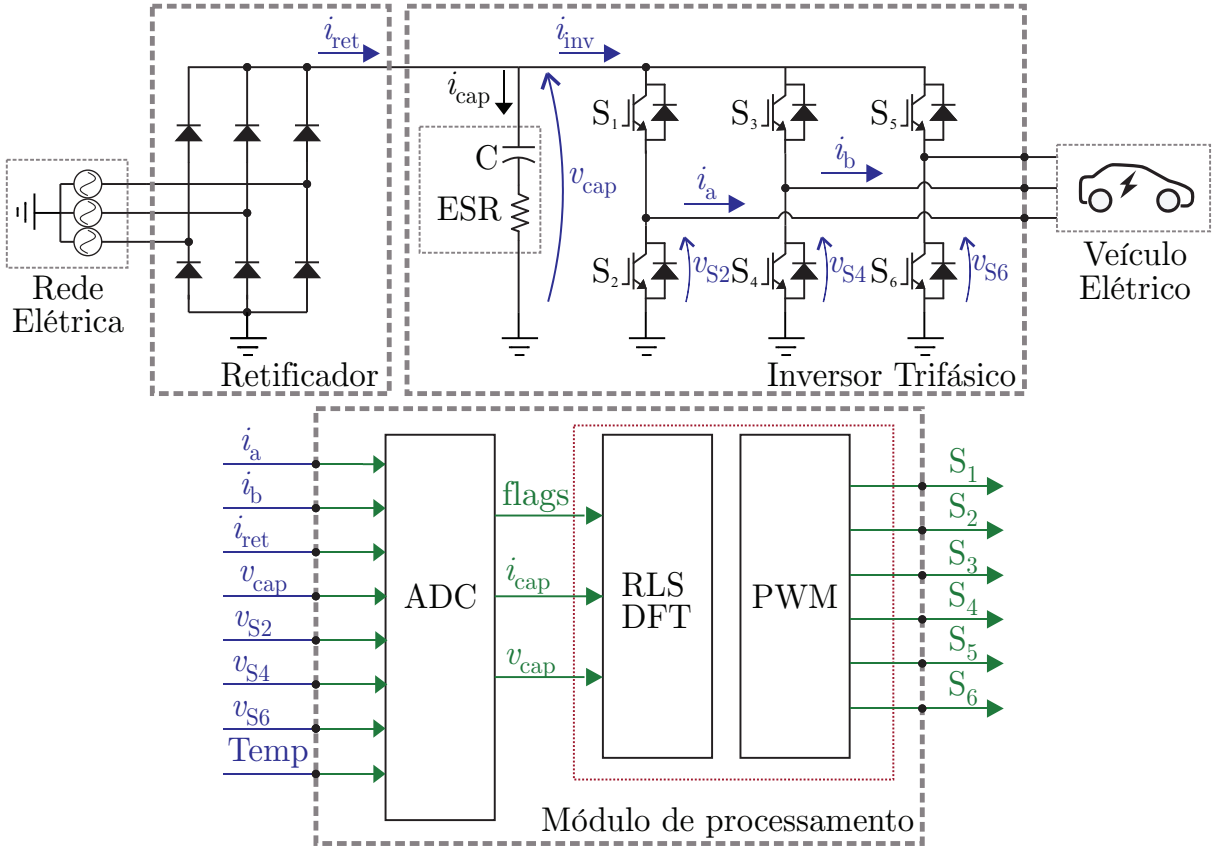
Para finalizar a dissertação, o capítulo 5 concentra as principais conclusões obtidas do estudo e apresenta sugestões para trabalhos futuros.

2 ANÁLISE E INVESTIGAÇÃO DAS TÉCNICAS DE ESTIMAÇÃO DOS PARÂMETROS INTERNOS E DE VIDA ÚTIL DE CAPACITORES

A estimativa de vida útil do capacitor inicia pela obtenção da relação parâmetro (ESR, C) x temperatura (°C) em uma câmara fechada. Esse procedimento é necessário, pois caso a influência da temperatura não seja considerada após a estimativa da ESR e C pelos métodos analisados, a previsão de vida útil será comprometida, confundindo a variação dos parâmetros por temperatura com o envelhecimento do componente. Após isso, neste capítulo será apresentado a análise e investigação dos procedimentos relacionados aos métodos estudados para estimativa da ESR e capacitância. Por fim, é realizada a predição de vida útil do capacitor utilizando técnicas de extrapolação.

O esquemático do sistema proposto é apresentado na Figura 12 onde os blocos inferiores incluem as variáveis necessárias para estimativa da corrente do capacitor (i_a, i_b, i_{ret}, v_{s2}, v_{s4}, v_{s6}), junto da temperatura que irá realizar a correção dos parâmetros e a tensão do barramento v_{cap}.

Figura 12 – Esquemático utilizado para implementação dos métodos RLS e DFT.



Fonte: do Autor.

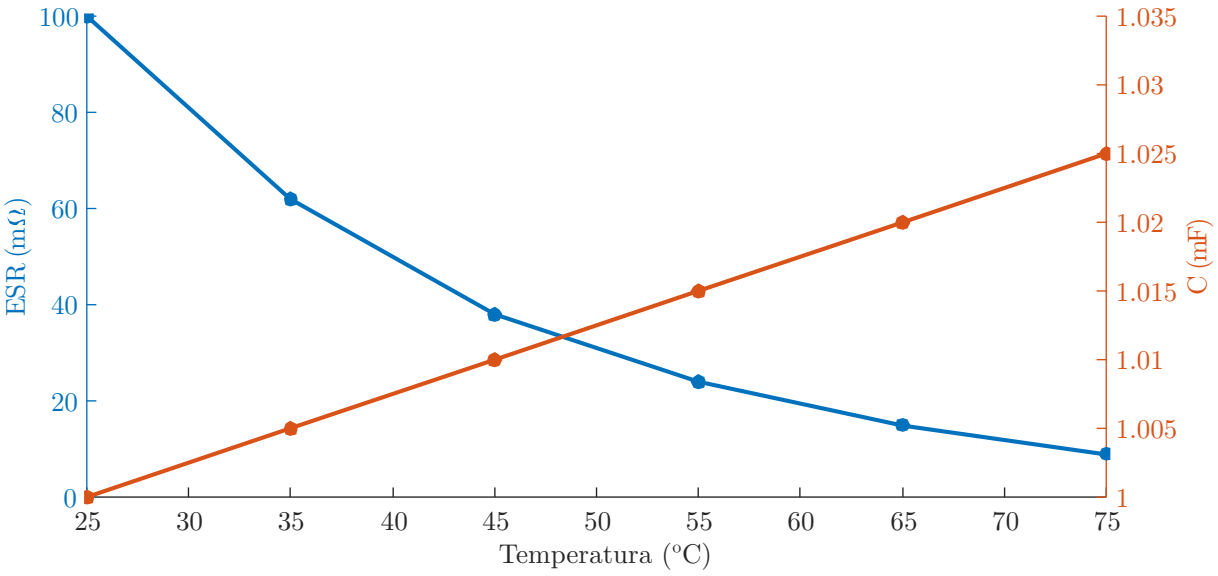
2.0.1 Procedimento inicial: obtenção da base de dados em câmara fechada

Para se obter uma base de dados inicial para predição da vida útil do capacitor, é necessário que o componente seja submetido a uma câmara fechada, com sensoriamento de temperatura e medição de impedância do capacitor em análise, onde valores de ESR e C serão extraídos para uma faixa de temperatura ($\hat{T}$) de 25 a 75°C na frequência desejada, sendo para esse trabalho desejado que as aquisições sejam realizadas em baixa frequência, especificamente em 360 Hz, introduzida pelo retificador trifásico que será abordado na próxima seção desse trabalho. Nesse sentido, os dados serão salvos em vetores na forma de:

$$\hat{T} = \{25,...,75\} \tag{1}$$

Da base de dados obtida, considerando haver amostras suficientes para reduzir a taxa de erro resultante, pode-se utilizar do método de regressão dos Mínimos Quadrados (LS - *Least Square*) para se obter uma aproximação de ESR e C para cada temperatura. Ressalta-se que a capacitância tende a ser mais estável para baixas frequências, onde tende a se aproximar a uma equação de reta, enquanto a ESR apresenta um comportamento exponencial como mostra a Figura 13, desenvolvida para exemplificar o mencionado.

Figura 13 – Curva genérica do comportamento da ESR e Capacitância para diversas temperaturas.



Fonte: do Autor.

O princípio do método LS teve sua formulação por Carl Friedrich Gauss no final do século XVIII, e tinha como objetivo determinar órbitas de planetas e asteroides. Gauss descobriu que, a partir de uma quantidade de dados, os parâmetros desconhecidos de um modelo matemático devem ser escolhidos de tal forma que a soma dos quadrados da

diferença entre os valores computados, multiplicados pelos números que medem o grau de precisão, é um mínimo (CLARKE, 1991).

Dito isto, nessa seção tem-se como base a equação exponencial de referência:

$$y = Ae^{Bx} \tag{2}$$

A equação exponencial naturalmente é não-linear. Assim, precisa-se aplicar um método de linearização. Aplicando o logaritmo natural em ambos os lados tem-se que:

$$ln(y) = ln(A) + Bx \tag{3}$$

Substituindo $ln(y)$ por y, $ln(A)$ por b e B por a, tem-se que:

$$y = ax + b \tag{4}$$

O Método LS consiste em justamente reduzir o erro encontrando constantes (a e b) para tal. De forma geral, o erro é definido como:

$$E(a,b) = \sum_{i=1}^{n} [(ax_i + b) - y_i]^2 \tag{5}$$

O mínimo global pode ser encontrado quando:

$$\nabla E(a,b) = \begin{bmatrix} \frac{\partial E}{\partial a} \\ \frac{\partial E}{\partial b} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \tag{6}$$

Fazendo a derivada parcial com relação a a:

$$\frac{\partial E}{\partial a} = 2 \sum_{i=1}^{n} [(ax_i + b) - y_i] x_i = 0 \tag{7}$$

O que resulta em:

$$\sum_{i=1}^{n} [(ax_i^2 + bx_i) - y_i x_i] = 0 \tag{8}$$

E para a derivada parcial com relação a "b":

$$\frac{\partial E}{\partial b} = 2 \sum_{i=1}^{n} [(ax_i + b) - y_i] = 0 \tag{9}$$

Ou seja:

$$\sum_{i=1}^{n} [(ax_i + b) - y_i] = 0 \tag{10}$$

Separando as constantes em relação aos seus somatórios, encontra-se que a resolução do problema depende de uma redução para um Sistema de Equações Lineares Algébricas 2x2:

$$\begin{cases} \left(\sum_{i=1}^{n} x_i^2\right) a + \left(\sum_{i=1}^{n} x_i\right) b = \sum_{i=1}^{n} x_i y_i \\ \left(\sum_{i=1}^{n} x_i\right) a + nb = \sum_{i=1}^{n} y_i \end{cases} \tag{11}$$

A resolução direta para as constantes a e b pode ser apresentada através da resolução da relação dos seguintes determinantes:

$$a = \frac{\begin{vmatrix} \sum_{i=1}^{n} x_i y_i & \sum_{i=1}^{n} x_i \\ \sum_{i=1}^{n} y_i & n \end{vmatrix}}{\begin{vmatrix} \sum_{i=1}^{n} x_i^2 & \sum_{i=1}^{n} x_i \\ \sum_{i=1}^{n} x_i & n \end{vmatrix}} \tag{12}$$

$$b = \frac{\begin{vmatrix} \sum_{i=1}^{n} x_i^2 & \sum_{i=1}^{n} x_i y_i \\ \sum_{i=1}^{n} x_i & \sum_{i=1}^{n} y_i \end{vmatrix}}{\begin{vmatrix} \sum_{i=1}^{n} x_i^2 & \sum_{i=1}^{n} x_i \\ \sum_{i=1}^{n} x_i & n \end{vmatrix}} \tag{13}$$

Substituindo pelas variáveis de interesse iniciais, substitui-se as constantes por:

$$a = ESR_0 \tag{14}$$

$$b = \frac{1}{A_0} \tag{15}$$

Por fim, a equação da ESR pode ser obtida retornando as constantes a e b obtidas para a forma da equação (2):

$$ESR_{\text{in}} = ESR_0 e^{\frac{T_0 - T_a}{A_0}} \tag{16}$$

onde:
ESR_0: ESR inicial;
ESR_{in}: ESR estimada em uma temperatura específica;
T_a: Temperatura lida;
T_0: Temperatura inicial;
A_0: Constante obtida pelo método LS;

Já a equação que representa a variação de capacitância com temperatura, pode ser obtida simplesmente encontrando as constantes de uma equação generalizada de uma reta, onde é obtida a partir da diferença entre a capacitância medida a uma determinada temperatura (T_a) e a capacitância medida a uma temperatura de referência (T_0), somada a capacitância de referência (C_0). A constante A_1 é a proporcionalidade da variação da capacitância medida em relação à variação de temperatura, encontrada pela seguinte definição:

$$A_1 = \left(\frac{\Delta C}{\Delta T}\right) C_0 \tag{17}$$

Por fim, manipulando a equação anterior, a equação final da capacitância medida em uma temperatura específica C_{in} pode ser expressa como:

$$C_{\text{in}} = C_0 + A_1(T_a - T_0) \tag{18}$$

onde:
C_0: Capacitância inicial;
C_{in}: Capacitância estimada em uma temperatura específica;
A_1: Constante obtida pelo método LS.

Após a aplicação dos algoritmos a serem explicados no decorrer do trabalho, os valores de ESR e C devem ser devidamente corrigidos considerando a temperatura do capacitor. Essa correção pode ser feita simplesmente compensando os parâmetros obtidos através dos métodos RLS ou DFT, somando-os com a diferença do valor inicial com o valor específico calculado pelas equações anteriores:

$$ESR_{\text{corrigida}} = ESR_{\text{DFT|RLS}} + ESR_0 - ESR_{\text{in}} \tag{19}$$

$$C_{\text{corrigida}} = C_{\text{DFT|RLS}} + C_0 - C_{\text{in}} \tag{20}$$

Onde $ESR_{\text{DFT|RLS}}$ e $C_{\text{DFT|RLS}}$ são as variáveis encontradas pelos métodos DFT ou RLS, que serão abordados a seguir nesse trabalho.

Como destacado anteriormente, os limites utilizados para realizar a troca do capacitor podem ser definidos como +100% e -20% para ESR e capacitância, respectivamente.

Dessa forma, os valores limites são dados como:

$$ESR_{\text{ft}} = 2ESR_{\text{in}} \tag{21}$$

$$C_{\text{ft}} = 0{,}8C_{\text{in}} \tag{22}$$

Finalmente, a situação de vida útil atual (PHS - *Present Health Status*) pode ser calculada por:

$$PHS_{\text{ESR}} = \frac{ESR - ESR_{\text{in}}}{ESR_{\text{ft}} - ESR_{\text{in}}} \tag{23}$$

$$PHS_C = \frac{C_{\text{in}} - C}{C_{\text{in}} - C_{\text{ft}}} \tag{24}$$

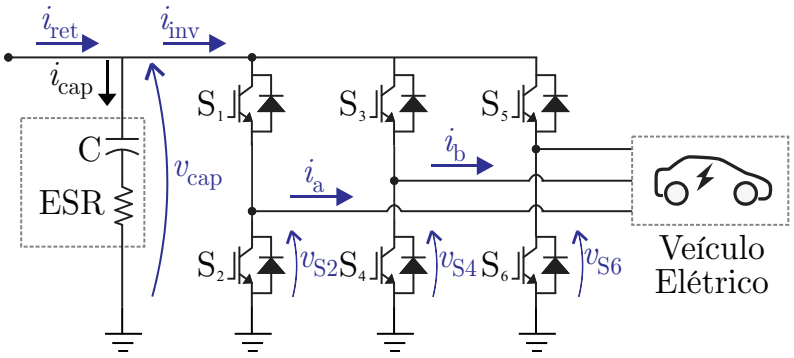
Assim que um dos limites atinge o valor unitário, pode-se considerar que o limite de vida útil do capacitor foi atingido e o mesmo deve ser substituído. Para obtermos uma previsão do tempo restante de vida útil do capacitor, poder-se-ia utilizar do mesmo algoritmo LS utilizado para interpolação como para extrapolação, entretanto, testes na ferramenta MATLAB demonstraram que conforme os dados previstos se distanciam da base de dados obtida, o erro é maior quando comparado a métodos diretos não-lineares, como o método de Gauss-Newton e o método de Levenberg-Marquardt, que serão desenvolvidos nas seções posteriores desse trabalho. Esse problema não afeta o resultado significativamente quando se há posse dos dados como para interpolação realizada em função da temperatura, além disso, a simplicidade do código para equações lineares também é extremamente reduzido.

2.0.2 Inversor e Retificador Trifásico: estimativa da corrente no capacitor

Do conceito geral, o inversor trifásico (também chamado de Inversor Fonte de Tensão) é um dispositivo que converte tensão contínua (CC) em tensão alternada (CA), podendo ter seu funcionamento como um conversor rebaixador (*buck*) em sua forma natural ou como elevador (*boost*) quando conectado a uma carga indutiva, nesse caso sendo o fluxo de energia revertido das saídas trifásicas para o barramento CC. Esse conversor possuí seis interruptores no total, onde as chaves inferiores devem ser comandadas de forma complementar às superiores de modo a evitar curto-circuito.

Duas variáveis são necessárias para obtenção dos parâmetros através dos métodos que serão estudados nessa dissertação, sendo elas a tensão e a corrente do capacitor, onde a tensão pode ser medida diretamente no barramento. Nos sistemas de predição apresentados na literatura (ZHAO et al., 2020) geralmente há medição direta da corrente no capacitor, ou mesmo das correntes de saída do retificador e de fases, proporcionando o cálculo da corrente no capacitor, método que será abordado nesse trabalho. Nesse caso, o estudo se concentra em inversores de caráter comercial onde não há acesso a corrente do capacitor de forma direta. Portanto, com o intuito se fazer a estimativa da corrente no capacitor, é necessário que seja realizada a medição de duas correntes de fase e das tensões em uma das chaves em cada braço do inversor. A Figura 14 mostra o esquemático do sistema a ser utilizado.

Figura 14 – Esquemático do inversor a ser utilizado com as medições de correntes e tensões.



Fonte: do Autor.

Fazendo o mapeamento dos possíveis estados em um inversor trifásico com base nas tensões das chaves inferiores é possível obter a Tabela 6, que se utilizando das correntes de fase medidas, obtém-se a composição da corrente de entrada do inversor (i_{inv}).

Tabela 6 – Tabela de estados do conversor trifásico.

V_{S2}	V_{S4}	V_{S6}	i_{inv}
0	0	0	0
0	0	1	i_c
0	1	0	i_b
0	1	1	$-i_a$
1	0	0	i_a
1	0	1	$-i_b$
1	1	0	$-i_c$
1	1	1	0

Fonte: Do Autor.

De outra forma, a seguinte equação também representa a tabela verdade:

$$i_{inv} = V_{S2} i_a + V_{S4} i_b + V_{S6} i_c \tag{25}$$

Na prática, pode-se utilizar apenas dois sensores para monitorar as correntes de fase com o intuito de reduzir o valor total do circuito de instrumentação. Dessa forma, considerando que apenas as correntes i_a e i_b serão medidas, a corrente i_c pode ser simplesmente calculada como:

$$i_c = -(i_a + i_b) \tag{26}$$

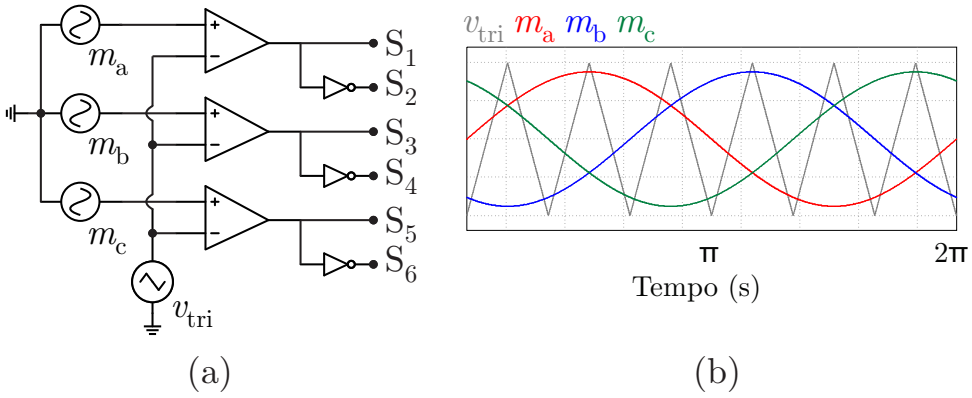
Então, a corrente no capacitor pode ser obtida por:

$$i_{cap} = i_{ret} - i_{inv} \tag{27}$$

Dentre as principais estratégias de modulação utilizadas para conversores trifásicos, destacam-se a modulação vetorial, que se utiliza de equações analíticas para definir os tempos de comutação, e a modulação Senoidal PWM (*Sinusoidal Pulse Width Modulation* - SPWM), que utiliza da comparação de uma portadora em alta frequência com funções de referência senoidais defasadas de 120°C para gerar os pulsos de comutação.

Nesse trabalho, a SPWM de 3 níveis foi escolhida principalmente por conta de sua simplicidade e facilidade de implementação. A Figura 15 apresenta um exemplo de geração da modulação.

Figura 15 – Modulação SPWM: (a) Esquemático, (b) Formas de onda.

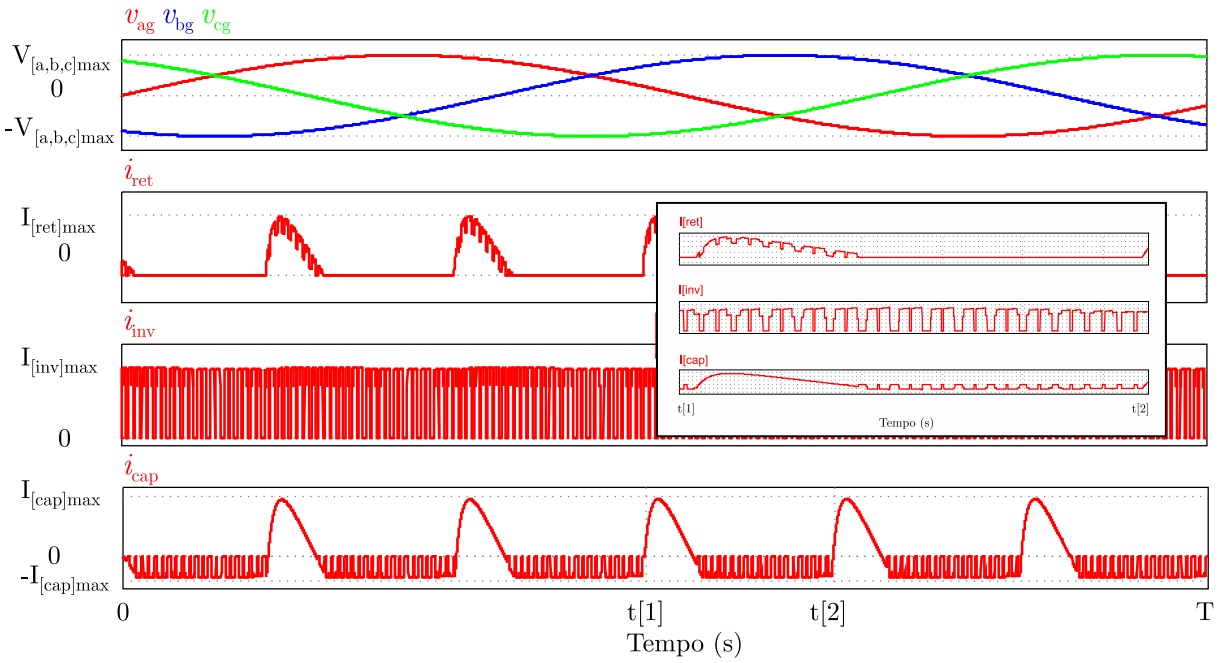


Fonte: do Autor.

Perceba que a implementação da estimativa da corrente i_{inv} independe da modulação aplicada, pois considera todos os possíveis estados que o conversor pode atuar. De qualquer forma, nesse trabalho essa modulação será utilizada para fins de demonstração, simulação e experimento.

A Figura 16 faz uma ilustração genérica das tensões de fase de entrada, correntes do retificador, do inversor e do capacitor respectivamente quando o barramento é conectado a um retificador trifásico de ponte completa.

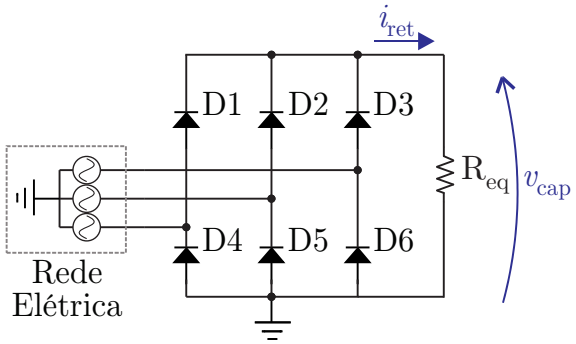
Figura 16 – Ilustração genérica das tensões de fase de entrada, correntes do retificador, do inversor e do capacitor respectivamente.



Fonte: do Autor.

Note que a corrente do capacitor possuí uma componente de baixa frequência alocada na sexta harmônica da frequência da rede elétrica. Essa será de suma importância para um dos métodos que será abordado na próxima seção, portanto, deve ser compreendida, pois é necessário ter conhecimento da componente harmônica mais significativa na tensão e corrente do barramento para aplicar o método DFT corretamente. No caso dessa dissertação, o estudo será baseado em um retificador trifásico de ponte completa a diodos conectado à rede elétrica. Para definir o surgimento dessa componente, deve-se, portanto, decompor o sinal de saída do retificador, o qual é apresentado em seu modelo genérico na Figura 17, onde o inversor trifásico (responsável pelas componentes de alta frequência do sinal) é substituído por uma carga resistiva equivalente.

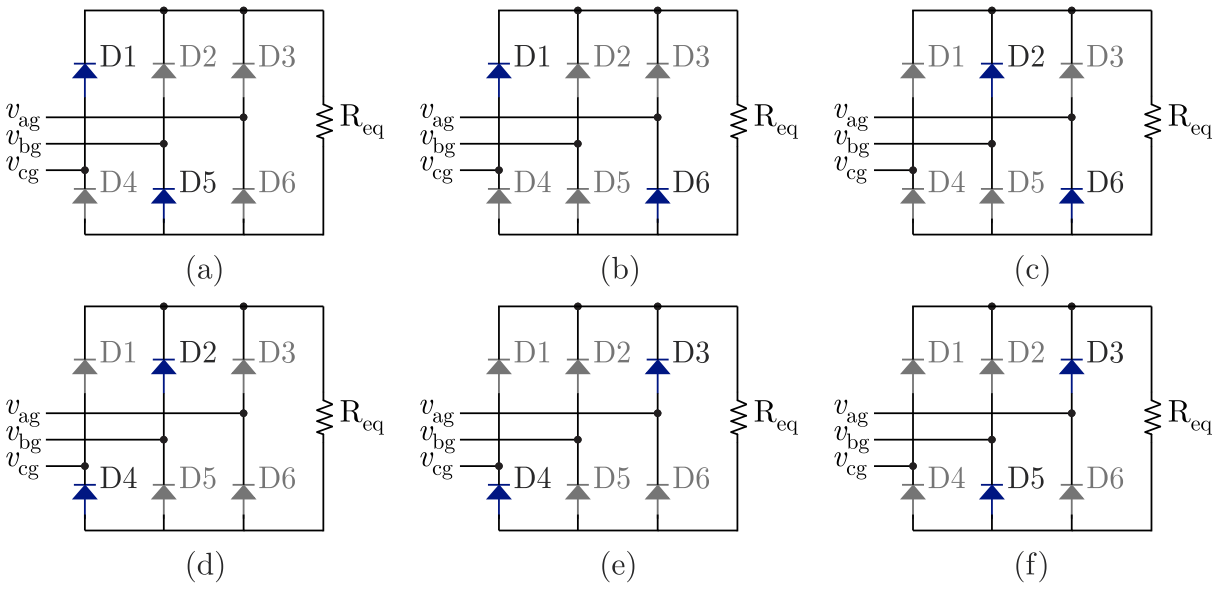
Figura 17 – Esquemático do retificador trifásico de ponte completa a ser utilizado.



Fonte: do Autor.

No total, divide-se em seis etapas de condução do retificador trifásico de onda completa, as quais são apresentadas na Figura 18. Em um primeiro momento a tensão v_{ag} é maior que v_{bg} e os diodos D1 e D5 passam a conduzir, como mostra a Figura 18(a). Até que a tensão v_{cg} se torna a menor das três e o diodo D6 entra em estado de condução, bloqueando D5, fato ilustrado na Figura 18(b). Em seguida, a maior tensão passa a ser v_{bg} e o diodo D2 passa a conduzir, como mostra a Figura 18(c). Nesse ponto, a tensão v_{ag} se torna a menor e o diodo D4 entra em condução, bloqueando D6, Figura 18(d). Então, o diodo D3 entra em condução quando a tensão v_{cg} se torna a maior, bloqueando D2, Figura 18(e). Por fim, a Figura 18(f) apresenta a última etapa, onde o diodo D5 entra em condução no momento em que v_{bg} se torna a menor tensão, bloqueando D4.

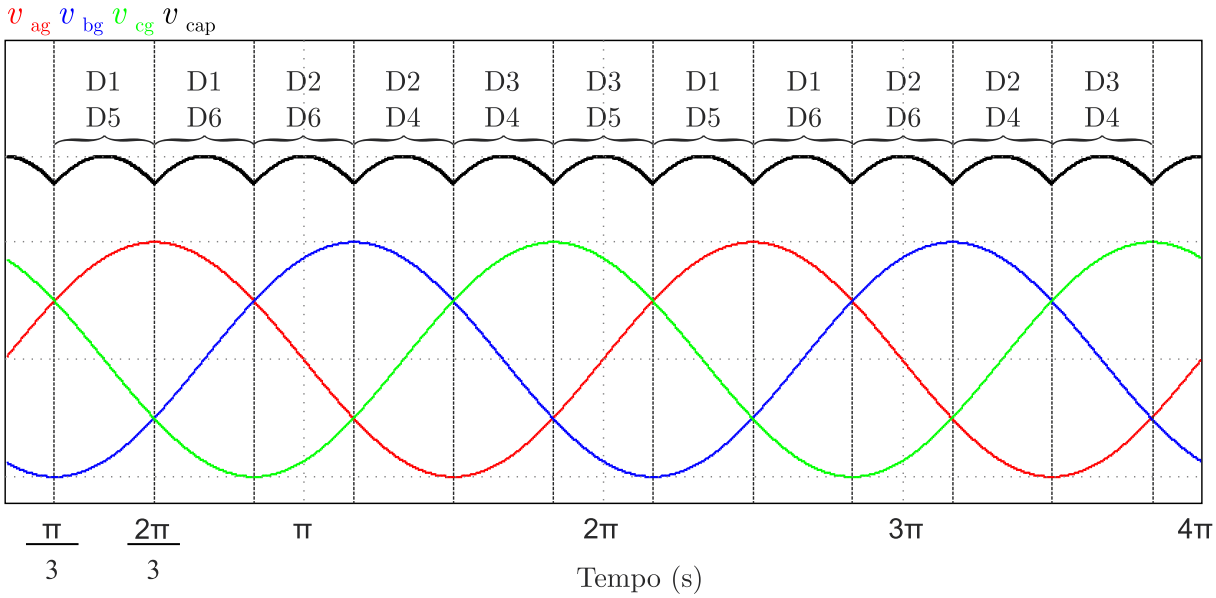
Figura 18 – Etapas de condução do retificador trifásico de ponte completa.



Fonte: do Autor.

Assim sendo, a Figura 19 mostra o comportamento das tensões de fase ($v_{a,b,c}$) e de saída (v_{cap}) do retificador, onde as etapas descritas podem ser visualizadas com maior facilidade.

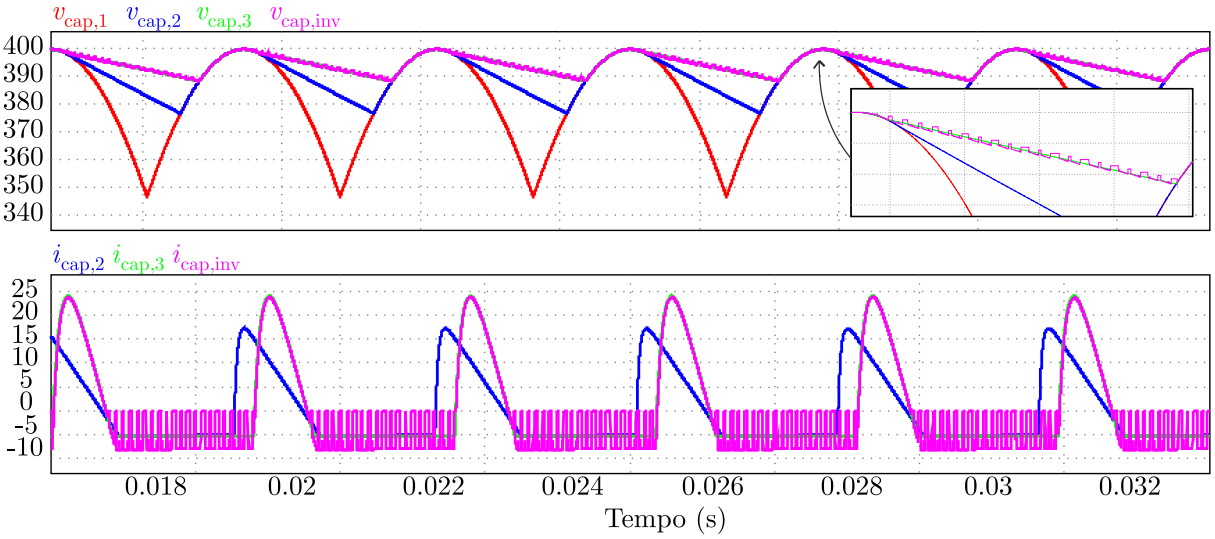
Figura 19 – Comportamento das tensões de fase e de saída no retificador trifásico.



Fonte: do Autor.

A dependência do comportamento da tensão para variadas capacitâncias é demonstrada na Figura 20, comparando as tensões entre o retificador e inversor trifásico conectado e uma carga resistiva equivalente. Nessa comparação a ESR é mantida fixa em 100 mΩ e a capacitância varia como (C_1, C_2, C_3) = (0 F, 400 µF, 1 mF), onde a capacitância C_3 é igual a utilizada para a conexão com o inversor C_{inv}. Ainda, vale ressaltar que a capacitância C_1 é considerada nula pois não há capacitor conectado no barramento para esse caso, portanto, não há corrente fluindo no ramo, e a tensão $v_{\text{cap},1}$ medida é a própria tensão na carga.

Figura 20 – Formas de onda da tensão de saída e corrente no capacitor.



Fonte: do Autor.

2.0.3 Método DFT: Estimativa da ESR e capacitância via extração de componentes harmônicas

Por definição, a DFT é uma adaptação da Transformada de Fourier (*Fourier Transform* - FT) (ALEXANDER; WILLIAMS, 2016) que funciona baseada em sinais discretos, o oposto da FT que trabalha com sinais contínuos. De forma geral, a FT decompõe o sinal de entrada em infinitas partes com equações senoidais. A saída da FT são amplitudes e frequências das componentes harmônicas, que podem ser usadas para processamento e manipulação do sinal (DIRLIK, 2013).

Teoricamente, a componente harmônica desejada do sinal analisado poderia ser obtida através da utilização de filtros passa-faixa, forma que muitos artigos já exploraram como mostrado em Zhao et al. (2020). Entretanto, circuitos analógicos de filtragem além de serem sensíveis à temperatura também podem estar suscetíveis aos efeitos da redução de vida útil dos componentes envolvidos, ocasionando erros na medição e consequentemente no processamento de dados. Para esse caso, a utilização da DFT supre as necessidades encontradas, podendo ser substituída pelo algoritmo de Goertzel (SUNDARARAJAN et al., 2020) para implementar o método de forma recursiva. Nesse trabalho, será utilizado apenas uma frequência de interesse: a sexta componente harmônica da rede elétrica. Por conta disso, pode-se facilmente implementar a DFT sem a necessidade de se obter componentes harmônicas em todo o espectro como é geralmente utilizada, e extrair apenas a componente de interesse.

Utilizando da definição da DFT, pode-se extrair as componentes harmônicas da frequência desejada, tanto da tensão medida, quanto da corrente obtida no capacitor. De

forma geral, a DFT é obtida por:

$$X_k = \sum_{n=0}^{N-1} x_n e^{-\frac{j2\pi kn}{N}} \tag{28}$$

Pelo teorema de Euler, tem-se que:

$$e^{j\theta} = cos(\theta) + jsin(\theta) \tag{29}$$

Então, a equação geral da DFT pode ser reescrita como:

$$X_k = \sum_{n=0}^{N-1} x_n \left[cos\left(\frac{2\pi}{N} kn\right) - jsin\left(\frac{2\pi}{N} kn\right) \right] \tag{30}$$

onde:
k: Variável referente a harmônica desejada ($k = 0,1,2...N-1$).
X: Variável analisada.
n: Passos da iteração (alterado conforme amostragem coletada).

Além disso, sendo N a quantidade de pontos armazenados que pode ser obtida através de uma simples relação com a frequência de amostragem do microcontrolador e a frequência de interesse (nesse caso, de 360 Hz).

$$N = \frac{f_s}{f_m} \tag{31}$$

onde:
f_s: Frequência de amostragem.
f_m: Frequência de interesse.

Ainda, considera-se o fator de dois referente a frequência de *Nyquist*, que diz que a frequência de amostragem deve ser pelo menos o dobro da frequência máxima considerada:

$$f_{\mathrm{s,min}} = 2f_m \tag{32}$$

Obtendo a parte real e imaginária:

$$X_{\mathrm{re}} = \sum_{n=0}^{N-1} x_n cos\left(\frac{2\pi}{N} kn\right) \tag{33}$$

$$X_{\mathrm{im}} = \sum_{n=0}^{N-1} -x_n jsin\left(\frac{2\pi}{N} kn\right) \tag{34}$$

No processo de implementação no Processador Digital de Sinal (*Digital Signal Processor* - DSP), pode-se reduzir o equacionamento utilizando algoritmos específicos derivados da DFT que possibilitam o processamento mais rápido do que a forma geral.

Esse algoritmos são subdivisões da então chamada Transformada Rápida de Fourier (*Fast Fourier Transform* - FFT), primeiramente introduzida em 1965 (COOLEY; TUKEY, 1965), computa a DFT de uma forma mais rápida. De forma geral, a FFT se utiliza do conceito "dividir e conquistar" e requer $O(Nlog_r(N))$ operações, diferentemente da DFT que requer $O(N^2)$. O "radix", r, é o número de partes em que o sinal será dividido (DIRLIK, 2013). Desses algoritmos, destaca-se os chamados "radix-2" e "radix-4" e serão explorados e desenvolvidos no decorrer dessa seção.

2.0.3.1 Radix-2

O algoritmo radix-2 depende de que o número de amostras seja potência de dois, já que basicamente divide a equação da DFT em duas partes. Cada parte é dividida entre partes menores e assim por diante, depois combina os dois resultados em um só, formando o resultado esperado. Para derivar sua equação, separa-se a equação geral da DFT em duas parcelas (BARBOSA, 2018):

$$X_k = \sum_{n=0}^{\frac{N}{2}-1} x_n e^{-\frac{j2\pi kn}{N}} + \sum_{n=\frac{N}{2}}^{N-1} x_n e^{-\frac{j2\pi kn}{N}} \tag{35}$$

Substituindo n por $n + \frac{N}{2}$ para deixar ambos somatórios com índices iguais:

$$X_k = \sum_{n=0}^{\frac{N}{2}-1} x_n e^{-\frac{j2\pi kn}{N}} + \sum_{n=0}^{\frac{N}{2}-1} x_{(n+\frac{N}{2})} e^{-\frac{j2\pi k(n+\frac{N}{2})}{N}} \tag{36}$$

Separando a parte exponencial da segunda parcela, tem-se que:

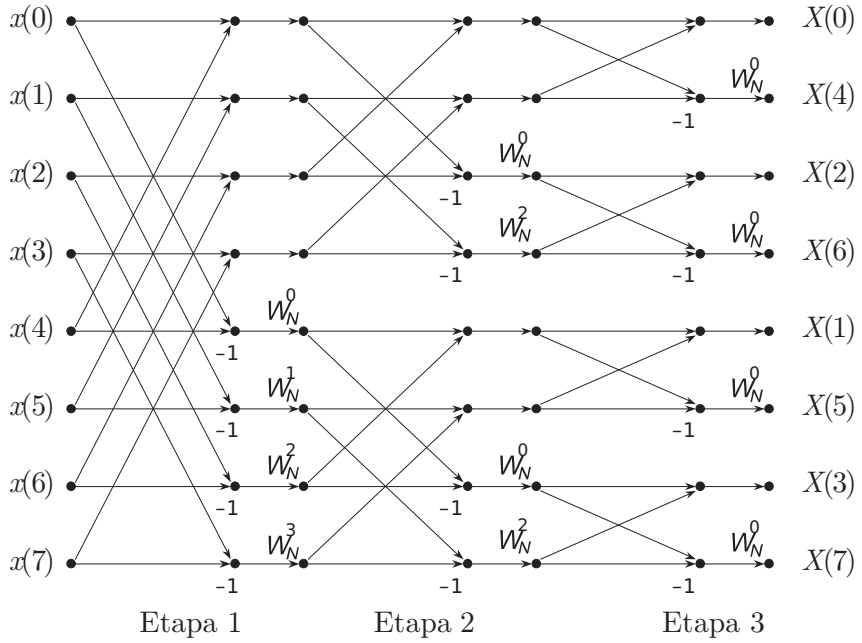
$$X_k = \sum_{n=0}^{\frac{N}{2}-1} x_n e^{-\frac{j2\pi kn}{N}} + e^{-j\pi k} \sum_{n=0}^{\frac{N}{2}-1} x_{(n+\frac{N}{2})} e^{-\frac{j2\pi kn}{N}} \tag{37}$$

Por fim, pode-se simplificar a equação substituindo $e^{-j\pi k}$ por -1 e juntando os somatórios:

$$X_k = \sum_{n=0}^{\frac{N}{2}-1} \left[x_n + (-1)^k x_{(n+\frac{N}{2})} \right] e^{-\frac{j2\pi kn}{N}} \tag{38}$$

A representação do radix-2 pode ser ilustrada através de um diagrama onde as conexões são chamadas de *butterflies*. De forma geral, o algoritmo é composto por $log_2(N)$ etapas, e cada etapa é dividida em $N/2$ *butterflies*. Cada *butterfly* consiste em duas adições para os dados de entrada e uma multiplicação para o *twiddle factor*, sendo uma constante W_N referente a parcela $e^{-\frac{j2\pi}{N}}$ das equações desenvolvidas (CHENG, 2000). O diagrama relativo ao algoritmo radix-2 para 8 amostras é ilustrado na Figura 21, onde círculos representam adições.

Figura 21 – Diagrama do algoritmo radix-2 para N=8.



Fonte: Adaptado de Cheng (2000).

2.0.3.2 Radix-4

Já o algoritmo radix-4 simplifica ainda mais o cálculo da DFT dividindo entre quatro partes. A razão pela qual vale a pena desenvolver uma implementação radix-4 em vez de simplesmente usar as FFT radix-2 é que o custo aritmético pode ser ainda menor (CHU; GEORGE, 1999). Por esse motivo, esse foi o algoritmo escolhido para implementação no DSP quando utilizado de forma *offline* (fora da interrupção principal). A lógica do equacionamento é a mesma utilizada anteriormente, portanto não será desenvolvida por completo novamente. A forma geral da equação do algoritmo radix-4 é dada como:

$$X_k = \sum_{n=0}^{\frac{N}{4}-1} x_n e^{-\frac{j2\pi kn}{N}} + \sum_{n=\frac{N}{4}}^{\frac{N}{2}-1} x_n e^{-\frac{j2\pi kn}{N}} + \sum_{n=\frac{N}{2}}^{\frac{3N}{4}-1} x_n e^{-\frac{j2\pi kn}{N}} + \sum_{n=\frac{3N}{4}}^{N-1} x_n e^{-\frac{j2\pi kn}{N}} \tag{39}$$

Substituindo na segunda equação n por $n + \frac{N}{4}$, na terceira n por $n + \frac{N}{2}$ e na quarta n por $n + \frac{3N}{4}$ e juntando os somatórios, tem-se que o formato radix-4 pode ser obtido por:

$$X_k = \sum_{n=0}^{\frac{N}{4}-1} \left[x_n + (-j)^k x(n + \frac{N}{4}) + (-1)^k x(n + \frac{N}{2}) + j^k x(n + \frac{3N}{4}) \right] e^{-\frac{j2\pi kn}{N}} \tag{40}$$

Perceba que se pode dividir a equação desenvolvida em quatro partes dependendo do resto (Q) da divisão da ordem do harmônico desejada, e essas são as equações que serão implementadas no DSP. Para Q = 0 (k = 0, 4, 8, ...), a equação tem a seguinte

forma:

$$X_0 = \sum_{n=0}^{\frac{N}{4}-1} \left[x_n + x(n+\frac{N}{4}) + x(n+\frac{N}{2}) + x(n+\frac{3N}{4}) \right] \tag{41}$$

E as partes real e imaginária:

$$X_{\text{re}} = \sum_{n=0}^{\frac{N}{4}-1} \left[x_n + x(n+\frac{N}{4}) + x(n+\frac{N}{2}) + x(n+\frac{3N}{4}) \right] \tag{42}$$

$$X_{\text{im}} = 0 \tag{43}$$

Já para Q = 1 (k = 1, 5, 9, ...), a equação se desenvolve como:

$$X_1 = \sum_{n=0}^{\frac{N}{4}-1} \left[x(n) - jx(n+\frac{N}{4}) - x(n+\frac{N}{2}) + jx(n+\frac{3N}{4}) \right] e^{-\frac{j2\pi n}{N}} \tag{44}$$

Substituindo (29) em (44) e separando em parte real e imaginária, simplifica-se a equação definindo as seguintes constantes:

$$D_0 = x(n) - x(n+\frac{N}{2}) \tag{45}$$

$$D_1 = -x(n+\frac{N}{4}) + x(n+\frac{3N}{4}) \tag{46}$$

Assim, a equação final a ser implementada tem as partes reais e imaginárias com forma de:

$$X_{\text{re}} = \sum_{n=0}^{\frac{N}{4}-1} D_0 cos\left(\frac{2\pi}{N}n\right) + D_1 sin\left(\frac{2\pi}{N}n\right) \tag{47}$$

$$X_{\text{im}} = -\sum_{n=0}^{\frac{N}{4}-1} D_0 sin\left(\frac{2\pi}{N}n\right) - D_1 cos\left(\frac{2\pi}{N}n\right) \tag{48}$$

Para Q = 2 (k = 2, 6, 10, ...), a equação tem forma de:

$$X_2 = \sum_{n=0}^{\frac{N}{4}-1} \left[x_n - x(n+\frac{N}{4}) + x(n+\frac{N}{2}) - x(n+\frac{3N}{4}) \right] e^{-\frac{j4\pi n}{N}} \tag{49}$$

Considerando uma constante C_2 para simplificação:

$$D_2 = x_n - x(n+\frac{N}{4}) + x(n+\frac{N}{2}) - x(n+\frac{3N}{4}) \tag{50}$$

E separando em termos de real e imaginário:

$$X_{\text{re}} = \sum_{n=0}^{\frac{N}{4}-1} D_2 cos\left(\frac{2\pi}{N}n\right) \tag{51}$$

$$X_{\text{im}} = -\sum_{n=0}^{\frac{N}{4}-1} D_2 sin\left(\frac{2\pi}{N}n\right) \tag{52}$$

Por fim, para Q = 3 (k = 3, 7, 11, ...) tem-se que:

$$X_3 = \sum_{n=0}^{\frac{N}{4}-1} \left[x_n + jx(n+\frac{N}{4}) - x(n+\frac{N}{2}) - jx(n+\frac{3N}{4})\right] e^{-\frac{j6\pi n}{N}} \tag{53}$$

Considerando constantes D_3 e D_4 para simplificação como:

$$D_3 = x_n - x(n+\frac{N}{2}) \tag{54}$$

$$D_4 = x(n+\frac{N}{4}) - x(n+\frac{3N}{4}) \tag{55}$$

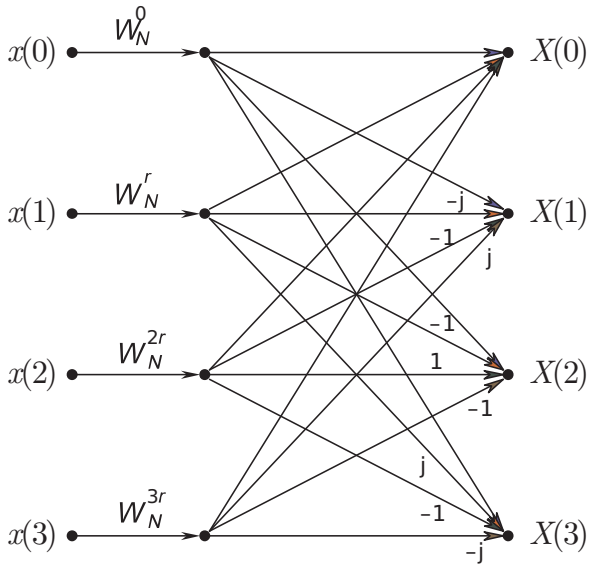
O que em termos de real e imaginário corresponde a:

$$X_{\text{re}} = \sum_{n=0}^{\frac{N}{4}-1} D_3 cos\left(\frac{2\pi}{N}n\right) + D_4 sin\left(\frac{2\pi}{N}n\right) \tag{56}$$

$$X_{\text{im}} = -\sum_{n=0}^{\frac{N}{4}-1} D_3 sin\left(\frac{2\pi}{N}n\right) - D_4 cos\left(\frac{2\pi}{N}n\right) \tag{57}$$

Uma vez que a FFT radix-4 requer menos estágios e *butterflies* do que a DFT radix-2, os cálculos podem ser melhorados ainda mais. Por exemplo, para calcular uma DFT de 16 pontos, o radix-2 leva $log_2(16) = 4$ estágios, mas o radix-4 leva apenas $log_4(16) = 2$ (CHENG, 2000). Da mesma forma que para o radix-2, o diagrama do algoritmo radix-4 é apresentado na Figura 22.

Figura 22 – Diagrama do algoritmo radix-4 para N=8.



Fonte: Adaptado de Cheng (2000).

2.0.3.3 Algoritmo de Goertzel

Ainda, pode-se utilizar outra abordagem chamada de algoritmo de Goertzel, apresentada pela primeira vez por Gerald Goertzel em 1958. Embora apresente os mesmos resultados, esta é uma metodologia de abordagem da DFT que pode ser implementada de forma recursiva e com apenas uma multiplicação por rotina, diferentemente da DFT que só pode ser computada após o armazenamento das N variáveis desejadas em um vetor, facilitando a implementação dentro da interrupção do ADC do DSP. Ressalta-se que apesar da recursividade a não-necessidade de armazenamento de variáveis de estado em vetores, o resultado desejado através do algoritmo de Goertzel só é considerado apenas após o cálculo para N amostras.

Um exemplo onde o algoritmo de Goertzel é amplamente utilizado é o DTMF (*Dual-Tone Multi-Frequency*). Esta aplicação requer o cálculo do conteúdo de frequência de oito frequências para detecção de dígitos discados na telefonia. O algoritmo pega a formulação de (28) e a converte em uma soma de convolução de um sistema IIR (*Infinite Impulse Response*) linear invariante no tempo (LTI) (KHAN, 2011).

Para iniciar o equacionamento do algoritmo, perceba que para $n = N$ em ${W_N}^{-kn}$ tem-se que:

$$ {W_N}^{-kN} = e^{\frac{j2\pi kN}{N}} = e^{j2\pi k} = 1 \tag{58} $$

O que denota periodicidade com o período N de ${W_N}^{-kn}$, tanto para n quanto k. Por conta disso, pode-se multiplicar o lado direito da equação (28) por ${W_N}^{-kN}$ sem afetar a mesma. Então:

$$ X_k = e^{-\frac{j2\pi kN}{N}} \sum_{n=0}^{N-1} x_n e^{-\frac{j2\pi kn}{N}} \tag{59} $$

Que pode ser rearranjada como:

$$ X_k = \sum_{n=0}^{N-1} x_n e^{-\frac{j2\pi k(n-N)}{N}} \tag{60} $$

Essa equação descreve uma convolução discreta de uma função com entrada de duração finita para $0 < n < (N-1)$, com a sequência infinita $e^{-\frac{j2\pi kn}{N}}$. Nessa etapa, percebe-se a semelhança com a resposta teórica de um filtro digital IIR que tem a equação geral dada como:

$$ y_k(m) = \sum_{n=-\inf}^{\inf} x(n) h_k(m-n) \tag{61} $$

Dessa forma, a definição de impulso infinito é:

$$ h_k(l) = e^{-\frac{j2\pi kl}{N}} u(l) \tag{62} $$

Portanto, reescrevendo a equação (61):

$$y_k(m) = \sum_{n=-\inf}^{\inf} x(n) e^{-\frac{j2\pi k(m-n)}{N}} u(m-n) \tag{63}$$

Comparando (60) e (61) se entende que o valor desejado de $X(k)$ é a N amostra da convolução (SYSEL; RAJMIC, 2012), ou seja:

$$X(k) = y_k(n)\Big|_{n=N} \tag{64}$$

E a transformada Z de $h(n)$:

$$H_k(z) = \sum_{n=-\inf}^{\inf} h_k(n) z^{-n} = \sum_{n=-\inf}^{\inf} e^{-\frac{j2\pi kn}{N}} u(n) z^{-n} \tag{65}$$

O que é idêntico a:

$$H_k(z) = \sum_{n=0}^{\inf} e^{-\frac{j2\pi kn}{N}} z^{-n} \tag{66}$$

Ou seja, a forma geral de $H_k(z)$ pode ser representada como:

$$H_k(z) = \sum_{n=0}^{\inf} \left(e^{-\frac{j2\pi k}{N}} z^{-1} \right) \tag{67}$$

Substituindo $e^{-\frac{j2\pi k}{N}}$ por W_N^{-kn} e abrindo a equação em termos, tem-se que:

$$H_k(z) = \sum_{n=0}^{\inf} \left(W_N^{-kn} z^{-1} \right) = 1 + W_N^{-k} z^{-1} + W_N^{-2k} z^{-2} + ... = \frac{1}{1 - W_N^{-2k} z^{-1}} \tag{68}$$

Essa estrutura precisa do mesmo número de multiplicações e somatórios que a DFT direta, porém $1/N$ do número de cálculos trigonométricos (CHASSAING; REAY, 2008). O algoritmo de Goertzel de segunda ordem pode ser obtido multiplicando o numerador e denominador por $1 - W_N^{-kn} z^{-1}$, o que resulta na seguinte função de transferência:

$$H(z) = \frac{1 - (W_N^k)^{-1}}{1 - 2cos\left(\frac{2\pi}{N} k\right) z^{-1} + z^{-2}} \tag{69}$$

O que na forma recursiva representa a seguinte equação:

$$y(n) = -y(n-2) + 2cos\left(\frac{2\pi k}{N}\right) y(n-1) + x(n) \tag{70}$$

As partes real e imaginária podem ser obtidas por:

$$X_{\text{re}} = \left[y(n-1) - y(n-2) cos\left(\frac{2\pi k}{N}\right) \right] \Big|_{n=N} \tag{71}$$

$$X_{\text{im}} = \left[y(n-2) sin\left(\frac{2\pi k}{N}\right) \right] \Big|_{n=N} \tag{72}$$

A Tabela 7 resume uma comparação entre os métodos descritos em relação ao poder de processamento a armazenamento necessário para uma rotina completa, onde se é desejado apenas uma componente harmônica para duas variáveis de estado (tensão e corrente). Perceba que por mais que os métodos radix-2 e radix-4 computem apenas $N/2$ e $N/4$ dos dados armazenados, o vetor deve ser preenchido por completo (N amostras), pois os cálculos internos dependem de variáveis futuras que estão fora da janela computada. Ainda, ressalta-se que dependendo da frequência de interesse escolhida (ordem da harmônica) os parâmetros da tabela podem mudar, como nesse trabalho existe apenas uma componente harmônica de interesse (a sexta harmônica da tensão da rede elétrica), considera-se essa como frequência base, ou seja, $k = 1$, o que é equivalente a razão $Q = 1$.

Tabela 7 – Comparação dos métodos radix-2, radix-4, Goertzel e DFT convencional para uma rotina de processamento completa de uma ordem harmônica individual.

Resto	**Algoritmo**	**Multiplicações**	**Senos e Cossenos**	**Variáveis Armazenadas**
$Q = 0$	DFT (radix-2)	$2N$	$2N$	N+8
$Q = (0,2)$	DFT (radix-4)	N	N	N+8
$Q = 1$	DFT (radix-2)	$4N$	$4N$	N+8
$Q = (1,3)$	DFT (radix-4)	$2N$	$2N$	N+8
	DFT	$4N$	$4N$	N+8
	Goertzel	N	0	14

Fonte: do Autor.

2.0.3.4 Cálculo da ESR e Capacitância

A magnitude e fase da componente obtida por qualquer um dos métodos apresentados podem ser obtidas através das seguintes equações:

$$X_{\text{amp}} = \sqrt{X_{\text{re}}^2 + X_{\text{im}}^2} \tag{73}$$

$$X_{\text{pha}} = atan\left(\frac{X_{\text{im}}}{X_{\text{re}}}\right)\frac{180}{\pi} \tag{74}$$

Além disso, é preciso também se preocupar com a quadratura da componente de fase obtida. Para quando a fase calculada for menor ou igual a zero, soma-se um ângulo de 90 graus para compensação. Em contraponto, caso o valor seja maior que zero, subtrai-se 90 graus.

Pode-se então calcular a ESR e a impedância capacitiva respectivamente:

$$ESR = \left|\frac{V_{\text{amp}} cos[(V_{\text{pha}} - I_{\text{pha}})\frac{\pi}{180}]}{I_{\text{amp}}}\right| \tag{75}$$

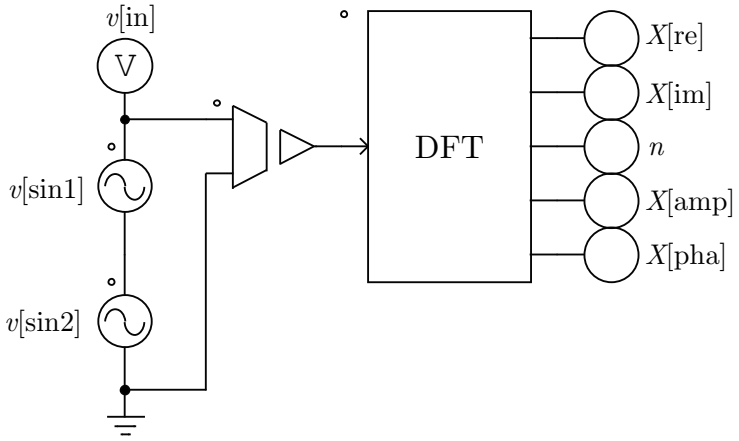
$$X_c = \frac{V_{\text{amp}} sin[(V_{\text{pha}} - I_{\text{pha}})\frac{\pi}{180}]}{I_{\text{amp}}} \tag{76}$$

Aplicando a condicional para quando X_c difere de zero, tem-se que a capacitância é dada por:

$$C = -\frac{1}{\omega X_c} \tag{77}$$

Na Figura 23 é ilustrado um exemplo de uma simulação na ferramenta PSIM com o somatório de duas entradas senoidais, uma com 0,8 de amplitude e 360 Hz e outra com 2 de amplitude e 720 Hz.

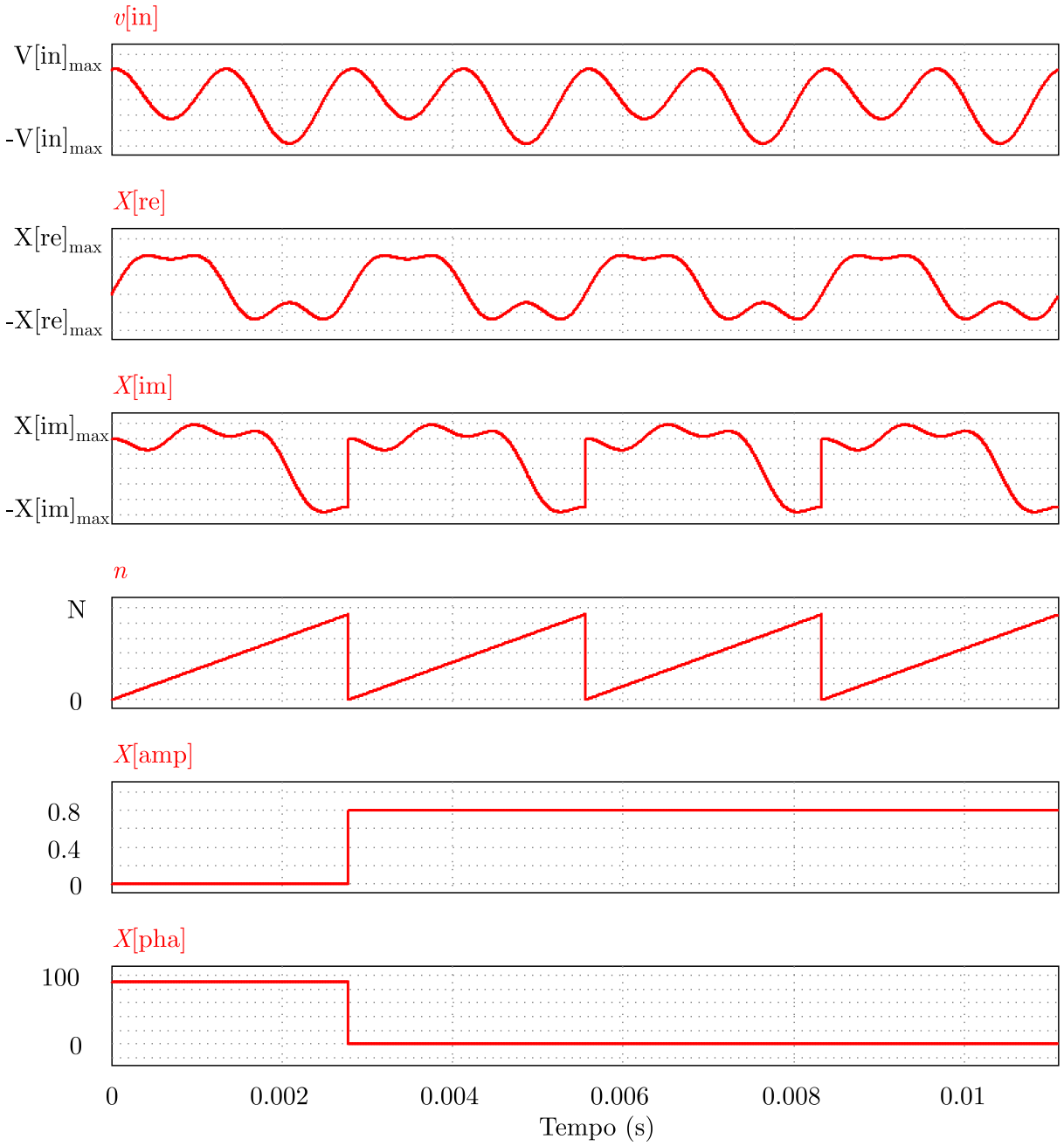
Figura 23 – Esquemático da simulação no PSIM para demonstração do funcionamento do módulo DFT.



Fonte: do Autor.

Na Figura 24 é mostrado de forma genérica para os parâmetros de configuração da DFT algumas formas de onda do processo para entendimento do funcionamento do algoritmo para $k = 1$, ou seja, deseja-se obter a resposta do algoritmo da DFT para a componente fundamental da frequência de interesse em 360 Hz. No topo, referente a variável $v[\text{in}]$, tem-se o sinal de entrada composto pela soma das duas entradas senoidais. As variáveis $X[\text{re}]$ e $X[\text{im}]$ representam as partes real e imaginária decompostas referentes as Equações (33) e (34) (seus valores máximos e mínimos dependem também da quantidade de pontos utilizada). A variável n é relativa ao passo de iteração do algoritmo, que conta de 0 ao seu máximo, como mostrado na Equação (31). Por fim, as variáveis $X[\text{amp}]$ e $X[\text{pha}]$ representam a amplitude e fase do sinal analisado em questão, devendo ser considerado seu valor final de convergência, após seu transiente inicial.

Figura 24 – Formas de onda de parâmetros necessários ao algoritmo da DFT, sendo tensão de entrada, termo real, termo imaginário, passo de iteração, amplitude e fase do sinal respectivamente para frequência de 360 Hz.



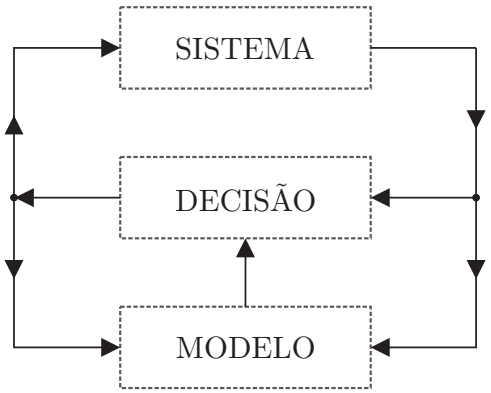
Fonte: do Autor.

2.0.4 Método RLS: Estimativa da ESR e capacitância via Método dos Mínimos Quadrados Recursivo

Sistemas que tenham característica de operação *online*, ou seja, que necessite ter seus parâmetros calculados em tempo real, podem ser chamados de "adaptativos". Dessa forma, fala-se controle adaptativo, filtro adaptativo, processamento de sinais adaptativo e

predição adaptativa (LJUNG, 1999). Esses podem ser representados através do diagrama ilustrado na Figura 25.

Figura 25 – Diagrama dos métodos adaptativos.

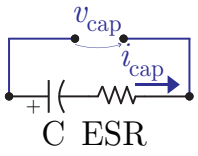


Fonte: Adaptado de Ljung (1999).

Nesta seção será desenvolvido matematicamente o Método dos Mínimos Quadrados Recursivo (*Recursive Least-Square* - RLS) que, como o nome sugere, tem origem no método LS, e o conceito da abordagem *offline* é praticamente o mesmo desenvolvido na seção 2.0.1, porém, agora aplicado ao modelo do capacitor. As equações serão manipuladas de tal forma que se possa realizar o processamento do método LS de forma adaptativa, ou seja, em funcionamento *online*, permitindo que os parâmetros de ESR e C sejam obtidos em tempo real. De modo geral, consideremos que em um algoritmo adaptativo os dados são tratados em forma de fluxo, ou seja, pequenos pacotes são processados a cada vez, obtendo um estimador e atualizando-o em cada iteração. De qualquer forma, do mesmo modo que no método LS, um valor arbitrário é escolhido como suposição inicial e o algoritmo trata de aproximar ao valor ideal reduzindo o erro.

Assim sendo, o primeiro passo para encontrar os parâmetros ESR e capacitância é desenvolver a função de transferência no domínio discreto do circuito equivalente do componente. A Figura 26 mostra o circuito genérico que representa a ESR, capacitância, corrente e tensão diferencial no capacitor.

Figura 26 – Diagrama de bode no domínio contínuo e discreto do sistema modelado.



Fonte: do Autor.

Daí, tem-se a equação diferencial que define a Lei de Kirchhoff das tensões no circuito:

$$-v_{\text{cap}}(t) + ESR i_{\text{cap}}(t) + \frac{1}{C}\int i_{\text{cap}}(t)dt = 0 \tag{78}$$

Aplicando a transformada de Laplace, tem-se que:

$$-v_{\text{cap}}(s) + ESR i_{\text{cap}}(s) + \frac{1}{sC} i_{\text{cap}}(s) = 0 \tag{79}$$

Manipulando e separando $v_{\text{cap}}(s)$ e $i_{\text{cap}}(s)$ em ambos os lados da equação:

$$v_{\text{cap}}(s) = i_{\text{cap}}(s)(ESR + \frac{1}{sC}) \tag{80}$$

Finalmente, a função de transferência do modelo em questão pode ser expressa como:

$$H(s) = \frac{v_{\text{cap}}(s)}{i_{\text{cap}}(s)} = \frac{ESRCs + 1}{Cs} \tag{81}$$

Utilizando o modelo correspondente a transformada Z para o domínio discreto com a transformada de Tustin, tem-se que:

$$H(z^{-1}) = \frac{(ESR + \frac{T_s}{2C}) + (\frac{T_s}{2C} - ESR)z^{-1}}{1 - z^{-1}} \tag{82}$$

Onde T_s é o período de amostragem.

Para simplificar, considera-se que:

$$b_0 = ESR + \frac{T_s}{2C} \tag{83}$$

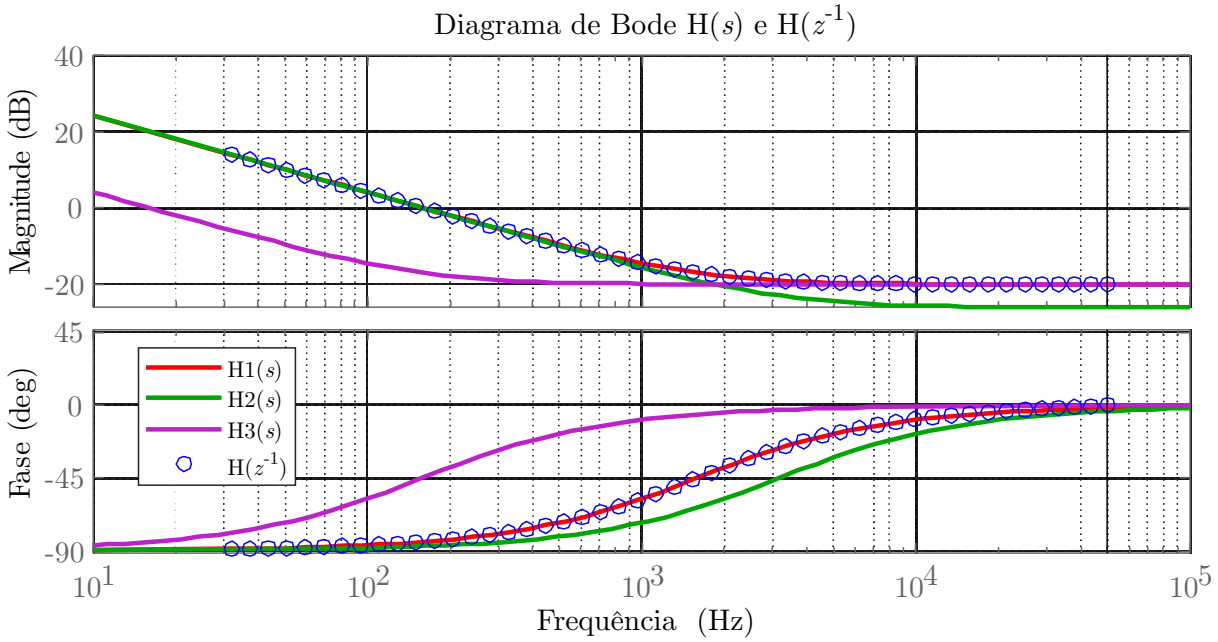
$$b_1 = \frac{T_s}{2C} - ESR \tag{84}$$

Então, representa-se a equação (82) como:

$$H(z^{-1}) = \frac{b_0 + b_1 z^{-1}}{1 - z^{-1}} \tag{85}$$

O Diagrama de Bode do modelo apresentado é mostrado na Figura 27, com uma frequência de amostragem de 100 kHz (sendo também o limite do gráfico, já que para maiores frequências teria de se considerar a parcela da indutância parasita do capacitor). Além disso, variações de ESR e capacitância são implementadas a fim de se verificar a dependência do comportamento da curva para quando cada variável é alterada.

Figura 27 – Diagrama de bode no domínio contínuo e discreto do sistema modelado com variações de resistência e capacitância.



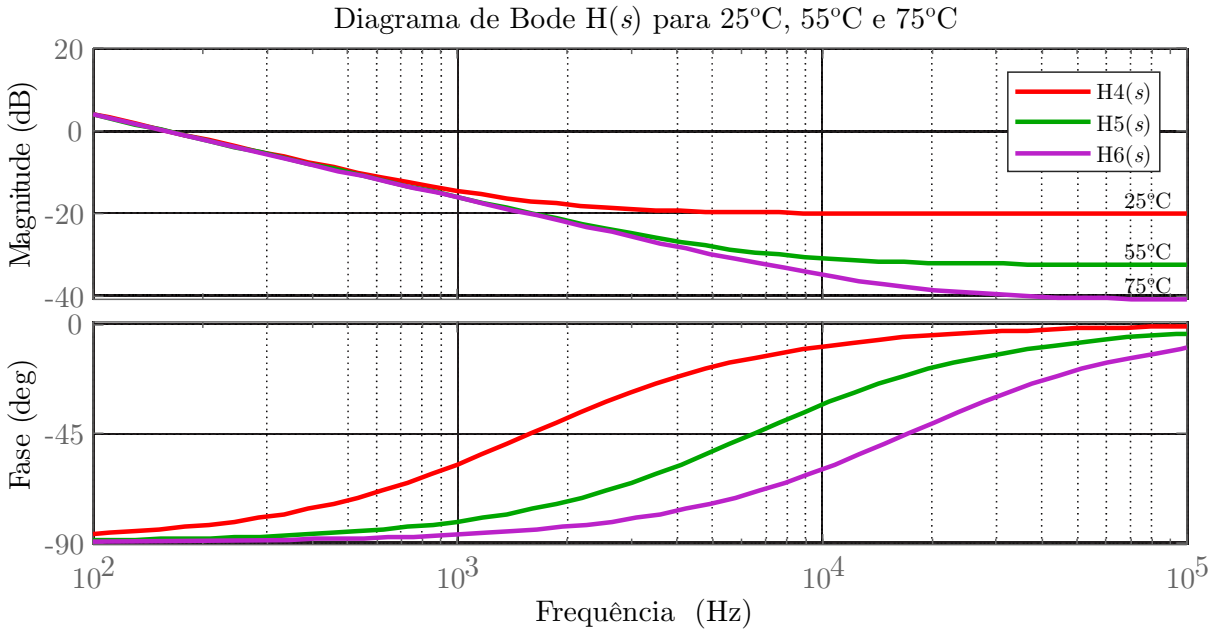
Fonte: do Autor.

Onde:

- H1(s) → (ESR,C) = (100 mΩ, 1 mF);
- H2(s) → (ESR,C) = (50 mΩ, 1 mF);
- H3(s) → (ESR,C) = (100 mΩ, 2 mF);
- H(z^{-1}) → Transformada de Tustin de H1(s).

Ainda, juntando a relação com temperatura com a equação da interpolação da Figura 13, pode-se obter o gráfico da amplitude do ganho *versus* frequência, onde ambos parâmetros são alterados. Este é representado na Figura 28.

Figura 28 – Diagrama de bode no domínio contínuo do sistema modelado com variações temperatura.



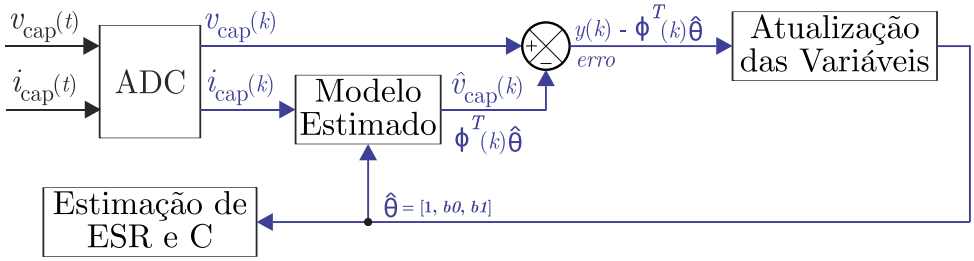
Fonte: do Autor.

Onde:

- H4(s) → (ESR, C, T) = (100 mΩ, 1 mF, 25°C);
- H5(s) → (ESR, C, T) = (24 mΩ, 1,015 mF, 55°C);
- H6(s) → (ESR, C, T) = (9 mΩ, 1,025 mF, 75°C).

O modelo geral em diagrama de blocos para o sistema a ser desenvolvido é ilustrado na Figura 29. Nesta, observa-se inicialmente o bloco que representa o modelo real do capacitor, tendo como entrada a corrente $i_{\text{cap}}(t)$ e a tensão $v_{\text{cap}}(t)$. O sinal da corrente em forma discretizada é enviado para o modelo estimado, que por fim gerará o sinal estimado de tensão no capacitor $\hat{y}$ (ou "$\phi^T(k)\hat{\theta}$" como deduzido a seguir nesta seção). Um erro é gerado quando comparado o sinal de tensão de saída discretizado do capacitor com o sinal estimado, que por sequência será enviado ao algoritmo de adaptação dos parâmetros para a próxima iteração. Com o sinal obtido e o sinal do modelo estimado é possível identificar os parâmetros de ESR e C.

Figura 29 – Diagrama de blocos do sistema a ser desenvolvido.



Fonte: do Autor.

De forma geral, a ideia principal do algoritmo LS como já abordado anteriormente na seção anterior, é de se minimizar o somatório quadrático da função erro. Nesse caso em particular, essa função erro pode ser descrita como a diferença entre o valor da tensão lida e a tensão estimada (parâmetro também conhecido por *inovação*):

$$\epsilon = y - \Phi\hat{\theta} \tag{86}$$

Onde y é a matriz que carrega os dados coletados da tensão no capacitor, Φ representa a matriz das funções de entrada e saída do sistema e $\hat{\theta}$ a matriz que carrega as variáveis de interesse e será utilizada como variável simbólica para encontrar a minimização da função custo:

$$y = \begin{bmatrix} y(2) & y(3) & \dots & y(N+1) \end{bmatrix}^T \tag{87}$$

$$\Phi = \begin{bmatrix} -y(1) & u(2) & u(1) \\ -y(2) & u(3) & u(2) \\ . & . & . \\ . & . & . \\ . & . & . \\ -y(N) & u(N+1) & u(N) \end{bmatrix} \tag{88}$$

A função custo a ser minimizada é representada pelo produto da transposta da função matricial do erro e a própria função matricial do erro:

$$J = \sum [\epsilon(k)]^2 = \epsilon^T\epsilon \tag{89}$$

Fazendo a derivada jacobiana $\frac{dJ}{d\hat{\theta}} = 0$, J será minimizado. Dessa forma, a matriz $\hat{\theta}$ é dada por:

$$\hat{\theta} = (\Phi^T\Phi)^{-1}\Phi^T y \tag{90}$$

A resolução desse método é viável para sistemas não dinâmicos e que sejam passíveis de serem analisados de forma *offline*. Nesse caso, como citado em Yu et al. (2012), o sistema em questão é submetido constantemente a variações de temperatura, envelhecimento e variações de carga, o que consequentemente leva a necessidade da utilização de um método recursivo.

Dito isto, em variáveis recursivas e discretas, as matrizes ϕ, $\hat{\theta}$ e a função erro podem ser representadas respectivamente como:

$$\phi^T = \begin{bmatrix} y(k-1) & u(k) & u(k-1) \end{bmatrix} \tag{91}$$

$$\hat{\theta} = \begin{bmatrix} 1 & b_0 & b_1 \end{bmatrix}^T \tag{92}$$

$$\epsilon(k) = y(k) - \phi^T(k)\hat{\theta} \tag{93}$$

Considerando que a equação (88) pode ser escrita como:

$$\Phi(k) = \begin{bmatrix} \phi^T(0) & \phi^T(1) & ... & \phi^T(k) \end{bmatrix}^T \tag{94}$$

Pode-se dizer que a definição da equação (90), em forma recursiva, é análoga a seguinte expressão:

$$\hat{\theta}(k) = \Big[\sum_{i=1}^{k} \phi(i-1)\phi^T(i-1) \Big]^{-1} \Big[\sum_{i=1}^{k} \phi(i-1)y(i) \Big]^{-1} \tag{95}$$

Buscando conceitos da teoria da probabilidade e na estatística, a covariância entre duas variáveis tem definição base pela seguinte equação:

$$P(k) = (\Phi^T\Phi)^{-1} = \Big[\sum_{i=1}^{k} \phi(i-1)\phi^T(i-1) \Big]^{-1} \tag{96}$$

Recursivamente, manipulando a equação é análogo também constatar que, separando o termo relativo a k no somatório, aplicando a inversa na variável $P(k)$ e somando com o termo restante relativo a $k-1$, tem-se que:

$$P(k)^{-1} = \sum_{i=1}^{k-1} \phi(i-1)\phi^T(i-1) + \phi(k-1)\phi^T(k-1) \tag{97}$$

Nessa etapa, substitui-se $P(k)^{-1}$ por $R(k)$. De forma análoga:

$$R(k) = R(k-1) + \phi(k-1)\phi^T(k-1) \tag{98}$$

Objetiva-se retornar a variável que representa a covariância, então, para estender a equação anterior utiliza-se da seguinte propriedade matemática de inversão de matrizes:

$$[A + BCD]^{-1} = A^{-1} - A^{-1}B[DA^{-1}B + C^{-1}]DA^{-1} \tag{99}$$

Onde $A = R(k-1)$, $B = D^T = \phi(k)$ e $C = 1$.

Substituindo as variáveis e retornando da representação de R para a variável P, encontra-se que a matriz de covariância é dada como:

$$P(k) = P(k-1) - \left[\frac{P(k-1)\phi(k)\phi(k)^T P(k-1)}{1 + \phi(k)^T P(k-1)\phi(k)}\right] \tag{100}$$

Tendo em mãos o conhecimento dessas manipulações matemáticas e da relação da variável da covariância, pode-se retornar a variável responsável pela atualização recursiva $\hat{\theta}$. Da equação (90) tem-se que:

$$\hat{\theta}(k) = P(k)\Big[\sum_{i=1}^{k} \phi(i-1)y(i)\Big]^{-1} \tag{101}$$

O que é análogo a:

$$\hat{\theta}(k) = R(k)^{-1}[R(k-1)\hat{\theta}(k-1) + \phi(k)y(k)] \tag{102}$$

Ou seja:

$$\hat{\theta}(k) = \hat{\theta}(k-1) + R(k)^{-1}\phi(k)[y(k) - \phi(k)^T\hat{\theta}(k-1)] \tag{103}$$

A variável $R(k)^{-1}\phi(k)$ representa a matriz de ganho e tem um papel importante na participação do algoritmo, sua definição é substituída pela variável K, sendo conhecida como Ganho de Kalman, sendo:

$$K(k) = R(k)^{-1}\phi(k) = P(k)\phi(k) = \frac{P(k-1)\phi(k)}{1 + \phi^T(k)P(k-1)\phi(k)} \tag{104}$$

Além disso, também é abordado em (LJUNG, 1999) a adição de um termo λ nas equações recursivas. Este, chamado de "fator de esquecimento", é responsável por considerar a influência das variáveis passadas nos cálculos de momento presente. O autor também demonstra que o termo em questão também está diretamente relacionado a analogia do método iterativo RLS com o Filtro de Kalman. A definição de Kalman para estimativa do estado do sistema é dada como:

$$x(k+1) = F(k)x(k) + \omega(k) \tag{105}$$

$$y(t) = H(k)x(k) + v(k) \tag{106}$$

O que para o modelo analisado nesse trabalho, traduz-se que:

$$\hat{\theta}(k+1) = \hat{\theta} \tag{107}$$

$$y(t) = \phi^T(k)\hat{\theta}(k) + v(t) \tag{108}$$

Aplicando a definição de filtro de Kalman, têm-se exatamente as definições das variáveis recursivas para quando λ tem valor unitário, refletindo em uma resposta não suscetível a variações rápidas de variáveis incontroláveis do sistema. Em resumo, considerando uma faixa de 0 a 1, quanto mais próximo de zero o fator de esquecimento, mais suscetível a resposta do algoritmo tende a corrigir variações externas, entretanto, também adiciona ruídos e imprecisão. Uma modificação para o caso em análise pode ser aplicada considerando o fator sendo variável, dependendo da temperatura do núcleo estimada. Como sugestão, essa alteração pode ser feita simplesmente criando uma relação com a taxa de variação de temperatura medida, já que no próximo capítulo será verificado que quanto maior a temperatura, maior será a variação e imprecisão do método. Sabe-se, entretanto, que sensores de temperatura adicionam ruídos aos valores medidos, então aconselha-se que ao menos um filtro média-móvel seja aplicado ao sinal recebido antes da modificação. No caso desse trabalho não será necessário, pois não se espera alterações bruscas de temperatura ao ponto de se necessitar um λ variável.

A descrição e evolução matemática com o termo λ é desenvolvida em Ljung (1999) e segue a mesma lógica aqui apresentada, porém incluindo o novo termo, portanto não será repetida nesse trabalho. Dessa forma, manipulando as equações descritas juntamente das definições encontradas com a inserção do fator de esquecimento nas mesmas, pode-se sumarizar as variáveis recursivas de interesse como:

$$K(k) = \frac{P(k-1)\phi(k)}{\lambda + \phi^T(k)P(k-1)\phi(k)} \tag{109}$$

$$\hat{\theta}(k) = \hat{\theta}(k-1) + K(k)\epsilon(k) \tag{110}$$

$$P(k) = \frac{(I - K(k)\phi^T)P(k-1)}{\lambda} \tag{111}$$

Com relação aos valores de $\hat{\theta}(k-1)$ e $P(k-1)$ iniciais, estes podem ser facilmente calculados com base em uma estimativa da ESR e C inicial, ou seja, define-se B_0 e B_1. Já a matriz de covariância $P(k-1)$ depende do quão confiável a estimativa inicial é, considerando um bom chute inicial para B_0 e B_1, pode-se definir a matriz como a multiplicação de um valor muito próximo de zero por sua matriz identidade 3x3.

Por fim, o resultado da ESR e capacitância finais são extraídos através das constantes b0 e b1 obtidas da matriz $\hat{\theta}(k)$. Dessa forma:

$$C = \frac{T_s}{b_1 + b_0} \tag{112}$$

$$ESR = b_0 - \frac{T_s}{2C} \tag{113}$$

2.0.5 Previsão de Vida Útil: Equação de Arrhenius

Nesta seção será exposto o desenvolvimento das equações de predição de vida útil utilizando Arrhenius, considerando individualmente e de forma total, a influência da temperatura, corrente e tensão no cálculo de vida útil do capacitor.

2.0.5.1 Fator de Temperatura

Em situações ideais onde o sistema possui um comportamento previsível e com ambiente controlado, a previsão da vida útil do capacitor pode ser baseada em variação de corrente, temperatura ambiente e tensão de barramento aplicada, segundo a equação de Arrhenius. A relação de difusão do eletrólito segue a Lei de Arrhenius:

$$k_{\mathrm{re}} = Ae^{-\frac{E}{RT}} \tag{114}$$

onde:
k_{re}: Constante da taxa de reação;
A: Fator de frequência;
E: Energia de ativação;
R: Constante de gás;
T: Temperatura absoluta (K);

A constante da taxa de reação pode ser diretamente relacionada a velocidade que a reação química ocorre, determinada pela relação do número de eventos ocorridos em um determinado intervalo de tempo e o número de eventos possíveis total. Nesse caso em específico, os "eventos" são a degradação do capacitor, e a constante k_{re} pode ser relacionada diretamente com a vida útil do capacitor medida L_o a uma temperatura T_o e a vida útil de referência do capacitor $L_{x,T}$ a uma temperatura T_x.

Onde L_o e $L_{x,T}$ podem ser definidos pela seguinte forma geral:

$$L = \frac{1}{k_{\mathrm{re}}} \tag{115}$$

Substituindo (114) em (115):

$$L = \frac{1}{Ae^{-\frac{E}{RT}}} \tag{116}$$

Sendo k_{re} a relação entre as duas variáveis:

$$k_{\mathrm{re}} = \frac{L_{x,T}}{L_o} \tag{117}$$

Pode-se substituir a relação $\frac{L_{x,T}}{L_o}$ por:

$$\frac{L_{x,T}}{L_o} = \frac{\frac{1}{Ae^{-\frac{E}{RT_0}}}}{\frac{1}{Ae^{-\frac{E}{RT_x}}}} \tag{118}$$

Aplicando o logaritmo natural em ambos os lados:

$$ln\left(\frac{L_{x,T}}{L_o}\right) = ln\left(\frac{\frac{1}{Ae^{-\frac{E}{RT_0}}}}{\frac{1}{Ae^{-\frac{E}{RT_x}}}}\right) \tag{119}$$

Simplificando o lado direito da equação:

$$ln\left(\frac{L_{x,T}}{L_o}\right) = ln\left(e^{\frac{E}{RT_0}-\frac{E}{RT_x}}\right) \tag{120}$$

E usando propriedades do logaritmo:

$$ln\left(\frac{L_{x,T}}{L_o}\right) = \frac{E}{RT_0} - \frac{E}{RT_x} \tag{121}$$

Manipulando a equação, tem-se que:

$$ln\left(\frac{L_{x,T}}{L_o}\right) = \frac{E}{R}\left(\frac{1}{T_0} - \frac{1}{T_x}\right) \tag{122}$$

Além disso, a constante R é substituída pela constante de Boltzmann (k_B) com a relação:

$$R = k_B N_{\text{av}} \tag{123}$$

Onde N_{av} é a constante de Avogadro e relaciona a quantidade de átomos ou moléculas presentes de uma determinada massa de uma substância. Essa constante não é mencionada explicitamente já que não tem influência no cálculo da vida útil de capacitores.

Substituindo as variáveis e adaptando a equação para a vida útil do capacitor tem-se que:

$$ln\left(\frac{L_{x,T}}{L_o}\right) = \frac{E}{k_B}\left(\frac{1}{T_0} - \frac{1}{T_x}\right) \tag{124}$$

onde:

k_B: Constante de Boltzmann;

$L_{x,T}$: Predição da vida útil considerando influência da temperatura;

L_o: Vida útil nominal especificada no *datasheet*;

Ou ainda:

$$L_{x,T} = L_o e^{\frac{E}{k_B}\left(\frac{1}{T_0}-\frac{1}{T_x}\right)} \tag{125}$$

Então, pode-se considerar que essa parcela confere a influência da temperatura na vida útil do capacitor, portanto, uma constante K_T relativa pode ser criada, simplificando a equação:

$$L_{x,T} = L_o K_T \tag{126}$$

Em capacitores eletrolíticos, a falha geralmente acontece devido à quebra da película de óxido nos eletrodos. A energia de ativação para este mecanismo de falha pode variar dependendo dos materiais específicos e da construção do capacitor, onde a energia de ativação para a quebra em capacitores eletrolíticos está na faixa de 0,5 eV a 1 eV. Em geral, utiliza-se como padrão o valor de 0,94 eV, enquanto a constante de Boltzman tem o valor de $k_B = 0{,}862\,\mu\text{eV/K}$. Essas definições resultam que a razão $Ea/k = 1{,}091 \cdot 10^4$ K. Simplificando a equação (125) e multiplicando as frações relacionadas à temperatura, tem-se que a equação pode ser refatorada como:

$$L_{x,T} = L_o e^{1{,}091 \cdot 10^4 \left(\frac{T_x - T_0}{T_x T_0}\right)} \tag{127}$$

Baseando-se que capacitores eletrolíticos geralmente não ultrapassam o valor nominal máximo de 125°C (398K), pode-se utilizá-lo como referência para simplificar a equação. Isso resulta em um padrão muito conhecido e utilizado na indústria, que diz que a cada aumento de 10°C na temperatura de operação, a vida útil do capacitor eletrolítico reduz pela metade (PARLER, S. G.; DUBILIER, 2004):

$$L_{x,T} = L_o e^{\frac{1{,}091 \cdot 10^5}{398^2}\left(\frac{T_x - T_0}{10}\right)} \approxeq L_o e^{\frac{ln(2)\Delta T}{10}} = L_o 2^{\frac{\Delta T}{10}} \tag{128}$$

2.0.5.2 Fator de Corrente

Quando considerado a influência do *ripple* de corrente no capacitor, uma versão bem estabelecida é apresentada como:

$$L_{x,i} = L_o K_r^{\frac{T_0 - T_x}{10}} \tag{129}$$

onde $L_{x,i}$ é a vida útil considerando a influência do *ripple* de corrente e K_r é um fator obtido experimentalmente (KHANDEBHARAD et al., 2015; SPANIK; FRIVALDSKY; KANOVSKY, 2014):

- $K_r = 2$ se $I_{\text{rms,x}} \leq I_{\text{rms,0}}$;
- $K_r = 4$ se $I_{\text{rms,x}} > I_{\text{rms,0}}$.

onde $I_{\text{rms,x}}$ é o valor RMS da corrente do capacitor em operação com os harmônicos corrigidos para a base do *datasheet* (geralmente 100 Hz) e $I_{\text{rms,0}}$ é o valor RMS da corrente do capacitor no valor nominal.

Com o intuito de se fazer uma simplificação, sabe-se que a potência dissipada pelo capacitor depende da ESR, dessa forma:

$$P = ESRI_{\text{rms}}^2 \tag{130}$$

Visando obter o equilíbrio entre geração e dissipação de calor, é derivada a seguinte equação:

$$ESRI_{\text{rms}}^2 = \beta A_r \Delta T \tag{131}$$

onde:
β: Constante de radiação (aproximadamente 1,5 a $2 \cdot 10^{-3}$ W/cm^2×°C);
A_r: Área de superfície (cm^2);
ΔT: Variação de temperatura devido ao *ripple* de corrente;

O que também é traduzido como:

$$ESRI_{\text{rms}}^2 = \frac{\Delta T}{R_{\text{th}}} \tag{132}$$

onde R_{th} representa a resistência térmica.

Para esse caso em específico, visando idealidade e simplificação do equacionamento, considera-se que a ESR e a resistência térmica não variam com a temperatura, e portanto, a relação entre a variação de temperatura de operação e nominal pode ser obtida como:

$$\frac{\Delta T_x}{\Delta T_0} = \frac{R_{\text{th}} ESRI_{\text{rms,x}}^2}{R_{\text{th}} ESRI_{\text{rms,0}}^2} = \left(\frac{I_{\text{rms,x}}}{I_{\text{rms,0}}}\right)^2 \tag{133}$$

Substituindo a relação encontrada em (129), é encontrado que a equação relacionando a variação de corrente é obtida como:

$$L_{x,i} = L_o K_r^{\left(1-\left(\frac{I_{\text{rms,x}}}{I_{\text{rms,0}}}\right)^2\right)\frac{\Delta T_0}{10}} \tag{134}$$

Então, uma nova constante K_I que denota a influência da variação de corrente no capacitor é dada por:

$$K_I = K_r^{\left(1-\left(\frac{I_{\text{rms,x}}}{I_{\text{rms,0}}}\right)^2\right)\frac{\Delta T_0}{10}} \tag{135}$$

A constante ΔT_0 pode variar nos seguintes valores:

$$\begin{aligned} 3{,}5K \leq \Delta T_0 \leq 5K \rightarrow T_0 = 378{,}15K\ (105^\circ\text{C}) \\ 5K \leq \Delta T_0 \leq 10K \rightarrow T_0 = 358{,}15K\ (85^\circ\text{C}) \end{aligned} \tag{136}$$

Assim, a vida útil do componente considerando a variação de corrente é dada como:

$$L_{x,i} = L_0 K_I \tag{137}$$

2.0.5.3 Fator de Tensão

Além disso, também é conhecido que a tensão aplicada nos terminais do capacitor pode alterar a vida útil remanescente do componente, já que pode causa estresse no dielétrico. Quanto mais a tensão operacional se aproxima da tensão nominal, mais os íons no eletrólito são consumidos para a reparação de pequenas falhas dentro da camada dielétrica chamadas de autocorreção (HAYEK et al., 2018). O consumo de eletrólitos também é exponencialmente dependente da temperatura. Ou seja, uma tensão de operação mais baixa que a tensão nominal pode prolongar significativamente a vida útil dos capacitores. A seguinte equação relaciona essa razão e foi derivada de forma empírica (ALBERTSEN, 2010):

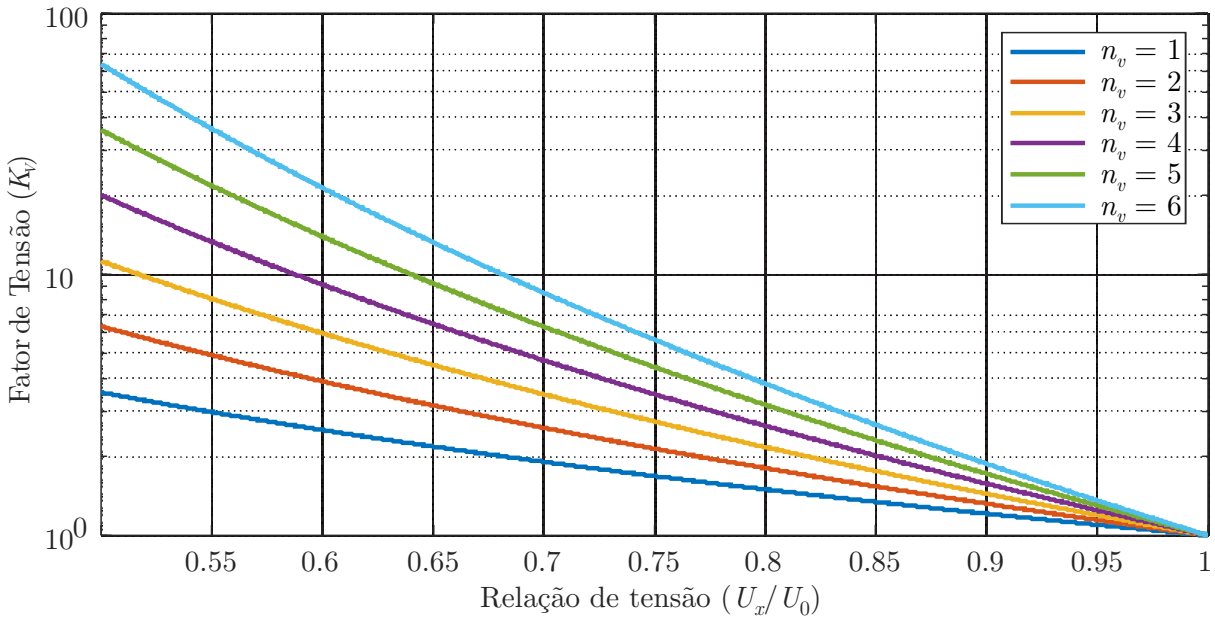
$$K_V = \left(\frac{U_x}{U_0}\right)^{-n_v} \tag{138}$$

onde:
U_0: Tensão nominal;
U_x: Tensão de operação.

Para capacitores eletrolíticos de tamanho radial pequeno, K_V é adotado como unitário. A variação de n_v depende exclusivamente da construção do componente, nem sempre é dada pelo fabricante e a equação que define o comportamento da constante pode mudar.

Graficamente, a equação (138) é representada na Figura 30 variando n_v na faixa de 1 a 6 considerando a relação de tensão e o fator de tensão para ilustrar o comportamento da equação.

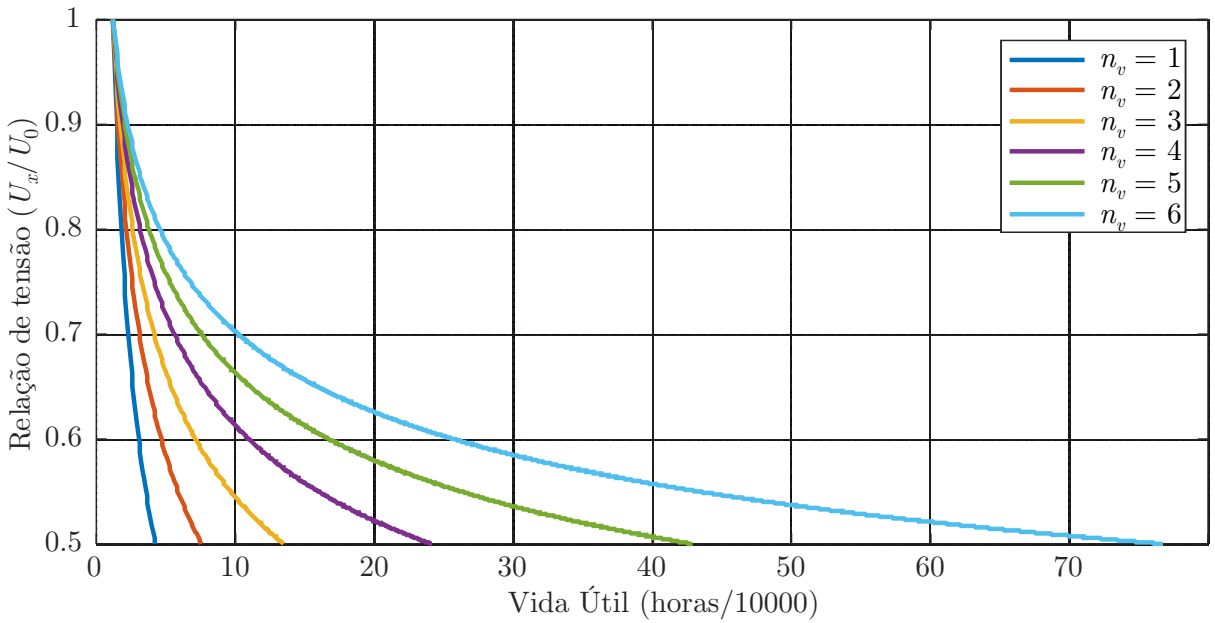
Figura 30 – Fator de tensão em função da relação de tensão variando n_v.



Fonte: do Autor.

Ainda, a equação (138) também é representada na Figura 31 variando n_v, porém, considerando a vida útil e a relação de tensão.

Figura 31 – Relação de tensão em função da vida útil remanescente variando n_v.



Fonte: do Autor.

Analisando os gráficos obtidos, as seguintes equações que relacionam a relação de

tensão podem ser utilizadas como base, caso não haja disponibilidade dos dados pelo fabricante (HAYEK et al., 2018):

$$\begin{aligned} 0 \leq \frac{U_x}{U_0} < 0{,}8 \rightarrow n_v = 3 \\ 0{,}8 \leq \frac{U_x}{U_0} \leq 1 \rightarrow n_v = 5 \end{aligned} \tag{139}$$

Finalmente, a equação de Arrhenius da predição da vida útil que considera os três principais fatores (equações (125), (135), (138)) é expressa como:

$$L_x = L_0 K_T K_I K_V = L_0 2^{\frac{T_0 - T_x}{10}} K_r^{\left[1-\left(\frac{I_{\text{rms,x}}}{I_{\text{rms,0}}}\right)^2\right]\frac{\Delta T_0}{10}} \left(\frac{U_x}{U_0}\right)^{-n_v} \tag{140}$$

2.0.5.4 Exemplo Prático

Para exemplificar as equações desenvolvidas, considere que um conversor está em operação com uma tensão de barramento de 400 V, onde o capacitor B43707 da EPCOS conectado uma tensão máxima de 400 V, tendo uma vida útil de referência de 12000 horas e uma capacitância de 2200 µF. A temperatura máxima desse componente é de 85°C (358,15K) e a de operação se encontra na faixa de 60°C (333,15K). Para o estresse de corrente, o valor máximo suportado é de 7,65 A e o valor de operação está na faixa de 4 A (já com as harmônicas corrigidas para a base de 100 Hz), com uma variação (ΔT_0) de 7 K.

A constante relativa a influência de temperatura pode ser calculada da seguinte forma:

$$K_T = 2^{\frac{358{,}15-333{,}15}{10}} = 5{,}657 \tag{141}$$

Já para a influência de corrente, tem-se que:

$$K_I = 2^{\left(1-\left(\frac{4}{7{,}65}\right)^2\right)\frac{7}{10}} = 1{,}423 \tag{142}$$

Finalmente, para a variação de tensão:

$$K_V = \left(\frac{400}{400}\right)^{-5} = 1 \tag{143}$$

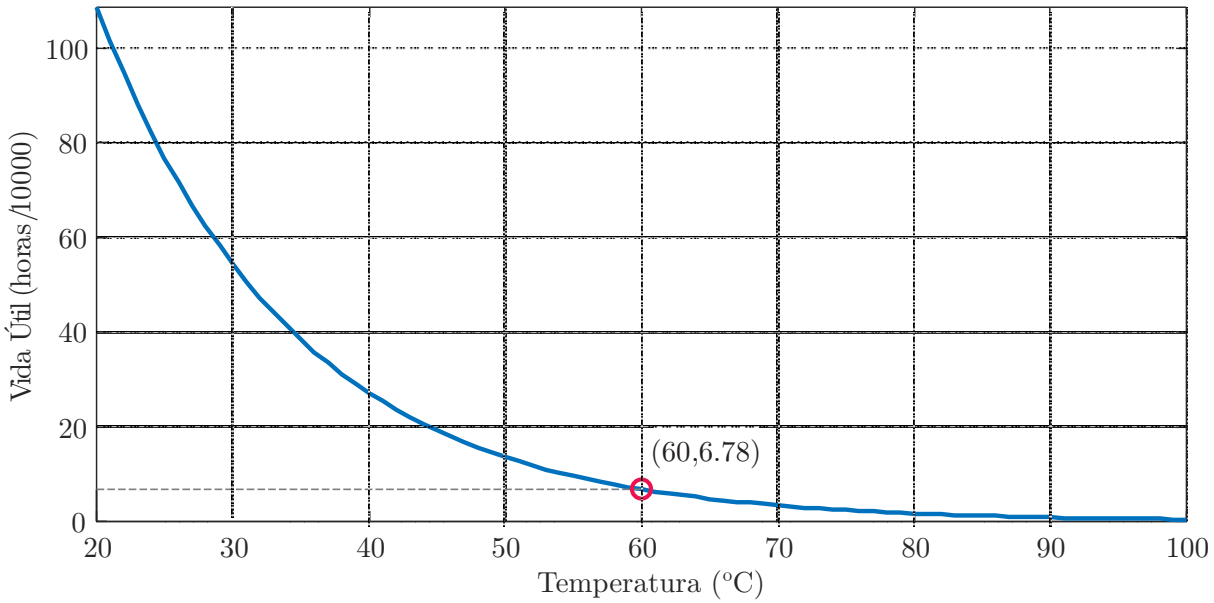
Unindo todas as constantes, tem-se que a vida útil remanescente resultante é dada como:

$$L_x = 12000 \cdot 5{,}657 \cdot 1{,}423 \cdot 1 = 96575\, horas \tag{144}$$

Perceba que a vida útil calculada para o capacitor é aproximadamente 8 vezes maior do que a mencionada no *datasheet*. Isso ocorre principalmente devido à influência da temperatura, que tem um impacto significativo no tempo de vida restante do componente.

Uma forma de visualizar facilmente essa influência é através de um gráfico. Considere que inicialmente as constantes referentes à corrente e à tensão têm valor unitário e que a análise será baseada apenas na variação de temperatura, de 20 a 100°C. A Figura 32 mostra essa relação, apresentando um comportamento exponencial, onde a vida útil é reduzida conforme a temperatura está abaixo da nominal especificada pelo *datasheet* e vice-versa. Utilizando o exemplo anterior, a vida útil se encontra em torno de 67000 horas quando a temperatura é de 60°C.

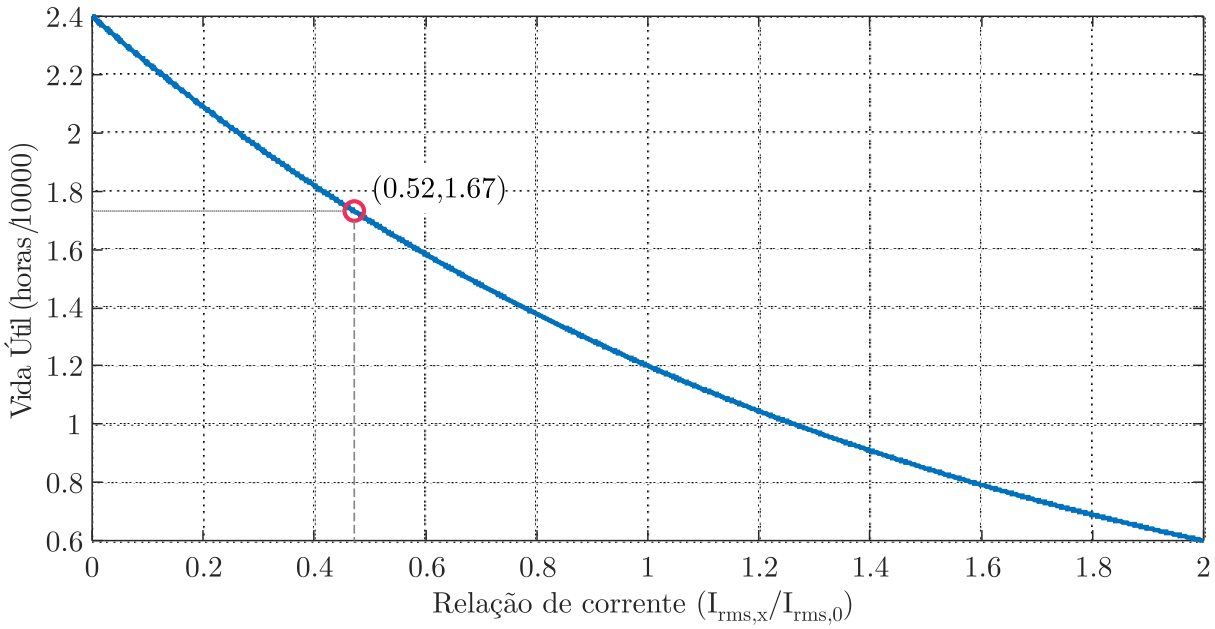
Figura 32 – Variação da vida útil em função da temperatura do capacitor, na condição de corrente e tensão nominal.



Fonte: do Autor.

Já para a variação de corrente, considere o valor de operação variando de 0 a 15,3 A, com a temperatura fixa em 85°C ($\Delta T_0 = 10K$), a Figura 33 ilustra essa situação.

Figura 33 – Variação da vida útil em função da corrente eficaz do capacitor, na condição de temperatura constante em 85 C.



Fonte: do Autor.

2.0.6 Previsão de Vida Útil: Método de Gauss-Newton

Por outro lado, em modelos não ideais e imprevisíveis onde as condições de operação e temperatura variam com o tempo, deve-se utilizar os dados armazenados de ESR e C calculados através dos métodos anteriores para se obter uma previsão mais precisa a longo prazo. Essa pode ser obtida através da extrapolação de dados utilizando algoritmos específicos.

Assim sendo, o método de extrapolação Levenberg-Marquadt será utilizado para prever a vida útil do capacitor analisado, sendo esse um dos métodos mais utilizados para essa finalidade. Recomenda-se que a implementação do algoritmo seja realizada paralelamente à rotina principal e com a supervisão externa frequente, uma vez que a previsão geralmente é realizada com intervalos de longo prazo. Cada caso deve ser avaliado individualmente, mas, em condições normais, a variação dos parâmetros internos do capacitor apresenta uma tendência lenta de mudança. Portanto, a coleta automática dos dados pode ser prejudicada devido ao longo período de tempo entre as amostras.

Para se ter compreensão do algoritmo Levenberg-Marquadt, deve-se primeiramente explanar sobre o método de Gauss-Newton, já que se trata de uma otimização do mesmo para casos em que não é possível se obter convergência.

O objetivo principal de se aplicar métodos de convergência para sistemas não-lineares é encontrar um vetor que minimiza a função erro, da forma:

$$F(x) = \sum_{i=1}^{m} v_i(x)^2 = ||v(x)^2|| = v(x)^T v(x) \tag{145}$$

Onde F(x) representa o erro total relacionado ao ajuste proporcionado por iterações. Da definição da função erro, tem-se que:

$$f_i = y_i - y(x_i) \tag{146}$$

Onde a convergência irá depender do erro limite, pré-definido no início da iteração do algoritmo implementado. Complementa-se que há casos em que funções podem convergir em diversos valores de mínimos diferentes. Nesse caso, a convergência irá ocorrer em função do valor inicial escolhido, que irá ser adequado automaticamente a cada iteração.

Em processo iterativo, descreve-se a atualização do parâmetro futuro como:

$$x_{k+1} = x_k - Z_k^{-1} g_k \tag{147}$$

Onde $Z_k = \nabla^2 F(x_k)$ e $g_k = \nabla F(x_k)$.

Assumindo a definição da função global $F(x)$, define-se que o "j_{th}" elemento do gradiente será:

$$[\nabla F(x)]_j = \frac{\partial F(x)}{\partial x_j} = 2 \sum_{i=1}^{m} v_i \frac{\partial v_i(x)}{\partial x_j} \tag{148}$$

Então, o gradiente pode ser escrito na forma matricial como:

$$\nabla F(x) = 2 J^T(x) V(x) \tag{149}$$

Onde:

$$\mathbf{J}(\mathbf{x}) = \begin{bmatrix} \frac{\partial v_1}{\partial x_1} & \frac{\partial v_1}{\partial x_2} & \cdots & \frac{\partial v_1}{\partial x_n} \\ \frac{\partial v_2}{\partial x_1} & \frac{\partial v_2}{\partial x_2} & \cdots & \frac{\partial v_2}{\partial x_n} \\ \cdot & \cdot & & \cdot \\ \cdot & \cdot & & \cdot \\ \cdot & \cdot & & \cdot \\ \frac{\partial v_n}{\partial x_1} & \frac{\partial v_n}{\partial x_2} & \cdots & \frac{\partial v_n}{\partial x_n} \end{bmatrix} \tag{150}$$

Onde $J(x)$ é a matriz jacobiana.

Além disso, pode-se também definir a matriz Hessiana. O (k,j) elemento da matriz é dado como:

$$[\nabla^2 F(x)]_{k,j} = \frac{\partial F(x)}{\partial x_k \partial x_j} = 2 \sum_{i=1}^{m} v_i \frac{\partial^2 v_i(x)}{\partial x_k \partial x_j} + \frac{v_i(x)}{\partial x_k} \frac{v_i(x)}{\partial x_j} \tag{151}$$

A matriz Hessiana pode ser expressa em forma matricial como:

$$\nabla^2 F(x) = 2J^T(x)J(x) + 2S(x) \tag{152}$$

Onde $S(x) = \sum_{i=1}^{m} v_i \nabla^2 v_i(x)$.

Considerando $S(x)$ suficientemente pequeno, a Hessiana pode ser reconsiderada:

$$\nabla^2 F(x) \approx 2J^T(x)J(x) \tag{153}$$

Então, o método Gauss-Newton pode ser obtido, em forma de processo iterativo, na forma de:

$$x_{k+1} = x_k - [J^T(x)J(x)]^{-1} J^T(x_k)v(x_k) \tag{154}$$

Perceba que caso o termo envolvendo a inversão matricial não seja possível de ser realizada, o método não irá convergir. Com base nisso, é apresentada uma otimização chamada de Levenberg-Marquadt que será apresentada a seguir.

2.0.7 Previsão de Vida Útil: Método de Levenberg-Marquadt

A otimização de Levenberg-Marquadt (PRESS et al., 2007) nada mais é do que uma modificação do algoritmo de Gauss-Newton para possibilitar a convergência do mesmo quando a multiplicação de matrizes internas é não-inversível, cuja detecção pode ser observada simplesmente adicionando uma condicional com fator determinante necessariamente diferente de 0. O termo a ser adicionado tem forma de:

$$G = H + \mu I \tag{155}$$

Suponha que os autovetores e autovalores de H são: $\lambda_1, \lambda_2, ..., \lambda_n,\ z_1, z_2, ...z_n$.

Os autovetores de G são os mesmos autovetores de H, e os autovalores são $(\lambda_i + \mu)$. Então, G pode ser positivo aumentando μ até que $(\lambda_i + \mu) > 0$ tornando a matriz inversível. Substituindo a definição na equação geral do algoritmo de Gauss-Newton, tem-se que:

$$x_{k+1} = x_k - [J^T(x)J(x) + u_k I]^{-1} J^T(x_k)v(x_k) \tag{156}$$

Em termos de implementação, o usuário deve definir um palpite inicial para μ e verificar se a matriz é inversível, se não, multiplica-se por um fator β até que seja possível a inversão. Perceba que para um decrescimento de μ a um fator insignificante, retorna-se ao método original.

Por definição física, o parâmetro da ESR tende a incrementar seu valor exponencialmente no decorrer do tempo, enquanto a capacitância decresce linearmente, podendo ser razoavelmente aproximada a uma equação de reta.

Recomenda-se que a definição geral da equação exponencial para descrever o comportamento da ESR seja formada por uma combinação linear de duas funções exponenciais, já que na forma de uma única equação exponencial não seria capaz de prever com precisão o comportamento do parâmetro, assim:

$$ESR(t_h) = E_0 e^{E_1 t_h} + E_2 e^{E_3 t_h} \tag{157}$$

Onde t_h é o tempo em horas.

Aplicando a definição de derivada parcial em função das constantes (E_0, E_1, E_2 e E_3):

$$\frac{\partial ESR(t_h)}{\partial E_{0,2}} = e^{E_{1,2} t_h(i)} \tag{158}$$

$$\frac{\partial ESR(t_h)}{\partial E_{1,3}} = E_{0,2} t_h e^{E_{1,3} t_h(i)} \tag{159}$$

E a Jacobiana é definida como:

$$\mathbf{J}(\mathbf{x}) = \begin{bmatrix} \frac{\partial ESR(th_i)}{\partial E_0} & \frac{\partial ESR(th_i)}{\partial E_1} & \frac{\partial ESR(th_i)}{\partial E_2} & \frac{\partial ESR(th_i)}{\partial E_3} \\ . & . & . & . \\ . & . & . & . \\ . & . & . & . \\ \frac{\partial ESR(th_n)}{\partial E_0} & \frac{\partial ESR(th_n)}{\partial E_1} & \frac{\partial ESR(th_n)}{\partial E_2} & \frac{\partial ESR(th_n)}{\partial E_3} \end{bmatrix} \tag{160}$$

E a matriz $V(x)$ representa a diferença entre o valor esperado e a função definida:

$$\mathbf{V}(\mathbf{x}) = \begin{bmatrix} ESR_i - ESR(t_i) \\ . \\ . \\ . \\ ESR_n - ESR(t_n) \end{bmatrix} \tag{161}$$

Considerando o gradiente ∇ como:

$$\nabla = [J^T(x)J(x)]^{-1} J^T(x_k) v(x_k) \tag{162}$$

E que a atualização iterativa será dada como $E_{x,k+1} = E_{x,k} + \nabla E_{x,k}$ onde $x = (0,1,2,3)$, então, os erros podem ser definidos por:

$$Erro_{Ax} = \left| \frac{\nabla E_x}{E_x + \nabla E_x} \right| \tag{163}$$

Apesar de a capacitância ser uma função linear e simples de ser obtida, o método aplicado também pode ser utilizado, minimizando ainda mais a taxa de erro resultante na obtenção da equação. No caso da capacitância, a equação geral pode ser representada como:

$$C(t) = -B_0 t_h + C(0) \tag{164}$$

Aplicando a definição de derivada parcial em função de B_0:

$$\frac{\partial C(t)}{\partial B_0} = -t_h \tag{165}$$

E a Jacobiana é definida como:

$$\mathbf{J}(\mathbf{x}) = \begin{bmatrix} \frac{\partial C(t_i)}{\partial B_0} \\ . \\ . \\ . \\ \frac{\partial C(t_n)}{\partial B_0} \end{bmatrix} \tag{166}$$

E a matriz $V(x)$:

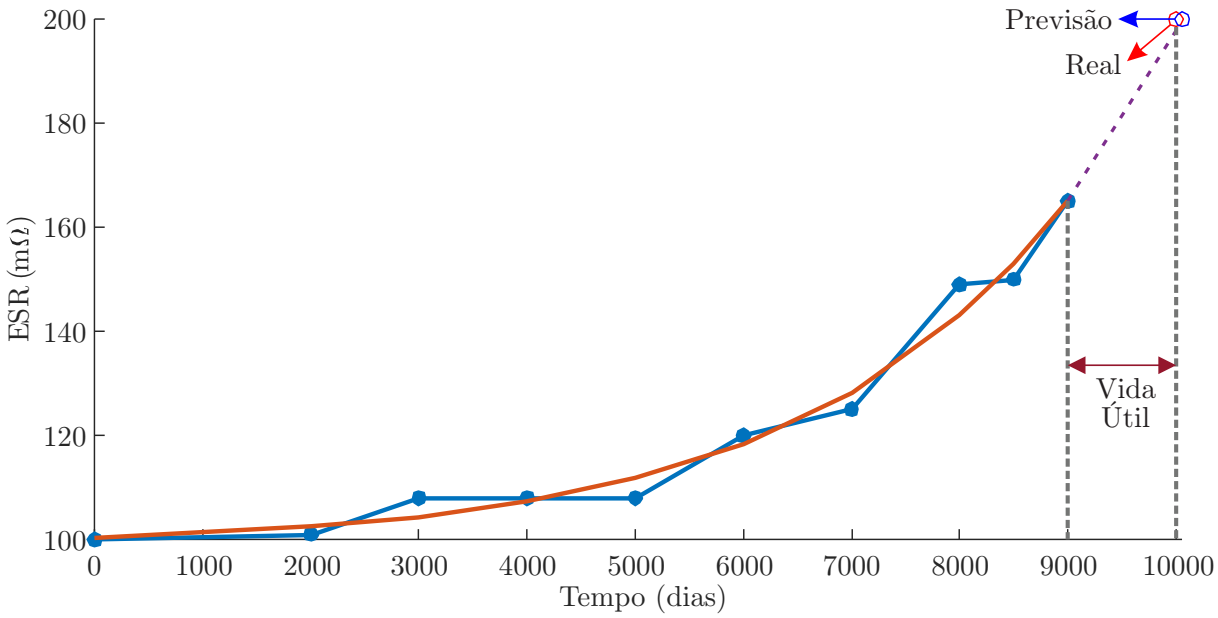
$$\mathbf{V}(\mathbf{x}) = \begin{bmatrix} C_i - C(t_i) \\ . \\ . \\ . \\ C_n - C(t_n) \end{bmatrix} \tag{167}$$

Por fim, o erro é definido como:

$$Erro_{B0} = \left| \frac{\nabla B_0}{B_0 + \nabla B_0} \right| \tag{168}$$

Considere como exemplo ideal um capacitor que teve um total de 10 medições ao longo de 9000h (375 dias). Após a extrapolação com o método Levenberg-Marquadt, a curva obtida tem comportamento como mostra a Figura 34.

Figura 34 – Extrapolação dos dados obtidos para o tempo de 9000 horas.



Fonte: do Autor.

Sabe-se que o primeiro ponto será a ESR de valor inicial, então, para $t_h = 0$ a soma das constantes E_0 e E_2 tem de ser igual a 100 mΩ nesse caso. Então, definindo um erro máximo de 0,1% e as constantes iniciais como (E_0, E_1, E_2 e E_3) = (99, 1μ, 1 e 500μ), aplicando a metodologia de Levenberg-Marquadt, a extrapolação dos dados armazenados entrega uma equação com forma de:

$$ESR(t_h) = E_0 e^{E_1 t_h} + E_2 e^{E_3 t_h} \tag{169}$$

Onde: (E_0, E_1, E_2 e E_3) = (98,3858, −1,7994μ, 1,9985 e 392,35μ).

Teoricamente, para fins de demonstração do funcionamento do algoritmo, adotemos que a ESR deva atingir o dobro do seu valor inicial em 10000 horas. Para prever o ponto nesse momento, basta substituir $ESR(t_h)$ pelo valor desejado, resultando em um t_h de aproximadamente 10057 horas.

2.1 CONSIDERAÇÕES FINAIS

Neste capítulo foi apresentada a sequência de passos para estimativa e predição da vida útil de um capacitor eletrolítico. Inicialmente o capacitor é submetido a uma câmara fechada onde a temperatura é controlada, variando de 25°C a 75°C, medindo a ESR e capacitância com um medidor de impedância na frequência de interesse. Após isso, para encontrar as equações referentes aos dados obtidos, essas variáveis são implementadas em um método de interpolação, o qual foi escolhido sendo o LS. Essa equação será importante para corrigir os valores de ESR e capacitância calculados pelos métodos RLS e DFT quando influenciados pela temperatura, já que pode confundir com as variações advindas do envelhecimento do componente.

Em sequência, é implementado a estimativa de corrente do capacitor baseando-se em variáveis internas que são passíveis de serem medidas facilmente com sensores no inversor e retificador. Algumas formas de onda são apresentadas para ilustrar o funcionamento do inversor mencionado.

Após isso, o método utilizando DFT é apresentado e todo o equacionamento é desenvolvido, desde a decomposição da forma geral até as equações como da magnitude e fase da harmônica do sinal na frequência de interesse. Esses dados são utilizados para calcular a ESR e C. Ainda, uma simulação é apresentada para demonstrar o funcionamento da DFT em termos ilustrativos.

Para o método RLS, o equacionamento tem início na análise do circuito e obtenção da função de transferência no domínio contínuo e discretizada com a transformada de Tustin. A metodologia dos mínimos quadrados recursivos é utilizada para encontrar os parâmetros da função de transferência obtida, que estão diretamente relacionados a ESR, capacitância e frequência de amostragem utilizada.

Por fim, o método de extrapolação Levenberg-Marquadt é utilizado para se obter

a previsão de quando o capacitor irá falhar, baseando-se em amostras obtidas ao longo do tempo. A aproximação estimada ficou próxima, mas, na prática, a quantidade das amostras obtidas será muito maior (cerca de centenas de pontos amostrados) o que irá facilmente convergir as equações de extrapolação para um resultado ainda mais próximo do valor real para o momento em que o capacitor irá de fato falhar, além de que capacitância e ESR serão analisadas simultaneamente, aumentando a confiabilidade da previsão.

3 RESULTADOS DE SIMULAÇÃO

Neste capítulo será apresentado o esquemático e resultados de simulação obtidos no software PSIM. Em uma tentativa de se aproximar do que foi montado em laboratório, os circuitos de condicionamento também foram considerados, porém, com componentes passivos ideais. Além disso, uma lógica semelhante à aquisição de dados realizada pelo DSP foi utilizada, onde não há memória e velocidade de processamento suficiente para realizar o cálculo do algoritmo RLS na interrupção do ADC. Portanto, duas simulações serão realizadas para cada método, uma sendo *online* com ambos os métodos em sua forma recursiva em tempo real (RLS e Goertzel) e outra de forma *offline*, ou seja, uma certa quantidade de dados deve ser armazenada para que posteriormente (fora da interrupção) os dados sejam processados e os métodos implementados. Resultados para implementação do algoritmo utilizando DFT e RLS são apresentados separadamente com uma variação de temperatura de 25 a 50°C.

3.1 RESULTADOS DE SIMULAÇÃO NO PSIM

A Figura 35 mostra o esquemático do circuito montado na ferramenta PSIM. Se trata de um retificador trifásico com entrada senoidal pura com tensão de linha de 400 V pico conectada a um inversor trifásico com carga R. Vale ressaltar que o componente do capacitor variável necessitou ser criado com fonte dependente de tensão respeitando a função de transferência do capacitor conforme equação (170), já que não há o componente na biblioteca da ferramenta. Além disso, o resistor variável foi implementado através de um componente não-linear modificado para funcionar como um resistor. As entradas dos blocos matemáticos se tratam das constantes A0, A1 e referências.

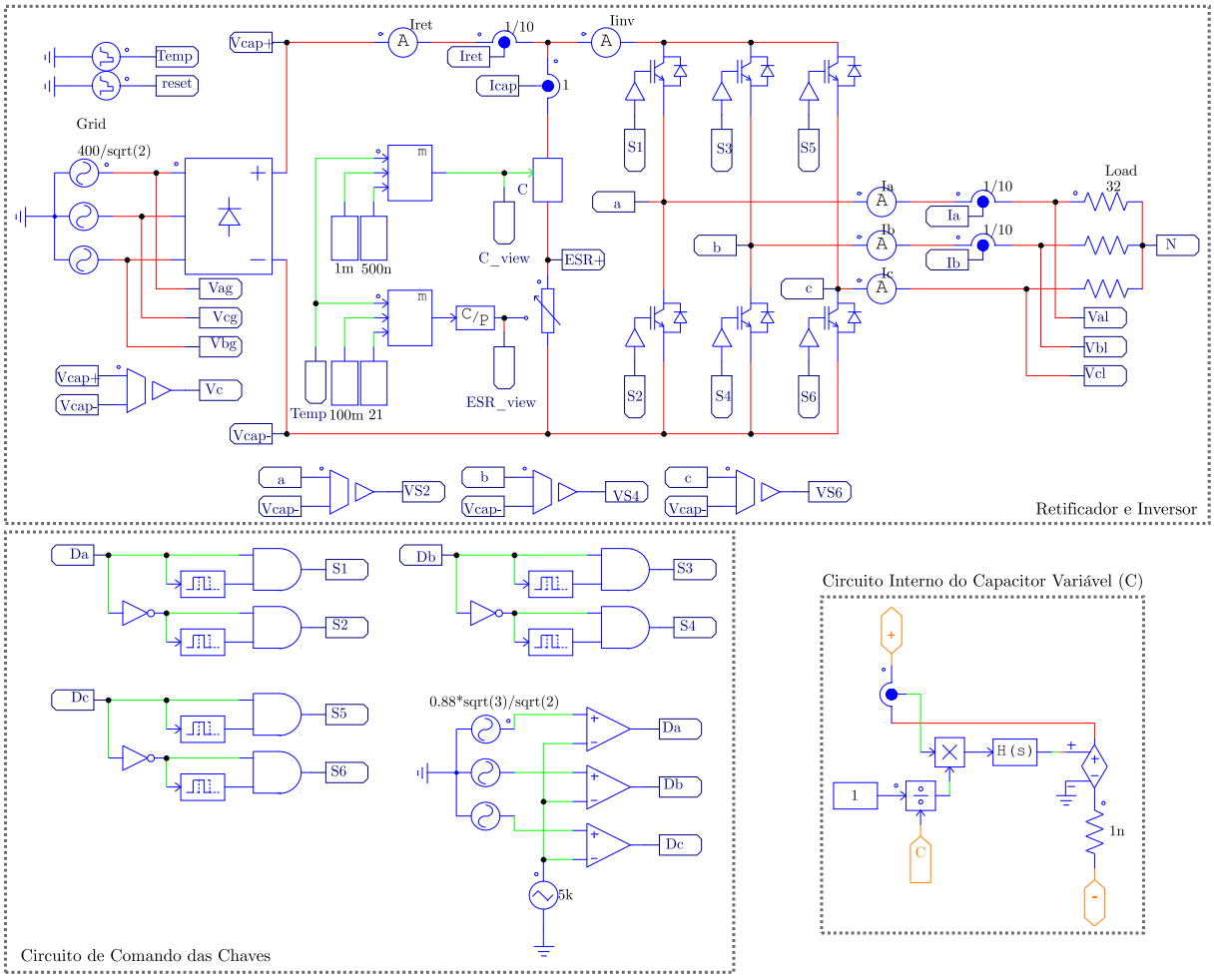
$$\frac{v_c(s)}{i_{\text{cap}}(s)} = \frac{1}{sC} \tag{170}$$

A modulação utilizada se trata da SPWM de 3 níveis, que utiliza sinais senoidais como referência de modulação comparando com uma portadora triangular. Na parte inferior esquerda da figura é apresentado o circuito que produz a modulação com índice de modulação de 0,88 com um tempo morto de 1 us. Apesar de ser recomendado a utilização de uma modulação vetorial para aproveitar o máximo do barramento CC, neste trabalho o objetivo é analisar os métodos de estimativa de capacitância e ESR, os quais independem da modulação utilizada e não alteram a precisão dos resultados.

Ressalta-se que o método foi testado para diferentes valores de tensão no barramento, inclusive com cargas variadas conectadas à saída do sistema, como filtro LC e Motor de Indução. Foi percebido que a precisão dos algoritmos sofrem apenas uma pequena influência do tipo de carga conectada, desde que haja algum tipo de variação na tensão do barramento para o método RLS funcionar ou alguma componente harmônica

de magnitude significativa para o método DFT realizar a aquisição dos dados e estimar as magnitudes e fases das componentes corretamente, esses resultados são discutidos e demonstrados brevemente na seção de considerações finais. Ainda, é de suma importância que a frequência de amostragem do ADC utilizado seja suficiente para se ter uma aquisição que preserve as características principais do sinal analisado.

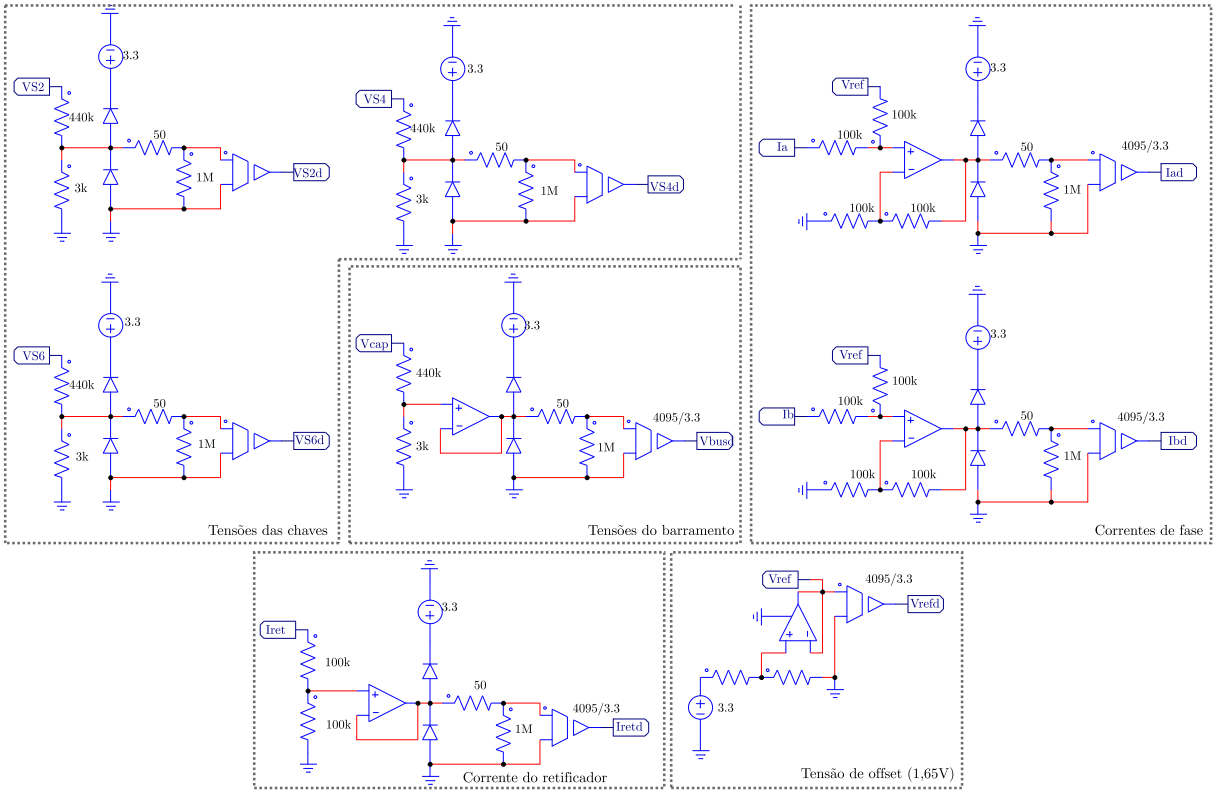
Figura 35 – Esquemático do circuito de potência no PSIM.



Fonte: do Autor.

A Figura 36 mostra o condicionamento dos sinais medidos, dos quais são as tensões nos interruptores $S_{2,4,6}$, tensão no barramento V_c, correntes de fase $I_{a,b}$, corrente do retificador I_{ret} e tensão de *offset* de 1,65 V (na prática, será obtido através de um regulador de tensão 5/3,3 V e um divisor de tensão com dois resistores de 100 kΩ). Além disso, caso haja utilização de filtragem no condicionamento, recomenda-se que seja escolhida uma frequência de corte alta o suficiente (a depender da frequência de amostragem) para não distorcer o sinal desejado, já que alterações nos sinais de entrada serão contabilizadas como penalidade quando processados pelos métodos RLS e DFT, reduzindo a precisão dos mesmos.

Figura 36 – Condicionamento dos sinais da simulação realizada no PSIM.



Fonte: do Autor.

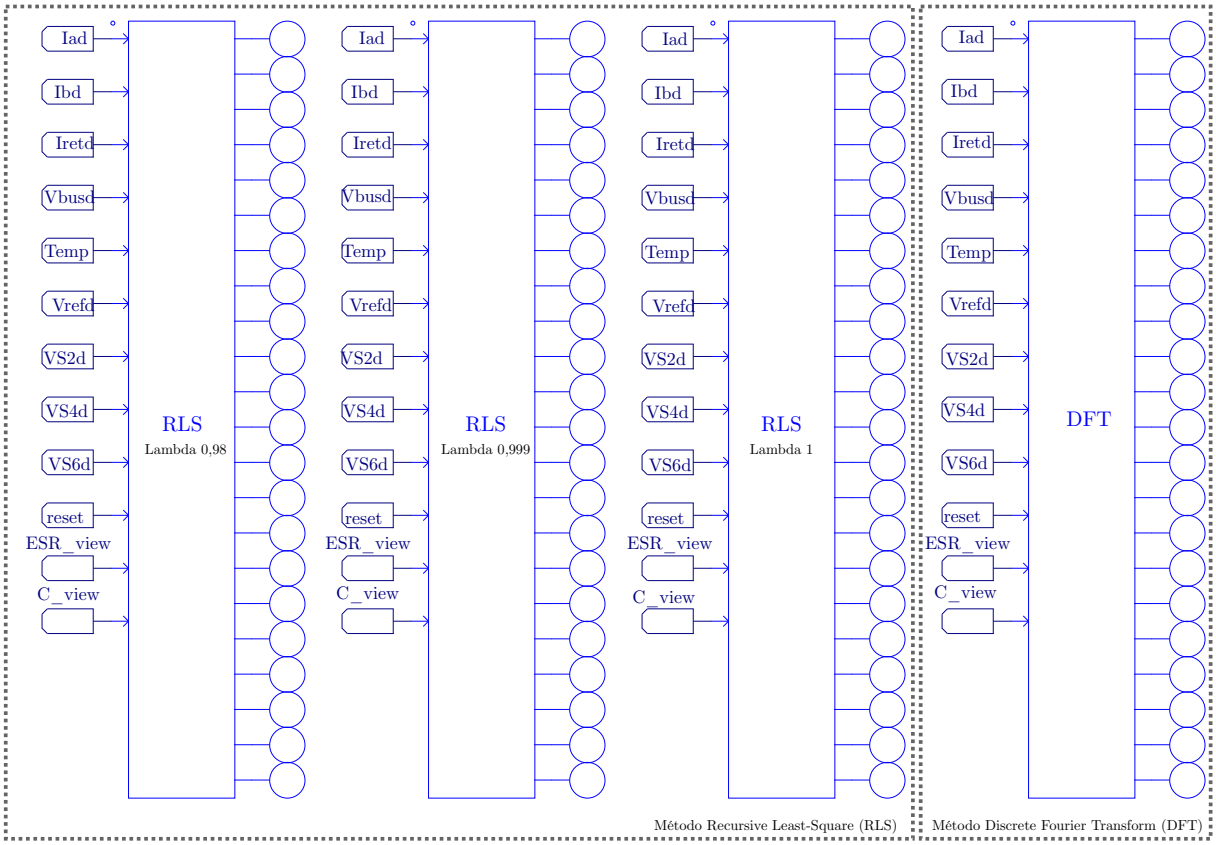
A Figura 37 mostra os blocos dos códigos dos métodos RLS e DFT, onde os códigos em si são apresentados nos apêndices deste trabalho. As entradas se tratam das variáveis lidas e condicionadas (tensões nos interruptores, tensão no barramento, correntes de fase, corrente no retificador, tensão de *offset* e temperatura) e também uma variável de "*reset*" para reinicializar a amostragem e armazenamento dos sinais. Além disso, diversos fatores de esquecimento (λ) foram escolhidos para demonstrar as consequências da resposta do sistema em relação a escolha desse parâmetro.

No caso de simulações *offline*, para o método RLS considera-se um passo de cálculo de 500 ns e frequência de amostragem de 100 kHz, um total de 1667 pontos são armazenados (resultando em 6 ciclos de 360 Hz ou 1 ciclo de 60 Hz), essa quantidade ficou limitada, pois no DSP não havia memória suficiente para se utilizar mais que 1 ciclo da rede para armazenar todas as variáveis e visualizá-las em tempo real, buscando manter o padrão, a simulação também considerou esse comportamento. Para se ter uma comparação justa com o método RLS, utilizou-se o método FFT Radix-4. Considera-se um passo de cálculo de 1 ns e frequência de amostragem de 92,16 kHz (ou seja, 256 amostras para 1 ciclo de 360 Hz).

Já para as simulações *online*, as mesmas taxas de amostragem foram mantidas, porém, no caso da DFT o método de Goertzel foi utilizado, também com 256 amostras.

Como abordado anteriormente, as tensões nos interruptores, corrente do retificador e correntes de fase serão utilizadas para estimar a corrente do capacitor.

Figura 37 – Blocos dos códigos dos algoritmos utilizando DFT e RLS.



Fonte: do Autor.

O sistema simulado tem como referência a tensão no barramento, e para se obter o valor de 400 V escolheu-se uma tensão eficaz para a entrada de 163,29 V com uma potência de 2,20 kW. Para fins de comparação, a simulação foi baseada no experimento real realizado em laboratório que será exposto no próximo capítulo, e por conta disso o barramento foi escolhido com esse nível específico de tensão. Os parâmetros de simulação são mostrados na Tabela 8.

3.1.1 Método *Recursive Least-Square* (RLS)

Nesta subseção serão apresentados os resultados obtidos na ferramenta PSIM utilizando o método RLS e suas características serão discutidas.

3.1.1.1 Modo *offline*

A Figura 38 mostra como foram realizados os passos sequenciados na simulação.

Tabela 8 – Parâmetros do sistema simulado.

Parâmetros	**Valores**
Tensão da rede elétrica eficaz	163,29 V
Frequência da rede elétrica	60 Hz
Potência do sistema	2,20 kW
Tensão do barramento CC	400 V
Corrente eficaz do retificador	11,43 A
Corrente eficaz de fase	4,75 A
Tensão eficaz de fase	152,31 V

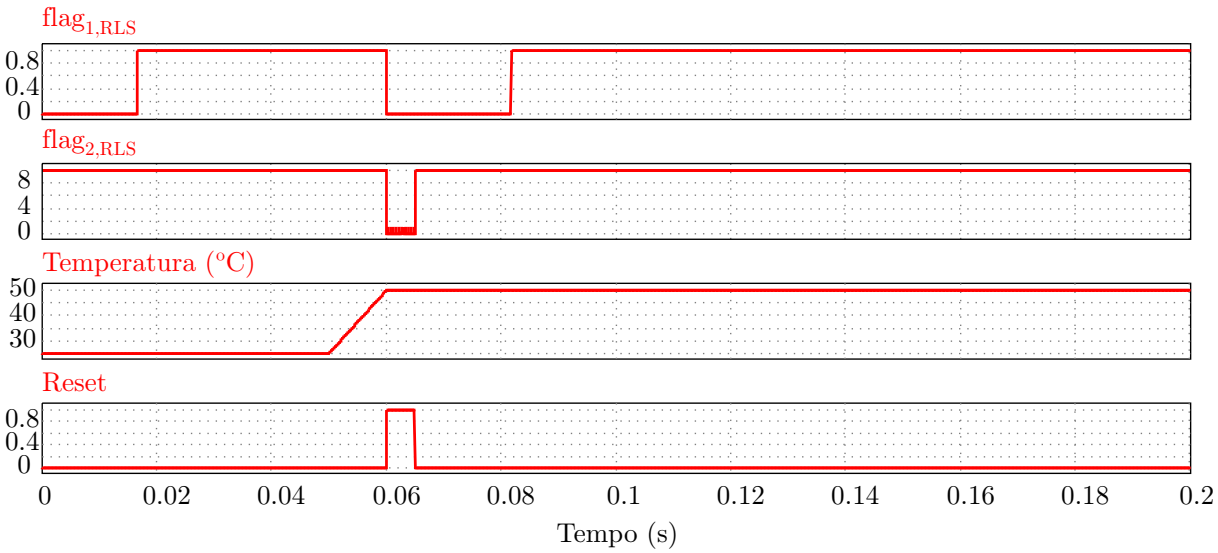
Fonte: do Autor.

Onde:

- $flag_{1,\mathrm{RLS}}$: é a *flag* que define o momento em que o vetor armazena a corrente estimada, ou seja, assume o valor 1, ou 0 para quando ainda está em processo de armazenamento;
- $flag_{2,\mathrm{RLS}}$: essa define uma certa quantidade de pontos a serem ignorados (com o intuito de ignorar transientes que possam modificar os resultados), nesse caso teve valor definido como 10 pontos;
- *Temperatura*: para a temperatura, foi imposta uma variação de 25°C para 50°C em 0,06 s para 0,065 s. Na prática, obviamente variações de temperatura tendem a ter uma dinâmica lenta e provável muito menores do que a aplicada. Entretanto, o objetivo aqui é de forçar o sistema para podermos observar como a variação de temperatura irá afetar na precisão dos resultados;
- *Reset*: Como o próprio nome sugere, essa *flag* tem como função reinicializar os métodos, limpar os vetores e permitir o armazenamento de novas variáveis lidas para próximas análises.

Em sequência, $flag_{2,\mathrm{RLS}}$ é incrementada até que a quantidade de pontos a serem ignorados sejam atingidos. Após isso, o armazenamento da corrente e tensão do capacitor é realizada, até que a $flag_{1,\mathrm{RLS}}$ é setada como 1, o que mostra que os vetores estão prontos para serem enviados ao código RLS e a ESR e capacitância estimadas. Em 0,05 s a temperatura é incrementada até 50°C. Após isso, uma nova medida pode ser requisitada (na prática, esse processo ocorre de forma automática) tornando a flag *Reset* ao nível alto para reinicializar as variáveis, recomeçando o ciclo de armazenagem de dados e estimativa novamente.

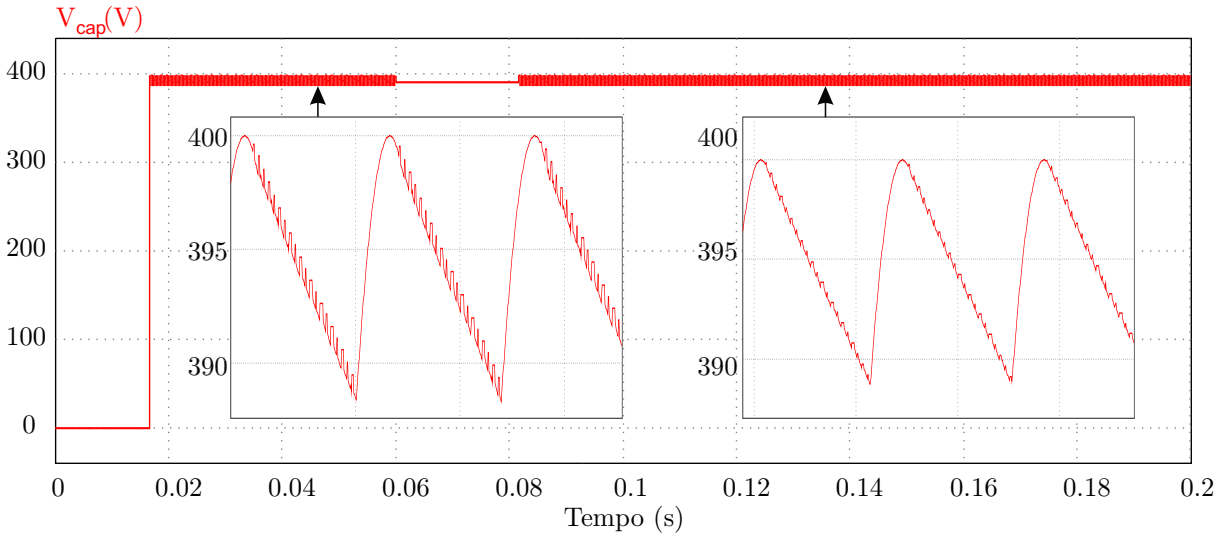
Figura 38 – Comportamento das *flags*, Temperatura e *Reset* respectivamente para o método RLS.



Fonte: do Autor.

As Figuras 39 e 40 mostram a tensão amostrada e a corrente no capacitor estimada respectivamente. Nesse caso, as figuras ilustram as variáveis estimadas na simulação do método RLS para a frequência de 100 kHz, porém a única diferença para o DFT é que a frequência de amostragem é de 92,16 kHz, resultando em estimações de corrente do capacitor muito semelhantes. Para o intervalo onde a temperatura se encontra em 25°C, a ESR tem um valor relativamente maior, o que faz que a ondulação de alta frequência seja maior nesse caso e tenha uma redução quando operando na temperatura de 50°C.

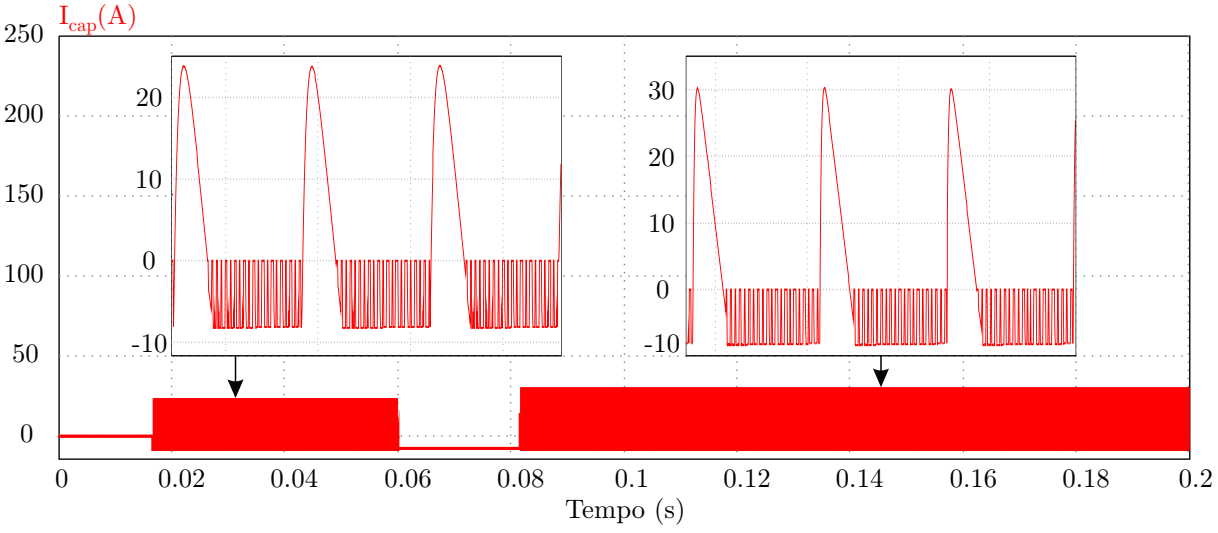
Figura 39 – Tensão no capacitor.



Fonte: do Autor.

Já para a corrente estimada, observa-se um aumento na magnitude de aproximadamente 24 A para 30 A com o aumento da temperatura, já que a ESR tem seu valor reduzido.

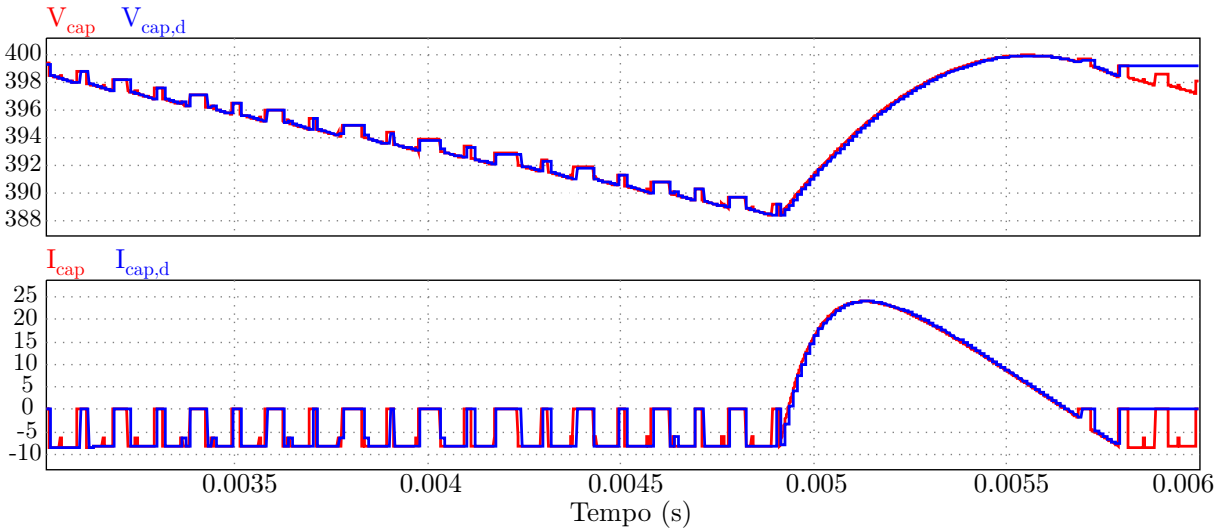
Figura 40 – Corrente no capacitor.



Fonte: do Autor.

A comparação da tensão e corrente no domínio contínuo e discreto são apresentados na Figura 41.

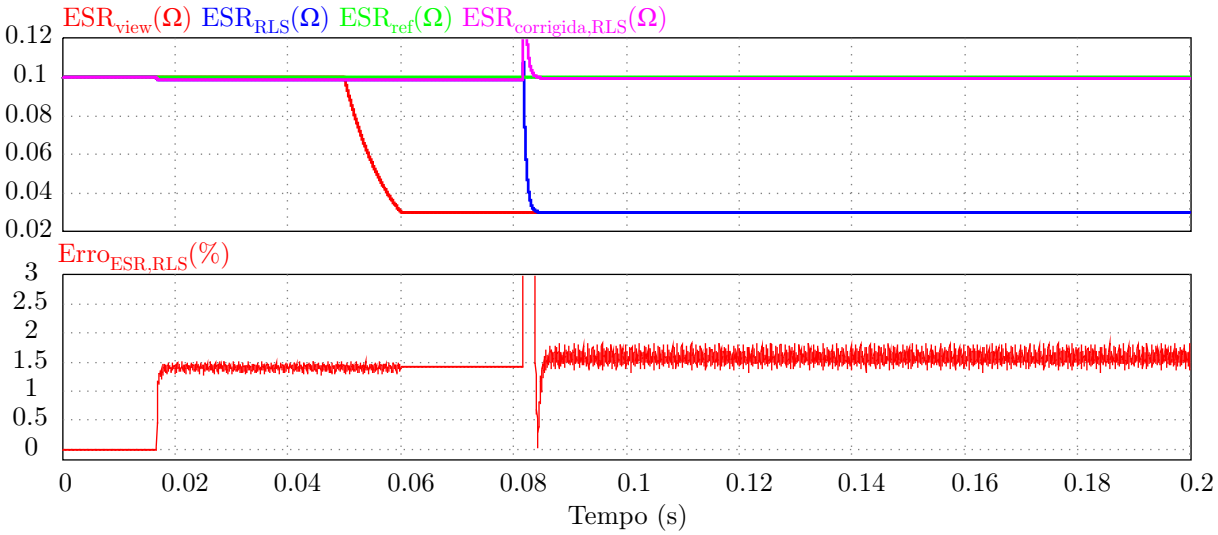
Figura 41 – Tensão e corrente no capacitor contínua e discretizada.



Fonte: do Autor.

Na Figura 42 é finalmente apresentado o comportamento da ESR para o método RLS utilizando um λ de 0,999.

Figura 42 – ESR obtida pelo método RLS.



Fonte: do Autor.

Onde:

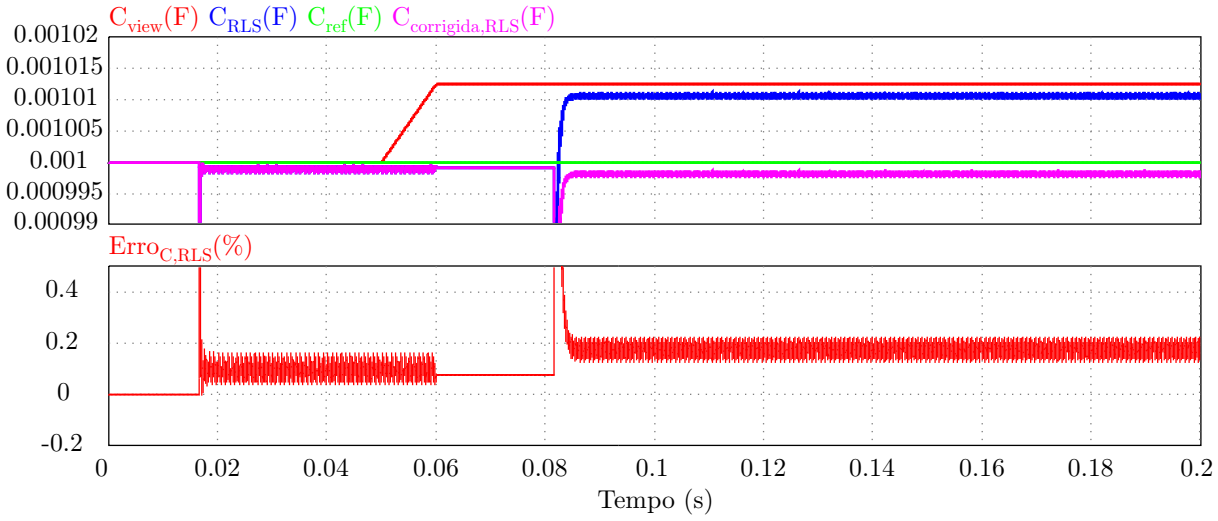
- ESR_{view}: ESR de referência medida diretamente nos componentes no PSIM;
- ESR_{RLS}: ESR calculada pelo método RLS;
- ESR_{ref}: ESR inicial de referência. Nesse caso de 100 mΩ;

- $ESR_{\text{corrigida,RLS}}$: ESR corrigida com base na equação (16);
- $Erro_{\text{ESR,RLS}}(\%)$: erro da RLS calculada (ESR_{RLS}) pela ESR medida ESR_{view}.

De forma geral, o comportamento da ESR está dentro do esperado, com um erro de aproximadamente 1,5% para ambas temperaturas, apesar de apresentar uma variação maior para a temperatura de 50°C, o que sugere que um aumento dinâmico de λ para maiores temperaturas e menores ESR podem apresentar melhores resultados. Ainda, observa-se que a correção da ESR com base na temperatura e a equação estimada mantém o parâmetro quase constante na ESR de referência, desconsiderando apenas o transiente inicial de cálculo.

Já para a capacitância o comportamento tem direção oposta, já que com o aumento da temperatura a capacitância tende a aumentar, como mostra a Figura 43. O erro em regime permanente se encontra próximo de 0,1% para 25°C e 0,2% para 50°C, extremamente aceitável para estimativa de vida útil posterior.

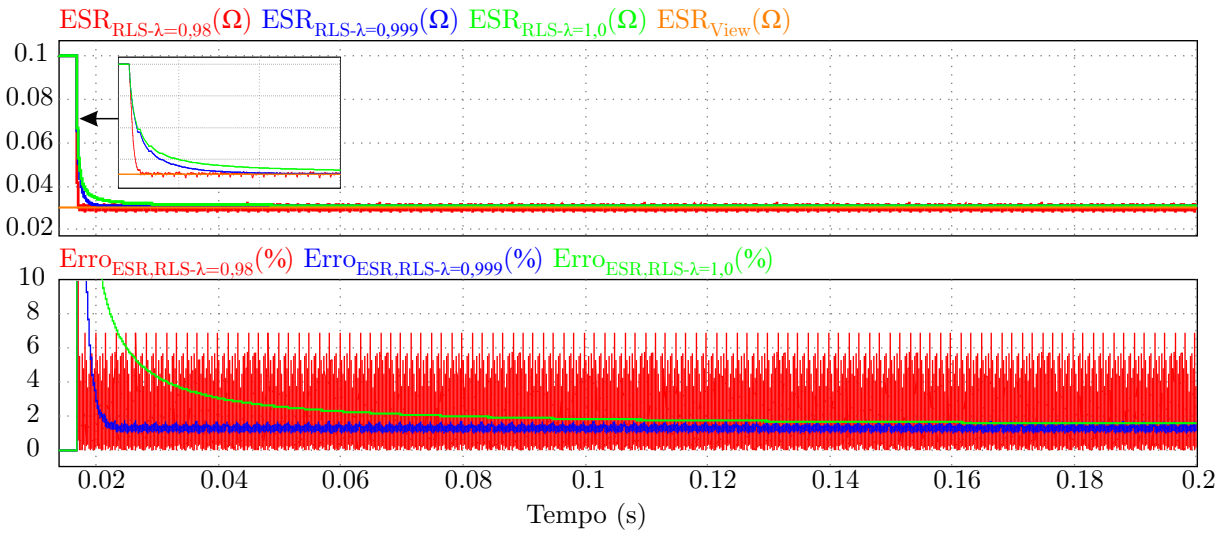
Figura 43 – Capacitância obtida pelo método RLS.



Fonte: do Autor.

Na Figura 44 é apresentada uma comparação do comportamento da ESR calculada com o método RLS para diversos fatores de esquecimento (λ) com a temperatura fixa em 50°C. Conforme estudado nos capítulos anteriores, percebe-se que para uma redução de λ verifica-se uma maior oscilação e, portanto, maior taxa de erro no resultado obtido, porém a recursividade tende a estabilizar rapidamente em seu valor final. Já um valor maior e próximo de 1 ou unitário, tende a ter uma dinâmica lenta.

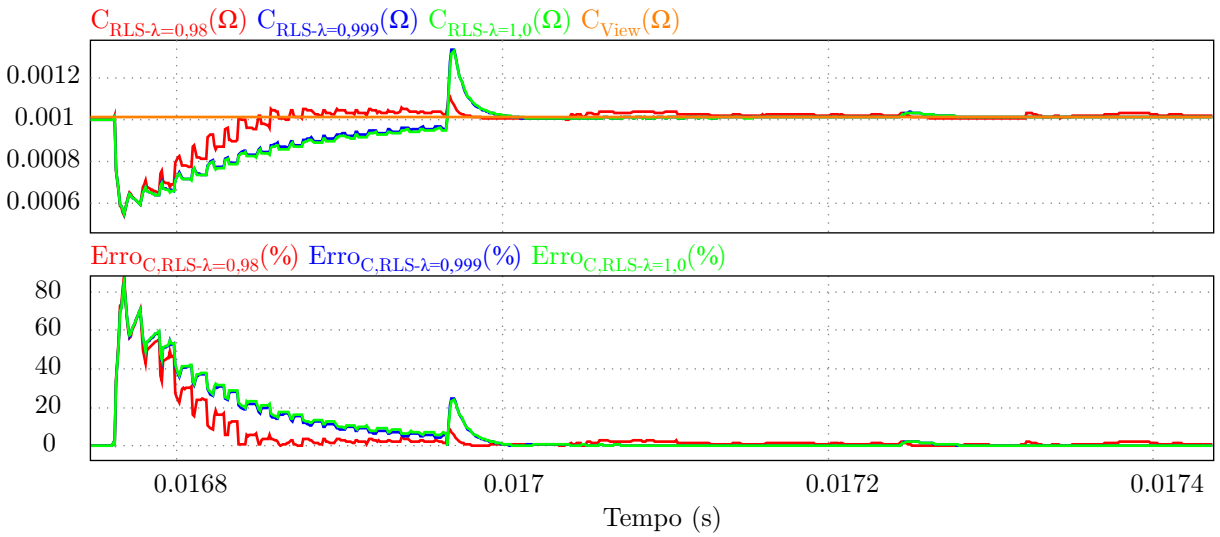
Figura 44 – Comparação do comportamento da ESR utilizando o método RLS com diferentes fatores de esquecimento.



Fonte: do Autor.

Para a capacitância apresentada na Figura 45 percebe-se que a influência de um λ baixo não interfere tanto em oscilações significativas como no cálculo da ESR, além de que um λ unitário apesar de apresentar uma dinâmica extremamente lenta para a ESR, para a capacitância tem um comportamento muito próximo do resultado para o λ de 0,999. Na prática, só é recomendado utilizar um valor abaixo de 1 caso o circuito de condicionamento de sinais e ADC sejam suficientemente precisos para não se ter variações abruptas. Assim, a oscilação em regime permanente pode ser corrigida aplicando um método RMS de janela móvel nos dados obtidos em um *software* externo.

Figura 45 – Comparação do comportamento da Capacitância na estimativa utilizando RLS com vários fatores de esquecimento.



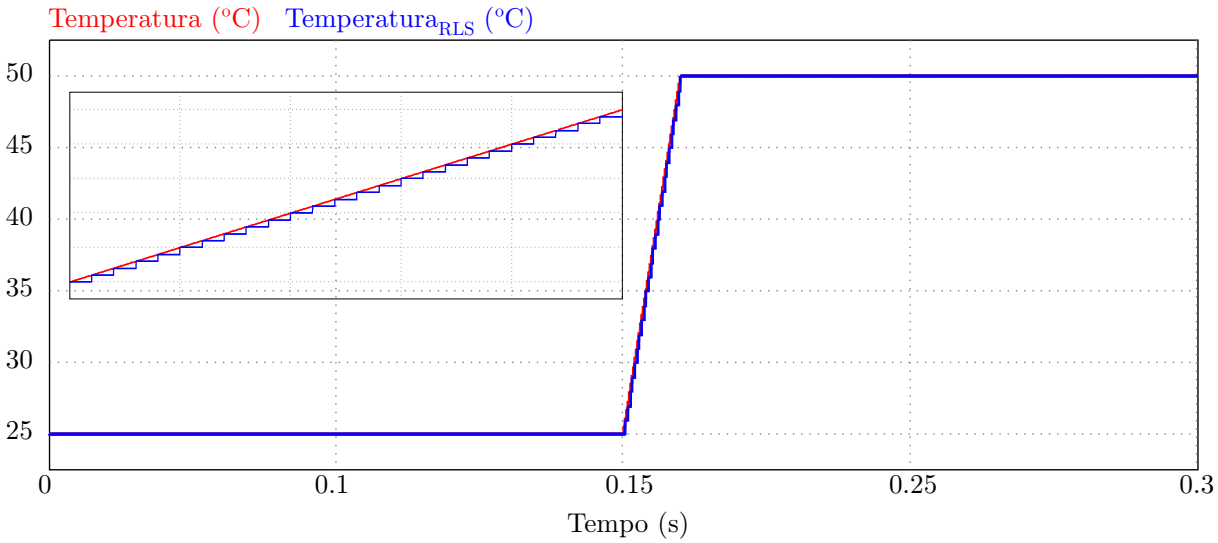
Fonte: do Autor.

3.1.1.2 Modo *online*

Para os resultados *online*, uma lógica semelhante à aquisição de dados foi realizada, onde a temperatura sofre uma variação gradual de 25°C para 50°C. Porém, agora a simulação é realizada em tempo real, o que representaria o algoritmo sendo implementado na interrupção do ADC do microcontrolador.

Um dos desafios principais na implementação desta previsão é a velocidade de processamento dos cálculos, que é afetada pela frequência de amostragem do ADC. Devido a cada iteração ocorrendo com essa frequência, a convergência dos cálculos é mais lenta em comparação com a implementação *offline*. Neste outro caso, os cálculos são realizados na frequência base do DSP, permitindo uma execução mais eficiente na rotina principal, fora da interrupção. Como mostra a Figura 46, a variação da temperatura acontece de 0,15 s para 0,16 s e a simulação dura 0,3 s.

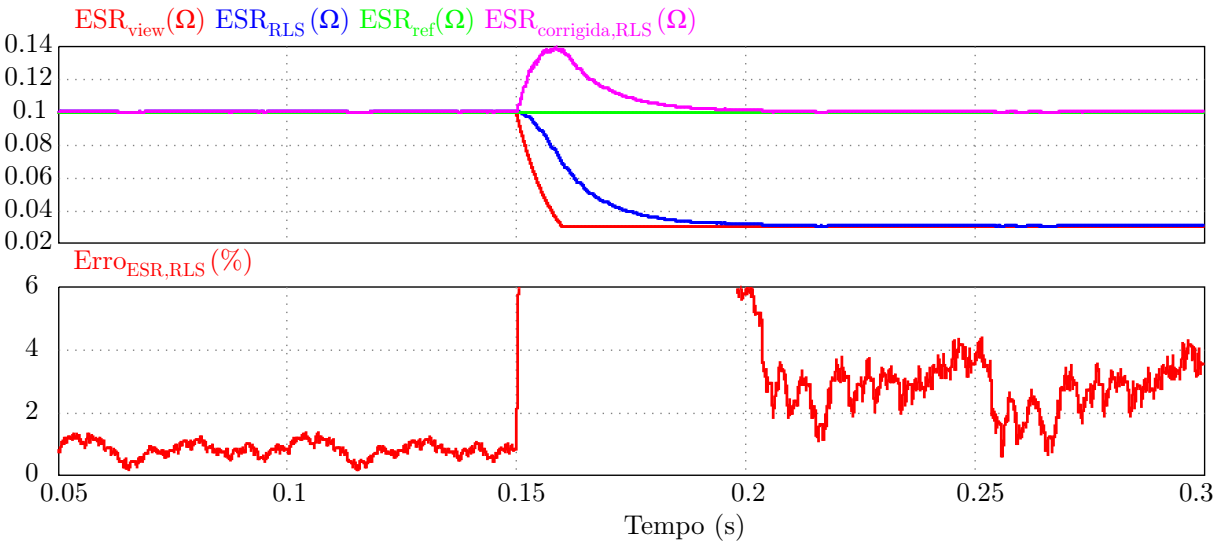
Figura 46 – Comportamento da temperatura real e amostrada em graus Celsius durante a simulação.



Fonte: do Autor.

Na Figura 47 é apresentado o resultado da ESR, onde a média do erro para 25°C e 50°C é de 0,82% e 2,36% respectivamente, em regime permanente.

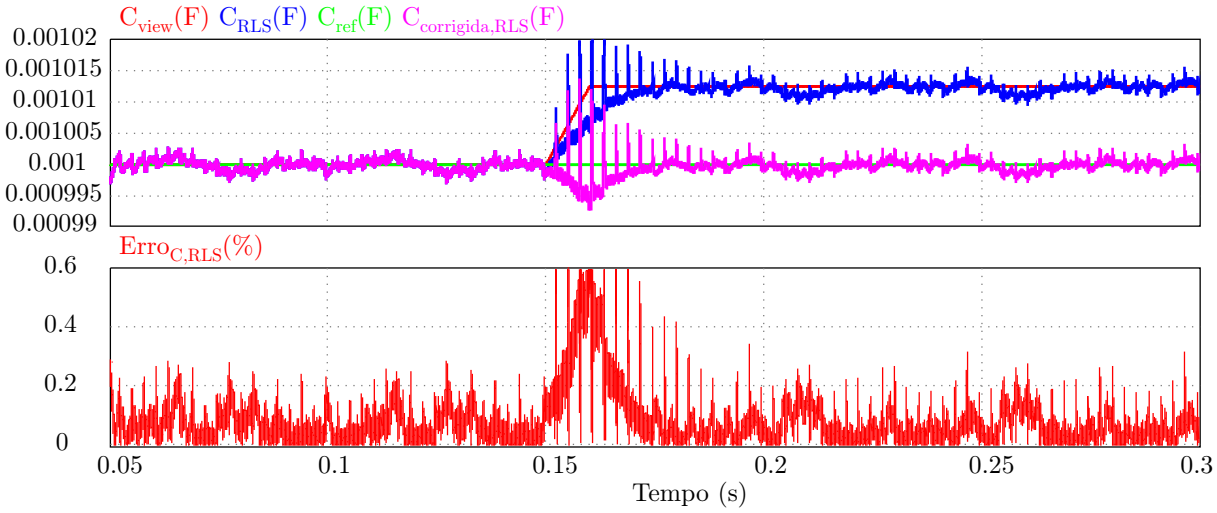
Figura 47 – ESR obtida pelo método RLS no modo *online*.



Fonte: do Autor.

Os resultados para a capacitância são ilustrados na Figura 48, onde a média do erro para regime permanente de 25°C e 50°C é de 0,065% e 0,064% respectivamente.

Figura 48 – C obtida pelo método RLS no modo *online*.



Fonte: do Autor.

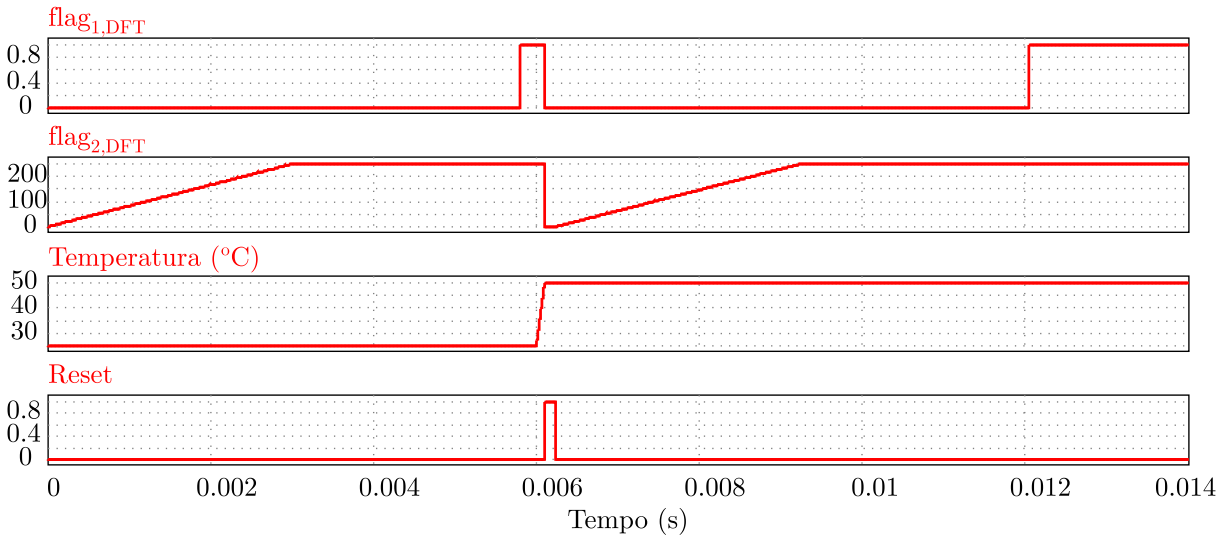
3.1.2 Método *Discrete Fourier Transform* (DFT)

Nessa subseção serão expostos e discutidos os resultados para o método DFT em modo *online* utilizando algoritmo de Goertzel e em modo *offline* com FFT Radix-4.

3.1.2.1 Modo *offline*

O comportamento das *flags*, Temperatura e *Reset* para o método DFT são mostrados na Figura 49 e a sequência segue a mesma lógica do que para o método RLS. A diferença é que para o método DFT não é necessário analisar um grande intervalo de tempo, já que assim que os cálculos são finalizados o resultado é prontamente exposto.

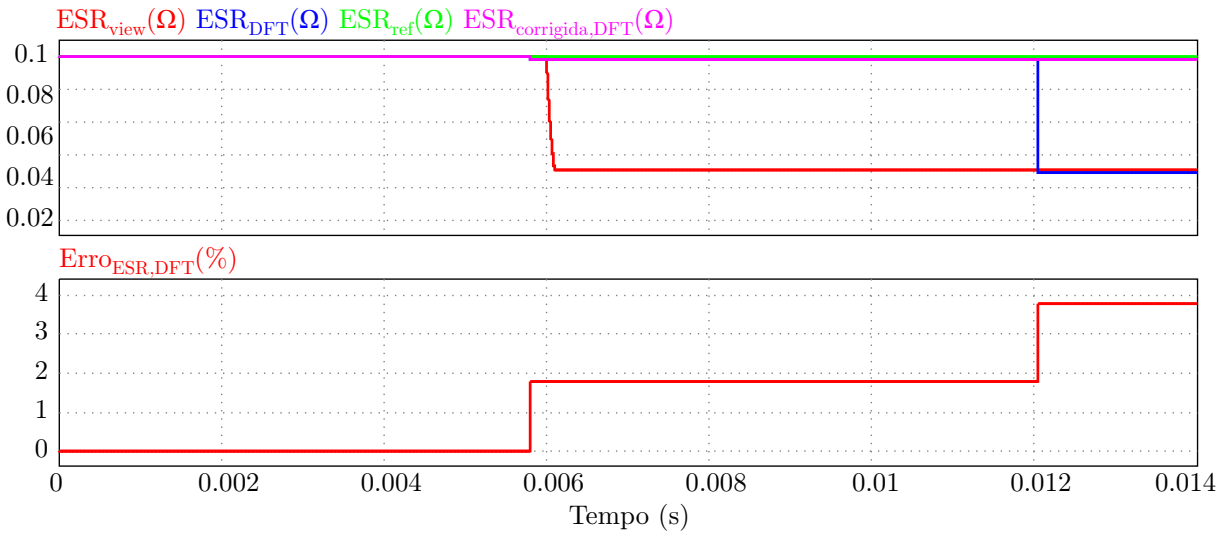
Figura 49 – Comportamento das *flags*, Temperatura e *Reset* respectivamente para o método DFT.



Fonte: do Autor.

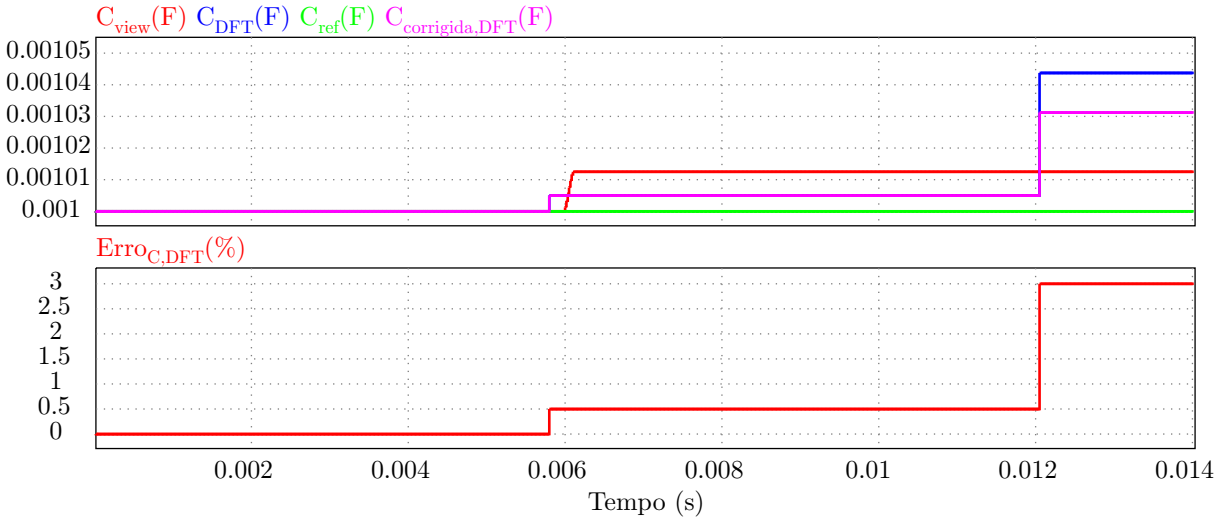
Nesse caso, a simulação para a temperatura de 25°C é finalizada em torno de 5 ms, quando se inicia para 50°C. A taxa de erro para a ESR e capacitância é de aproximadamente 1,9% e 0,5% para 25°C respectivamente e 3,7% e 3% para a temperatura de 50°C. Para ambos os casos o método DFT se mostrou ser aceitável para ser implementado. Para ilustrar o mencionado, as Figuras 50-51 apresentam os resultados para ESR e capacitância respectivamente.

Figura 50 – ESR obtida pelo método DFT.



Fonte: do Autor.

Figura 51 – Capacitância obtida pelo método DFT.

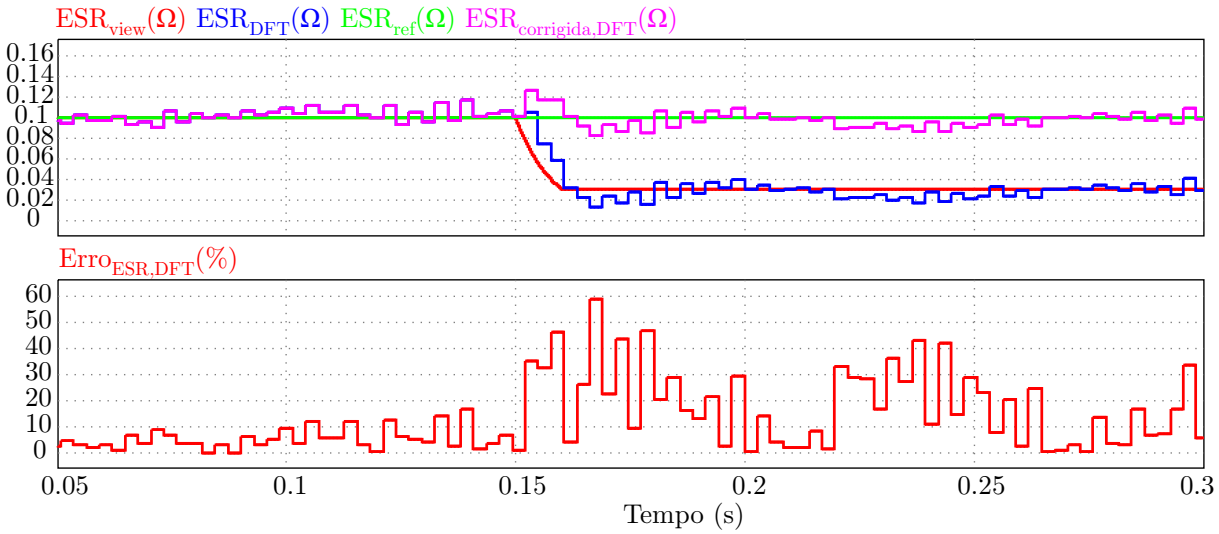


Fonte: do Autor.

3.1.2.2 Modo *online*

Para a simulação do modo *online* da DFT foi utilizado a variação do algoritmo de Goertzel, onde o comportamento da temperatura é o mesmo das simulações para o método RLS *online*. O resultado para a ESR é apresentado na Figura 52, onde se verifica um erro médio de 5,49% para 25°C e 20,55% para 50°C.

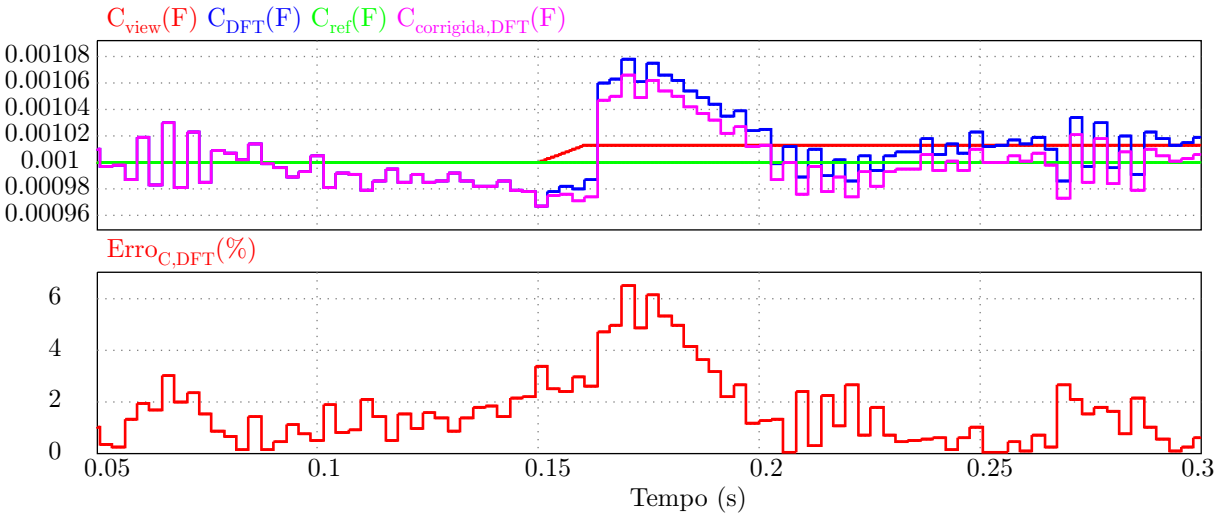
Figura 52 – ESR obtida pelo método DFT *online*.



Fonte: do Autor.

Na Figura 53 é apresentado o resultado para a capacitância, que teve uma média de 1,30% para 25°C e 1,76% para 50°C.

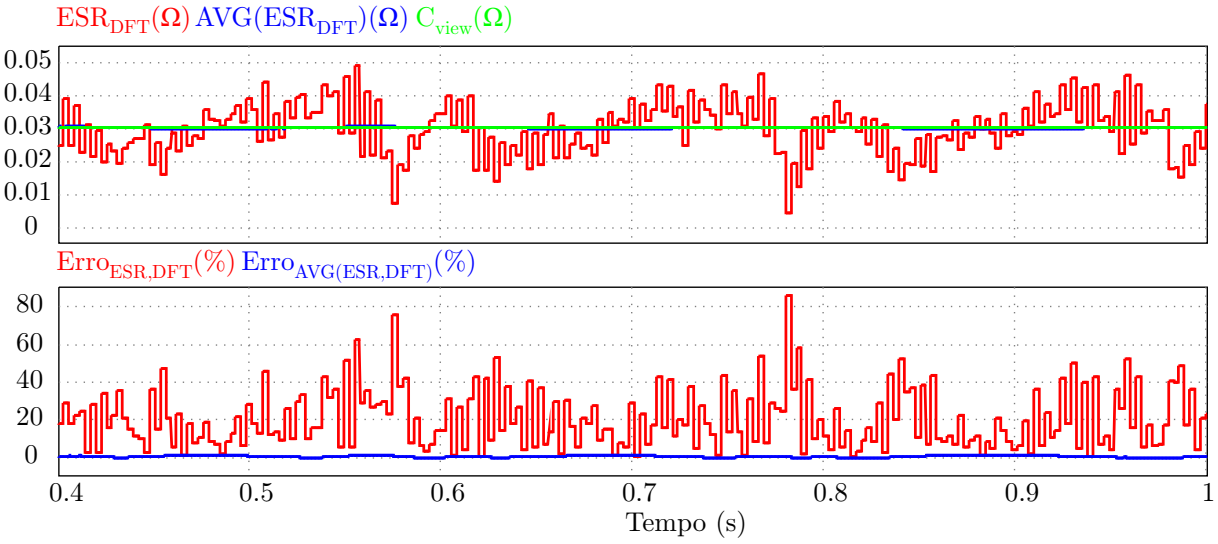
Figura 53 – Capacitância obtida pelo método DFT.



Fonte: do Autor.

Ainda, diferentemente do método RLS, um dos pontos negativos do método DFT é que os resultados variam significativamente dependendo do ponto inicial em que a amostragem se inicia. Para se verificar essa condição, considere a simulação a seguir onde a temperatura é fixada em 50°C. Na Figura 54 é apresentado a ESR variando durante 0,6 s. Nessa situação se percebe que, na prática, é recomendado a utilização de uma forma de tratamento dos dados como o RMS de janela móvel ou média móvel. Sem a utilização do método de tratamento o erro chegou a alcançar aproximadamente 85% próximo de 0,77 s, enquanto que com a média móvel o erro ficou concentrado em torno de 0,65%.

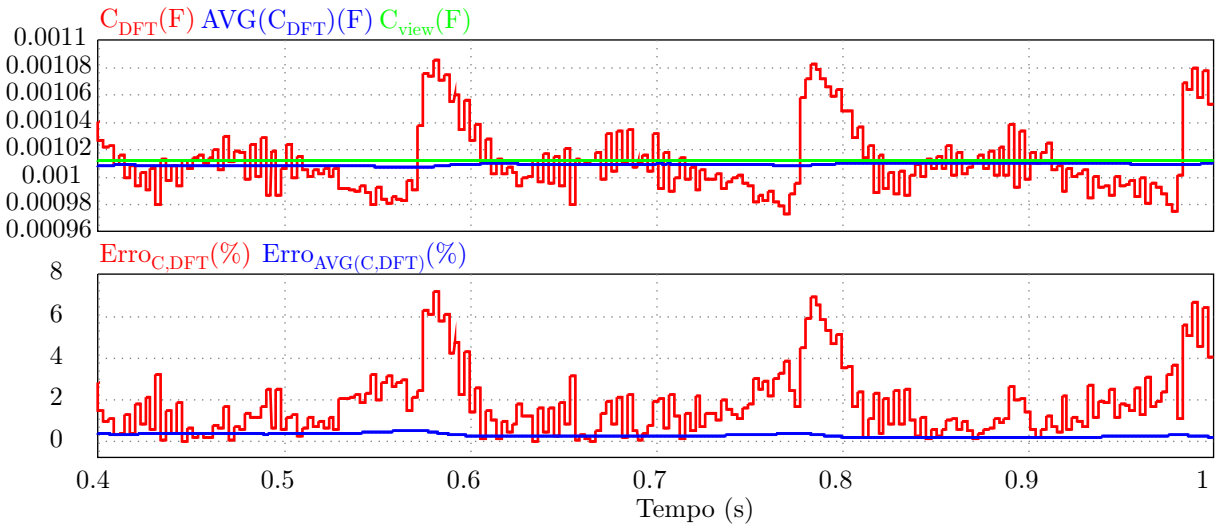
Figura 54 – ESR obtida pelo método DFT de forma recursiva.



Fonte: do Autor.

Já na Figura 55 é apresentado os resultados para a capacitância. Perceba que para o resultado bruto o erro chega a alcançar 7,2% próximo de 0,57 s. Enquanto que com a média móvel o erro ficou concentrado em torno de 0,3%.

Figura 55 – Capacitância obtida pelo método DFT de forma recursiva.



Fonte: do Autor.

3.2 CONSIDERAÇÕES FINAIS

Neste capítulo foram expostos resultados de simulações para ambos os métodos RLS e DFT estudados neste trabalho. Com base no apresentado, conclui-se que de forma geral ambos os métodos obtiveram resultados precisos e com taxas de erro extremamente baixas e confiáveis para se ter uma estimativa de vida útil a longo prazo em capacitores. Entretanto, ressalta-se que o método DFT necessita obrigatoriamente do tratamento dos dados após as estimativas dos parâmetros, visto que depende da posição em que a amostragem e estimativa do sinal tem início. A Tabela 9 mostra um resumo das taxas de erro obtidos.

Tabela 9 – Resumo da comparação dos erros dos métodos RLS e DFT para os modos *online* e *offline* em porcentagem (%).

Método	*Online*		*Offline*	
	ESR (25°C | 50°C)	C (25°C | 50°C)	ESR (25°C | 50°C)	C (25°C | 50°C)
RLS	0,8200 | 2,3600	0,065 | 0,064	1,5000 | 1,5000	0,1000 | 0,2000
DFT	5,4900 | 20,5500	1,300 | 1,7600	1,9000 | 3,7000	0,5000 | 3,0000
$DFT_{mov,avg}$	0,7500 | 0,6500	0,3300 | 0,3000	-	-

Ainda, ressalta-se que nas simulações realizadas essas taxas podem variar a depender do passo de cálculo utilizado no *software*, portanto, os resultados aqui apresentados devem ser considerados subjetivos e aproximados, possuindo um peso maior em demonstrar

o funcionamento do sistema do que resultados práticos comparativos, que serão validados experimentalmente no próximo capítulo.

Além dos resultados obtidos para a simulação principal, diversos tipos de simulações foram realizadas paralelamente para verificar resultados com diferentes tipos de carga conectados a saída. Na Tabela 10 é apresentado as taxas de erros para carga R, carga R com filtro LC ($L_f = 5$ mH, $C_f = 60$ nF) e um motor de indução em operação nominal para a temperatura de 25°C, cujos parâmetros são mostrados na Tabela 11.

Tabela 10 – Respostas de simulação para diferentes tipos de cargas.

	Erro (ESR) %		Erro (C) %	
Carga	RLS	DFT_{AVG}	RLS	DFT_{AVG}
R	1,50	0,61	0,10	0,60
R com filtro LC	1,45	0,85	0,03	1,10
Motor de Indução (nominal)	0,43	1,15	0,34	0,17

Fonte: do Autor.

Tabela 11 – Parâmetros do motor de indução simulado em regime permanente.

Resistência do estator	294 mΩ	Resistência do Rotor	156 mΩ
Indutância do estator	1,39 mH	Indutância do rotor	740 mH
Indutância magnetizante	41 mH	Número de polos	6

4 PROTÓTIPO IMPLEMENTADO E RESULTADOS EXPERIMENTAIS

4.1 INTRODUÇÃO

A estrutura do protótipo montado exige uma série de cuidados e componentes relativos aos utilizados na simulação na ferramenta PSIM realizado no capítulo anterior. Entre eles, transformador, módulo retificador, inversor, drivers de acionamento dos interruptores, amplificadores operacionais, sensores de corrente e tensão e circuitos de proteção para os dispositivos LAUNCHXL-F28069M (ou apenas referido por F28069M) e ESP32.

O dispositivo F28069M é um kit de desenvolvimento vendido pela *Texas Instrument* que conta com o microcontrolador TMS320F28069M. Suas principais características são (INSTRUMENTS, 2011):

- CPU C28x de 90 MHz;
- 256 kB de Flash;
- 100 kB de RAM;
- 16 ADC de 12 bits;
- 16 canais PWM;
- 3 *timers* de 32 bits;
- Tensão de alimentação de 3,3 V;
- Sensor de temperatura;
- Conexão USB 2.0;
- Ponto flutuante;
- Programável em C, C++ e Assemly;

Já o ESP32 é um microcontrolador de baixo custo e alta performance desenvolvido pela *Espressif Systems*. Algumas de suas características incluem (ESPRESSIF, 2022):

- CPU dual-core de 160 MHz a 240 MHz;
- 4 MB a 16 MB de Flash;
- 520 kB a 520 kB de RAM;
- 2 canais ADC de 12 bits;
- 16 canais PWM;
- 4 *timers* de 32 bits;
- Tensão de alimentação de 2,5 V a 3,6 V;
- Wi-Fi e Bluetooth integrados;
- Conexão USB 2.0 ou conexão serial;
- Ponto flutuante;

- Programável em C, C++ e MicroPython.

Após a definição da estrutura do protótipo, os resultados experimentais são analisados e discutidos.

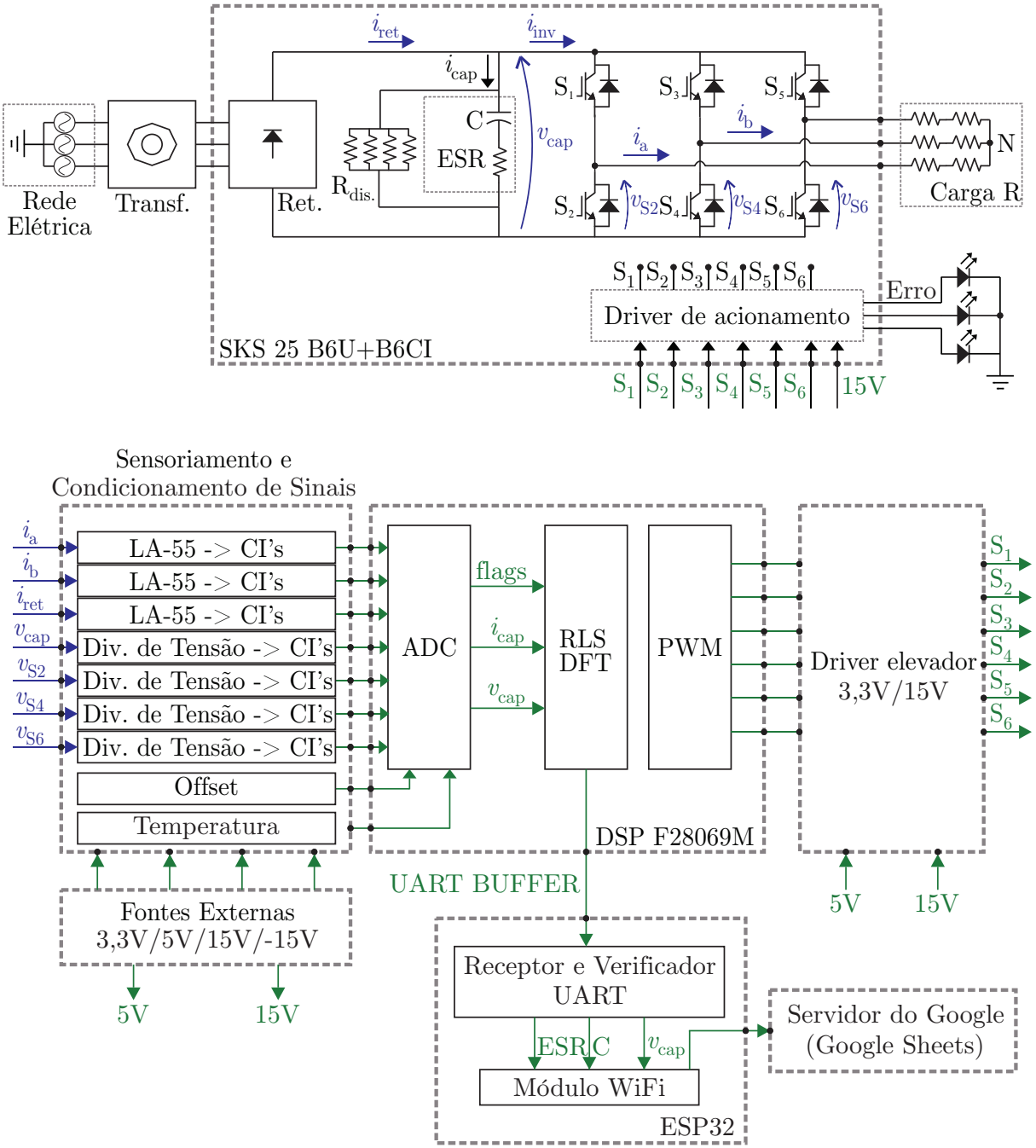
4.2 ESTRUTURA E FUNCIONAMENTO DO PROTÓTIPO

Para realizar a experimentação e montagem do protótipo foi utilizado o módulo inversor industrial e comercial SKS 25 B6U+B6CI vendido pela *Semikron*, do qual já possuí um retificador trifásico não controlado na entrada do barramento. Além disso, o *driver* de acionamento dos semicondutores embutido do modelo SKHI 23/12 R (também fabricado pela *Semikron*) conta com tempo morto (necessitando apenas dos pulsos de modulação na tensão adequada). A Figura 56 mostra um esquemático geral de como a estrutura foi montada no laboratório.

Inicialmente, o conversor é acionado através do módulo PWM com uma modulação SPWM como discutido nos capítulos anteriores. Um transformador variável foi conectado à rede elétrica para se ter controle manual da magnitude da tensão injetada no sistema, um ponto que vale ressaltar é que esse transformador acaba por adicionar uma indutância de entrada, distorcendo a forma de onda de corrente de saída do retificador. Em termos de resultados, a distorção da forma de onda não altera a precisão dos métodos significativamente. Após a certificação de que o sistema opera normalmente e após a elevação gradual da tensão do barramento para 400 V, o dispositivo ESP32 é finalmente conectado, fazendo uma *flag* interna ser enviada ao F28069M como uma forma de inicializar o processamento de dados. Após isso, com os sensoriamento e condicionamento dos sinais funcionando adequadamente, as variáveis são lidas pelo ADC do DSP e um *offset* de 1,65 V é adicionado para possibilitar a leitura das correntes alternadas. As leituras são então processadas e a corrente do capacitor é estimada, acionando os códigos RLS/DFT para se obter a ESR e capacitância do capacitor conectado no barramento do sistema. Nota-se também que o módulo utilizado possui um banco de resistores de 22 kΩ em paralelo ao capacitor para descarregá-los rapidamente em caso de interrupção da alimentação elétrica. Durante a operação nominal, não há corrente circulando nesses resistores, tendo sua função de suprimir transições rápidas de tensão no barramento.

Os parâmetros de ESR e C, juntamente da tensão de barramento medida no momento de análise, são enviados por um *buffer* de 8 bits através da comunicação UART do DSP para o ESP32, que faz uma tradução desses dados e envia por conexão WiFi para o Google Sheets, esse ciclo é repetido por um intervalo de tempo definido até que a quantidade de amostras desejadas sejam obtidas. Após isso, os dados são extraídos em CSV manualmente e processados no software MATLAB para uma análise mais profunda.

Figura 56 – Esquemático do protótipo montado.

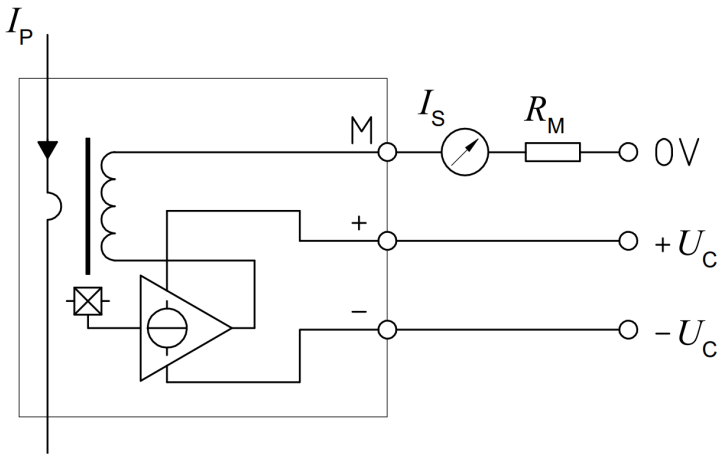


Fonte: do Autor.

4.3 PLACA DE CONDICIONAMENTO DE SINAIS E PULSOS DE COMANDO

O sensoriamento das correntes necessárias são realizadas através do sensor de corrente LA 55-P. Esse sensor possuí uma relação entre corrente lida e de saída (I_s) de 1:1000. As tensões de alimentação são de +15 V e -15 V e o nível máximo de corrente é de 50 A. A Figura 57 mostra uma representação do sensor utilizado.

Figura 57 – Sensor LA 55-P.



Fonte: (LEM, 2021)

Na prática, foram utilizadas duas espiras para cada sensor com 100 Ω no resistor R_m. Considerando uma corrente máxima de 8,9 A e 13 A para as correntes de fase e do retificador no ponto de operação em 400 V, tem-se tensões V_{RM} que serão enviadas aos CI's de condicionamento de:

$$V_{\mathrm{RM,Ifase}} = \frac{R_m n_s I_{\mathrm{ab,max}}}{1000} = 1{,}78\,V \tag{171}$$

$$V_{\mathrm{RM,Iret}} = \frac{R_m n_s I_{\mathrm{ret,max}}}{1000} = 2{,}6\,V \tag{172}$$

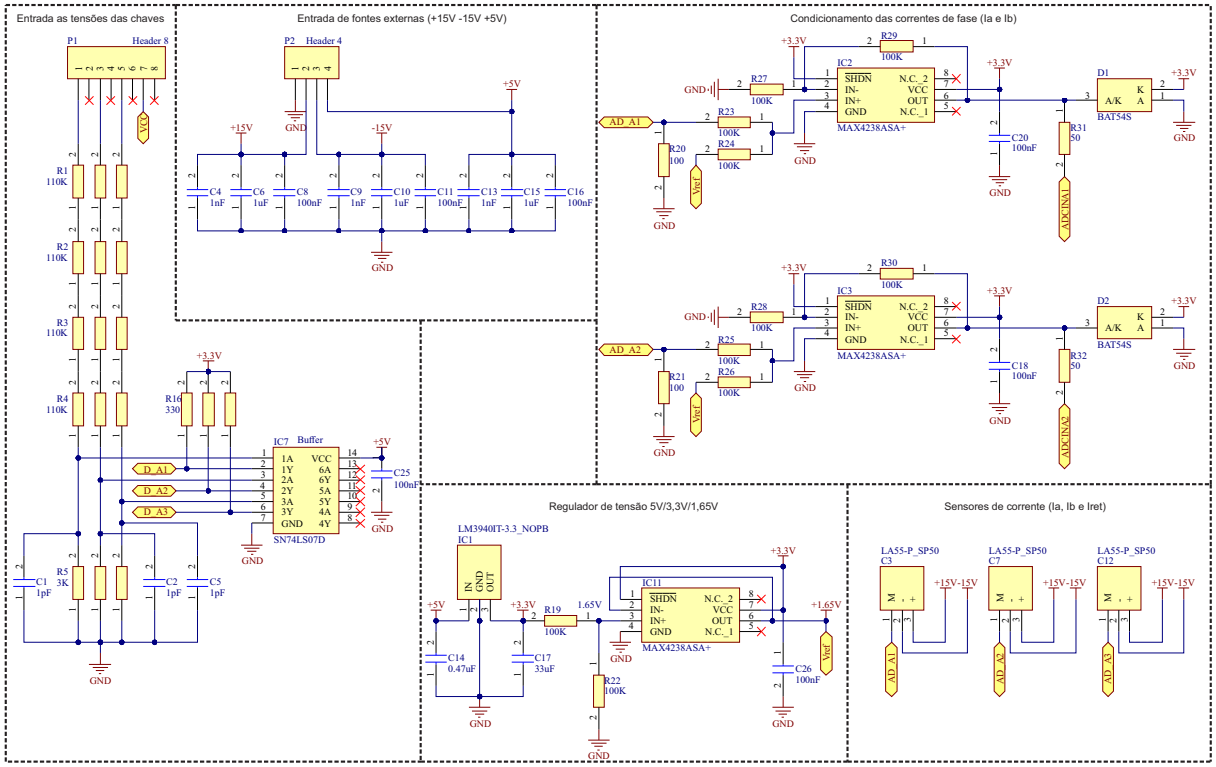
O condicionamento dos sinais são realizados com o amplificador operacional (amp-op) MAX4238. Para os sinais de corrente de fase, inicialmente após a leitura pelo sensor e o resistor R_m, o sinal passa por um divisor de tensão de ambos 100 kΩ, enquanto na outra ponta do divisor um *offset* de 1,65 V é adicionado para tornar a corrente medida positiva e possibilitar a amostragem pelo DSP. Esse *offset* é obtido através de um divisor de tensão advinda de um regulador que reduz a tensão da fonte de 5 V para 3,3 V. O sinal também é aplicado a um seguidor de tensão com um amp-op para que outras partes do circuito não influenciem na tensão de 1,65 V e a mesma se mantenha isolada. Depois disso, o sinal de tensão referente a corrente medida entra no amp-op com uma configuração não inversor com ganho 2, corrigindo a amplitude do sinal novamente. Por fim, o sinal é enviado ao ADC do DSP conectado no ponto central de dois diodos em série do modelo BAT54S para proteger o dispositivo de tensões maiores que 3,3 V. O sinal de corrente do retificador não necessita de *offset* pois por ser retificada será sempre maior que 0 A.

Para os sinais de tensão dos interruptores é adicionado um divisor de tensão de 440 kΩ e 3 kΩ. Os sinais são enviados ao circuito integrado SN74LS07 que é um *driver* não-inversor de coletor aberto. Esse dispositivo irá transformar as oscilações presentes

nos níveis altos dos sinais, em sinal constante. Na prática, foi percebido que além de um atraso no sinal, ainda adiciona um pequeno *offset* no sinal de saída, portanto, os sinais poderiam ser conectados diretamente aos amp-op's (apenas por segurança de isolamento) e ao DSP, pois a entrada digital do mesmo irá automaticamente converter o sinal em 0 ou 1. O sinal de tensão do barramento é medida com o mesmo divisor de tensão e enviada diretamente ao seguidor de tensão com o amp-op e enviado ao ADC do DSP. As Figuras 58-59 mostram os esquemáticos com os circuitos mencionados.

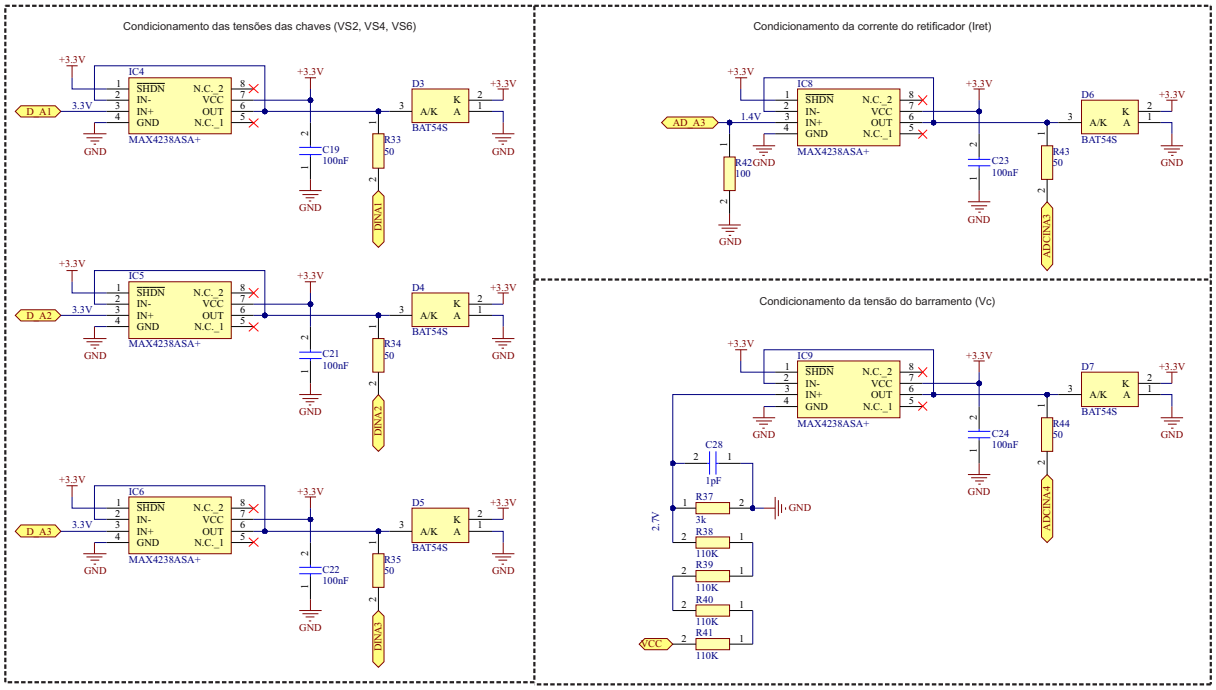
Já na Figura 60 observa-se na esquerda os dispositivos ESP32 com os pinos RX e TX da comunicação UART conectados ao DSP F28069M. Enquanto na direita verifica-se o sensor de temperatura externo e os pulsos de comando do PWM do inversor. Para o acionamento dos interruptores, o mesmo *driver* SN74LS07 é utilizado, mas nesse caso para elevar a tensão chaveada de 3,3 V para 15 V por uma fonte externa.

Figura 58 – Esquemático do circuito de condicionamento do protótipo montado em bancada: parte 1.



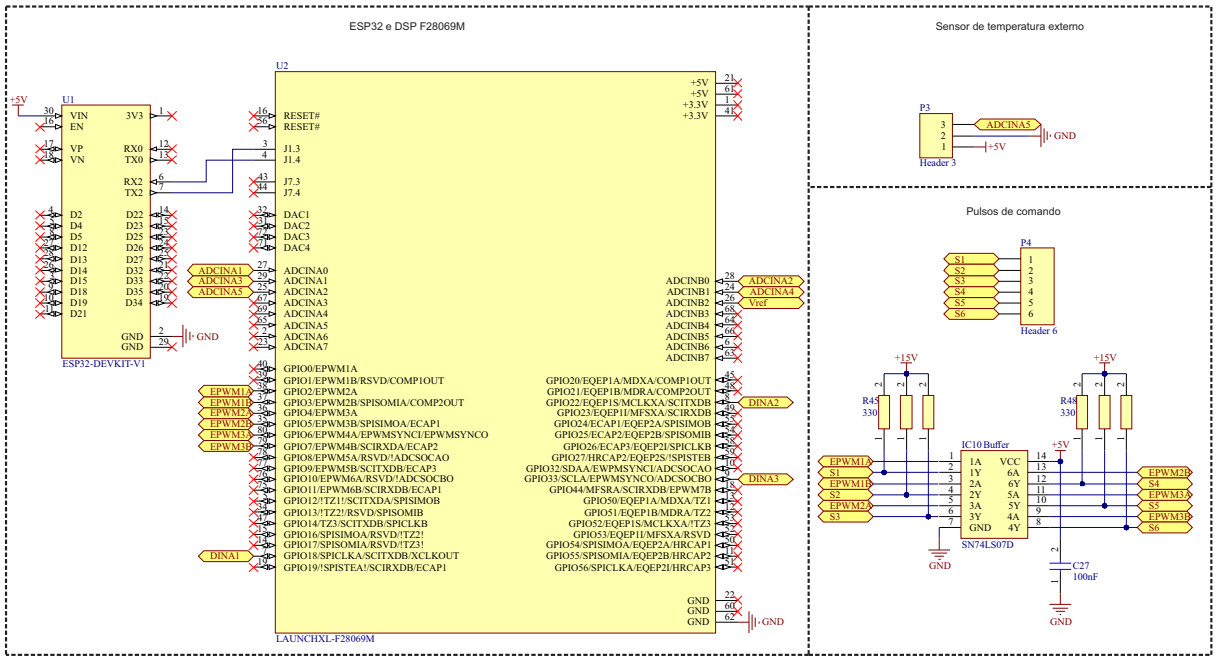
Fonte: do Autor.

Figura 59 – Esquemático do circuito de condicionamento do protótipo montado em bancada: parte 2.



Fonte: do Autor.

Figura 60 – Esquemático do circuito de condicionamento do protótipo montado em bancada: parte 3.



Fonte: do Autor.

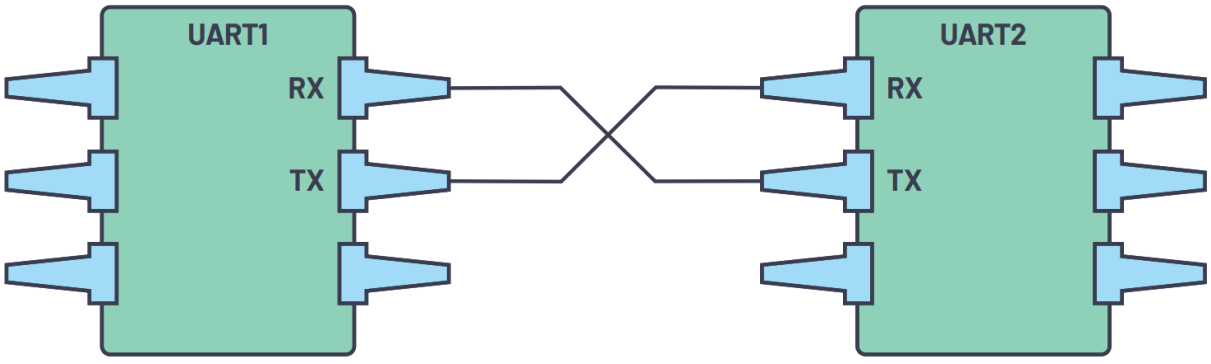
4.4 TRANSMISSÃO DE DADOS ENTRE OS DISPOSITIVOS (F28069M - ESP32 - GOOGLE SHEETS)

Por conta da dificuldade de encontrar artigos que demonstrem o passo a passo para se elaborar essa comunicação entre os três dispositivos/*server*, nessa seção será apresentado em detalhes o funcionamento e descrição do mesmo. Inicialmente a comunicação acontece entre os dispositivos F28069M e ESP32 que utiliza a comunicação UART para enviar os dados de ESR e capacitância, salvando os dados em um *buffer* binário de 8 bits no dispositivo F28069M e enviando através de dois pinos específicos (RX-TX). Assim, o ESP32 recebe e converte esse *buffer* para os valores de referência. Após isso, o dispositivo se encarrega de enviar os dados em novos *buffer's* via WiFi para o Google Sheets, do qual utiliza de um *script* na linguagem JavaScript para organizar e salvar os dados em uma planilha previamente salva na conta do usuário no Google Sheets.

4.4.1 Comunicação via *Universal Asynchronous Receiver and Transmitter* (UART)

Neste trabalho foi escolhida a comunicação via *Universal Asynchronous Receiver and Transmitter* (UART) que em tradução livre significa "Receptor e Transmissor Assíncrono Universal", que basicamente é um protocolo de comunicação serial utilizada em circuitos digitais. Essa comunicação é uma das mais usadas quando se trata de envio e recepção de dados em distâncias curtas, baixa velocidade e custo mínimo (GUPTA et al., 2020). A Figura 61 mostra como deve ser feita uma comunicação UART bidirecional entre dois dispositivos.

Figura 61 – Conexões de pinos na comunicação UART.

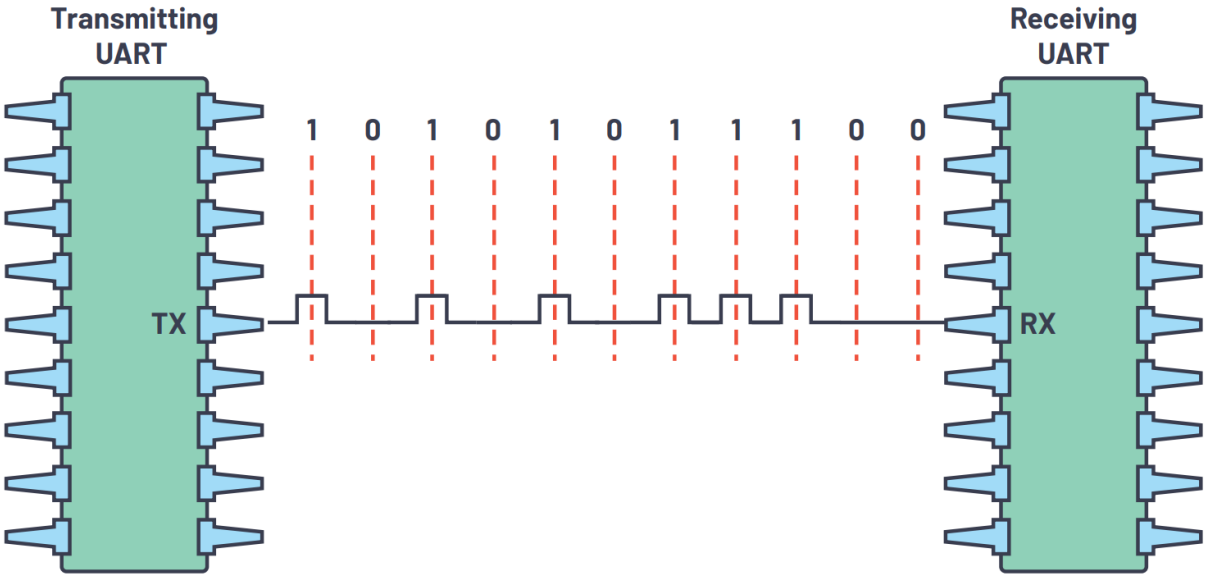


Fonte: Peña e Legaspi (2020)

A transmissão de dados via UART funciona com o envio de bits sequenciais em uma determinada taxa de transmissão (*Baud Rate*) entre um dispositivo e outro, onde o transmissor é programado para converter os dados a serem enviados por uma sequência de bits passível de ser interpretada no receptor, onde apenas um pino é exclusivamente

dedicado para transmissão e outro para recepção. As taxas de transmissão variam entre 9600, 19200, 38400, 57600, 115200, 230400, 460800, 921600, 1000000 e 1500000. A Figura 62 mostra como esse pacote de bits é enviado de um dispositivo a outro.

Figura 62 – Bits enviados do transmissor para o receptor.



Fonte: Peña e Legaspi (2020)

4.4.1.1 F28069M

O dispositivo F28069M possuí internamente um módulo dedicado a UART, onde suporta comunicação digital entre CPU e dispositivos externos assíncronos. Os módulos UART suportam comunicações digitais entre a CPU e outros periféricos assíncronos que usam o formato padrão sem retorno a zero (*non-return-to-zero* - NRZ). O receptor e o transmissor UART têm um FIFO (*first in first out* - os dados gravados no *buffer* na primeira posição são os primeiros a sair) de 4 níveis e cada um tem seus próprios bits de habilitação e interrupção separados. Ambos podem ser operados independentemente para comunicação *half-duplex* (envio e recepção de dados por vez) ou simultaneamente para comunicação *full-duplex* (comunicação bidirecional simultânea). Além disso, para especificar a integridade dos dados, o módulo verifica os dados recebidos para detecção de quebra, paridade, saturação e erros. A taxa de bits é programável para diferentes velocidades através de um registrador de seleção de transmissão de 16 bits (INSTRUMENTS, 2011).

Os recursos do módulo SCI incluem (INSTRUMENTS, 2011):

- Dois pinos externos (ambos os pinos podem ser usados como GPIO se não forem usados para UART):

  SCITXD: pino de saída de transmissão UART;

SCIRXD: pino de entrada de recepção UART;

- Taxa de transmissão programável para 64 mil taxas diferentes;
- Formato "*Data-word* (byte)"

  Um bit de início

  Comprimento da palavra de dados programável de um a oito bits

  Bit opcional par/ímpar/sem paridade

  Um ou dois bits de parada

  Um bit extra para distinguir endereços de dados (somente modo de bit de endereço)
- Quatro sinalizadores de detecção de erros: paridade, saturação, enquadramento e detecção de interrupção;
- Dois modos de multiprocessador de ativação: linha ociosa e bit de endereço;
- Operação *half* ou *full-duplex*;
- Funções de recepção e transmissão com *buffer* duplo;
- As operações do transmissor e do receptor podem ser realizadas por meio de algoritmos controlados por interrupção ou pesquisados com sinalizadores de status;
- Bits de habilitação separados para interrupção de transmissor e receptor (exceto BRKDT - uma das *flags* de interrupção do módulo UART)
- Formato NRZ (sem retorno a zero).

A forma de envio utilizando os registradores do módulo UART é realizada por um *buffer* de 8 bytes, sendo cada posição do vetor composta por 8 bits. Cada número (ou dado enviado) ocupa duas posições do vetor, onde a primeira posição é referente aos bits mais significativos para representação do sinal, e a segunda para os menos significativos respectivamente. Nesse caso os valores utilizados sempre terão magnitudes positivas, porém, esse método mais completo foi utilizado para comunicação pensando em possíveis alterações no código futuramente.

Inicialmente, o valor é convertido para uma amplitude aceitável que permita a visualização das casas decimais em variável inteira. Após, o mesmo é comparado com uma lógica AND e sofre deslocamento para a direita. Como são utilizados 2 bytes para cada número, cada variável enviada pode ter amplitude de -32767 até 32767. Por exemplo, considerando que nas primeiras duas posições serão enviadas o valor de ESR de 100,54 mΩ e que o fator de conversão $conv_{\text{factor}}$ seja de 100000, a equação geral da primeira posição utilizada no DSP tem forma de:

$$buffer_{\text{tx}}[0] = ((int)(ESR_{\text{tx}} conv_{\text{factor}}) \& 0xFF00) >> 8 \tag{173}$$

Substituindo os valores, tem-se que:

$$buffer_{\text{tx}}[0] = ((int)(100{,}54m \cdot 100k)\&0xFF00) >> 8 \tag{174}$$

Ou seja:

$$buffer_{\text{tx}}[0] = (10054_{DEC}\&0xFF00) >> 8 \tag{175}$$

O número 10054 tem forma binária de '0010011101000110' ou '2746' em hexadecimal. Realizando a lógica AND:

$$buffer_{\text{tx}}[0] = 2700_{HEX} >> 8 \tag{176}$$

Então, deslocando os bits (ou dividindo por 256_{DEC}):

$$buffer_{\text{tx}}[0] = 27_{HEX} = 39_{DEC} \tag{177}$$

Para a segunda posição, o valor é apenas aplicado à lógica AND com um valor de 0xFF (ou 255 decimal).

$$buffer_{\text{tx}}[1] = (int)(ESR_{\text{tx}}conv_{\text{factor}})\&0x00FF \tag{178}$$

O resultado é expresso como:

$$buffer_{\text{tx}}[1] = 46_{HEX} = 70_{DEC} \tag{179}$$

Os códigos completos para funcionamento do módulo UART no DSP podem ser encontrados nos apêndices desse trabalho.

4.4.1.2 ESP32

O dispositivo ESP32 possui três controladores UART, cada um com um conjunto idêntico de registradores para simplificar a programação e aumentar a flexibilidade. Cada controlador UART é configurável independentemente com parâmetros como taxa de transmissão, comprimento de bits de dados, ordenação de bits, número de bits de parada, bit de paridade, etc. Todos os controladores são compatíveis com dispositivos habilitados para UART de vários fabricantes e também podem suportar protocolos de associação de dados infravermelhos (*Infrared Data Association* - IrDA), realizando tranquilamente uma troca de dados *full-duplex* entre dispositivos (ESPRESSIF, 2022).

Os recursos do módulo da comunicação UART do ESP32 incluem (ESPRESSIF, 2022):

- Taxa de transmissão programável (até 115200);
- 1024 × 8 bits de RAM compartilhada por três UART transmitir-FIFOs e receber-FIFOs;

- Suporta autoverificação da taxa de transmissão de entrada;
- Suporta 5/6/7/8 bits de comprimento de dados;
- Suporta 1/1,5/2/3 bits de PARADA;
- Suporta bit de paridade;
- Suporta protocolo RS485;
- Suporta protocolo IrDA;
- Suporta DMA para comunicar dados em alta velocidade;
- Suporta despertar UART;
- Suporta controle de fluxo de *software* e *hardware*.

Para o ESP32, o processo de recepção é inverso ao de transmissão. Ou seja, a posição 0 do *buffer* é multiplicada por 256, somada com a posição 1, e o resultado é dividido pelo fator de conversão. Resgatando o exemplo anterior, a equação geral tem forma de:

$$ESR_{\text{rx}} = \frac{buffer_{\text{rx}}[0] \cdot 256 + buffer_{\text{rx}}[1]}{conv_{\text{factor}}} \tag{180}$$

Substituindo os valores, tem-se que:

$$ESR_{\text{rx}} = \frac{39 \cdot 256 + 70}{100000} = 100{,}54\,\text{m}\Omega \tag{181}$$

4.4.2 Envio e recepção de dados no Google Sheets

Nesta subseção será tratado do envio de dados para o Google Sheets através do módulo WiFi do ESP32 e também da recepção de dados e organização dos mesmos utilizando a linguagem JavaScript.

4.4.2.1 Envio de dados via WiFi no ESP32

O dispositivo ESP32 foi escolhido para ser o intermédio entre a transmissão de dados para a tabela no Google Sheets justamente por conta do módulo WiFi que é embutido de fábrica no modelo padrão do mesmo. Além disso, a facilidade de conexão e programação é o seu ponto mais forte, já que conta com uma vasta gama de bibliotecas prontas. O módulo WiFi do ESP32 tem as seguintes características (ESPRESSIF, 2022):

- Modo de estação (também conhecido como modo STA ou modo de cliente Wi-Fi). O ESP32 se conecta a um ponto de acesso.
- Modo AP (também conhecido como modo *Soft-AP* ou modo *Access Point*). As estações se conectam ao ESP32.
- Modo de coexistência estação/AP (ESP32 é simultaneamente um ponto de acesso e uma estação conectada a outro ponto de acesso).

- Vários modos de segurança para o acima (WPA, WPA2, WEP, etc.)
- Varredura de pontos de acesso (varredura ativa e passiva).
- Modo promíscuo para monitoramento de pacotes Wi-Fi IEEE802.11.

No processo de enviar dados através do módulo WiFi também é necessário tratar com vetores. Em uma visão macroscópica do algoritmo, os dados são enviados em pacotes a cada 10 variáveis salvas, ou seja, após a recepção dessa quantidade pelo módulo UART, elas são salvas em vetores específicos e transformadas em *strings* para compor uma parte do endereço *web* que será enviado. O endereço URL padrão de um *script* do Google Sheets tem forma de:

```
https://script.google.com/macros/s/" + GOOGLE_SCRIPT_ID + "/exec?" + "
    value1=" + var1 + "&value2=" + var2 + "&value3=" + var3;
```

Lista 4.1 – Endereço padrão de *script* do Google Sheets.

Onde o *GOOGLE_SCRIPT_ID* tem forma de 72 caracteres compostos de letras, números e caracteres especiais e são únicos para cada planilha criada. Esse ID é obtido quando o *script* é implantado.

Nesse caso, as variáveis ESR, capacitância e tensão do barramento são substituídas pelas variáveis var1, var2 e var3 no endereço respectivamente.

Assim que uma *flag* (*flagSendToGoogle*) sinaliza que os dados estão prontos para serem processados e enviados pelo módulo WiFi, uma função customizada (*prepareDataToGoogleSheets()*) é chamada para transformar as variáveis dos vetores salvos do tipo número para *string*, plotando no serial interno do ESP32 uma mensagem de notificação para confirmar que os dados estão sendo enviados. Após a varredura do vetor por completo, uma condicional que compara o índice de varredura com o tamanho do vetor é aplicada, onde verifica se os dados foram enviados e reinicializa a *flag* para novos envios. Esse processo, de forma generalizada, é implementado como:

```
void prepareDataToGoogleSheets() {
  if (flagSendToGoogle == 1) {
    for (i = 0; i <= 9; i++) {
      String ESR_s(ESR_v[i], 6);
      String C_s(C_v[i], 6);
      String Vc_s(Vc_v[i], 6);
      String param;
      param = "value1=" + ESR_s;
      param += "&value2=" + C_s;
      param += "&value3=" + Vc_s;
      Serial.println("Sending parameters (...) index: " + String(i));
      sendData(param);
      Serial.print("\n");

      if (i == lengthESRData - 1) {
        Serial.println("The data was sent to Google Sheets successfully");
```

```
17          flagSendToGoogle = 0;
18        }
19      }
20    }
21 }
```

Lista 4.2 – Função que prepara os dados para envio via WiFi.

Onde:

i: Índice de varredura do vetor;

X_v: Vetor de 10 amostras recebidos pelo módulo UART;

X_s: Variável tipo *string* que converte o vetor de amostras conforme o índice de varredura;

Além disso, perceba que outra função com nome de ***sendData(param)*** também é chamada internamente, onde "***param***" se trata de uma variável do tipo *string* que concatena parte do endereço referente às variáveis a serem enviadas. Essa função é responsável por lidar com o protocolo HTTP e enviar os dados ao servidor. Maiores detalhes podem ser verificados nos apêndices desse trabalho.

4.4.2.2 Recepção e tratamento de dados com JavaScript no Google Sheets

Após a recepção dos dados pelo servidor, os mesmos passam por um *script* de verificação na linguagem JavaScript para tratamento. Inicialmente, após passar por verificação de uma função de nome *doGet(e)* que é executada quando uma requisição HTTP é recebida, o ID da planilha alvo deve ser substituída na variável *sheet_id* relacionada. Esse ID vem da URL padrão com 44 caracteres também composta por letras, números e caracteres especiais. A forma geral é dada como:

```
1 https://docs.google.com/spreadsheets/d/sheet_id
```

Lista 4.3 – URL padrão de uma planilha no Google Sheets onde é obtido o *sheet_id*.

Após isso, uma estrutura condicional *switch* é implementada para separar a *string* "*param*" recebida em partes de um vetor com nome de ***rowData***[], que irá ser responsável por armazenar as variáveis que serão plotadas na planilha. As posições 0 e 1 do vetor são salvas a partir da chamada da função construtora "*Date()*" própria do JavaScript, onde são extraídos dia e hora respectivamente. As outras posições são definidas como:

```
1 switch (param) {
2  case 'value1': //Parameter 1
3  rowData[2] = value; //Value in column A (ESR)
4  result = 'Written on column A';
5  break;
6  case 'value2': //Parameter 2
7  rowData[3] = value; //Value in column B (C)
8  result += ' Written on column B';
9  break;
```

```
  case 'value3': //Parameter 3
  rowData[4] = value; //Value in column C (Vc)
  result += ' Written on column C';
  break;
  default:
  result = "unsupported parameter";
}

```

Lista 4.4 – Estrutura condicional *switch* que trata de separar os dados da *string* "*param*" recebida do ESP32.

Após isso, finalmente os dados são escritos na planilha do Google Sheets com a função *setValues*, da forma:

```
var newRange = sheet.getRange(newRow, 1, 1, rowData.length);
newRange.setValues([rowData]);
```

Lista 4.5 – Escrevendo valores na planilha

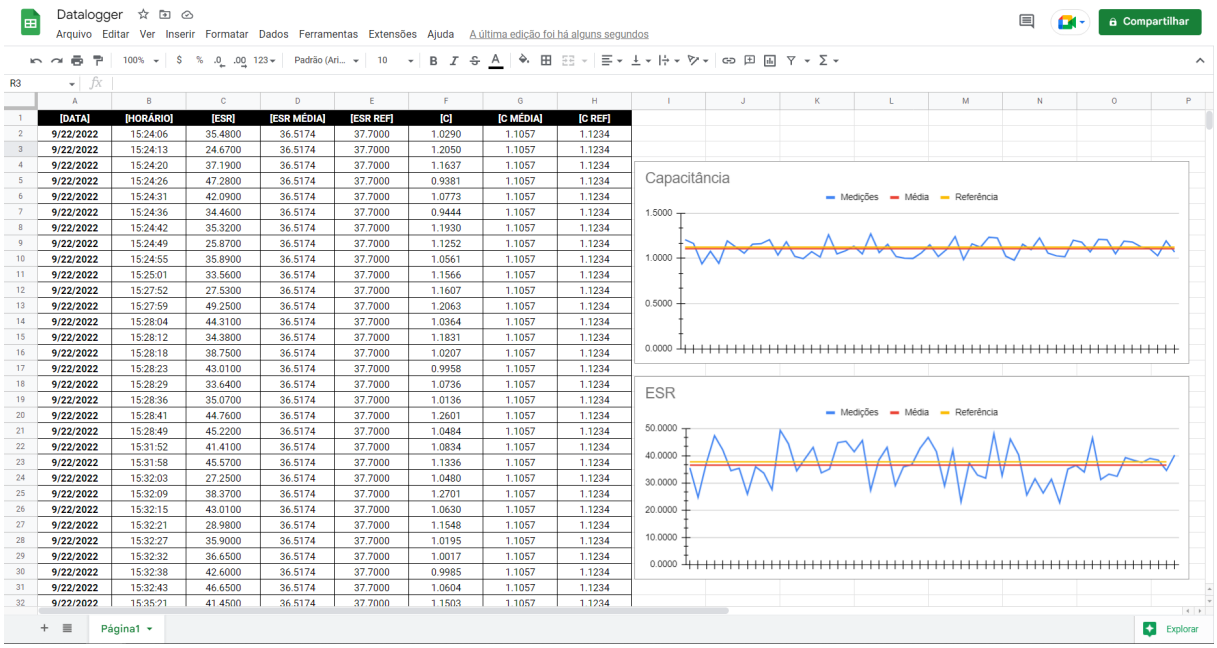
Se tudo funcionar como esperado, o envio e resposta da solicitação HTTP no serial do ESP32 deve retornar como mostra uma representação do monitor serial a seguir:

```
  Sending parameters (...) index: 10
  https://script.google.com/macros/s/GOOGLE_SCRIPT_ID/exec?value1=ESR&
    value2=C&value3=Vc
  Postring data to Google Sheet
  HTTP Status Code: 200
  Payload: Written on column A Written on column B Written on column C
  The data was sent to Google Sheets successfully
```

Lista 4.6 – Representação do monitor serial durante e após o envio dos 10 parâmetros para o Google Sheets

A Figura 63 mostra como deve parecer a planilha no Google Sheets após envio dos dados.

Figura 63 – Planilha no Google Sheets.



[DATA]	[HORÁRIO]	[ESR]	[ESR MÉDIA]	[ESR REF]	[C]	[C MÉDIA]	[C REF]
9/22/2022	15:24:06	35.4800	36.5174	37.7000	1.0290	1.1057	1.1234
9/22/2022	15:24:13	24.6700	36.5174	37.7000	1.2050	1.1057	1.1234
9/22/2022	15:24:20	37.1900	36.5174	37.7000	1.1637	1.1057	1.1234
9/22/2022	15:24:26	47.2800	36.5174	37.7000	0.9381	1.1057	1.1234
9/22/2022	15:24:31	42.0900	36.5174	37.7000	1.0773	1.1057	1.1234
9/22/2022	15:24:36	34.4600	36.5174	37.7000	0.9444	1.1057	1.1234
9/22/2022	15:24:42	35.3200	36.5174	37.7000	1.1930	1.1057	1.1234
9/22/2022	15:24:49	25.8700	36.5174	37.7000	1.1252	1.1057	1.1234
9/22/2022	15:24:55	35.8900	36.5174	37.7000	1.0561	1.1057	1.1234
9/22/2022	15:25:01	33.5600	36.5174	37.7000	1.1566	1.1057	1.1234
9/22/2022	15:27:52	27.5300	36.5174	37.7000	1.1607	1.1057	1.1234
9/22/2022	15:27:59	49.2500	36.5174	37.7000	1.2063	1.1057	1.1234
9/22/2022	15:28:04	44.3100	36.5174	37.7000	1.0364	1.1057	1.1234
9/22/2022	15:28:12	34.3800	36.5174	37.7000	1.1831	1.1057	1.1234
9/22/2022	15:28:18	38.7500	36.5174	37.7000	1.0207	1.1057	1.1234
9/22/2022	15:28:23	43.0100	36.5174	37.7000	0.9958	1.1057	1.1234
9/22/2022	15:28:29	33.6400	36.5174	37.7000	1.0736	1.1057	1.1234
9/22/2022	15:28:36	35.0700	36.5174	37.7000	1.0136	1.1057	1.1234
9/22/2022	15:28:41	44.7600	36.5174	37.7000	1.2601	1.1057	1.1234
9/22/2022	15:28:49	45.2200	36.5174	37.7000	1.0484	1.1057	1.1234
9/22/2022	15:31:52	41.4100	36.5174	37.7000	1.0834	1.1057	1.1234
9/22/2022	15:31:58	45.5700	36.5174	37.7000	1.1336	1.1057	1.1234
9/22/2022	15:32:03	27.2500	36.5174	37.7000	1.0480	1.1057	1.1234
9/22/2022	15:32:09	38.3700	36.5174	37.7000	1.2701	1.1057	1.1234
9/22/2022	15:32:15	43.0100	36.5174	37.7000	1.0630	1.1057	1.1234
9/22/2022	15:32:21	28.9800	36.5174	37.7000	1.1548	1.1057	1.1234
9/22/2022	15:32:27	35.9000	36.5174	37.7000	1.0195	1.1057	1.1234
9/22/2022	15:32:32	36.6500	36.5174	37.7000	1.0017	1.1057	1.1234
9/22/2022	15:32:38	42.6000	36.5174	37.7000	0.9985	1.1057	1.1234
9/22/2022	15:32:43	46.6500	36.5174	37.7000	1.0604	1.1057	1.1234
9/22/2022	15:35:21	41.4500	36.5174	37.7000	1.1503	1.1057	1.1234

Fonte: do Autor.

4.5 ESTRUTURA FINAL DO PROTÓTIPO

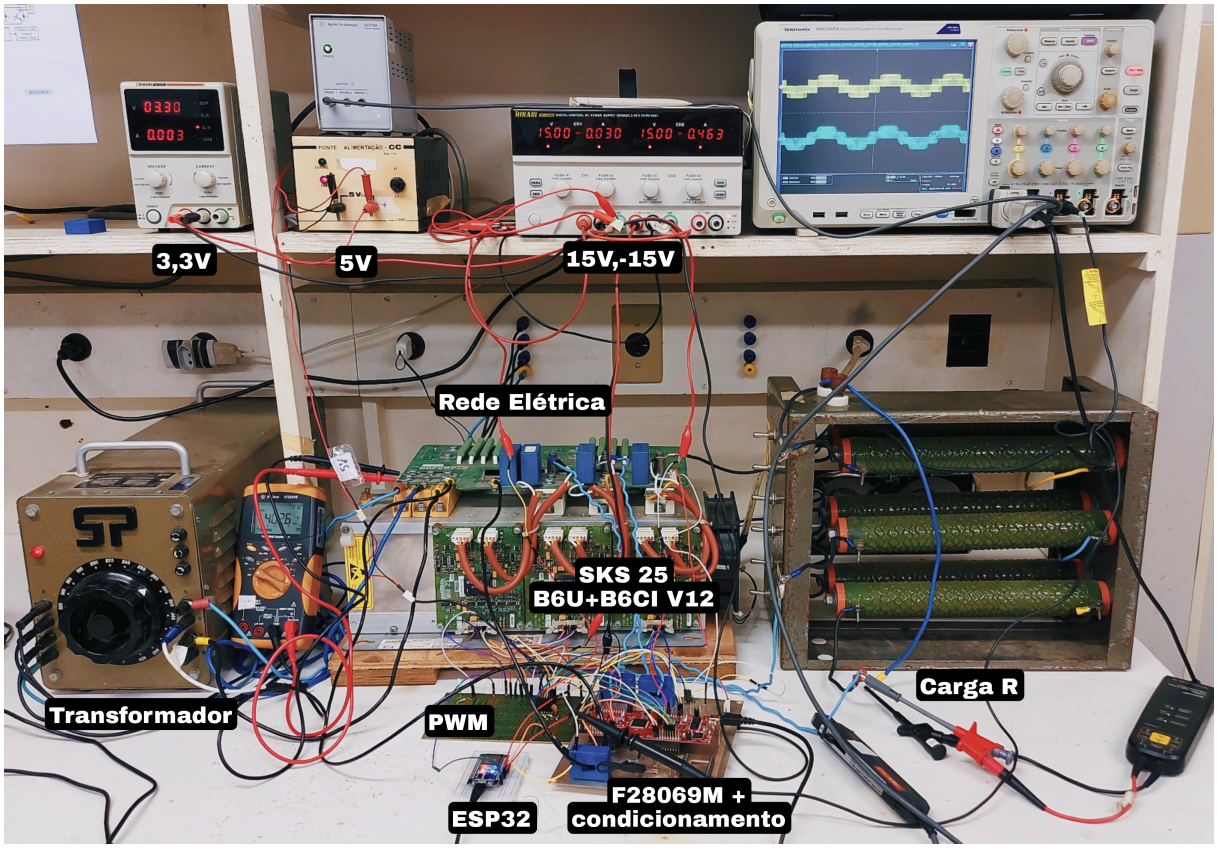
Na estrutura final do protótipo foram realizadas algumas modificações, como a medida das tensões dos interruptores sendo emuladas internamente com o próprio sinal de PWM do DSP, já que a presença do *offset* inserida pelo CI SN74LS07 e a baixa taxa de *Slew-Rate* dos amp-ops MAX4238 estavam distorcendo os sinais percebidos pelos pinos digitais do dispositivo, em aplicações reais, a medição das tensões dos interruptores deve ser realizada, os resultados esperados devem ser praticamente os mesmos, considerando apenas um pequeno atraso adicionado pelos possíveis CI's a serem utilizados. O sensoriamento de temperatura também não foi implementado, já que seria apenas necessário para a etapa de previsão de vida útil do capacitor e compensação de variáveis, o que exigiria análise semanal do sistema funcionando por várias horas diárias ou uma aceleração artificial do envelhecimento do componente. Nesse caso em específico, o experimento foi conduzido em ambiente controlado, assim como a medição dos parâmetros para comparação com o medidor de impedância, e portanto, a compensação das variáveis por temperatura não foi necessária.

Ainda, o DSP utilizado não suportou realizar os cálculos do método RLS na interrupção do ADC para a frequência de amostragem desejada sem extrapolar o tempo de cálculo máximo, e por conta disso, os dados foram armazenados em vetores e depois processados na rotina principal. Assim, para deixar as comparações justas, o algoritmo da DFT também realizou os cálculos fora da interrupção, sendo escolhido o formato radix-4,

podendo ser substituído pelo algoritmo de Goertzel para implementações recursivas na interrupção.

Além disso, a placa de comando e ESP32 foram implementados externamente à placa principal de condicionamento por conta de imprevistos em relação ao dimensionamento da placa. A Figura 64 mostra o protótipo montado.

Figura 64 – Protótipo montado.



Fonte: do Autor.

4.6 RESULTADOS EXPERIMENTAIS

Após a montagem do protótipo basta extrair os resultados. Os parâmetros utilizados para o experimento são apresentados na Tabela 12.

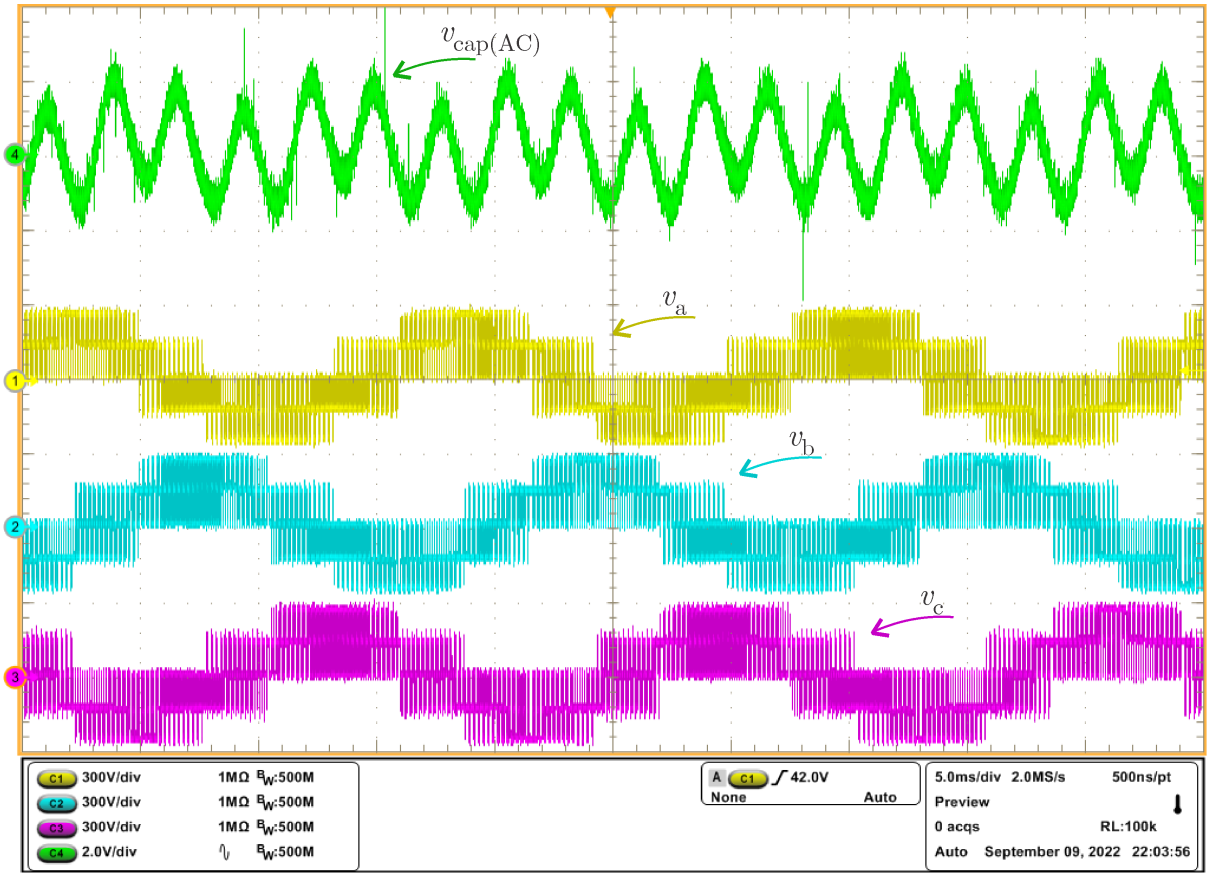
Tabela 12 – Parâmetros do protótipo montado em bancada.

Parâmetros	**Valores**
Frequência da rede elétrica	60 Hz
Potência do sistema	2,20 kW
Tensão do barramento CC	400 V

Fonte: do Autor.

A Figura 65 mostra no topo o valor AC da tensão do barramento em 400 V, onde se verifica um comportamento oscilatório por conta das distorções de forma de onda da tensão da rede elétrica e da influência do transformador adicionado na entrada. A variação de tensão pico a pico é de 4 V, o que resulta em 270 mV chegando no ADC do DSP. Em sequência, as tensões de fase ABC são ilustradas respectivamente, com um valor de pico de 276 V e valor eficaz de 158,4 V.

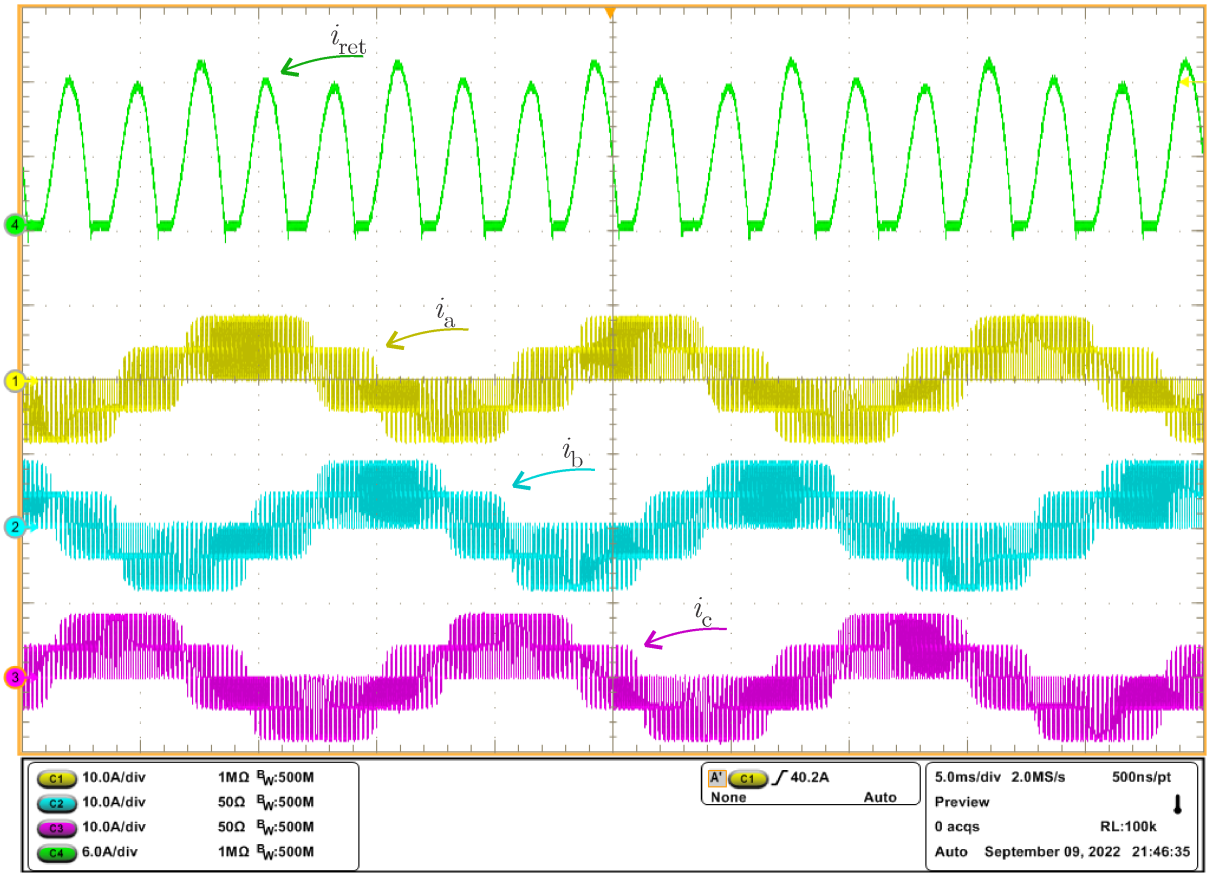
Figura 65 – Componente AC da tensão do barramento em 400V e tensões de fase ABC respectivamente.



Fonte: do Autor.

Já a Figura 66 mostra a corrente na saída do retificador no topo, com um valor máximo de 13 A, com uma oscilação por consequência das distorções da rede elétrica. A seguir, as correntes de fase ABC são mostradas em ordem, com um valor de pico de 8,9 A e corrente eficaz de 4,78 A.

Figura 66 – Corrente na saída do retificador e correntes de fase ABC respectivamente.



Fonte: do Autor.

4.7 VARIÁVEIS LIDAS NO CCS E GOOGLE SHEETS

Nesta seção serão apresentadas as análises dos dados adquiridos pelo *Code Composer Studio* - CCS onde ficam armazenados os dados lidos pelo ADC do DSP F28069M. Além disso, também são apresentados os dados lidos diretamente do Google Sheets e processados pelo MATLAB. As leituras realizadas pelo ESP32 são feitas a cada 15 s por aproximadamente 2,5 minutos (o que totalizam 10 amostras). No total, foram obtidas 60 amostras para cada método. Observa-se que nem todas as amostras são enviadas no tempo exato por conta de erros na comunicação, o que levou o processo de amostragem ter uma duração em torno de 20 minutos no total para cada método.

A Figura 67 mostra a tensão do barramento lida pelo ADC do DSP e armazenada em um vetor de visualização no software CCS. Posteriormente foi adicionada a forma de onda do RMS obtida com janela móvel de 120 amostras. Esse mesmo procedimento de análise é utilizado nas próximas variáveis, se tratando de uma manipulação de uma função interna chamada de '*movmean(array, window)*' do MATLAB. O modelo utilizado é o mais básico e realiza a média de uma determinada janela a partir de um valor central. Para um

exemplo de aplicação, considere que se defina um vetor A com uma janela de 3 variáveis, ou seja:

```
A = [4 8 6 -1 -2 -3 -1 3 4 5]
rmsWindow = 3
M = movmean(A,rmsWindow)
```

Lista 4.7 – Arquivo "Exemplo_movmean.m".

O resultado será baseado na média de cada 3 variáveis com um valor central de cada vez. Para a primeira variável do vetor resultante a média será de:

$$M(1) = \frac{null + 4 + 8}{rmsWindow - N_{\text{null}}} = 6 \tag{182}$$

Onde a média é sempre realizada pelo tamanho da janela escolhida menos a quantidade de números vazios, ou seja, o resto de valores que englobam a janela mas se encontram fora do vetor definido. Nesse caso,

$$rmsWindow - N_{\text{null}} = 2 \tag{183}$$

Assim, o vetor M terá resultado de:

```
M = [6.00 6.00 4.33 1.00 -2.00 -2.00 -0.33 2.00 4.00 4.50]
```

Lista 4.8 – Arquivo "Exemplo_movmean.m".

Já para adaptar esse mesmo comando para se obter o vetor de RMS com janela móvel, considera-se que a equação geral que define o valor de *Root Mean Square* é de:

$$x_{\text{rms}} = \sqrt{\frac{1}{n}(x_1^2 + x_2^2 + ... + x_n^2)} \tag{184}$$

Ou seja, adaptando o comando para realizar a mesma operação, define-se como:

```
M_movrms = sqrt(movmean(A.^2, rmsWindow))
```

Lista 4.9 – Arquivo "Exemplo_movmean.m".

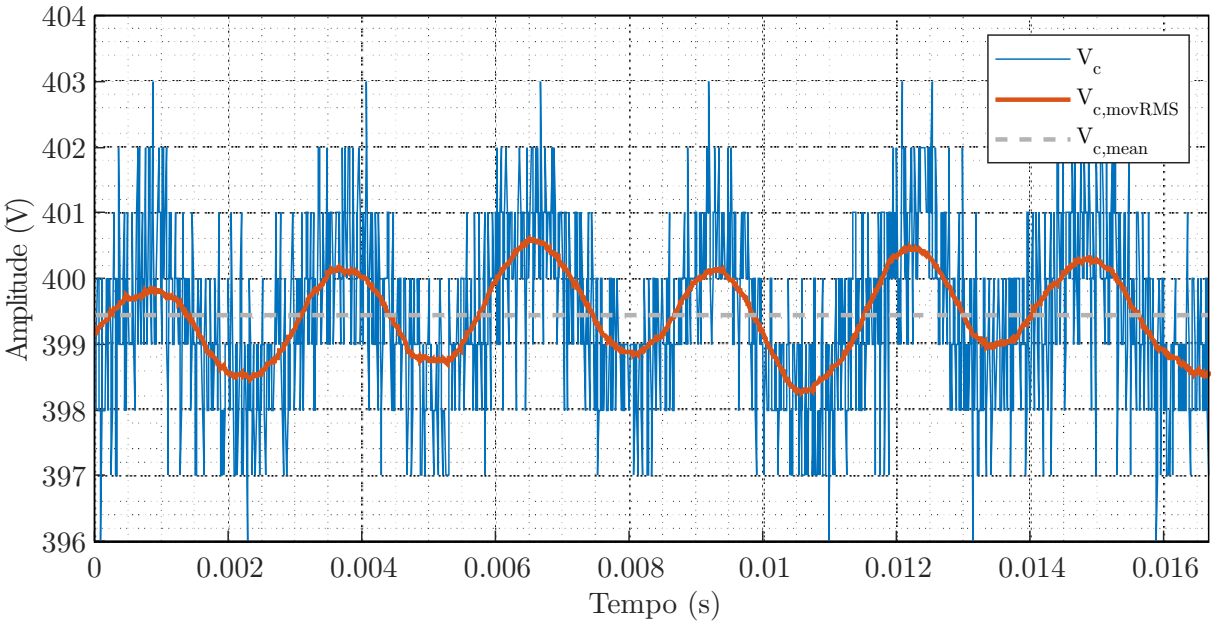
O que resulta em:

```
M_movrms = [6.32 6.22 5.80 3.70 2.16 2.16 2.51 2.94 4.08 4.53]
```

Lista 4.10 – Arquivo "Exemplo_movmean.m".

Infelizmente por conta da amplitude pequena do sinal que chega no DSP a quantidade de ruídos é considerável, entretanto, pode-se observar perfeitamente os 4 V de *ripple* observados no osciloscópio dos resultados experimentais. O RMS de janela móvel também foi aplicado para se verificar claramente a forma de onda relativa ao sinal de tensão obtido.

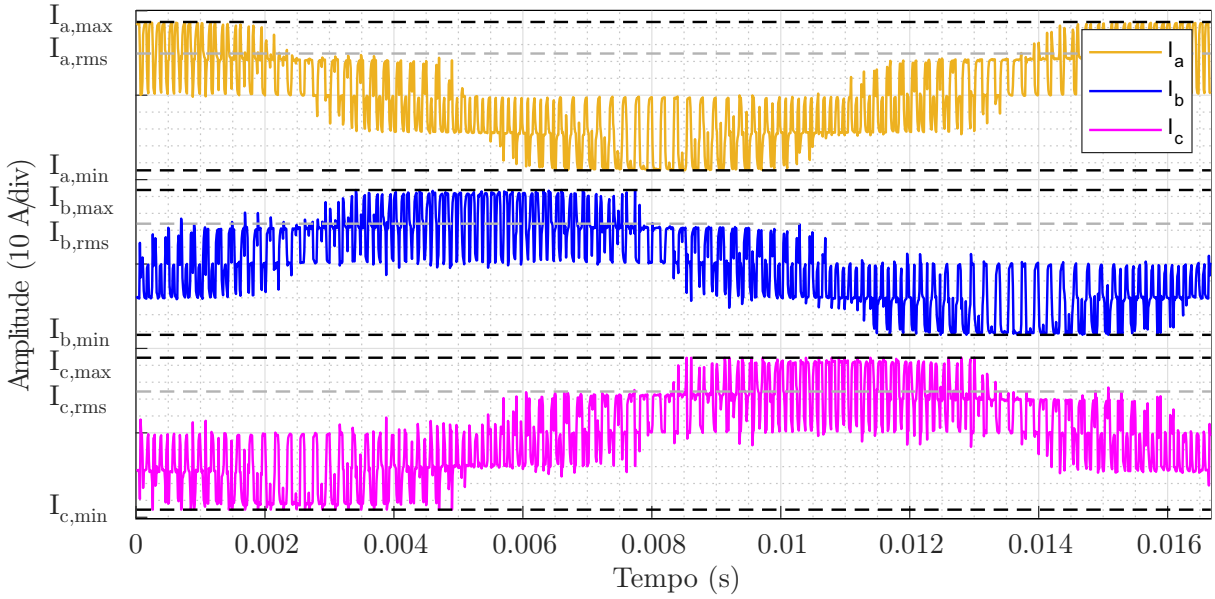
Figura 67 – Tensão do barramento lida no DSP.



Fonte: do Autor.

As correntes de fase A, B e estimada C são mostradas na Figura 68, que apesar de alguns ruídos presentes em pontos arbitrários do sinal, apresentou semelhança considerável com os resultados observados nos resultados do osciloscópio mostrados anteriormente.

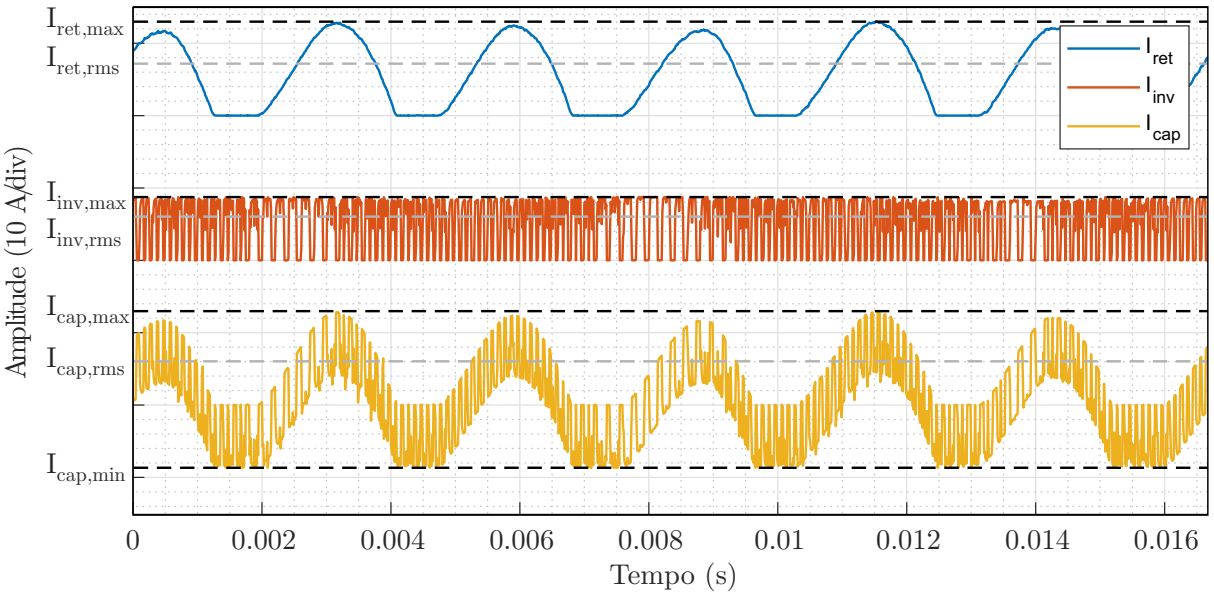
Figura 68 – Correntes de fase A e B lidas no DSP respectivamente e corrente da fase C estimada.



Fonte: do Autor.

Na Figura 69 podem ser observadas a corrente do retificador armazenada no topo, e as correntes I_{inv} e I_{cap} estimadas pelo DSP.

Figura 69 – Corrente na saída do retificador lida no DSP e correntes da entrada do inversor e do capacitor estimadas.

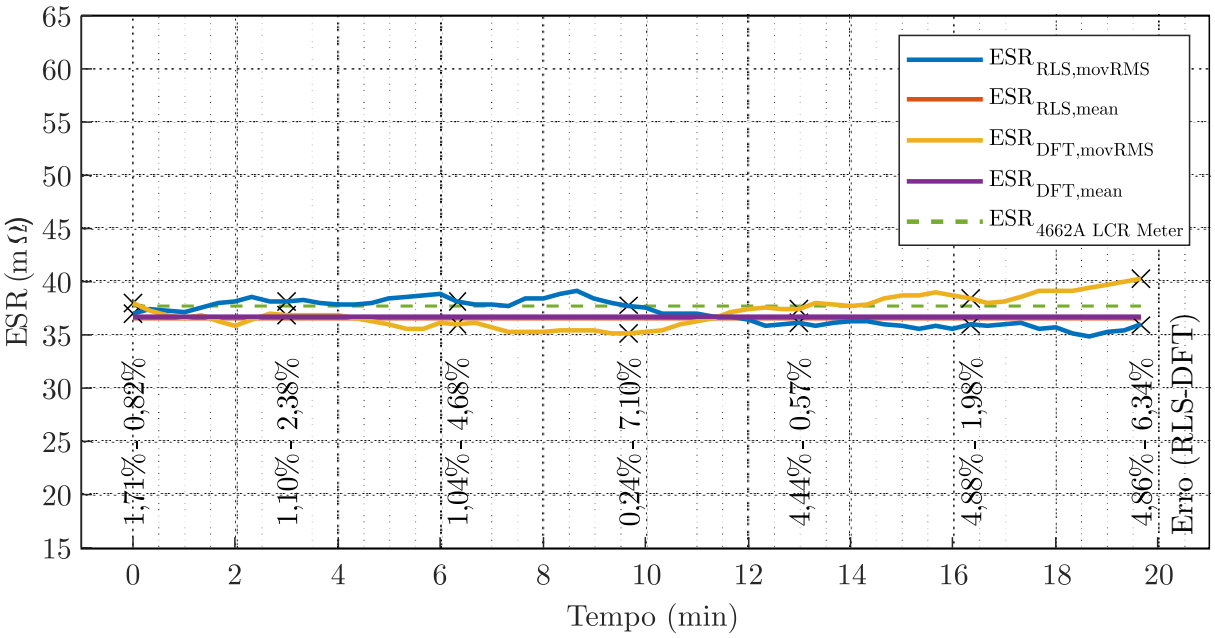


Fonte: do Autor.

4.7.1 Comparação entre os métodos RLS e DFT utilizando RMS com janela móvel

A Figura 70 mostra o comportamento da ESR para os métodos RLS e DFT durante um intervalo de 20 minutos e 60 amostras. O mesmo procedimento para RMS de janela móvel foi aplicada com janela de 30 amostras, onde os erros máximos da ESR foram de 7,68% e 7,10% e valores médios de 36,52 Ω e 36,65 Ω para os métodos RLS e DFT respectivamente. Além disso, o equipamento 4262A Agilent LCR Meter com a frequência selecionada de 120 Hz (o equipamento não permite sintonizar em 360 Hz) foi utilizado como referência para os capacitores utilizados, onde a ESR e Capacitância de referência foram observados sendo de 37,7 Ω e 1,12 mF respectivamente.

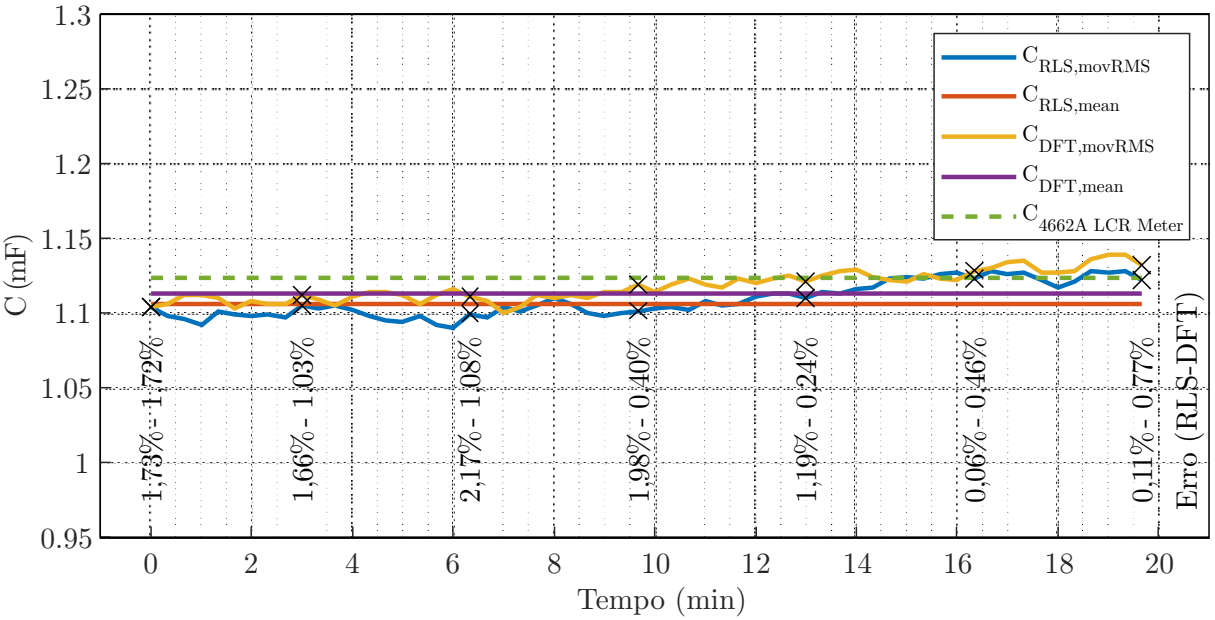
Figura 70 – Comparação da ESR entre os métodos RLS e DFT.



Fonte: do Autor.

Já a Figura 71 mostra o comportamento da capacitância para o mesmo intervalo e quantidade de amostras que o anterior. Os valores de erros máximos foram de 3,09% e 2,11% e médios de 1,10 mF e 1,11 mF para os métodos RLS e DFT respectivamente.

Figura 71 – Comparação da capacitância entre os métodos RLS e DFT.



Fonte: do Autor.

4.8 CONSIDERAÇÕES FINAIS

Neste capítulo foi exposto o processo de desenvolvimento e resultados obtidos do experimento realizado, desde os resultados brutos obtidos diretamente do osciloscópio, os dados importados do *software* CCS a partir da leitura com o DSP F28069M, até a análise dos dados importados do Google Sheets, enviados pelo dispositivo ESP32.

Com base nos resultados primários obtidos, percebe-se que alguns componentes como o circuito integrado MAX4238 que possui um *Slew Rate* de 0,35 $V\mu s$ (sendo a taxa para que o sinal de saída do CI se acomode em regime permanente) pode ser substituído por um componente com valor maior, o que reproduziria certamente resultados ainda mais precisos, já que um valor pequeno pode distorcer as componentes de frequências mais altas do sinal analisado. Além disso, sugere-se a utilização de um ADC externo com maior resolução e frequência de amostragem maior.

Apesar disso, é possível concluir que mesmo com as imprecisões obtidas com os circuitos integrados escolhidos, os resultados tiveram uma taxa de erro máxima aceitável. Infelizmente a estimativa de vida útil a longo prazo não foi possível de ser realizada, pois o processo de experimentação e amostragem necessitaria de ter uma duração de no mínimo semanas com o módulo inversor funcionando diariamente, forçando o envelhecimento e desgaste dos componentes do conversor. A Tabela 13 mostra a comparação dos resultados entre os dados lidos pelo DSP e diretamente no osciloscópio durante o experimento (Exp.), com o valor máximo, mínimo, RMS, médio, erro máximo, erro mínimo e erro RMS respectivamente. Os dados analisados tiveram valores muito próximos, tendo uma taxa de erro maior de 8,1353% no valor mínimo da corrente I_b. Os valores relativos a corrente do inversor (I_{inv}) e corrente do capacitor (I_{cap}) não foram possíveis de serem obtidas no experimento, pois não há acesso direto a esses sinais, o que deixa apenas a estimativa realizada pelo DSP como referência.

Tabela 13 – Comparação de resultados entre dados armazenados no DSP e visualizados no osciloscópio durante o experimento.

		Máximo	**Mínimo**	**RMS**	**Médio**	**Erro (Máx)**	**Erro (Mín)**	**Erro (RMS)**
$\mathbf{I_a}$	**DSP**	8,6730	-8,9000	4,9366	–	5,5459	7,3820	5,8866
	Exp.	8,1920	-8,2430	4,6460	–			
$\mathbf{I_b}$	**DSP**	8,7700	-8,3900	4,7774	–	2,0659	8,1353	0,0962
	Exp.	8,9250	-9,1330	4,7820	–			
$\mathbf{I_c}$	**DSP**	8,9000	-9,1000	4,8979	–	4,5281	3,9670	5,0614
	Exp.	8,4970	-8,7390	4,6500	–			
$\mathbf{I_{ret}}$	**DSP**	12,9860	0,0000	7,1804	5,4218	3,4961	–	3,4302
	Exp.	13,4400	0,000	0,000	5,3410			
$\mathbf{I_{inv}}$	**DSP**	8,7400	0,0000	6,3700	–	–	–	–
	Exp.	–	–	–	–			
$\mathbf{I_{cap}}$	**DSP**	12,9770	-8,6700	6,0438	–	–	–	–
	Exp.	–	–	–	–			

Fonte: do Autor

5 CONCLUSÕES

De forma geral, neste trabalho foi desenvolvido um estudo acerca de dois métodos para estimativa da resistência interna (ESR) e capacitância de um capacitor eletrolítico, podendo também ser aplicado a capacitores MPPF com ressalvas, permitindo a estimativa de vida útil do componente, prevendo o momento em que o componente irá falhar baseando-se em amostras obtidas ao longo do tempo.

Inicialmente, ressaltou-se a importância do estudo de veículos elétricos atualmente, onde se busca principalmente a substituição dos combustíveis derivados de petróleo. Foi apresentado também que um dos principais dispositivos que compõe a conversão de energia elétrica em energia cinética em um VE se trata do inversor CC/CA, responsável por realizar a interface entre os sistemas da bateria e o motor elétrico. Além disso, também foi mostrado que nesse dispositivo a maior causa de falhas é a temperatura, e que dentre os componentes presentes o que mais aponta problemas recorrentes são os capacitores.

Com base nisso, uma extensa revisão bibliográfica foi realizada, e os métodos utilizados na literatura para previsão de vida útil desse componente foram comparados e analisados. Nesse trabalho, um dos objetivos principais tem como base a premissa de que o sistema a ser utilizado com esse intuito deve atuar apenas como um observador, de forma não invasiva ao sistema, ou seja, implementando apenas sensoriamento externo e não modificando a estrutura interna do conversor, como modulação, injeção de sinais ou controle. Da revisão realizada, dois principais artigos se destacaram, onde um deles trata da estimativa da ESR e capacitância baseando-se em um Método dos Mínimos Quadrados Recursivo (RLS) (YU et al., 2012) e o outro baseado na utilização de uma modificação da transformada discreta de Fourier chamada de algoritmo de Goertzel (SUNDARARAJAN et al., 2020). Com base nesses artigos, essa dissertação teve como proposta estudá-los, desenvolver os métodos matematicamente de forma mais completa, melhorá-los, compará-los via simulação, e por fim, validá-los experimentalmente em bancada no laboratório.

O desenvolvimento matemático realizado iniciou-se no capítulo 2, onde o procedimento inicial sugerido é submeter o capacitor analisado a uma câmara fechada, onde será controlada a temperatura e os parâmetros de interesse serão medidos com um analisador de impedância. Esses dados obtidos devem ser armazenados em um vetor específico dentro do DSP de preferência, que fará a interpolação dos mesmos encontrando constantes referentes a suas equações com base no Método dos Mínimos Quadrados. Essas equações obtidas são importantes para a correção dos parâmetros de ESR e capacitância, já que a temperatura exerce uma influência direta nesses, podendo confundir o algoritmo de previsão de vida útil que irá considerar essa variação como envelhecimento do componente e não uma alteração pontual devido à temperatura.

Após isso, é destacado que em inversores comerciais geralmente não há possibilidade de se ter acesso direto a corrente de entrada do inversor ou do capacitor. Dessa forma,

propõe-se a estimativa da corrente do inversor baseando-se na medição das tensões das chaves inferiores e duas das correntes de fase de saída, que realizando a diferença com a corrente de saída do retificador, resulta na corrente do capacitor.

A partir disso, iniciou-se a análise dos métodos propostos a serem estudados e desenvolvidos. O primeiro método a ser estudado é baseado na Transformada Discreta de Fourier (DFT) e suas derivações. Esse utiliza como base a extração de harmônicas específicas (magnitude e fase) para calcular os parâmetros do capacitor. Ainda, três métodos derivados da DFT convencional foram apresentados, sendo o radix-2, radix-4 e algoritmo de Goertzel, que é basicamente uma tradução da DFT em forma recursiva. Esses métodos foram comparados em termos de exigência de processamento e memória, e concluiu-se que as melhores escolhas a serem implementadas no DFT tratou-se dos métodos radix-4 por possuir menos multiplicações e utilização de senos e cossenos nos formatos de DFT *offline*, e algoritmo de Goertzel por sua recursividade, que além disso, não necessita de armazenamento de variáveis em vetores, o que facilita a implementação na interrupção do ADC do DSP.

Na próxima seção é apresentado o Método dos Mínimos Quadrados Recursivo, que inicia com a obtenção da função de transferência do circuito base do capacitor e sua ESR. Essa é equacionada em um formato que permite aplicar o método dos mínimos quadrados de forma discreta, minimizando o erro. Após o equacionamento desenvolvido e a equivalência do método com o filtro de Kalman através do fator de esquecimento, as variáveis são retornadas as suas respectivas formas e os parâmetros são obtidos.

Por fim, após o armazenamento dos parâmetros de ESR e capacitância, é sugerido uma modificação do método de Gauss-Newton chamado de algoritmo de Levenberg-Marquadt para realizar a extrapolação dos dados e obter o momento próximo em que o capacitor irá falhar, possibilitando a substituição do componente antes que o mesmo cause maiores danos ao sistema. A equação de Arrhenius também foi abordada, essa teoria fornece uma relação matemática entre a taxa de reação e a temperatura, estresse de corrente e de tensão, mas sua aplicação é restrita a sistemas ideais. Devido às variações dinâmicas nos parâmetros do sistema experimental, essa equação pode ser utilizada como uma referência geral, mas não deve ser considerada um resultado confiável em situações reais.

No capítulo 3 é apresentado os resultados de simulação no PSIM, onde os principais são obtidos com uma carga R na saída. Em simulação, os resultados são de certa forma subjetivos, já que demonstraram depender não só da frequência de amostragem, mas também do passo de cálculo usado na simulação. Entretanto, mesmo com essas limitações intrínsecas do *software*, os resultados foram extremamente aceitáveis para previsão da vida útil do capacitor, não ultrapassando 1,5% em nenhum caso, mesmo com uma variação de temperatura exagerada de 25ºC para 50ºC (considerando os dados dos resultados do método DFT já tratados com média móvel).

Por fim, no capítulo 4 é finalmente apresentado o protótipo implementado na bancada do laboratório e os resultados experimentais são obtidos. Todo o processo envolvendo a montagem do protótipo e da placa de condicionamento foi exposta, incluindo detalhamento sobre o protocolo de comunicação utilizado para transmissão de dados entre os dispositivos F28069M e ESP32, cujo objetivo foi enviar os dados via WiFi para uma tabela em tempo real nos servidores do Google Sheets. Diferentemente da simulação, os resultados experimentais apresentaram um erro de até 7,68% para a ESR. Entretanto, ressaltou-se que alguns componentes escolhidos para a placa de condicionamento poderiam ser a causa desse erro maior, principalmente o CI MAX4238, responsável pela maioria do tratamento dos sinais, que acabou por adicionar atrasos na medição final pelo ADC do DSP. De qualquer forma, mesmo com as limitações enfrentadas no experimento, os resultados se demonstraram adequados, validando os algoritmos estudados.

Como contribuições à literatura, essa dissertação trouxe em detalhes o estudo dessas duas técnicas e pontos importantes para o processo de se estimar a vida útil de um capacitor eletrolítico ou mesmo o de filme. Os métodos foram desenvolvidos matematicamente por completo e após isso foram validados via simulação e experimento, onde foi exposto em detalhes todo o processo envolvido.

De forma geral, os objetivos propostos previamente de prever falhas e estimar a vida útil de capacitores foram atingidos com sucesso. Pontos positivos e negativos acerca dos métodos utilizados foram mencionados e a eficácia e dificuldades para implementação prática foram discutidos. Por fim, expõem-se sugestões para trabalhos futuros:

- Refazer o experimento na bancada com circuitos integrados que possuam respostas mais rápidas do que os utilizados nessa dissertação;
- A implementação do método RLS em DSP se demonstrou ser impraticável na interrupção do ADC para altas frequências para se obter resultados em tempo real e aproveitar a recursividade. Dessa forma, sugere-se o estudo de alternativas para simplificar o equacionamento (principalmente multiplicações e divisões), já que o método depende de uma amostragem em alta frequência para se obter erros menores.
- Neste trabalho foram estudados e analisados algoritmos que já existiam previamente na literatura, apesar dos métodos serem o suficiente para prever a vida útil do capacitor, sugere-se o estudo da possibilidade em se propor novos métodos para se obter maior variedade de algoritmos para comparação;
- Simulações com cargas diferentes foram realizadas para demonstrar a eficiência dos métodos, mostrando que o funcionamento independe da saída conectada. Sugere-se, portanto, implementações experimentais com diferentes tipos de cargas para comprovar as simulações realizadas.
- Paralelamente a essa dissertação, um breve estudo foi realizado conectando um

conversor bidirecional (*Dual Active Bridge* - DAB) no lugar do retificador a diodos, alimentado por uma bateria. Esse sistema representa o que tipicamente seria a conexão do carro elétrico em funcionamento, onde a bateria fornece energia ao motor elétrico. Como visto nos estudos desenvolvidos, os métodos dependem de que haja componentes harmônicas no sinal de tensão do capacitor e portanto um *ripple* muito pequeno produzido pelo conversor CC/CC impossibilitaria a implementação. Entretanto, foi constatado em simulação que com circuitos integrados de precisão é possível de se obter bons resultados com *ripples* de até 1 V pico a pico. Sugere-se a implementação e análise aprofundada dos métodos para sistemas do tipo bateria - conversor CC/CC - inversor e sua implementação prática para validação.

REFERÊNCIAS

ABDENNADHER, Karim; VENET, Pascal; ROJAT, Gérard; RETIF, Jean-Marie; ROSSET, Christophe. A Real-Time Predictive-Maintenance System of Aluminum Electrolytic Capacitors Used in Uninterrupted Power Supplies. **IEEE Transactions on Industry Applications**, v. 46, p. 1644–1652, 4 jul. 2010. ISSN 0093-9994.

ABO-KHALIL, A.G. Current injection-based DC-link capacitance estimation using support vector regression. **IET Power Electronics**, v. 5, p. 53, 1 2012. ISSN 17554535.

AELOIZA, E.; KIM, Jang-Hwan; ENJETI, P.; RUMINOT, P. A Real Time Method to Estimate Electrolytic Capacitor Condition in PWM Adjustable Speed Drives and Uninterruptible Power Supplies. In: p. 2867–2872.

AHMAD, Md. Waseem; KUMAR, P. Nandha; ARYA, Abhinav; ANAND, Sandeep. Noninvasive Technique for DC-Link Capacitance Estimation in Single-Phase Inverters. **IEEE Transactions on Power Electronics**, v. 33, p. 3693–3696, 5 mai. 2018. ISSN 0885-8993.

ALBERTSEN, Arne. **Electrolytic Capacitor Lifetime Estimation**. [S.l.], 2010. Disponível em: <www.jianghai-europe.com>.

ALEXANDER, Winser; WILLIAMS, Cranos M. **Digital Signal Processing: Principles, Algorithms and System Design**. 1st. USA: Academic Press, Inc., 2016. ISBN 0128045477.

BARBOSA, Callebe Soares. **IMPLEMENTAÇÃO DO ALGORITMO RADIX-2 PARA CÁLCULO DA FFT EM FPGA**. 2018. Universidade Tecnológica Federal do Paraná.

BHATT, Devang Kirtikumar; EL. DARIEBY, Mohamed. An Assessment of Batteries form Battery Electric Vehicle Perspectives. In: 2018 IEEE International Conference on Smart Energy Grid Engineering (SEGE). [S.l.: s.n.], 2018. P. 255–259.

BROWN, Robert W. Equivalent series and parallel equivalent circuits for metallized film capacitors. In: p. 1–4.

CHAN, C.C. The state of the art of electric and hybrid vehicles. **Proceedings of the IEEE**, Institute of Electrical e Electronics Engineers Inc., v. 90, p. 247–275, 2 2002. ISSN 00189219.

CHASSAING, Rulph; REAY, Donald. **Digital Signal Processing and Applications with the TMS320C6713 and TMS320C6416 DSK**. [S.l.]: Wiley, abr. 2008. ISBN 9780470138663.

CHEMI-CON, Nippon. **Aluminum capacitors catalogue-Technical note on judicious use of aluminum electrolytic capacitors, 2013**. [S.l.], 2013.

CHENG, Yao-Ting. **Application Report Autoscaling Radix-4 FFT for TMS320C6000™**. [S.l.], 2000.

CHU, Eleanor; GEORGE, Alan. **Inside the FFT Black Box**. [S.l.]: CRC Press, nov. 1999. ISBN 9780367802332.

CLARKE, D.W. Adaptive control. **Automatica**, v. 27, p. 207–208, 1 jan. 1991. ISSN 00051098.

COOLEY, James W.; TUKEY, John W. An algorithm for the machine calculation of complex Fourier series. **Mathematics of Computation**, v. 19, p. 297–301, 90 1965. ISSN 0025-5718.

DANG, Hoang-Long; KWAK, Sangshin. Review of Health Monitoring Techniques for Capacitors Used in Power Electronics Converters. **Sensors**, v. 20, p. 3740, 13 jul. 2020. ISSN 1424-8220.

DIRLIK, S. **A comparison of FFT processor designs**. [S.l.: s.n.], dez. 2013.

ESPRESSIF. **ESP32 Datasheet**. [S.l.], 2022. Disponível em: <https://www.espressif.com/>.

GAO, Yimin; EHSANI, M.; MILLER, J.M. Hybrid Electric Vehicle: Overview and State of the Art. In: p. 307–316.

GASPERI, M.L. Life Prediction Modeling of Bus Capacitors in AC Variable-Frequency Drives. **IEEE Transactions on Industry Applications**, v. 41, p. 1430–1435, 6 nov. 2005. ISSN 0093-9994.

GUDAVALLI, Ganesh Sainadh; DHAKAL, Tara P. Simple parallel-plate capacitors to high-energy density future supercapacitors: A materials review. **Emerging Materials for Energy Conversion and Storage**, Elsevier, p. 247–301, jan. 2018.

GUPTA, Ashok Kumar; RAMAN, Ashish; KUMAR, Naveen; RANJAN, Ravi. Design and Implementation of High-Speed Universal Asynchronous Receiver and Transmitter (UART). **2020 7th International Conference on Signal Processing and Integrated Networks (SPIN)**, 2020.

HARADA, K.; KATSUKI, A.; FUJIWARA, M. Use of ESR for deterioration diagnosis of electrolytic capacitor. **IEEE Transactions on Power Electronics**, v. 8, p. 355–361, 4 out. 1993. ISSN 0885-8993.

HAYEK, Antoine El; VENET, Pascal; MITOVA, Radoslava; WANG, Miao-Xin; CLERC, Guy; SARI, Ali. **Aging laws of electrolytic capacitors Aging laws of electrolytic capacitors. Evolution of Functional Performance and Expected Lifetime of Electrical Equipments Aging laws of electrolytic capacitors**. [S.l.], 2018. P. 16–17. Disponível em: <https://hal.archives-ouvertes.fr/hal-01922188>.

IEA. **International Energy Agency - Global EV Outlook 2022**. [S.l.], 2022. Disponível em:
<https://www.iea.org/reports/global-ev-outlook-2022>.

IMAM, A.M.; HABETLER, T.G.; HARLEY, R.G.; DIVAN, D.M. Condition Monitoring of Electrolytic Capacitor in Power Electronic Circuits using Adaptive Filter Modeling. In: p. 601–607.

IMAM, Afroz M.; DIVAN, Deepak M.; HARLEY, Ronald G.; HABETLER, Thomas G. Real-Time Condition Monitoring of the Electrolytic Capacitors for Power Electronics Applications. In: p. 1057–1061.

INSTRUMENTS, Texas. **Technical Reference Manual TMS320x2806x**. [S.l.], 2011.

JING, Wanqing; WANG, Qing; RAN, Hanzheng. Analysis on Frequency-Sensitive Failure Mechanism of Multi-Layer Ceramic Capacitor. In: p. 1–4.

KEBRIAEI, Mohammad; NIASAR, Abolfazl Halvaei; ASAEI, Behzad. Hybrid electric vehicles: An overview. In: p. 299–305.

KHAN, Shoab Ahmed. **Digital Design of Signal Processing Systems**. [S.l.]: Wiley, fev. 2011. ISBN 9780470741832.

KHANDEBHARAD, A. R.; DHUMALE, R. B.; LOKHANDE, S. S.; LOKHANDE, S. D. Real time remaining useful life prediction of the electrolytic capacitor. In: p. 631–636.

KIEFERNDORF, F.D.; FORSTER, M.; LIPO, T.A. Reduction of DC bus capacitor ripple current with PAM/PWM converter. In: p. 2371–2377.

KIM, Sang-Hun; KIM, Seok-Min; PARK, Sungmin; LEE, Kyo-Beum. Switch Open-Fault Detection for a Three-Phase Hybrid Active Neutral-Point-Clamped Rectifier. **Electronics**, v. 9, n. 9, 2020. ISSN 2079-9292.

LAHYANI, A.; VENET, P.; GRELLET, G.; VIVERGE, P.-J. Failure prediction of electrolytic capacitors during operation of a switchmode power supply. **IEEE Transactions on Power Electronics**, v. 13, p. 1199–1207, 6 nov. 1998. ISSN 0885-8993.

LEE, D.C.; LEE, K.J.; SEOK, J.K.; CHOI, J.W. Online capacitance estimation of DC-link electrolytic capacitors for three-phase AC DC AC PWM converters using recursive least squares method. **IEE Proceedings Electric Power Applications**, v. 152, p. 1503, 6 2005. ISSN 13502352.

LEE, Kwang-Woon; KIM, Myungchul; YOON, Jangho; LEE, Sang Bin; YOO, Ji-Yoon. Condition Monitoring of DC-Link Electrolytic Capacitors in Adjustable-Speed Drives. **IEEE Transactions on Industry Applications**, v. 44, p. 1606–1613, 5 set. 2008. ISSN 0093-9994.

LEE, Sang Bin et al. A New Strategy for Condition Monitoring of Adjustable Speed Induction Machine Drive Systems. **IEEE Transactions on Power Electronics**, v. 26, p. 389–398, 2 fev. 2011. ISSN 0885-8993.

LEM. **Current Transducer LA 55-P**. [S.l.], 2021. Disponível em: <https://www.lem.com/sites/default/files/products_datasheets/la_55-p_e.pdf>.

LI, Ting; CHEN, Jie; CONG, Peicheng; DAI, Xiaoteng; RUICHANG, Qiu; LIU, Zhigang. Online Condition Monitoring of DC-Link Capacitor for AC/DC/AC PWM Converter. **IEEE Transactions on Power Electronics**, v. 37, p. 865–878, 1 jan. 2022. ISSN 0885-8993.

LJUNG, L. **System Identification: Theory for the User**. [S.l.]: Prentice Hall PTR, 1999. (Prentice Hall information and system sciences series). ISBN 9780136566953.

MAKDESSI, Maawad; SARI, Ali; VENET, Pascal; BEVILACQUA, Pascal; JOUBERT, Charles. Accelerated Ageing of Metallized Film Capacitors Under High Ripple Currents Combined With a DC Voltage. **IEEE Transactions on Power Electronics**, v. 30, p. 2435–2444, 5 mai. 2015. ISSN 0885-8993.

MENG, Tianze; ZHANG, Pinjia. An on-line DC-link Capacitance Estimation method for Motor Drive System Based on Intermittent Active Control Strategy. In: p. 5118–5123.

MEZAROBA, Mateus Nava. **Eletroposto Móvel Para Carga de Baterias de Veículos Elétricos Alimentadas em Corrente Contínua e Alternada**. 2021.

NGUYEN, Thanh Hai; LEE, Dong-Choon. Deterioration Monitoring of DC-Link Capacitors in AC Machine Drives by Current Injection. **IEEE Transactions on Power Electronics**, v. 30, p. 1126–1130, 3 mar. 2015. ISSN 0885-8993.

PANDAY, Aishwarya; BANSAL, Hari Om. A Review of Optimal Energy Management Strategies for Hybrid Electric Vehicle. **International Journal of Vehicular Technology**, v. 2014, p. 1–19, nov. 2014. ISSN 1687-5702.

PARLER, S.G. Improved Spice models of aluminum electrolytic capacitors for inverter applications. In: p. 2411–2418.

PARLER, Sam G; DUBILIER, P E Cornell. **Deriving Life Multipliers for Electrolytic Capacitors**. v. 16. [S.l.], 2004. P. 11–12.

PEÑA, Eric; LEGASPI, Mary Grace. UART: A Hardware Communication Protocol Understanding Universal Asynchronous Receiver/Transmitter. **VISIT ANALOG.COM**, v. 54, 4 2020.

PRESS, William H.; TEUKOLSKY, Saul A.; VETTERLING, William T.; FLANNERY, Brian P. **Numerical Recipes: The Art of Scientific Computing**. [S.l.]: Cambridge University Press, 2007. ISBN 0521880688, 9780521880688.

PU, Xing-Si; NGUYEN, Thanh Hai; LEE, Dong-Choon; LEE, Kyo-Beum; KIM, Jang-Mok. Fault Diagnosis of DC-Link Capacitors in Three-Phase AC/DC PWM Converters by Online Estimation of Equivalent Series Resistance. **IEEE Transactions on Industrial Electronics**, v. 60, p. 4118–4127, 9 set. 2013. ISSN 0278-0046.

PU, Xingsi; NGUYEN, Thanh-Hai; LEE, Dong-Choon; LEE, Suk-Gyu. Capacitance estimation of DC-link capacitors for single-phase PWM converters. In: p. 1656–1661.

SALMASI, Farzad Rajaei. Control Strategies for Hybrid Electric Vehicles: Evolution, Classification, Comparison, and Future Trends. **IEEE Transactions on Vehicular Technology**, v. 56, p. 2393–2404, 5 set. 2007. ISSN 0018-9545.

SELVAKUMAR, Sesha Gopal. Electric and Hybrid Vehicles – A Comprehensive Overview. In: p. 1–6.

SHOYAMA, M.; NAKA, T.; NINOMIYA, T. Resonant switched capacitor converter with high efficiency. In: p. 3780–3786.

SOLIMAN, Hammam; WANG, Huai; BLAABJERG, Frede. A Review of the Condition Monitoring of Capacitors in Power Electronic Converters. **IEEE Transactions on Industry Applications**, v. 52, p. 4976–4989, 6 nov. 2016. ISSN 0093-9994.

SPANIK, Pavol; FRIVALDSKY, Michal; KANOVSKY, Andrej. Life time of the electrolytic capacitors in power applications. In: p. 233–238.

STEWART, Joshua; NEELY, Jason; DELHOTAL, Jarod; FLICKER, Jack. DC link bus design for high frequency, high temperature converters. In: p. 809–815.

SUN, Yuting. Metallized Polypropylene Film Capacitor Condition Monitoring in MMC Based on Impedance Measurement. In: p. 69–73.

SUNDARARAJAN, Prasanth; SATHIK, Mohamed Halick Mohamed; SASONGKO, Firman; TAN, Chuan Seng; POU, Josep; BLAABJERG, Frede; GUPTA, Amit Kumar. Condition Monitoring of DC-Link Capacitors Using Goertzel Algorithm for Failure Precursor Parameter and Temperature Estimation. **IEEE Transactions on Power Electronics**, v. 35, p. 6386–6396, 6 jun. 2020. ISSN 0885-8993.

SUNDARARAJAN, Prasanth; SATHIK, Mohamed Halick Mohamed; SASONGKO, Firman; TAN, Chuan Seng; TARIQ, Mohd; SIMANJORANG, Rejeki. Online Condition Monitoring System for DC-Link Capacitor in Industrial Power Converters. **IEEE Transactions on Industry Applications**, v. 54, p. 4775–4785, 5 set. 2018. ISSN 0093-9994.

SYSEL, Petr; RAJMIC, Pavel. Goertzel algorithm generalized to non-integer multiples of fundamental frequency. **EURASIP Journal on Advances in Signal Processing**, v. 2012, p. 56, 1 dez. 2012. ISSN 1687-6180.

VENET, P; DAMAND, H; GRCLLCT, G. DETECTION OF FAULTS OF FILTER CAPACITORS IN A CONVERTER. APPLICATION T O PREDICTIVE MAINTENANCE. APPLICATION DE LA MAINTENANCE PREDICTIVE. **Proceedings of Intelec 93: 15th International Telecommunications Energy Conference**, v. 2, p. 229–234, 1993.

VENET, P.; PERISSE, F.; EL-HUSSEINI, M.H.; ROJAT, G. Realization of a smart electrolytic capacitor circuit. **IEEE Industry Applications Magazine**, v. 8, p. 16–20, 1 2002. ISSN 10772618.

WANG, Huai; BLAABJERG, Frede. Reliability of Capacitors for DC-Link Applications in Power Electronic Converters—An Overview. **IEEE Transactions on Industry Applications**, Institute of Electrical e Electronics Engineers Inc., v. 50, p. 3569–3578, 5 set. 2014. ISSN 0093-9994.

WECHSLER, Andrew; MECROW, Barrie C.; ATKINSON, David J.; BENNETT, John W.; BENAROUS, Maamar. Condition monitoring of DC-link capacitors in aerospace drives. **IEEE Transactions on Industry Applications**, v. 48, p. 1866–1874, 6 2012. ISSN 00939994.

YU, Yong; ZHOU, Tao; ZHU, Mingjun; XU, Dianguo. Fault Diagnosis and Life Prediction of DC-link Aluminum Electrolytic Capacitors Used in Three-phase AC/DC/AC Converters. In: p. 825–830.

ZHAO, Zhaoyang; DAVARI, Pooya; LU, Weiguo; WANG, Huai; BLAABJERG, Frede. An Overview of Condition Monitoring Techniques for Capacitors in DC-Link Applications. **IEEE Transactions on Power Electronics**, Institute of Electrical e Electronics Engineers Inc., v. 36, p. 3692–3716, 4 abr. 2020. ISSN 19410107.

APÊNDICE A – CÓDIGO RLS PARA O DSP F28069M

O código 'main.c' é apresentado em sequência.

```
1
2  /**********************************************************************
3  INEP-UFSC
4  ______________________________________________________________________
5
6  FILE : main.c
7  TITLE : RLS Code for ESR and Capacitance estimation
8  DESCRIPTION : This code is divided into 7 files made by the developer and
       also 14 default files that must be added.
9
10 Files made by the developer:
11 ADC.c, EPWM.c, GPIO.c, main.c, Peripheral_Setup.h, Senos_1666.h, UART
12
13 Default files:
14 F28069x_(PieCtrl.c, Adc.c, CodeStartBranch.asm, DefaultIsr.c, EPwm.c,
       GlovalVariableDefs.c, Gpio.c, Headers_nonBIOS.cmd, PieVect.c, Sci.c,
       SysCtrl.c, usDelay.asm, SYSTEM.c)
15
16 Notice that the RAM configuration file (28069_RAM_Ink.cmd) must be changed
       to distribute more memory into '.ebss' section in PAGE 1, otherwise the
       program will not run.
17
18 How a typical application works:
19 - First, the DSP needs to be on to make the PWM control the 3 phase
       inverter that the measurements will be taken. The RLS and DFT
       measurements will start immediately (it is recommended that the user
       check the arrays that storage the measured variables on CCS realtime
       plot). When everything is working as expected and the BUS voltage is
       stable, it is time to turn on ESP32. When ESP32 is ready to receive data
        it will send a flag (flagESP32DataSend) that will say that the DSP is
       allowed to send the 8 bits word (buffers that contain the ESR,
       Capacitance and BUS voltage). Then, the DSP will prepare and send the
       buffer, also blocking it to not send a buffer more than one time at once
       . The rest of the work will be made by ESP32 to decrypt and translate
       the message.
20
21 ______________________________________________________________________
22
23 DATE: 03/10/2022
24 VERSION: 1.0
25 ______________________________________________________________________
26 AUTHOR: LEONARDO A. RODRIGUES
27 ______________________________________________________________________
28
```

```
29 CONNECTED PINS:
30 ADCINA0 - Ia - P27 - ADCRESULT0 - ADCIN_0
31 ADCINB0 - Ib - P28 - ADCRESULT1 - ADCIN_1
32 ADCINA1 - Iret - P29 - ADCRESULT2 - ADCIN_2
33 ADCINB1 - Vc - P24 - ADCRESULT3 - ADCIN_3
34 ADCINA2 - Temp - P25 - ADCRESULT4 - ADCIN_4
35 ADCINB2 - Vref - P26 - ADCRESULT5 - ADCIN_5
36 DINA1 - VS2 - P7
37 DINA2 - VS4 - P8
38 DINA3 - VS6 - P9
39 DAC1 - P32
40 DAC3 - P72
41
42 PWM:
43 EPWM1A - P40
44 EPWM1B - P39
45 EPWM2A - P38
46 EPWM2B - P37
47 EPWM3A - P36
48 EPWM3B - P35
49 EPWM4A - P80
50 EPWM4B - P79
51
52 UART:
53 RX - P3
54 TX - P4
55 GPIO55 - P11 (flagESP32Reset)
56 GPIO51 - P12 (flagESP32DataSend)
57 */
58
59 //---------------------------------------------------------------------------
60 // Header Files
61 //---------------------------------------------------------------------------
62 #include "DSP28x_Project.h"
63 #include <math.h>
64 #include <Senos_1666.h>
65 #include "Peripheral_Setup.h"
66
67 //---------------------------------------------------------------------------
68 // Functions Prototypes
69 //---------------------------------------------------------------------------
70
71 // PWM Functions
72 void InitEPwm1(void); // Set the EPwm1 module
73 void InitEPwm2(void); // Set the EPwm1 module
74 void InitEPwm3(void); // Set the EPwm1 module
75 void InitEPwm4(void); // Set the EPwm1 module
```

```
76 void InitEPwm7(void); // Set the EPwm7 DAC1 module
77 void InitEPwm8(void); // Set the EPwm8 DAC2 module
78
79 // Define function to initialize SOC and ADC Interruption
80 void InitSOC(void);             // Set SOC
81 __interrupt void adc_isr(void); // EOC routine (Control routine)
82
83 void GPIO28069(void); // GPIOs configuration function
84
85 void RLS(void); // Recursive Least-Square function
86 void DAC(void); // Digital Analogic Converter function (to verify the
       Sinusoidal PWM signals on testing purposes)
87
88 // Functions to start Serial Communication w/ ESP32
89 void scia_init(void);
90 void scia_fifo_init(void);
91 void sendBufferTX(void);
92 void prepareBufferTX(void);
93
94 //---------------------------------------------------------------------
95 // Global Variables
96 //---------------------------------------------------------------------
97 int arraysSize, arraysLoaded;
98
99 // ADC, Digital and DAC pins related variables
100 unsigned int ADCIN_0, ADCIN_1, ADCIN_2, ADCIN_3[N], ADCIN_4[N], ADCIN_5; //
        ( ADCIN_0 PIN 27 / ADCIN_1 - PIN 28)
101 int DIGIN_0, DIGIN_1, DIGIN_2; // (VS2 - VS4 - VS6)
102 double DAC_write_A, DAC_write_B; // (DAC PIN 32 - 72)
103
104 // PWM Variables
105 unsigned int nPWM; // Size of PWM array
106 double nADC = 12; // Number of acquisition cicles
107
108 // Capacitor current estimation variables:
109 double Ia, Ib, Ic, Iret, Iinv, Icap[N], Vc[N];
110 double ESR, C, ESR_m, C_m;
111 int VS2, VS4, VS6;
112
113 // Temperature correction
114 double ESR_ref, ESR_fixed, C_ref, C_fixed, T_ref, T_core, Vref, A0, A1;
115
116 // Variables to send data to ESP32
117 int flag1 = 0; // Flag that allows data acquisition
118 int flagDataSent = 0; // Flag that warns the data was sent
119 int flagESP32Reset, flagESP32DataSend; // Flags received by ESP32
120 double ESR_tx = 0, C_tx = 0;
```

```
121
122 //---------------------------------------------------------------------------
123 // Main code
124 //---------------------------------------------------------------------------
125 void main(void)
126 {
127   //-------------------------------------
128   // System configuration
129   //-------------------------------------
130   InitSysCtrl();
131
132   GPIO28069();
133
134   InitEPwm1Gpio(); // ADC PWM
135   InitEPwm2Gpio(); // SPWM
136   InitEPwm3Gpio(); // SPWM
137   InitEPwm4Gpio(); // SPWM
138   InitEPwm7Gpio(); // DAC 1
139   InitEPwm8Gpio(); // DAC 2
140
141   scia_init();
142   scia_fifo_init(); // Initialize the SCI FIFO
143   InitSciaGpio();   // Init GPIO for Serial Communication
144
145   DINT;
146   InitPieCtrl();
147   IER = 0x0000;
148   IFR = 0x0000;
149   InitPieVectTable();
150
151   EALLOW; // This is needed to write to EALLOW protected registers
152   PieVectTable.ADCINT1 = &adc_isr;
153   EDIS; // This is needed to disable write to EALLOW protected registers
154
155   InitSOC(); // Init SOC function
156
157   EALLOW;
158   SysCtrlRegs.PCLKCR0.bit.TBCLKSYNC = 0;
159   EDIS;
160
161   InitEPwm1(); // PWM ADC
162   InitEPwm2(); // SPWM
163   InitEPwm3(); // SPWM
164   InitEPwm4(); // SPWM
165   InitEPwm7(); // PWM DAC 1
166   InitEPwm8(); // PWM DAC 2
167
```

```
168   EALLOW;
169   SysCtrlRegs.PCLKCR0.bit.TBCLKSYNC = 1;
170   EDIS;
171
172   IER |= M_INT1;
173
174   PieCtrlRegs.PIEIER1.bit.INTx1 = 1;
175
176   EINT; // Enable Global interrupt INTM
177   ERTM; // Enable Global realtime interrupt DBGM
178
179   // Variables
180
181   arraysSize = N;
182   nPWM = (round((double)F_ADC_PWM / 60)) - 1; // Size of PWM array
183
184   for (;;)
185   {
186     RLS();
187
188     // Receiving ESP32 flag to reset the measurements
189     EALLOW;
190     flagESP32Reset = GpioDataRegs.GPBDAT.bit.GPIO55;
191     flagESP32DataSend = GpioDataRegs.GPBDAT.bit.GPIO51;
192     EDIS;
193
194     // Sending data with UART
195
196     // esp32 needs to send the flag flagESP32DataSend on high level to this
      device to allow new data communication.
197     // flagDataSent block the DSP to not send more than 1 byte at time, so
     when the flag is 1 means that 1 byte was already sent.
198     if (flagESP32DataSend == 1 && flagDataSent == 0)
199     {
200       prepareBufferTX();
201       sendBufferTX();
202       flagDataSent = 1;
203       DELAY_US(100000); // Force a 100ms to the next buffer
204     }
205     if (flagESP32DataSend == 0)
206     {
207       flagDataSent = 0;
208     }
209   }
210 }
211
```

Lista A.1 – Arquivo "main.c".

O código 'GPIO.c' é apresentado em sequência.

```
//-----------------------------------//
// Define GPIO Pins
//-----------------------------------//
#include "DSP28x_Project.h"
void GPIO28069(void)
{
  EALLOW;
  //PWM1
  //MUX: 0 = GPIO0, 1 = EPWM1A, 2 = RESERVED, 3 = RESERVED
  GpioCtrlRegs.GPAMUX1.bit.GPIO10 = 1; // Set as EPWM1A
  GpioCtrlRegs.GPADIR.bit.GPIO10 = 1; // Set as output
  GpioCtrlRegs.GPAPUD.bit.GPIO10 = 1; // Turn off intern PULL-UP
  EDIS;

  // Input Digital Pins
  EALLOW;
  GpioCtrlRegs.GPAMUX2.bit.GPIO18 = 0;  // GPIO18
  GpioCtrlRegs.GPADIR.bit.GPIO18 = 0;   // GPIO18 = Input
  GpioCtrlRegs.GPAMUX2.bit.GPIO22 = 0;  // GPIO22
  GpioCtrlRegs.GPADIR.bit.GPIO22 = 0;   // GPIO22 = Input
  GpioCtrlRegs.GPBMUX1.bit.GPIO33 = 0;  // GPIO33
  GpioCtrlRegs.GPBDIR.bit.GPIO33 = 0;   // GPIO33 = Input

  GpioCtrlRegs.GPBMUX2.bit.GPIO55 = 0;  // GPIO55 = GPIO55
  GpioCtrlRegs.GPBDIR.bit.GPIO55 = 0;   // GPIO55 = input / PIN 11

  GpioCtrlRegs.GPBMUX2.bit.GPIO51 = 0;  // GPIO51 = GPIO51
  GpioCtrlRegs.GPBDIR.bit.GPIO51 = 0;   // GPIO51 = input / PIN 12

  GpioCtrlRegs.GPBMUX2.bit.GPIO56 = 0;  // GPIO56 = GPIO56 // PIN 51
  GpioCtrlRegs.GPBDIR.bit.GPIO56 = 1;   // GPIO56 = output
  EDIS;


} /* EOF*/


```

Lista A.2 – Arquivo "GPIO.c".

O código 'ADC.c' é apresentado em sequência.

```
// ADC File
//-----------------------------------//
#include "DSP28x_Project.h"
#include "Peripheral_Setup.h"

//-----------------------------------
```

```
7  // Define variables
8  //-------------------------------------
9
10 // Import Global Variables
11 extern unsigned int ADCIN_0, ADCIN_1, ADCIN_2, ADCIN_3[N], ADCIN_4[N],
       ADCIN_5[N];
12 extern int DIGIN_0, DIGIN_1, DIGIN_2;
13 extern double Ia, Ib, Ic, Iret, Iinv, Icap[N], Vc[N], Vref[N];
14 extern int VS2, VS4, VS6;
15 extern double DAC_write_A, DAC_write_B;
16 extern double PWM_freq;
17 double Pwm2_duty, Pwm3_duty, Pwm4_duty, ma;
18 extern unsigned int sine_a[N_SINE], sine_b[N_SINE], sine_c[N_SINE];
19 extern unsigned int N_pwm;
20 extern int slowdown_index;
21 extern int arraysSize, arraysLoaded, flagESP32Reset;
22 extern int flag1;
23
24 // Local variables
25 int Ia_view[N], Ib_view[N], Ic_view[N], Iret_view[N], Icap_view[N] = {0},
       Iinv_view[N] = {0};
26 int VS2_view[N], VS4_view[N], VS6_view[N];
27 int resetTest;
28 int i = 0, wt = 0;
29
30 int flag2 = 0; // Flag to ignore initial transient
31 int ignoreTransient = 1666; // Values to be ignored on first transient
32
33 //-------------------------------------
34 // ADC Interruption
35 //-------------------------------------
36 __interrupt void adc_isr()
37 {
38   // The resetTest can be used if ESP32 is not connected to make testing
       measurements (it can be changed on CCS debug interface)
39   if ((flagESP32Reset == 1) || (resetTest == 1))
40   {
41     flag1 = 0;
42     flag2 = 0;
43     arraysLoaded = 0;
44   }
45
46   // Flag 1 -> Turn on this routine to fulfill the ADC vectors
47   if ((flagESP32Reset == 0) || (resetTest == 0))
48   {
49     if (flag2 == ignoreTransient)
50     {
```

```
51      if (flag1 == 0)
52      {
53        EALLOW;
54        GpioDataRegs.GPBDAT.bit.GPIO56 = 0;   // Load output latch
55        EDIS;
56        // Call reading variables function to get ADC and digital variables
57        readingVariables();
58
59        // Call the current estimation function to estimate the capacitor
    current
60        currentEstimation();
61
62        // The DAC can be turned on to see variables in a external
    osciloscope
63        // DAC
64        // DAC_write_A = ADCIN_3[i] * 0.0002442;
65        // DAC_write_B = ADCIN_4[i] * 0.0002442;
66        // DAC();
67
68        // Increment index and verify if routine is ended
69
70        ++i;
71        if (i == arraysSize)
72        {
73          EALLOW;
74          GpioDataRegs.GPBDAT.bit.GPIO56 = 1;   // Load output latch
75          EDIS;
76          i = 0;
77          flag1 = 1;
78          arraysLoaded = 1;
79        }
80      }
81    }
82    else
83    {
84      flag2++;
85    }
86  }
87
88  ++wt;
89
90  // PWM
91  ma = 0.88; // Modulation index
92  Pwm2_duty = sine_a[wt] * 0.00002 * ma;
93  Pwm3_duty = sine_b[wt] * 0.00002 * ma;
94  Pwm4_duty = sine_c[wt] * 0.00002 * ma;
95
```

```
96    EALLOW;
97    EPwm2Regs.CMPA.half.CMPA = (int)(Pwm2_duty * 9000); // Pwm2_duty *
        (90000000 / (PWM_freq))
98    EPwm3Regs.CMPA.half.CMPA = (int)(Pwm3_duty * 9000); // Pwm3_duty *
        (90000000 / (PWM_freq))
99    EPwm4Regs.CMPA.half.CMPA = (int)(Pwm4_duty * 9000); // Pwm4_duty *
        (90000000 / (PWM_freq))
100   EDIS;
101
102   if (wt == (N_pwm - 1))
103   {
104     wt = 0;
105   }
106
107   AdcRegs.ADCINTFLGCLR.bit.ADCINT1 = 1;   // Clear ADCINT1 flag
        reinitialize for next SOC
108   PieCtrlRegs.PIEACK.all = PIEACK_GROUP1; // Acknowledge interrupt to PIE
109
110   return;
111 }
112
113 //---------------------------------------
114 // Functions
115 //---------------------------------------
116 readingVariables()
117 {
118   // Reading ADC registers
119   ADCIN_0 = AdcResult.ADCRESULT0;    // Ia
120   ADCIN_1 = AdcResult.ADCRESULT1;    // Ib
121   ADCIN_2 = AdcResult.ADCRESULT2;    // Iret
122   ADCIN_3[i] = AdcResult.ADCRESULT3; // Vc
123   ADCIN_4[i] = AdcResult.ADCRESULT4; // Temp
124   ADCIN_5[i] = AdcResult.ADCRESULT5; // Vref
125
126   // Digital PINs (Switching voltages) - It can be turned on if the user
        don't have access to the PWM pins.
127   EALLOW;
128   //  DIGIN_0 = GpioDataRegs.GPADAT.bit.GPIO18; // VS2 - GPIO18 - P7
129   //  DIGIN_1 = GpioDataRegs.GPADAT.bit.GPIO22; // VS4 - GPIO22 - P8
130   //  DIGIN_2 = GpioDataRegs.GPBDAT.bit.GPIO33; // VS6 - GPIO33 - P9
131
132   // Backloop for internal PWM pins - It can be turned off if the user don'
        t have access to the PWM pins.
133   DIGIN_0 = GpioDataRegs.GPADAT.bit.GPIO3; // VS2 - GPIO18 - P7
134   DIGIN_1 = GpioDataRegs.GPADAT.bit.GPIO5; // VS4 - GPIO22 - P8
135   DIGIN_2 = GpioDataRegs.GPADAT.bit.GPIO7; // VS6 - GPIO33 - P9
136   EDIS;
```

```
137 }
138
139 currentEstimation()
140 {
141   // Replacing ADC and digital readings to actual variables vectors
142   // 3.3/4095 = 8.0586e-04
143   Vref[i] = ((float)ADCIN_5[i]) * 0.0008058608;
144
145   Ia = (((float)ADCIN_0) * 0.0008058608 - Vref[i]) * 10;
146   Ib = (((float)ADCIN_1) * 0.0008058608 - Vref[i]) * 10;
147   Ic = -(Ia + Ib);
148   Ia_view[i] = (int)(Ia * 1000);
149   Ib_view[i] = (int)(Ib * 1000);
150   Ic_view[i] = (int)(Ic * 1000);
151
152   Iret = (((float)ADCIN_2) * 0.0008058608 * 10);
153
154   VS2 = DIGIN_0;
155   VS4 = DIGIN_1;
156   VS6 = DIGIN_2;
157   //  VS2_view[i] = VS2;
158   //  VS4_view[i] = VS4;
159   //  VS6_view[i] = VS6;
160
161   // Ia = -(Ic+Ib)
162   // Ib = -(Ic+Ia)
163
164   // Creating comparison table using switching states to estimate capacitor
        current
165   if (VS2 == 0 && VS4 == 0 && VS6 == 0)
166   {
167     Iinv = 0;
168   }
169   if (VS2 == 0 && VS4 == 0 && VS6 == 1)
170   {
171     Iinv = Ic;
172   }
173   if (VS2 == 0 && VS4 == 1 && VS6 == 0)
174   {
175     Iinv = Ib;
176   }
177   if (VS2 == 0 && VS4 == 1 && VS6 == 1)
178   {
179     Iinv = Ic + Ib;
180   }
181   if (VS2 == 1 && VS4 == 0 && VS6 == 0)
182   {
```

```
183     Iinv = Ia;
184   }
185   if (VS2 == 1 && VS4 == 0 && VS6 == 1)
186   {
187     Iinv = Ic + Ia;
188   }
189   if (VS2 == 1 && VS4 == 1 && VS6 == 0)
190   {
191     Iinv = -Ic;
192   }
193   if (VS2 == 1 && VS4 == 1 && VS6 == 1)
194   {
195     Iinv = 0;
196   }
197   Iinv = -Iinv;
198
199   Icap[i] = Iret - Iinv;
200
201   Icap_view[i] = (int)(Icap[i] * 1000);
202   Iret_view[i] = (int)(Iret * 1000);
203   Iinv_view[i] = (int)(Iinv * 1000);
204
205 }
206
207 DAC()
208 {
209   // Limiters to force DAC to not cross 0 or 1.
210   if (DAC_write_A > 1)
211   {
212     DAC_write_A = 1;
213   }
214   if (DAC_write_A < 0)
215   {
216     DAC_write_A = 0;
217   }
218   if (DAC_write_B > 1)
219   {
220     DAC_write_B = 1;
221   }
222   if (DAC_write_B < 0)
223   {
224     DAC_write_B = 0;
225   }
226
227   EALLOW;
228   EPwm7Regs.CMPA.half.CMPA = DAC_write_A * 900; //(90000000 / ADC_PWM_freq)
229   EPwm8Regs.CMPA.half.CMPA = DAC_write_B * 900; //(90000000 / ADC_PWM_freq)
```

```
230   EDIS;
231 }
232
233 // InitSOC
234
235 InitSOC()
236 {
237   // SOC0 (ADCINA0 - Ia - P27 - ADCRESULT0 e ADCINB0 - Ib - P28 -
       ADCRESULT1)
238   // SOC2 (ADCINA1 - Iret - P29 - ADCRESULT2 e ADCINB1 - Vc - P24 -
       ADCRESULT3)
239   // SOC4 (ADCINA2 - Temp - P25 - ADCRESULT4)
240
241   InitAdc();
242   AdcOffsetSelfCal();
243
244   EALLOW;
245   AdcRegs.ADCCTL2.bit.ADCNONOVERLAP = 1; // Enable nonoverlap mode
246
247   AdcRegs.ADCCTL1.bit.INTPULSEPOS = 1; // INT pulse generation occurs 1
       cycle prior to ADC result latching into its result register
248   AdcRegs.INTSEL1N2.bit.INT1E = 1;     // Enabled ADCINT1
249   AdcRegs.INTSEL1N2.bit.INT1CONT = 0;  // Disable ADCINT1 Continuous mode
250   AdcRegs.INTSEL1N2.bit.INT1SEL = 1;   // EOC1 is trigger for ADCINT1
251
252   // Ia, Ib, Iret, Vc, Temp;
253   // AdcRegs.ADCSAMPLEMODE.all=0x0000;
254   //  SOC0 (ADCINA0 - Ia - P27)
255   AdcRegs.ADCSAMPLEMODE.bit.SIMULEN0 = 0; // Set simultaneous sample for
       SOC0
256   AdcRegs.ADCSOC0CTL.bit.CHSEL = 0;       // Set ADCINA0 to be converted
257   AdcRegs.ADCSOC0CTL.bit.TRIGSEL = 5;     // Set EPWM1 ADCSOCA to trigger
       SOC0
258   AdcRegs.ADCSOC0CTL.bit.ACQPS = 6;       // Set SOC0 S/H Window to 7 ADC
       Clock Cycles, (6 ACQPS plus 1)
259   // SOC1 (ADCINB0 - Ib - P28)
260   AdcRegs.ADCSOC1CTL.bit.CHSEL = 8;   // Set ADCINB0 to be converted
261   AdcRegs.ADCSOC1CTL.bit.TRIGSEL = 5; // Set EPWM1 ADCSOCA to trigger SOC0
262   AdcRegs.ADCSOC1CTL.bit.ACQPS = 6;   // Set SOC0 S/H Window to 7 ADC Clock
        Cycles, (6 ACQPS plus 1)
263   // SOC2 (ADCINA1 - Iret - P29)
264   AdcRegs.ADCSAMPLEMODE.bit.SIMULEN2 = 0; // Set simultaneous sample for
       SOC1
265   AdcRegs.ADCSOC2CTL.bit.CHSEL = 1;       // Set ADCINA1 to be converted
266   AdcRegs.ADCSOC2CTL.bit.TRIGSEL = 5;     // Set EPWM1 ADCSOCA to trigger
       SOC1
267   AdcRegs.ADCSOC2CTL.bit.ACQPS = 6;       // Set SOC0 S/H Window to 7 ADC
```

```
    Clock Cycles, (6 ACQPS plus 1)
268   // SOC3 (ADCINB1 - Vc - P24)
269   AdcRegs.ADCSOC3CTL.bit.CHSEL = 9;   // Set ADCINB1 to be converted
270   AdcRegs.ADCSOC3CTL.bit.TRIGSEL = 5; // Set EPWM1 ADCSOCA to trigger SOC1
271   AdcRegs.ADCSOC3CTL.bit.ACQPS = 6;   // Set SOC0 S/H Window to 7 ADC Clock
        Cycles, (6 ACQPS plus 1)
272   // SOC4 (ADCINA2 - Temp - P25)
273   AdcRegs.ADCSAMPLEMODE.bit.SIMULEN4 = 0; // Set simultaneous sample for
        SOC1 and SOC1
274   AdcRegs.ADCSOC4CTL.bit.CHSEL = 2;       // Set ADCINA1 and ADCINB1 to be
        converted
275   AdcRegs.ADCSOC4CTL.bit.TRIGSEL = 5;     // Set EPWM1 ADCSOCA to trigger
        SOC1
276   AdcRegs.ADCSOC4CTL.bit.ACQPS = 6;       // Set SOC0 S/H Window to 7 ADC
        Clock Cycles, (6 ACQPS plus 1)
277   // SOC5 (ADCINB2 - Vref - P26)
278   AdcRegs.ADCSOC5CTL.bit.CHSEL = 0xA; // Set ADCINA1 and ADCINB1 to be
        converted
279   AdcRegs.ADCSOC5CTL.bit.TRIGSEL = 5; // Set EPWM1 ADCSOCA to trigger SOC1
280   AdcRegs.ADCSOC5CTL.bit.ACQPS = 6;   // Set SOC0 S/H Window to 7 ADC Clock
        Cycles, (6 ACQPS plus 1)
281   EDIS;
282
283   // Assumes EPWM1 clock is already enabled in InitSysCtrl();
284   EPwm1Regs.ETSEL.bit.SOCAEN = 1;  // Enable SOC on A group
285   EPwm1Regs.ETSEL.bit.SOCASEL = 2; // Set SOC to occur when TBCTR = TBPRD
286   EPwm1Regs.ETPS.bit.SOCAPRD = 1;  // Generate pulse on 1st event
287 }
288
```

Lista A.3 – Arquivo "ADC.c".

O código 'EPWM.c' é apresentado em sequência.

```
1  //-----------------------------------
2  // EPWM Configuration
3  //-----------------------------------
4  #include "DSP28x_Project.h"
5  #include "Peripheral_Setup.h"
6
7  void InitEPwm1() // ADC Module
8  {
9    EPwm1Regs.TBPRD = (90000000 / (double)F_ADC_PWM);                // Set
       the frequency
10   EPwm1Regs.CMPA.half.CMPA = (90000000 / (2 * (double)F_ADC_PWM)); // Set
       50% fixed duty for EPWM1A
11   EPwm1Regs.TBPHS.half.TBPHS = 0;                                  // Set
       Phase register to zero
```

```
12   EPwm1Regs.TBCTL.bit.CTRMODE = TB_COUNT_UP;                          //
      Asymmetrical mode
13   EPwm1Regs.TBCTL.bit.PHSEN = TB_DISABLE;                             //
      Master module
14   EPwm1Regs.TBCTL.bit.PRDLD = TB_SHADOW;
15   EPwm1Regs.TBCTL.bit.SYNCOSEL = TB_CTR_ZERO; // Sync downstream module
16   EPwm1Regs.TBCTL.bit.CLKDIV = TB_DIV1;       // Set timebase clock
      prescale to one
17   EPwm1Regs.TBCTL.bit.HSPCLKDIV = TB_DIV1;    // Set timebase clock
      prescale to one
18   EPwm1Regs.CMPCTL.bit.SHDWAMODE = CC_SHADOW;
19   EPwm1Regs.CMPCTL.bit.SHDWBMODE = CC_SHADOW;
20   EPwm1Regs.CMPCTL.bit.LOADAMODE = CC_CTR_ZERO; // load on CTR=Zero
21   EPwm1Regs.CMPCTL.bit.LOADBMODE = CC_CTR_ZERO; // load on CTR=Zero
22   EPwm1Regs.AQCTLA.bit.ZRO = AQ_SET;            // set actions for EPWM1A
23   EPwm1Regs.AQCTLA.bit.CAU = AQ_CLEAR;
24   EPwm1Regs.DBCTL.bit.OUT_MODE = DB_FULL_ENABLE; // enable Deadband module
25   EPwm1Regs.DBCTL.bit.POLSEL = DB_ACTV_HIC;      // Active Hi complementary
26   EPwm1Regs.DBFED = 32;                          // FED = 32 TBCLKs
27   EPwm1Regs.DBRED = 32;                          // RED = 32 TBCLKs
28 }
29
30 //
31 // InitEPwm2
32 //
33 void InitEPwm2() // Sine_a - P38
34 {
35   EPwm2Regs.TBPRD = (90000000 / (2 * (double)F_PWM)); // Set the frequency
36   EPwm2Regs.TBCTR = 0x0000;
37   EPwm2Regs.CMPA.half.CMPA = 4500;               // Set 50% fixed duty
      EPWM2A
38   EPwm2Regs.TBPHS.half.TBPHS = 0;                // Set Phase register to
      zero initially
39   EPwm2Regs.TBCTL.bit.CTRMODE = TB_COUNT_UPDOWN; // Asymmetrical mode
40   EPwm2Regs.TBCTL.bit.PHSEN = TB_DISABLE;        // Slave module
41   EPwm2Regs.TBCTL.bit.PRDLD = TB_SHADOW;
42   EPwm2Regs.TBCTL.bit.SYNCOSEL = TB_DISABLE; // sync flowthrough
43   EPwm2Regs.TBCTL.bit.CLKDIV = TB_DIV1;      // Set timebase clock prescale
       to one
44   EPwm2Regs.TBCTL.bit.HSPCLKDIV = TB_DIV1;   // Set timebase clock prescale
       to one
45   EPwm2Regs.CMPCTL.bit.SHDWAMODE = CC_SHADOW;
46   EPwm2Regs.CMPCTL.bit.SHDWBMODE = CC_SHADOW;
47   EPwm2Regs.CMPCTL.bit.LOADAMODE = 2;      // load on CTR=Zero
48   EPwm2Regs.CMPCTL.bit.LOADBMODE = 2;      // load on CTR=Zero
49   EPwm2Regs.AQCTLA.bit.ZRO = AQ_NO_ACTION; // set actions for EPWM2A
50   EPwm2Regs.AQCTLA.bit.PRD = AQ_NO_ACTION;
```

```
51   EPwm2Regs.AQCTLA.bit.CAU = AQ_CLEAR;
52   EPwm2Regs.AQCTLA.bit.CAD = AQ_SET;
53   EPwm2Regs.DBCTL.bit.OUT_MODE = DB_FULL_ENABLE; // enable Deadband module
54   EPwm2Regs.DBCTL.bit.POLSEL = DB_ACTV_HIC;      // Active Hi complementary
55   EPwm2Regs.DBFED = 32;                          // FED = 32 TBCLKs
56   EPwm2Regs.DBRED = 32;                          // RED = 32 TBCLKs
57 }
58
59 //
60 // InitEPwm3
61 //
62 void InitEPwm3() // Sine_b - P36
63 {
64   EPwm3Regs.TBPRD = (90000000 / (2 * (double)F_PWM)); // Set the frequency
65   EPwm3Regs.TBCTR = 0x0000;
66   EPwm3Regs.CMPA.half.CMPA = 0.5 * (90000000 / (double)F_PWM); // Set 50%
       fixed duty EPWM2A
67   EPwm3Regs.TBPHS.half.TBPHS = 0;                              // Set Phase
       register to zero initially
68   EPwm3Regs.TBCTL.bit.CTRMODE = TB_COUNT_UPDOWN;               //
       Asymmetrical mode
69   EPwm3Regs.TBCTL.bit.PHSEN = TB_DISABLE;                      // Slave
       module
70   EPwm3Regs.TBCTL.bit.PRDLD = TB_SHADOW;
71   EPwm3Regs.TBCTL.bit.SYNCOSEL = TB_DISABLE; // sync flowthrough
72   EPwm3Regs.TBCTL.bit.CLKDIV = TB_DIV1;      // Set timebase clock prescale
       to one
73   EPwm3Regs.TBCTL.bit.HSPCLKDIV = TB_DIV1;   // Set timebase clock prescale
       to one
74   EPwm3Regs.CMPCTL.bit.SHDWAMODE = CC_SHADOW;
75   EPwm3Regs.CMPCTL.bit.SHDWBMODE = CC_SHADOW;
76   EPwm3Regs.CMPCTL.bit.LOADAMODE = 2;      // load on CTR=Zero
77   EPwm3Regs.CMPCTL.bit.LOADBMODE = 2;      // load on CTR=Zero
78   EPwm3Regs.AQCTLA.bit.ZRO = AQ_NO_ACTION; // set actions for EPWM2A
79   EPwm3Regs.AQCTLA.bit.PRD = AQ_NO_ACTION;
80   EPwm3Regs.AQCTLA.bit.CAU = AQ_CLEAR;
81   EPwm3Regs.AQCTLA.bit.CAD = AQ_SET;
82   EPwm3Regs.DBCTL.bit.OUT_MODE = DB_FULL_ENABLE; // enable Deadband module
83   EPwm3Regs.DBCTL.bit.POLSEL = DB_ACTV_HIC;      // Active Hi complementary
84   EPwm3Regs.DBFED = 32;                          // FED = 32 TBCLKs
85   EPwm3Regs.DBRED = 32;                          // RED = 32 TBCLKs
86 }
87
88 //
89 // InitEPwm4
90 //
91 void InitEPwm4() // Sine_c P80
```

```
92 {
93   EPwm4Regs.TBPRD = (90000000 / (2 * (double)F_PWM)); // Set the frequency
94   EPwm4Regs.TBCTR = 0x0000;
95   EPwm4Regs.CMPA.half.CMPA = 0.5 * (90000000 / (double)F_PWM); // Set 50%
      fixed duty EPWM2A
96   EPwm4Regs.TBPHS.half.TBPHS = 0;                              // Set Phase
       register to zero initially
97   EPwm4Regs.TBCTL.bit.CTRMODE = TB_COUNT_UPDOWN;               //
      Asymmetrical mode
98   EPwm4Regs.TBCTL.bit.PHSEN = TB_DISABLE;                      // Slave
      module
99   EPwm4Regs.TBCTL.bit.PRDLD = TB_SHADOW;
100  EPwm4Regs.TBCTL.bit.SYNCOSEL = TB_DISABLE; // sync flowthrough
101  EPwm4Regs.TBCTL.bit.CLKDIV = TB_DIV1;      // Set timebase clock prescale
       to one
102  EPwm4Regs.TBCTL.bit.HSPCLKDIV = TB_DIV1;   // Set timebase clock prescale
       to one
103  EPwm4Regs.CMPCTL.bit.SHDWAMODE = CC_SHADOW;
104  EPwm4Regs.CMPCTL.bit.SHDWBMODE = CC_SHADOW;
105  EPwm4Regs.CMPCTL.bit.LOADAMODE = 2;      // load on CTR=Zero
106  EPwm4Regs.CMPCTL.bit.LOADBMODE = 2;      // load on CTR=Zero
107  EPwm4Regs.AQCTLA.bit.ZRO = AQ_NO_ACTION; // set actions for EPWM2A
108  EPwm4Regs.AQCTLA.bit.PRD = AQ_NO_ACTION;
109  EPwm4Regs.AQCTLA.bit.CAU = AQ_CLEAR;
110  EPwm4Regs.AQCTLA.bit.CAD = AQ_SET;
111  EPwm4Regs.DBCTL.bit.OUT_MODE = DB_FULL_ENABLE; // enable Deadband module
112  EPwm4Regs.DBCTL.bit.POLSEL = DB_ACTV_HIC;      // Active Hi complementary
113  EPwm4Regs.DBFED = 32;                          // FED = 32 TBCLKs
114  EPwm4Regs.DBRED = 32;                          // RED = 32 TBCLKs
115 }
116
117 //
118 // InitEPwm7 DAC1
119 //
120 void InitEPwm7()
121 {
122   EPwm7Regs.TBPRD = (90000000 / (double)F_ADC_PWM);               // Set
      the frequency
123   EPwm7Regs.CMPA.half.CMPA = (90000000 / (2 * (double)F_ADC_PWM)); // Set
      50% fixed duty EPWM2A
124   EPwm7Regs.TBPHS.half.TBPHS = 0;                                 // Set
      Phase register to zero initially
125   EPwm7Regs.TBCTL.bit.CTRMODE = TB_COUNT_UP;                      //
      Asymmetrical mode
126   EPwm7Regs.TBCTL.bit.PHSEN = TB_ENABLE;                          // Slave
       module
127   EPwm7Regs.TBCTL.bit.PRDLD = TB_SHADOW;
```

```
128   EPwm7Regs.TBCTL.bit.SYNCOSEL = TB_SYNC_IN; // sync flowthrough
129   EPwm7Regs.TBCTL.bit.CLKDIV = TB_DIV1;      // Set timebase clock prescale
        to one
130   EPwm7Regs.TBCTL.bit.HSPCLKDIV = TB_DIV1;   // Set timebase clock prescale
        to one
131   EPwm7Regs.CMPCTL.bit.SHDWAMODE = CC_SHADOW;
132   EPwm7Regs.CMPCTL.bit.SHDWBMODE = CC_SHADOW;
133   EPwm7Regs.CMPCTL.bit.LOADAMODE = CC_CTR_ZERO; // load on CTR=Zero
134   EPwm7Regs.CMPCTL.bit.LOADBMODE = CC_CTR_ZERO; // load on CTR=Zero
135   EPwm7Regs.AQCTLA.bit.ZRO = AQ_SET;            // set actions for EPWM2A
136   EPwm7Regs.AQCTLA.bit.CAU = AQ_CLEAR;
137   EPwm7Regs.DBCTL.bit.OUT_MODE = DB_FULL_ENABLE; // enable Deadband module
138   EPwm7Regs.DBCTL.bit.POLSEL = DB_ACTV_HIC;      // Active Hi complementary
139   EPwm7Regs.DBFED = 32;                          // FED = 32 TBCLKs
140   EPwm7Regs.DBRED = 32;                          // RED = 32 TBCLKs
141 }
142
143 //
144 // InitEPwm8 DAC3
145 //
146 void InitEPwm8()
147 {
148   EPwm8Regs.TBPRD = (90000000 / (double)F_ADC_PWM);                // Set
        the frequency
149   EPwm8Regs.CMPA.half.CMPA = (90000000 / (2 * (double)F_ADC_PWM)); // Set
        50% fixed duty EPWM2A
150   EPwm8Regs.TBPHS.half.TBPHS = 0;                                  // Set
        Phase register to zero initially
151   EPwm8Regs.TBCTL.bit.CTRMODE = TB_COUNT_UP;                       //
        Asymmetrical mode
152   EPwm8Regs.TBCTL.bit.PHSEN = TB_ENABLE;                           // Slave
        module
153   EPwm8Regs.TBCTL.bit.PRDLD = TB_SHADOW;
154   EPwm8Regs.TBCTL.bit.SYNCOSEL = TB_SYNC_IN; // sync flowthrough
155   EPwm8Regs.TBCTL.bit.CLKDIV = TB_DIV1;      // Set timebase clock prescale
        to one
156   EPwm8Regs.TBCTL.bit.HSPCLKDIV = TB_DIV1;   // Set timebase clock prescale
        to one
157   EPwm8Regs.CMPCTL.bit.SHDWAMODE = CC_SHADOW;
158   EPwm8Regs.CMPCTL.bit.SHDWBMODE = CC_SHADOW;
159   EPwm8Regs.CMPCTL.bit.LOADAMODE = CC_CTR_ZERO; // load on CTR=Zero
160   EPwm8Regs.CMPCTL.bit.LOADBMODE = CC_CTR_ZERO; // load on CTR=Zero
161   EPwm8Regs.AQCTLA.bit.ZRO = AQ_SET;            // set actions for EPWM2A
162   EPwm8Regs.AQCTLA.bit.CAU = AQ_CLEAR;
163   EPwm8Regs.DBCTL.bit.OUT_MODE = DB_FULL_ENABLE; // enable Deadband module
164   EPwm8Regs.DBCTL.bit.POLSEL = DB_ACTV_HIC;      // Active Hi complementary
165   EPwm8Regs.DBFED = 32;                          // FED = 32 TBCLKs
```

```
166   EPwm8Regs.DBRED = 32;                          // RED = 32 TBCLKs
167 }
```

Lista A.4 – Arquivo "EPWM.c".

O código 'RLS.c' é apresentado em sequência.

```
1  //-------------------------------------//
2  // RLS Method Code
3  //-------------------------------------//
4  #include "DSP28x_Project.h"
5  #include "Peripheral_Setup.h"
6  #include <math.h>
7
8  //-------------------------------------
9  // RLS Variables
10 //-------------------------------------
11
12 // Importing Global Variables
13 extern double Icap[N], Vc[N];
14 extern unsigned int ADCIN_0, ADCIN_1, ADCIN_2, ADCIN_3[N], ADCIN_4[N],
       ADCIN_5;
15 extern double DAC_write_A, DAC_write_B;
16 extern double ESR_ref, ESR_fixed, C_ref, C_fixed, T_ref, T_core, Vref, A0,
       A1;
17 extern unsigned char buffer_tx[];
18 extern int flag1;
19 extern int arraysSize, arraysLoaded;
20 extern double y;
21 extern double ESR, C, ESR_m, C_m;
22
23 // Local Variables
24 int j;
25 double denK, b0, b1, u, up, yp;
26 double lambda = 1, Ts;
27 double P11, P12, P13, P21, P22, P23, P31, P32, P33;
28 double K11, K21, K31;
29 double Ts;
30 double teta11, teta21, teta31;
31 double teste;
32 double teste2;
33
34 void RLS()
35 {
36   if (arraysLoaded == 0)
37   {
38     // Init/Reset variables
39     Ts = 1 / ((double)F_ADC_PWM);
40     P11 = 6.561175216846932e-09;
```

```
41      P12 = 2.433906386362213e-09;
42      P13 = 3.998812655022957e-09;
43      P21 = 2.767056198250937e-09;
44      P22 = 3.271921453778275e-05;
45      P23 = -3.108184221339390e-05;
46      P31 = 3.063515225368884e-09;
47      P32 = -3.120423905395961e-05;
48      P33 = 3.280822195908541e-05;
49      teta11 = -1;
50      teta21 = 0.0123;
51      teta31 = -0.0077;
52      K11 = -2.584084709401899e-06;
53      K21 = 9.076320218062094e-06;
54      K31 = 8.750101616645840e-06;
55      ESR = 10e-3;
56      C = 2.2e-3;
57      y = 400;
58      u = 6.22;
59      yp = 400;
60      up = 6.22;
61      j = 0;
62    }
63
64    if (flag1 == 1)
65    {
66      if (arraysLoaded == 1)
67      {
68        y = (((double)ADCIN_3[j]) * 8.0586e-04) * 147.667; // Vc
69        u = Icap[j]; // Icap
70
71        // K Matrix:
72        denK = lambda + u * (P22 * u + P32 * up - P12 * yp) + up * (P23 * u +
     P33 * up - P13 * yp) - yp * (P21 * u + P31 * up - P11 * yp);
73
74        if (denK != 0)
75        {
76          K11 = (P12 * u + P13 * up - P11 * yp) / denK;
77          K21 = (P22 * u + P23 * up - P21 * yp) / denK;
78          K31 = (P32 * u + P33 * up - P31 * yp) / denK;
79        }
80
81        // Theta Matrix:
82        teta11 = teta11 + K11 * (y - teta21 * u - teta31 * up + teta11 * yp);
83        teta21 = teta21 + K21 * (y - teta21 * u - teta31 * up + teta11 * yp);
84        teta31 = teta31 + K31 * (y - teta21 * u - teta31 * up + teta11 * yp);
85
86        // P Matrix:
```

```
87      P11 = -((K11 * P21 * u - P11 * (K11 * yp + 1) + K11 * P31 * up) / (
    lambda));
88      P12 = -((K11 * P22 * u - P12 * (K11 * yp + 1) + K11 * P32 * up) / (
    lambda));
89      P13 = -((K11 * P23 * u - P13 * (K11 * yp + 1) + K11 * P33 * up) / (
    lambda));
90      P21 = -((P21 * (K21 * u - 1) - K21 * P11 * yp + K21 * P31 * up) / (
    lambda));
91      P22 = -((P22 * (K21 * u - 1) - K21 * P12 * yp + K21 * P32 * up) / (
    lambda));
92      P23 = -((P23 * (K21 * u - 1) - K21 * P13 * yp + K21 * P33 * up) / (
    lambda));
93      P31 = -((P31 * (K31 * up - 1) - K31 * P11 * yp + K31 * P21 * u) / (
    lambda));
94      P32 = -((P32 * (K31 * up - 1) - K31 * P12 * yp + K31 * P22 * u) / (
    lambda));
95      P33 = -((P33 * (K31 * up - 1) - K31 * P13 * yp + K31 * P23 * u) / (
    lambda));
96
97      // b0 and b1:
98      b0 = teta21;
99      b1 = teta31;
100
101     if (b0 > 1)
102     {
103       b0 = 1;
104     }
105     if (b0 < -1)
106     {
107       b0 = -1;
108     }
109     if (b1 > 1)
110     {
111       b1 = 1;
112     }
113     if (b1 < -1)
114     {
115       b1 = -1;
116     }
117
118     // Calculating C and ESR:
119     // Condition to calculate C and ESR:
120     if ((b1 + b0) != 0)
121     {
122       //          ESR = 10e-3; // You can define this to make initial
    tests with UART
123       //          C = 2.2e-3;
```

```
124         C = Ts / (b1 + b0);
125         ESR = b0 - Ts / (2 * C);
126 
127         // Limits to not saturate:
128         if (C > 10e-3) C = 10e-3;
129         if (C < 0) C = -C;
130         if (ESR > 1) ESR = 1;
131         if (ESR < 0) ESR = -ESR;
132 
133         C_m = 1000*C;
134         ESR_m = 1000*ESR;
135       }
136 
137       // ESR Temperature Correction (in this case temperature sensors was
    not used):
138       // ESR_ref = ESR_inicial; // Reference ESR
139       // T_ref = 25; // Reference Temperature
140       // A0 = 21; // Experimental Value
141       // ESR_fixed = ESR + abs(ESR_ref - ESR_ref * exp((T_ref - T_core) /
    A0));
142       // // Fixing Capacitance
143       // C_ref = C_init; // Reference Capacitance
144       // A1 = 5e-7; // Experimental Value
145       // C_fixed = C - abs(C_ref - C_ref + A1 * (T_core - T_ref));
146 
147       // Viewing ESR and C on DAC:
148       // DAC_write_A = C*500;
149       // DAC_write_B = ESR*5;
150       // DAC();
151 
152       // Updating yp e up
153       yp = y; // Ypast
154       up = u; // Upast
155 
156       j++;
157       if (j == N && arraysLoaded == 1)
158       {
159         teste = j;
160         j = 0;
161       }
162     }
163   }
164 }
```

Lista A.5 – Arquivo "RLS.c"

```
1 //-------------------------------------//
2 // UART Configuration File
```

```
3  //-------------------------------------//
4  #include "DSP28x_Project.h"
5
6  // Import extern variables
7  extern double ESR_tx, C_tx;
8  extern double ESR, C, y;
9
10 // Locale variables
11 int bufferSize = 8;
12 int ESR_fact_conv = 1000, conv_factor = 100;
13 unsigned char buffer_tx[] = {'a', 'b', 'c', 'd', 'e', 'f', '\n', '\r'};
14
15 // SCI Functions
16 void scia_init()
17 {
18   //
19   // Note: Clocks were turned on to the SCIA peripheral
20   // in the InitSysCtrl() function
21   //
22
23   //
24   // 1 stop bit,  No loopback, No parity,8 char bits, async mode,
25   // idle-line protocol
26   //
27   SciaRegs.SCICCR.all = 0x0007; // Communications Control Register
28   // bit 7: Determines the number of stop bits transmitted one stop bit if
       cleared to 0 or two stop bits if set to 1.
29
30   //
31   // enable TX, RX, internal SCICLK, Disable RX ERR, SLEEP, TXWAKE
32   //
33   SciaRegs.SCICTL1.all = 0x0003;
34   // The SCI port first sets the TXWAKE bit (SCICTL1, bit 3) to 1 before
       writing to the  SCITXBUF register.
35   // This sends an idle time of exactly 11 bits. In this method, the serial
         communications line is not idle any longer than necessary.
36
37   SciaRegs.SCICTL2.bit.TXINTENA = 0;
38   SciaRegs.SCICTL2.bit.RXBKINTENA = 0;
39
40   //
41   // 9600 baud @LSPCLK = 22.5MHz (90 MHz SYSCLK)
42   //
43
44   SciaRegs.SCIHBAUD = 0x0001; // Baud rate (high) register
45   SciaRegs.SCILBAUD = 0x0024; // Baud rate (low) register
46   // SCI Asynchronous Baud = LSPCLK / ((BRR + 1) *8)
```

```
47    SciaRegs.SCICCR.bit.LOOPBKENA = 0; // Enable loop back
48    SciaRegs.SCICTL1.all = 0x0023;     // Relinquish SCI from Reset
49  }
50
51  void prepareBufferTX(void)
52  {
53    ESR_tx = ESR * ESR_fact_conv;
54    C_tx = C * ESR_fact_conv * 100;
55    buffer_tx[0] = ((((int)(ESR_tx * conv_factor)) & 0xFF00) >> 8);
56    buffer_tx[1] = (((int)(ESR_tx * conv_factor)) & 0x00FF);
57    buffer_tx[2] = ((((int)(C_tx * conv_factor)) & 0xFF00) >> 8);
58    buffer_tx[3] = (((int)(C_tx * conv_factor)) & 0x00FF);
59    buffer_tx[4] = ((((int)(y)) & 0xFF00) >> 8);
60    buffer_tx[5] = (((int)(y)) & 0x00FF);
61  }
62
63  // Function that will run over the buffer array and send data via TX Pin
64  void sendBufferTX(void)
65  {
66    unsigned char i;
67    for (i = 0; i < bufferSize; i++)
68    {
69      while (!SciaRegs.SCICTL2.bit.TXRDY)
70      {
71      }
72      SciaRegs.SCITXBUF = buffer_tx[i];
73    }
74  }
75
76
77  // scia_fifo_init - Initalize the SCI FIFO
78  void scia_fifo_init()
79  {
80    SciaRegs.SCIFFTX.all = 0xE040;
81    SciaRegs.SCIFFRX.all = 0x2044;
82    SciaRegs.SCIFFCT.all = 0x0;
83  }
84
85  //
86  // End of File
87  //
88
```

Lista A.6 – Arquivo "UART.c"

O arquivo "Peripheral_Setup.h"é apresentado a seguir.

```
1  /*
2  * Peripheral_Setup.h
```

```
3  *
4  *  Created on: jun 21 2022
5  *  Author: Leonardo Rodrigues
6  */
7
8  #ifndef PERIPHERAL_SETUP_H_
9  #define PERIPHERAL_SETUP_H_
10 #include "DSP28x_Project.h"
11
12 #define N 1667
13 #define F_BASE 360
14 #define F_PWM 5000
15 #define F_ADC_PWM 100000
16 #define N_SINE 1667
17
18 // sample Time N*freq base
19
20 #endif /* PERIPHERAL_SETUP_H_ */
21
```

Lista A.7 – Arquivo "Peripheral_Setup.h"

A modificação do arquivo padrão "28069_RAM_lnk.cmd"é apresentada a seguir.

```
1   /*
2   // TI File $Revision: /main/3 $
3   // Checkin $Date: March 3, 2011   13:45:43 $
4   //###########################################################################
5   //
6   // FILE:    28069_RAM_lnk.cmd
7   //
8   // TITLE:   Linker Command File For F28069 examples that run out of RAM
9   //
10  //          This ONLY includes all SARAM blocks on the F28069 device.
11  //          This does not include flash or OTP.
12  //
13  //          Keep in mind that L0,L1,L2,L3 and L4 are protected by the
      code
14  //          security module.
15  //
16  //          What this means is in most cases you will want to move to
17  //          another memory map file which has more memory defined.
18  //
19  //###########################################################################
20  // $TI Release: $
21  // $Release Date: $
22  //###########################################################################
23  */
24
```

```
25  /* ======================================================
26  // For Code Composer Studio V2.2 and later
27  // ---------------------------------------
28  // In addition to this memory linker command file,
29  // add the header linker command file directly to the project.
30  // The header linker command file is required to link the
31  // peripheral structures to the proper locations within
32  // the memory map.
33  //
34  // The header linker files are found in <base>\F2806x_headers\cmd
35  //
36  // For BIOS applications add:      F2806x_Headers_BIOS.cmd
37  // For nonBIOS applications add:   F2806x_Headers_nonBIOS.cmd
38  ======================================================== */
39
40  /* ======================================================
41  // For Code Composer Studio prior to V2.2
42  // --------------------------------------
43  // 1) Use one of the following -l statements to include the
44  // header linker command file in the project. The header linker
45  // file is required to link the peripheral structures to the proper
46  // locations within the memory map                                  */
47
48  /* Uncomment this line to include file only for non-BIOS applications */
49  /* -l F2806x_Headers_nonBIOS.cmd */
50
51  /* Uncomment this line to include file only for BIOS applications */
52  /* -l F2806x_Headers_BIOS.cmd */
53
54  /* 2) In your project add the path to <base>\F2806x_headers\cmd to the
55  library search path under project->build options, linker tab,
56  library search path (-i).
57  /*========================================================== */
58
59  /* Define the memory block start/length for the F2806x
60  PAGE 0 will be used to organize program sections
61  PAGE 1 will be used to organize data sections
62
63  Notes:
64  Memory blocks on F28069 are uniform (ie same
65  physical memory) in both PAGE 0 and PAGE 1.
66  That is the same memory region should not be
67  defined for both PAGE 0 and PAGE 1.
68  Doing so will result in corruption of program
69  and/or data.
70
71  Contiguous SARAM memory blocks can be combined
```

```
72  if required to create a larger memory block.
73  */
74
75  MEMORY
76  {
77    PAGE 0 :
78    /* BEGIN is used for the "boot to SARAM" bootloader mode   */
79
80    BEGIN       : origin = 0x000000, length = 0x000002
81    RAMM0       : origin = 0x000050, length = 0x0003B0
82    RAML0_L3    : origin = 0x008000, length = 0x002000   /* RAML0-3
     combined for size of .text */
83    /* in Example_F2806xSWPrioritezedInterrupts */
84    RESET       : origin = 0x3FFFC0, length = 0x000002
85    FPUTABLES   : origin = 0x3FD860, length = 0x0006A0   /* FPU Tables in
     Boot ROM */
86    IQTABLES    : origin = 0x3FDF00, length = 0x000B50    /* IQ Math Tables
      in Boot ROM */
87    IQTABLES2   : origin = 0x3FEA50, length = 0x00008C    /* IQ Math Tables
      in Boot ROM */
88    IQTABLES3   : origin = 0x3FEADC, length = 0x0000AA   /* IQ Math Tables
     in Boot ROM */
89
90    BOOTROM    : origin = 0x3FF3B0, length = 0x000C10
91
92
93    PAGE 1 :
94
95    BOOT_RSVD   : origin = 0x000002, length = 0x00004E     /* Part of M0,
     BOOT rom will use this for stack */
96    RAMM1       : origin = 0x000400, length = 0x000400     /* on-chip RAM
     block M1 */
97    RAML4       : origin = 0x00A000, length = 0x002000     /* on-chip RAM
     block L4 */
98    RAML5       : origin = 0x00C000, length = 0x002000     /* on-chip RAM
     block L5 */
99    RAML6       : origin = 0x00E000, length = 0x002000     /* on-chip RAM
     block L6 */
100   RAML7       : origin = 0x010000, length = 0x002000     /* on-chip RAM
     block L7 */
101   RAML8       : origin = 0x012000, length = 0x002000     /* on-chip RAM
     block L8 */
102   USB_RAM     : origin = 0x040000, length = 0x000800     /* USB RAM
     */
103 }
104
105
```

```
106  SECTIONS
107  {
108    /* Setup for "boot to SARAM" mode:
109    The codestart section (found in DSP28_CodeStartBranch.asm)
110    re-directs execution to the start of user code.  */
111    codestart        : > BEGIN,      PAGE = 0
112
113    #ifdef __TI_COMPILER_VERSION__
114    #if __TI_COMPILER_VERSION__ >= 15009000
115    .TI.ramfunc : {} > RAMM0,      PAGE = 0
116    #else
117    ramfuncs         : > RAMM0,      PAGE = 0
118    #endif
119    #endif
120
121    .text            : > RAML0_L3,   PAGE = 0
122    .cinit           : > RAML4,      PAGE = 1
123    .pinit           : > RAMM0,      PAGE = 0
124    .switch          : > RAMM0,      PAGE = 0
125    .reset           : > RESET,      PAGE = 0, TYPE = DSECT /* not used, */
126
127    .stack           : > RAMM1,      PAGE = 1
128    .ebss            : >> RAML4 | RAML5 | RAML6 | RAML7 | RAML8,      PAGE
     = 1
129    .econst          : > RAML4,      PAGE = 1
130    .esysmem         : > RAML4,      PAGE = 1
131
132    IQmath           : > RAML0_L3,   PAGE = 0
133    IQmathTables     : > IQTABLES,   PAGE = 0, TYPE = NOLOAD
134
135    /* Allocate FPU math areas: */
136    FPUmathTables    : > FPUTABLES,  PAGE = 0, TYPE = NOLOAD
137
138    //   DMARAML5         : > RAML5,      PAGE = 1
139    //   DMARAML6         : > RAML6,      PAGE = 1
140    //   DMARAML7         : > RAML7,      PAGE = 1
141    //   DMARAML8         : > RAML8,      PAGE = 1
142
143    /* Uncomment the section below if calling the IQNexp() or IQexp()
144    functions from the IQMath.lib library in order to utilize the
145    relevant IQ Math table in Boot ROM (This saves space and Boot ROM
146    is 1 wait-state). If this section is not uncommented, IQmathTables2
147    will be loaded into other memory (SARAM, Flash, etc.) and will take
148    up space, but 0 wait-state is possible.
149    */
150    /*
151    IQmathTables2    : > IQTABLES2, PAGE = 0, TYPE = NOLOAD
```

```
152     {
153
154       IQmath.lib<IQNexpTable.obj> (IQmathTablesRam)
155
156     }
157     */
158     /* Uncomment the section below if calling the IQNasin() or IQasin()
159     functions from the IQMath.lib library in order to utilize the
160     relevant IQ Math table in Boot ROM (This saves space and Boot ROM
161     is 1 wait-state). If this section is not uncommented, IQmathTables2
162     will be loaded into other memory (SARAM, Flash, etc.) and will take
163     up space, but 0 wait-state is possible.
164     */
165     /*
166     IQmathTables3    : > IQTABLES3, PAGE = 0, TYPE = NOLOAD
167     {
168
169       IQmath.lib<IQNasinTable.obj> (IQmathTablesRam)
170
171     }
172     */
173
174   }
175
176   /*
177   //===========================================================================
178   // End of file.
179   //===========================================================================
180   */
181
```

Lista A.8 – Arquivo "28069_RAM_lnk.cmd"

APÊNDICE B – CÓDIGO DFT PARA O DSP F28069M

As partes externas ao código DFT são muito semelhantes ao código RLS portanto não serão apresentadas aqui. Em resumo, no arquivo "main.c" o usuário deve substituir a chamada da função "void RLS(void)" por "void DFT(void)". No arquivo "ADC.c" deve ser recalculado a constante de configuração dos registradores "EPwm2Regs.CMPA.half.CMPA", "EPwm3Regs.CMPA.half.CMPA" e "EPwm4Regs.CMPA.half.CMPA" para a nova frequência de 4608 Hz. Ou seja, substitui-se 9000 por 9765,625. Altera-se também a constante dos registradores "EPwm7Regs.CMPA.half.CMPA" e "EPwm8Regs.CMPA.half.CMPA" de 900 para 976,563 por conta da alteração da frequência de amostragem do ADC de 100 kHz para 92,16 kHz.

```
1  //-------------------------------------//
2  // DFT Method Code
3  //-------------------------------------//
4  #include "DSP28x_Project.h"
5  #include "Peripheral_Setup.h"
6  #include <math.h>
7
8  //-------------------------------------
9  // DFT Variables
10 //-------------------------------------
11
12 // Import Global Variables
13 extern unsigned char buffer_tx[];
14 extern int i, j;
15 extern int arraysSize, arraysLoaded, flag1;
16 extern double Icap[N], Vc[N];
17 extern unsigned int ADCIN_0, ADCIN_1, ADCIN_2, ADCIN_3[N], ADCIN_4[N],
      ADCIN_5[N];
18 extern double DAC_write_A, DAC_write_B;
19 extern double ESR_ref, ESR_corrigida, C_ref, C_corrigida, T_ref, T_core,
      Vref[N], A0, A1;
20
21 // Local Variables
22 double ESR, C, ESR_m, C_m;
23 double X_re_Vc, X_im_Vc, X_amp_Vc = 6, X_pha_Vc, X_phafinal_Vc, temp0_Vc,
      temp1_Vc;
24 double X_re_Icap, X_im_Icap, X_amp_Icap, X_pha_Icap, X_phafinal_Icap,
      temp0_Icap, temp1_Icap;
25 double AMP_I, AMP_V, PHASE_I, PHASE_V, w, XC;
26 double Wn, order;
27 double PI2 = 6.2832;
28 int k;
29
```

```
30 double test;
31 int Vc_view[N];
32
33 void DFT()
34 {
35   if (arraysLoaded == 0)
36   {
37     // Loading Vc and Ic vector
38     for(k = 0; k < N; ++k){
39       Vc[k] = (((double)ADCIN_3[k]) * 8.0586e-04) * 147.667; // Vc e Trocar
     1.65 por Vref
40       Vc_view[k] = (int)(Vc[k]);
41     }
42
43   }
44   if (flag1 == 1)
45   {
46     if (arraysLoaded == 1)
47     {
48       order = 1;
49       X_amp_Vc = 0;
50       X_re_Vc = 0;
51       X_im_Vc = 0;
52       X_amp_Icap = 0;
53       X_re_Icap = 0;
54       X_im_Icap = 0;
55
56       for (k = 0; k < (N >> 2); ++k)
57       {
58         Wn = 2.45436926E-2 * (float)(k * order);
59         // Vc DFT
60         temp0_Vc = (Vc[k] - Vc[k + (N >> 1)]);
61         temp1_Vc = (-Vc[k + (N >> 2)] + Vc[k + 3 * (N >> 2)]);
62         X_re_Vc += cos(Wn) * temp0_Vc + sin(Wn) * temp1_Vc;
63         X_im_Vc -= sin(Wn) * temp0_Vc - cos(Wn) * temp1_Vc;
64
65         // Icap DFT
66         temp0_Icap = (Icap[k] - Icap[k + (N >> 1)]);
67         temp1_Icap = (-Icap[k + (N >> 2)] + Icap[k + 3 * (N >> 2)]);
68         X_re_Icap += cos(Wn) * temp0_Icap + sin(Wn) * temp1_Icap;
69         X_im_Icap -= sin(Wn) * temp0_Icap - cos(Wn) * temp1_Icap;
70       }
71
72       // Vc Magnitude and Phase
73       X_amp_Vc = sqrt(X_re_Vc * X_re_Vc + X_im_Vc * X_im_Vc) * 2 / N;
74       test = sqrt(X_re_Vc * X_re_Vc + X_im_Vc * X_im_Vc) * 2 / N;
75       X_pha_Vc = atan(X_im_Vc / X_re_Vc) * 180 * 2 / PI2;
```

```
76
77      if (X_re_Vc >= 0)
78      {
79        X_phafinal_Vc = X_pha_Vc + 90;
80      }
81      else
82      {
83        X_phafinal_Vc = X_pha_Vc - 90;
84      }
85
86      // Icap Magnitude and Phase
87      X_amp_Icap = sqrt(X_re_Icap * X_re_Icap + X_im_Icap * X_im_Icap) * 2
    / N;
88      X_pha_Icap = atan(X_im_Icap / X_re_Icap) * 180 * 2 / PI2;
89
90
91      if (X_re_Icap >= 0)
92      {
93        X_phafinal_Icap = X_pha_Icap + 90;
94      }
95      else
96      {
97        X_phafinal_Icap = X_pha_Icap - 90;
98      }
99
100     // Saving final amplitude variables:
101     AMP_V = X_amp_Vc;
102     PHASE_V = X_phafinal_Vc;
103     AMP_I = X_amp_Icap;
104     PHASE_I = X_phafinal_Icap;
105
106     w = 2 * 3.1415 * 360;
107
108     // Calculating C and ESR if current Amplitude is different from 0:
109     if (AMP_I != 0)
110     {
111       ESR = fabs(AMP_V * cos((PHASE_V - PHASE_I) * 3.1415 / 180) / AMP_I)
    ;
112       ESR_m = 1000 * ESR;
113
114       XC = AMP_V * sin((PHASE_V - PHASE_I) * 3.1415 / 180) / AMP_I;
115
116       if (XC != 0)
117       {
118         C = fabs(-1 / (w * XC));
119         C_m = 1000 * C;
120       }
```

```
121       }
122
123       // Limits for ESR and C
124
125       if (C > 10e-3)
126       {
127         C = 10e-3;
128       }
129       if (C < 0)
130       {
131         C = 0;
132       }
133       if (ESR > 1)
134       {
135         ESR = 1;
136       }
137       if (ESR < 0)
138       {
139         ESR = 0;
140       }
141
142       // DAC
143       //      DAC_write_A = C * 500;
144       //      DAC_write_B = ESR * 5;
145       //      DAC();
146
147       // ESR Temperature Correction (in this case temperature sensors was
    not used):
148       // ESR_ref = ESR_inicial; // Reference ESR
149       // T_ref = 25; // Reference Temperature
150       // A0 = 21; // Experimental Value
151       // ESR_fixed = ESR + abs(ESR_ref - ESR_ref * exp((T_ref - T_core) /
    A0));
152       // // Fixing Capacitance
153       // C_ref = C_init; // Reference Capacitance
154       // A1 = 5e-7; // Experimental Value
155       // C_fixed = C - abs(C_ref - C_ref + A1 * (T_core - T_ref));
156     }
157   }
158 }
```

Lista B.1 – Arquivo "DFT.c"

O arquivo "Peripheral_Setup.h"é apresentado a seguir.

```
1 /*
2 * Peripheral_Setup.h
3 *
4 *  Created on: jun 21 2022
```

```
5  *  Author: Leonardo Rodrigues
6  */
7
8  #ifndef PERIPHERAL_SETUP_H_
9  #define PERIPHERAL_SETUP_H_
10 #include "DSP28x_Project.h"
11
12 #define N 256
13 #define F_BASE 360
14 #define F_PWM 4608
15 #define F_ADC_PWM 92160
16 #define N_SINE 1536
17
18 // sample Time N*freq base
19
20 #endif /* PERIPHERAL_SETUP_H_ */
```

Lista B.2 – Arquivo "Peripheral_Setup.h"

APÊNDICE C – CÓDIGO RLS PARA A SIMULAÇÃO NO PSIM

O código C contido no bloco RLS mostrado na Figura é apresentado a seguir.

```
// RLS Variables
// Variables for current estimation
#define arraySize 1666
#define adcFreq 100e3
#define simFreq 300e3
static int ignoreTransient = 10;
static double Ia, Ib, Ic, Iret, Icap[arraySize], Iinv, Vref;
static double VS2, VS4, VS6;

// RLS Method:
static double lambda = 0.999;
static double denK, b0, b1, u, up = 7, y, yp = 400;
static double ESR = 100e-3, C = 1e-3;
static double P11 = 1e-3, P12 = 0, P13 = 0, P21 = 0, P22 = 1e-3, P23 = 0,
    P31 = 0, P32 = 0, P33 = 1e-3;
static double K11, K21, K31;
static double teta11 = -1, teta21 = 0.105, teta31 = -0.095;
static double ESR_ref = 100e-3, C_ref = 1e-3, T_ref = 25, T_core, A0, A1,
    ESR_fixed = 100e-3, C_fixed = 1e-3;
static double error_ESR, error_C, ESR_view, C_view;

// Array ADC variables
static double ADCIN_0, ADCIN_1, ADCIN_2, ADCIN_3[arraySize], ADCIN_4[
    arraySize], ADCIN_5;
static double DIGIN_0, DIGIN_1, DIGIN_2;
static double Vc, Idc;
static int flag_1 = 0, flag_2 = 0;

// ADC
static int i = 0, j = 0, z = 0, k = 0;
int Tper, Tper_2;
static double Ts = 1 / adcFreq, Ts_2 = 1 / simFreq;
Tper = floor(Ts / delt);
Tper_2 = floor(Ts_2 / delt);

static double reset;

reset = x10;

if (reset == 1)
{
  flag_1 = 0;
  flag_2 = 0;
}
```

```
42
43 // ADC reading loop
44 if (i == 0)
45 {
46   if (flag_2 == ignoreTransient)
47   {
48     if (flag_1 == 0)
49     {
50       // Reading ADC variables
51       ADCIN_0 = x1;     // Ia
52       ADCIN_1 = x2;     // Ib
53       ADCIN_2 = x3;     // Iret
54       ADCIN_3[k] = x4; // Vc
55       ADCIN_4[k] = x5; // Temp
56       ADCIN_5 = x6;     // Vref
57       DIGIN_0 = x7;
58       DIGIN_1 = x8;
59       DIGIN_2 = x9;
60       ESR_view = x11;
61       C_view = x12;
62
63       // Current estimation
64       Vref = ((float)ADCIN_5) * 0.0008058608;
65       Ia = (((float)ADCIN_0) * 0.0008058608 - Vref) * 10;
66       Ib = (((float)ADCIN_1) * 0.0008058608 - Vref) * 10;
67       Ic = -(Ia + Ib);
68       Iret = (((float)ADCIN_2) * 0.0008058608) * 20;
69
70       VS2 = DIGIN_0;
71       VS4 = DIGIN_1;
72       VS6 = DIGIN_2;
73
74       // Converting digital inputs
75       if (VS2 > 0.001)
76       {
77         VS2 = 1;
78       }
79       if (VS2 <= 0.001)
80       {
81         VS2 = 0;
82       }
83       if (VS4 > 0.001)
84       {
85         VS4 = 1;
86       }
87       if (VS4 <= 0.001)
88       {
```

```
89        VS4 = 0;
90      }
91      if (VS6 > 0.001)
92      {
93        VS6 = 1;
94      }
95      if (VS6 <= 0.001)
96      {
97        VS6 = 0;
98      }
99
100     // Creating comparison table using switching states to estimate
    capacitor current
101     if (VS2 == 0 && VS4 == 0 && VS6 == 0)
102     {
103       Iinv = 0;
104     }
105     if (VS2 == 0 && VS4 == 0 && VS6 == 1)
106     {
107       Iinv = Ic;
108     }
109     if (VS2 == 0 && VS4 == 1 && VS6 == 0)
110     {
111       Iinv = Ib;
112     }
113     if (VS2 == 0 && VS4 == 1 && VS6 == 1)
114     {
115       Iinv = Ic + Ib;
116     }
117     if (VS2 == 1 && VS4 == 0 && VS6 == 0)
118     {
119       Iinv = Ia;
120     }
121     if (VS2 == 1 && VS4 == 0 && VS6 == 1)
122     {
123       Iinv = Ic + Ia;
124     }
125     if (VS2 == 1 && VS4 == 1 && VS6 == 0)
126     {
127       Iinv = -Ic;
128     }
129     if (VS2 == 1 && VS4 == 1 && VS6 == 1)
130     {
131       Iinv = 0;
132     }
133
134     Icap[k] = Iret - Iinv;
```

```
135
136       //
137       ++k;
138       ++i;
139       if (k == arraySize)
140       {
141         k = 0;
142         flag_1 = 1;
143       }
144     }
145   }
146   else
147   {
148     flag_2 = flag_2 + 1;
149   }
150 }
151
152 // Comparator conditionals for ADC reading
153 if (i <= Tper)
154 {
155   i = i + 1;
156 }
157 if (i == Tper + 1)
158 {
159   i = 0;
160 }
161
162 // Z loop can be used to control calculation frequency
163 // Loop to emulate DSP while(1)
164 //if (z == 0)
165 //{
166   if (flag_1 == 1)
167   {
168     y = (((double)ADCIN_3[j]) * 8.0586e-04) * 147.667;
169     u = Icap[j];
170     T_core = ((double)ADCIN_4[j]);
171
172     // RLS Code
173     // Iniciando com a matriz K:
174     denK = lambda + u * (P22 * u + P32 * up - P12 * yp) + up * (P23 * u +
    P33 * up - P13 * yp) - yp * (P21 * u + P31 * up - P11 * yp);
175
176     if (denK != 0)
177     {
178       K11 = (P12 * u + P13 * up - P11 * yp) / denK;
179       K21 = (P22 * u + P23 * up - P21 * yp) / denK;
180       K31 = (P32 * u + P33 * up - P31 * yp) / denK;
```

```
181     }
182
183     // Iniciando matriz teta:
184     teta11 = teta11 + K11 * (y - teta21 * u - teta31 * up + teta11 * yp);
185     teta21 = teta21 + K21 * (y - teta21 * u - teta31 * up + teta11 * yp);
186     teta31 = teta31 + K31 * (y - teta21 * u - teta31 * up + teta11 * yp);
187
188     // Matriz P:
189     P11 = -((K11 * P21 * u - P11 * (K11 * yp + 1) + K11 * P31 * up) / (
        lambda));
190     P12 = -((K11 * P22 * u - P12 * (K11 * yp + 1) + K11 * P32 * up) / (
        lambda));
191     P13 = -((K11 * P23 * u - P13 * (K11 * yp + 1) + K11 * P33 * up) / (
        lambda));
192     P21 = -((P21 * (K21 * u - 1) - K21 * P11 * yp + K21 * P31 * up) / (
        lambda));
193     P22 = -((P22 * (K21 * u - 1) - K21 * P12 * yp + K21 * P32 * up) / (
        lambda));
194     P23 = -((P23 * (K21 * u - 1) - K21 * P13 * yp + K21 * P33 * up) / (
        lambda));
195     P31 = -((P31 * (K31 * up - 1) - K31 * P11 * yp + K31 * P21 * u) / (
        lambda));
196     P32 = -((P32 * (K31 * up - 1) - K31 * P12 * yp + K31 * P22 * u) / (
        lambda));
197     P33 = -((P33 * (K31 * up - 1) - K31 * P13 * yp + K31 * P23 * u) / (
        lambda));
198
199     // Calculando b0 e b1
200     b0 = teta21;
201     b1 = teta31;
202
203     if (b0 > 1)
204     {
205       b0 = 1;
206     }
207     if (b0 < -1)
208     {
209       b0 = -1;
210     }
211     if (b1 > 1)
212     {
213       b1 = 1;
214     }
215     if (b1 < -1)
216     {
217       b1 = -1;
218     }
```

```
219
220     // ESR and C
221     // Condition to proceed w/o errors
222     if ((b1 + b0) != 0)
223     {
224       C = Ts / (b1 + b0);
225       ESR = b0 - Ts / (2 * C);
226     }
227
228     // ESR Temperature Correction (in this case temperature sensors was not
        used):
229     A0 = 21; // Experimental Value
230     ESR_fixed = ESR + abs(ESR_ref - ESR_ref * exp((T_ref - T_core) / A0));
231     // Fixing Capacitance
232     A1 = 5e-7; // Experimental Value
233     //C_fixed = C - abs(C_ref - C_ref + A1 * (T_core - T_ref));
234     C_fixed = C - abs(A1 * (T_core - T_ref));
235
236     // Limits
237     if (C_fixed > 1.5e-3)
238     {
239       C = 1.5e-3;
240       C_fixed = 1.5e-3;
241     }
242     if (C_fixed < 0)
243     {
244       C = 0;
245       C_fixed = 0;
246     }
247     if (ESR_fixed > 1)
248     {
249       ESR = 0;
250       ESR_fixed = 1;
251     }
252     if (ESR_fixed < 0)
253     {
254       ESR = 0;
255       ESR_fixed = 0;
256     }
257
258     if((ESR_fixed && C_fixed) > 0)
259     {
260       error_ESR = 100*abs(1-ESR_view/ESR);
261       error_C = 100*abs(1-C_view/C);
262     }
263
264     // update yp and up
```

```
    yp = y; // Ypast
    up = u; // Upast

    ++j;

    if (j == arraySize)
    {
      j = 0;
    }
  }
  //}
if (z <= Tper_2)
{
  z = z + 1;
}
if (z == Tper_2 + 1)
{
  z = 0;
}

y1 = y;
y2 = u;
y3 = i;
y4 = z;
y5 = flag_1;
y6 = flag_2;
y7 = j;
y8 = ESR;
y9 = C;
y10 = Vref;
y11 = Ia;
y12 = Ib;
y13 = Iret;
y14 = Iinv;
y15 = VS2;
y16 = VS4;
y17 = VS6;
y18 = T_core;
y19 = ESR_fixed;
y20 = C_fixed;
y21 = error_ESR;
y22 = error_C;
```

Lista C.1 – Código RLS para PSIM

APÊNDICE D – CÓDIGO DFT (RADIX-4) PARA A SIMULAÇÃO NO PSIM

O código C contido no bloco que processa os cálculos da DFT é apresentado a seguir.

```
// RLS Variables
// Variables for current estimation
#define arraySize 256
#define adcFreq 92.16e3
#define simFreq 300e3
static int ignoreTransient = 50;
static double Ia, Ib, Ic, Iret, Icap[arraySize], Vc[arraySize], Iinv, Vref,
     Icap_view;
static double VS2, VS4, VS6;

// DFT Method:
static double Wn = 0, order, PI2 = 6.2832;
static double X_re_Vc, X_im_Vc, X_amp_Vc, X_pha_Vc, X_phafinal_Vc, temp0_Vc
    , temp1_Vc;
static double X_re_Icap, X_im_Icap, X_amp_Icap, X_pha_Icap, X_phafinal_Icap
    , temp0_Icap, temp1_Icap;
static double ESR_ref = 100e-3, C_ref = 1e-3, T_ref = 25, T_core, A0, A1,
    ESR_fixed = 100e-3, C_fixed = 1e-3;
static double error_ESR, error_C, ESR_view, C_view;

static double AMP_V, PHASE_V, AMP_I, PHASE_I, w = 2 * 3.1415 * 360, ESR =
    100e-3, XC, C = 1e-3, PHASYS;

// Array ADC variables
static double ADCIN_0, ADCIN_1, ADCIN_2, ADCIN_3[arraySize], ADCIN_4[
    arraySize], ADCIN_5;
static double DIGIN_0, DIGIN_1, DIGIN_2;
static int flag_1 = 0, flag_2 = 0;

// ADC
static int i = 0, j = 0, z = 0, k = 0;
int Tper, Tper_2;
static double Ts = 1 / adcFreq, Ts_2 = 1 / simFreq;
Tper = floor(Ts / delt);
Tper_2 = floor(Ts_2 / delt);

static double reset;

reset = x10;

if (reset == 1)
{
  k = 0;
```

```
38    flag_1 = 0;
39    flag_2 = 0;
40  }
41
42  // ADC reading loop
43  if (i == 0)
44  {
45    if (flag_2 == ignoreTransient)
46    {
47      if (flag_1 == 0)
48      {
49        // Reading ADC variables
50        ADCIN_0 = x1;     // Ia
51        ADCIN_1 = x2;     // Ib
52        ADCIN_2 = x3;     // Iret
53        ADCIN_3[k] = x4; // Vc
54        ADCIN_4[k] = x5; // Temp
55        ADCIN_5 = x6;     // Vref
56        DIGIN_0 = x7;
57        DIGIN_1 = x8;
58        DIGIN_2 = x9;
59        ESR_view = x11;
60        C_view = x12;
61
62        // Current estimation
63        Vref = ((float)ADCIN_5) * 0.0008058608;
64        Ia = (((float)ADCIN_0) * 0.0008058608 - Vref) * 10;
65        Ib = (((float)ADCIN_1) * 0.0008058608 - Vref) * 10;
66        Ic = -(Ia + Ib);
67        Iret = (((float)ADCIN_2) * 0.0008058608) * 20;
68
69        VS2 = DIGIN_0;
70        VS4 = DIGIN_1;
71        VS6 = DIGIN_2;
72
73        // Converting digital inputs
74        if (VS2 > 0.001)
75        {
76          VS2 = 1;
77        }
78        if (VS2 <= 0.001)
79        {
80          VS2 = 0;
81        }
82        if (VS4 > 0.001)
83        {
84          VS4 = 1;
```

```
85      }
86      if (VS4 <= 0.001)
87      {
88        VS4 = 0;
89      }
90      if (VS6 > 0.001)
91      {
92        VS6 = 1;
93      }
94      if (VS6 <= 0.001)
95      {
96        VS6 = 0;
97      }
98
99      // Creating comparison table using switching states to estimate
    capacitor current
100     if (VS2 == 0 && VS4 == 0 && VS6 == 0)
101     {
102       Iinv = 0;
103     }
104     if (VS2 == 0 && VS4 == 0 && VS6 == 1)
105     {
106       Iinv = Ic;
107     }
108     if (VS2 == 0 && VS4 == 1 && VS6 == 0)
109     {
110       Iinv = Ib;
111     }
112     if (VS2 == 0 && VS4 == 1 && VS6 == 1)
113     {
114       Iinv = Ic + Ib;
115     }
116     if (VS2 == 1 && VS4 == 0 && VS6 == 0)
117     {
118       Iinv = Ia;
119     }
120     if (VS2 == 1 && VS4 == 0 && VS6 == 1)
121     {
122       Iinv = Ic + Ia;
123     }
124     if (VS2 == 1 && VS4 == 1 && VS6 == 0)
125     {
126       Iinv = -Ic;
127     }
128     if (VS2 == 1 && VS4 == 1 && VS6 == 1)
129     {
130       Iinv = 0;
```

```
131       }
132
133       Icap[k] = Iret - Iinv;
134       Icap_view = Icap[k];
135
136       //
137       ++k;
138       ++i;
139       if (k == arraySize)
140       {
141         k = 0;
142         flag_1 = 1;
143       }
144     }
145   }
146   else
147   {
148     flag_2 = flag_2 + 1;
149   }
150 }
151
152 // Comparator conditionals for ADC reading
153 if (i <= Tper)
154 {
155   i = i + 1;
156 }
157 if (i == Tper + 1)
158 {
159   i = 0;
160 }
161
162 // Z loop can be used to control calculation frequency
163 // Loop to emulate DSP while(1)
164 // if (z == 0)
165 //{
166   if (flag_1 == 1)
167   {
168     for (k = 0; k < arraySize; ++k)
169     {
170       Vc[k] = (((double)ADCIN_3[k]) * 8.0586e-04) * 147.667; // Vc e Trocar
      1.65 por Vref
171     }
172
173     T_core = ((double)ADCIN_4[j]);
174
175     order = 1;
176     X_amp_Vc = 0;
```

```
177     X_re_Vc = 0;
178     X_im_Vc = 0;
179     X_amp_Icap = 0;
180     X_re_Icap = 0;
181     X_im_Icap = 0;
182
183     for (k = 0; k < (arraySize >> 2); ++k)
184     {
185       Wn = 2.45436926E-2 * (float)(k * order);
186       // Vc DFT
187       temp0_Vc = (Vc[k] - Vc[k + (arraySize >> 1)]);
188       temp1_Vc = (-Vc[k + (arraySize >> 2)] + Vc[k + 3 * (arraySize >> 2)])
    ;
189       X_re_Vc += cos(Wn) * temp0_Vc + sin(Wn) * temp1_Vc;
190       X_im_Vc -= sin(Wn) * temp0_Vc - cos(Wn) * temp1_Vc;
191
192       // Icap DFT
193       temp0_Icap = (Icap[k] - Icap[k + (arraySize >> 1)]);
194       temp1_Icap = (-Icap[k + (arraySize >> 2)] + Icap[k + 3 * (arraySize
    >> 2)]);
195       X_re_Icap += cos(Wn) * temp0_Icap + sin(Wn) * temp1_Icap;
196       X_im_Icap -= sin(Wn) * temp0_Icap - cos(Wn) * temp1_Icap;
197     }
198
199     // Vc Mag e Fase
200     X_amp_Vc = sqrt(X_re_Vc * X_re_Vc + X_im_Vc * X_im_Vc) * 2 / arraySize;
201     X_pha_Vc = atan(X_im_Vc / X_re_Vc) * 180 * 2 / PI2;
202
203     if (X_re_Vc >= 0)
204     {
205       X_phafinal_Vc = X_pha_Vc + 90;
206     }
207     else
208     {
209       X_phafinal_Vc = X_pha_Vc - 90;
210     };
211
212     // Icap Mag e Fase
213     X_amp_Icap = sqrt(X_re_Icap * X_re_Icap + X_im_Icap * X_im_Icap) * 2 /
    arraySize;
214     X_pha_Icap = atan(X_im_Icap / X_re_Icap) * 180 * 2 / PI2;
215
216     if (X_re_Icap >= 0)
217     {
218       X_phafinal_Icap = X_pha_Icap + 90;
219     }
220     else
```

```
221     {
222       X_phafinal_Icap = X_pha_Icap - 90;
223     };
224 
225     AMP_V = X_amp_Vc;
226     PHASE_V = X_phafinal_Vc;
227     AMP_I = X_amp_Icap;
228     PHASE_I = X_phafinal_Icap;
229 
230     if (AMP_I != 0)
231     {
232       ESR = fabs(AMP_V * cos((PHASE_V - PHASE_I) * 3.1415 / 180) / AMP_I);
233       XC = AMP_V * sin((PHASE_V - PHASE_I) * 3.1415 / 180) / AMP_I;
234 
235       if (XC != 0)
236       {
237         C = fabs(-1 / (w * XC));
238       }
239     }
240 
241     // ESR Temperature Correction (in this case temperature sensors was not
      used):
242     A0 = 21; // Experimental Value
243     ESR_fixed = ESR + abs(ESR_ref - ESR_ref * exp((T_ref - T_core) / A0)) +
      0.1e-3;
244     // Fixing Capacitance
245     A1 = 5e-7; // Experimental Value
246     // C_fixed = C - abs(C_ref - C_ref + A1 * (T_core - T_ref));
247     C_fixed = C - abs(A1 * (T_core - T_ref));
248 
249     // Limits
250     if (C_fixed > 1.5e-3)
251     {
252       C = 1.5e-3;
253       C_fixed = 1.5e-3;
254     }
255     if (C_fixed < 0)
256     {
257       C = 0;
258       C_fixed = 0;
259     }
260     if (ESR_fixed > 1)
261     {
262       ESR = 0;
263       ESR_fixed = 1;
264     }
265     if (ESR_fixed < 0)
```

```
266     {
267       ESR = 0;
268       ESR_fixed = 0;
269     }
270 
271     if ((ESR && C) > 0)
272     {
273       error_ESR = 100 * abs(1 - ESR_view / ESR);
274       error_C = 100 * abs(1 - C_view / C);
275     }
276   }
277   //}
278 if (z <= Tper_2)
279 {
280   z = z + 1;
281 }
282 if (z == Tper_2 + 1)
283 {
284   z = 0;
285 }
286 
287 y1 = AMP_V;
288 y2 = AMP_I;
289 y3 = i;
290 y4 = Icap_view;
291 y5 = flag_1;
292 y6 = flag_2;
293 y7 = k;
294 y8 = ESR;
295 y9 = C;
296 y10 = Vref;
297 y11 = Ia;
298 y12 = Ib;
299 y13 = Iret;
300 y14 = Iinv;
301 y15 = VS2;
302 y16 = VS4;
303 y17 = VS6;
304 y18 = T_core;
305 y19 = ESR_fixed;
306 y20 = C_fixed;
307 y21 = error_ESR;
308 y22 = error_C;
```

Lista D.1 – Código DFT (Radix-4) para PSIM

APÊNDICE E – CÓDIGO DFT (GOERTZEL) PARA A SIMULAÇÃO NO PSIM

```
#include <math.h>
static double targetFreq = 360, N = 256, k, w;
static double X_re_Vcap, X_im_Vcap, X_amp_Vcap, X_pha_Vcap, X_phafinal_Vcap
    , count = 0, coeff;
static double X_re_Icap, X_im_Icap, X_amp_Icap, X_pha_Icap, X_phafinal_Icap
    ;
static double Ia, Ib, Ic, Iret, Iinv, Vref;
static double VS2, VS4, VS6;

static double cosine, sine, PI2 = 6.283;
static double Q0_Vcap = 0, Q1_Vcap = 0, Q2_Vcap = 0, Vcap;
static double Q0_Icap = 0, Q1_Icap = 0, Q2_Icap = 0, Icap;

static double ESR_ref = 100e-3, C_ref = 1e-3, T_ref = 25, T_core, A0, A1,
    ESR_fixed = 100e-3, C_fixed = 1e-3;
static double error_ESR, error_C, ESR_view, C_view;
static double AMP_V, PHASE_V, AMP_I, PHASE_I, ESR, C, XC;

// ADC variables
static int ADCIN_0, ADCIN_1, ADCIN_2, ADCIN_3, ADCIN_4, ADCIN_5;
static double DIGIN_0, DIGIN_1, DIGIN_2;

static double sampleFreq = N * targetFreq;

k = (int)(0.5 + (N * targetFreq) / sampleFreq);

w = PI2 * k / N;
cosine = cos(w);
sine = sin(w);
coeff = 2 * cosine;

// ADC
static int z = 0;
int Tper;
static double Ts = 1 / sampleFreq;
Tper = floor(Ts / delt);

if (z == 0)
{
  // Reading ADC variables
  ADCIN_0 = x1; // Ia
  ADCIN_1 = x2; // Ib
  ADCIN_2 = x3; // Iret
  ADCIN_3 = x4; // Vcap
  ADCIN_4 = x5; // Temp
```

```
43  ADCIN_5 = x6; // Vref
44  DIGIN_0 = x7;
45  DIGIN_1 = x8;
46  DIGIN_2 = x9;
47  ESR_view = x10;
48  C_view = x11;
49
50  // Current estimation
51  Vref = ((float)ADCIN_5) * 0.0008058608;
52  Ia = (((float)ADCIN_0) * 0.0008058608 - Vref) * 10;
53  Ib = (((float)ADCIN_1) * 0.0008058608 - Vref) * 10;
54  Ic = -(Ia + Ib);
55  Iret = (((float)ADCIN_2) * 0.0008058608) * 20;
56
57  VS2 = DIGIN_0;
58  VS4 = DIGIN_1;
59  VS6 = DIGIN_2;
60
61  // Converting digital inputs
62  if (VS2 > 0.001)
63  {
64    VS2 = 1;
65  }
66  if (VS2 <= 0.001)
67  {
68    VS2 = 0;
69  }
70  if (VS4 > 0.001)
71  {
72    VS4 = 1;
73  }
74  if (VS4 <= 0.001)
75  {
76    VS4 = 0;
77  }
78  if (VS6 > 0.001)
79  {
80    VS6 = 1;
81  }
82  if (VS6 <= 0.001)
83  {
84    VS6 = 0;
85  }
86  T_core = ((double)ADCIN_4);
87  Vcap = (((double)ADCIN_3) * 8.0586e-04) * 147.667;
88  Q0_Vcap = coeff * Q1_Vcap - Q2_Vcap + Vcap;
89  Q2_Vcap = Q1_Vcap;
```

```
90  Q1_Vcap = Q0_Vcap;
91
92  Iinv = VS2 * Ia + VS4 * Ib + VS6 * Ic;
93  Icap = Iret - Iinv;
94  Q0_Icap = coeff * Q1_Icap - Q2_Icap + Icap;
95  Q2_Icap = Q1_Icap;
96  Q1_Icap = Q0_Icap;
97  count++;
98
99  if (count == N)
100 {
101   X_re_Vcap = (Q1_Vcap - Q2_Vcap * cosine);
102   X_im_Vcap = (Q2_Vcap * sine);
103
104   // Vcap Mag e Fase
105   X_amp_Vcap = sqrt(X_re_Vcap * X_re_Vcap + X_im_Vcap * X_im_Vcap) * 2 /
      N;
106   X_pha_Vcap = atan(X_im_Vcap / X_re_Vcap) * 180 * 2 / PI2;
107
108   if (X_re_Vcap >= 0)
109   {
110     X_phafinal_Vcap = X_pha_Vcap + 90;
111   }
112   else
113   {
114     X_phafinal_Vcap = X_pha_Vcap - 90;
115   };
116
117   X_re_Icap = (Q1_Icap - Q2_Icap * cosine);
118   X_im_Icap = (Q2_Icap * sine);
119
120   // Icap Mag e Fase
121   X_amp_Icap = sqrt(X_re_Icap * X_re_Icap + X_im_Icap * X_im_Icap) * 2 /
      N;
122   X_pha_Icap = atan(X_im_Icap / X_re_Icap) * 180 * 2 / PI2;
123
124   if (X_re_Icap >= 0)
125   {
126     X_phafinal_Icap = X_pha_Icap + 90;
127   }
128   else
129   {
130     X_phafinal_Icap = X_pha_Icap - 90;
131   };
132
133   Q2_Vcap = 0;
134   Q1_Vcap = 0;
```

```
135     Q2_Icap = 0;
136     Q1_Icap = 0;
137     count = 0;
138
139     // Calculating ESR
140     AMP_V = X_amp_Vcap;
141     PHASE_V = X_phafinal_Vcap;
142     AMP_I = X_amp_Icap;
143     PHASE_I = X_phafinal_Icap;
144     w = PI2 * targetFreq;
145
146     if (AMP_I != 0)
147     {
148
149       ESR = fabs(AMP_V * cos((PHASE_V - PHASE_I) * 3.1415 / 180) / AMP_I);
150       XC = AMP_V * sin((PHASE_V - PHASE_I) * 3.1415 / 180) / AMP_I;
151
152       if (XC != 0)
153       {
154         C = fabs(-1 / (w * XC));
155       }
156     }
157
158     // ESR Temperature Correction (in this case temperature sensors was not
        used):
159     A0 = 21; // Experimental Value
160     ESR_fixed = ESR + abs(ESR_ref - ESR_ref * exp((T_ref - T_core) / A0));
161     // Fixing Capacitance
162     A1 = 5e-7; // Experimental Value
163     // C_fixed = C - abs(C_ref - C_ref + A1 * (T_core - T_ref));
164     C_fixed = C - abs(A1 * (T_core - T_ref));
165
166     // Limits
167     if (C_fixed > 1.5e-3)
168     {
169       C = 1.5e-3;
170       C_fixed = 1.5e-3;
171     }
172     if (C_fixed < 0)
173     {
174       C = 0;
175       C_fixed = 0;
176     }
177     if (ESR_fixed > 1)
178     {
179       ESR = 0;
180       ESR_fixed = 1;
```

```
    }
    if (ESR_fixed < 0)
    {
      ESR = 0;
      ESR_fixed = 0;
    }

    if ((ESR_view && C_view) > 0)
    {
      error_ESR = 100 * abs(1 - ESR / ESR_view);
      error_C = 100 * abs(1 - C / C_view);
    }
  }
}

if (z <= Tper)
{
  z = z + 1;
}
if (z == Tper + 1)
{
  z = 0;
}

y1 = AMP_V;
y2 = AMP_I;
y3 = Vcap;
y4 = Icap;
y5 = k;
y6 = ESR;
y7 = C;
y8 = Vref;
y9 = Ia;
y10 = Ib;
y11 = Iret;
y12 = Iinv;
y13 = VS2;
y14 = VS4;
y15 = VS6;
y16 = T_core;
y17 = ESR_fixed;
y18 = C_fixed;
y19 = error_ESR;
y20 = error_C;
```

Lista E.1 – Código DFT (Goertzel) para PSIM

APÊNDICE F – RECEBENDO DADOS VIA UART NO ESP32 E ENVIANDO PARA O GOOGLE SHEET

O arquivo a seguir mostra como foi realizada a recepção de dados no ESP32 através da interface do Arduino para receber os dados do DSP F28069M para a tabela no Google Sheets.

```
// UART & Other Libraries + Variables
#include <HardwareSerial.h>
#include <stdlib.h>
#include <stdlib.h>
#include <string.h>
#include <errno.h>

#define RXp 16 //(RX2)
#define TXp 17 //(TX2)
#define ONBOARD_LED  2
#define RESET_PIN_FLAG 18
#define RECEIVEDATA_PIN_FLAG 19 // flag that will be on when receiving data from F28069M
#define PIN_TEST 21

#define INTERVAL 15000 //ms
#define INTERVAL2 13000 //ms

#define lengthESRData 10

// TX2 17
// RX2 16

int Received_Data;
const int BUFFER_SIZE = 8;
byte buf[BUFFER_SIZE];
float conv_factor = 100, ESR_conv_factor = 1, C_conv_factor = 100;
int lengthData, flag;

// Received Data
static int i = 0, k = 0;
unsigned int t0, t1, dt, dt2;
double ESR_v[lengthESRData], C_v[lengthESRData], Vc_v[lengthESRData], ESR, C, Vc;

// WiFi Libraries and variables
#include <WiFi.h>
#include <HTTPClient.h>
#include <math.h>
//Things to change
```

```
39 const char* ssid = "--paste your custom SSID here--";
40 const char* password = "--paste your password here--";
41
42 static int flagSendToGoogle = 0, flagDataReceived = 0;
43
44 String GOOGLE_SCRIPT_ID = "--paste your custom ID here--";
45
46
47
48 WiFiClientSecure client;
49
50 void setup() {
51   Serial.begin(9600);
52   Serial2.begin(9600, SERIAL_8N1, RXp, TXp);
53   pinMode(ONBOARD_LED, OUTPUT);
54   pinMode(RESET_PIN_FLAG, OUTPUT);
55   pinMode(PIN_TEST, OUTPUT);
56   pinMode(RECEIVEDATA_PIN_FLAG, OUTPUT);
57   digitalWrite(ONBOARD_LED, HIGH);
58   digitalWrite(RESET_PIN_FLAG, LOW);
59   digitalWrite(PIN_TEST, LOW);
60   digitalWrite(RECEIVEDATA_PIN_FLAG, LOW);
61   t0 = millis();
62
63   delay(10);
64
65   WiFi.mode(WIFI_STA);
66   WiFi.begin(ssid, password);
67
68   Serial.println("Started");
69   Serial.println("Connecting");
70   while (WiFi.status() != WL_CONNECTED) {
71     delay(500);
72     Serial.println("Connection Failed");
73   }
74
75   Serial.println("Wi-Fi Connected");
76 }
77
78 // Receiving data
79 // How data conversion works: The data comes with 1 byte array (8 bits),
     that is -> an array tx[1,2,3,4,5,6,7,8]. Example: Where two first bits
     are data sent by the other device, the conversion is made like: (tx
     [1]*256 + tx[2])/conv_factor
80
81 void loop() {
82
```

```
83    sendResetFlagToF28069M();
84    digitalWrite(PIN_TEST, HIGH);
85    ReceiveDataFromF28069M();
86    digitalWrite(PIN_TEST, LOW);
87    if (flagDataReceived == 1) {
88      digitalWrite(RECEIVEDATA_PIN_FLAG, LOW);
89    }
90    if (flagSendToGoogle == 1) {
91      Serial.println("BUFFER's ready to send to Google Sheets");
92      prepareDataToGoogleSheets();
93    }
94  }
95
96  void sendResetFlagToF28069M() {
97    t1 = millis();
98    dt = (t1 - t0) % INTERVAL;
99
100   if (dt == 0) {
101     digitalWrite(RESET_PIN_FLAG, HIGH);
102     digitalWrite(RECEIVEDATA_PIN_FLAG, LOW);
103     Serial.println("Reset Flag sent");
104     delay(100);
105     digitalWrite(RESET_PIN_FLAG, LOW);
106     delay(INTERVAL2);
107     digitalWrite(RECEIVEDATA_PIN_FLAG, HIGH);
108     flagDataReceived = 0;
109     delay(10);
110   }
111   else {
112     digitalWrite(RESET_PIN_FLAG, LOW);
113   }
114 }
115
116 void ReceiveDataFromF28069M() {
117   if (Serial2.available() > 0 && Received_Data != 255 && Received_Data !=
      -1) {
118
119     getDataLength();
120
121     if (flag == 0) {
122       Serial.println("Received Data:");
123       for (int i = 0; i < lengthData; i++) {
124         Serial.print(buf[i]);
125         Serial.print(",");
126       }
127       // Decrypting received data
128
```

```
129       ESR = (( buf[0] * 256 + buf[1]) / conv_factor ) / ESR_conv_factor;
130       C = (( buf[2] * 256 + buf[3]) / conv_factor) / C_conv_factor;
131       Vc = ( buf[4] * 256 + buf[5]);
132
133       if ((ESR && C) > 0) {
134         ESR_v[k] = ESR;
135         C_v[k] = C;
136         Vc_v[k] = Vc;
137
138         flagDataReceived = 1;
139
140         Serial.print("Decripted protocol: ESR = ");
141         Serial.print(ESR);
142         Serial.print(", C = ");
143         Serial.print(C);
144         Serial.print(", Vc = ");
145         Serial.print(Vc);
146         Serial.print(", K index = ");
147         Serial.println(k);
148         Serial.print("\n");
149         k++;
150
151         if (k == (lengthESRData)) {
152           k = 0;
153           flagSendToGoogle = 1;
154         }
155       }
156       else {
157         flagDataReceived = 1;
158         Serial.print("\n");
159         Serial.print("Decripted protocol ((ESR || C) <= 0): ESR = ");
160         Serial.print(ESR);
161         Serial.print(", C = ");
162         Serial.println(C);
163         Serial.println("Algorithm Stopped");
164         Serial.print("\n");
165       }
166     }
167   }
168 }
169
170 // Functions
171 void getDataLength() {
172   flag = 0;
173   // Get the array length
174   lengthData = Serial2.readBytes(buf, BUFFER_SIZE);
175 }
```

```
176
177 void prepareDataToGoogleSheets() {
178   if (flagSendToGoogle == 1) {
179     for (i = 0; i <= 9; i++) {
180       String ESR_s(ESR_v[i], 6);
181       String C_s(C_v[i], 6);
182       String Vc_s(Vc_v[i], 6);
183       String param;
184       param = "value1=" + ESR_s;
185       param += "&value2=" + C_s;
186       param += "&value3=" + Vc_s;
187       Serial.println("Sending parameters (...) index: " + String(i));
188       sendData(param);
189       Serial.print("\n");
190
191       if (i == lengthESRData - 1) {
192         Serial.println("The data was sent to Google Sheets successfully");
193         flagSendToGoogle = 0;
194       }
195     }
196   }
197 }
198
199 void sendData(String params) {
200   HTTPClient http;
201   String url = "https://script.google.com/macros/s/" + GOOGLE_SCRIPT_ID + "
       /exec?" + params;
202   //String url = "https://script.google.com/macros/s/" + GOOGLE_SCRIPT_ID +
       "/exec?" + "value1=" + temp_s + "&value2=" + lightpercentage;
203
204   Serial.println(url);
205   Serial.println("Postring data to Google Sheet");
206   //---------------------------------------------------------
207   //starts posting data to google sheet
208   http.begin(url.c_str());
209   http.setFollowRedirects(HTTPC_STRICT_FOLLOW_REDIRECTS);
210   int httpCode = http.GET();
211   Serial.print("HTTP Status Code: ");
212   Serial.println(httpCode);
213   //---------------------------------------------------------
214   //getting response from google sheet
215   String payload;
216   if (httpCode > 0) {
217     payload = http.getString();
218     Serial.println("Payload: " + payload);
219   }
220   //---------------------------------------------------------
```

```
221   http.end();
222 }
```

Lista F.1 – Código recepção UART para recepção de dados no ESP32 e transmissão para o Google Sheets

APÊNDICE G – RECEBENDO E ORGANIZANDO OS DADOS NO GOOGLE SHEETS EM JAVASCRIPT

Nesse capítulo é apresentado o código em Javascript para receber e organizar os dados nas planilhas do Google Sheets.

```
function doGet(e) {
  Logger.log( JSON.stringify(e) ); // view parameters
  var result = 'Ok'; // assume success
  if (e.parameter == 'undefined') {
    result = 'No Parameters';
  }
  else {
    var sheet_id = '1BYC7FngqQJ12daVny2nR1lwclm2PsD5hVJ5RD-ZWeiQ';  // Spreadsheet ID
    var sheet = SpreadsheetApp.openById(sheet_id).getActiveSheet(); // get Active sheet
    var newRow = sheet.getLastRow() + 1;
    var rowData = [];
    d=new Date();
    rowData[0] = d; // Timestamp in column A
    rowData[1] = d.toLocaleTimeString(); // Timestamp in column A

    for (var param in e.parameter) {
      Logger.log('In for loop, param=' + param);
      var value = stripQuotes(e.parameter[param]);
      Logger.log(param + ':' + e.parameter[param]);
      switch (param) {
        case 'value1': //Parameter 1, It has to be updated in Column in Sheets in the code, orderwise
        rowData[2] = value; //Value in column A
        result = 'Written on column A';
        break;
        case 'value2': //Parameter 2, It has to be updated in Column in Sheets in the code, orderwise
        rowData[3] = value; //Value in column B
        result += ' Written on column B';
        break;
        case 'value3': //Parameter 3, It has to be updated in Column in Sheets in the code, orderwise
        rowData[4] = value; //Value in column C
        result += ' Written on column C';
        break;
        default:
        result = "unsupported parameter";
      }
    }
```

```
    Logger.log(JSON.stringify(rowData));
    // Write new row below
    var newRange = sheet.getRange(newRow, 1, 1, rowData.length);
    newRange.setValues([rowData]);
  }
  // Return result of operation
  return ContentService.createTextOutput(result);
}
function stripQuotes( value ) {
  return value.replace(/^["']|['"]$/g, "");
}
```

Lista G.1 – Arquivo "Script.gs".