UNIVERSIDADE DA INTEGRAÇÃO INTERNACIONAL DA LUSOFONIA AFRO-BRASILEIRA

INSTITUTO DE ENGENHARIAS E DESENVOLVIMENTO SUSTENTÁVEL

CURSO DE ENGENHARIA DE ENERGIAS

SIMÃO SILVA FREITAS

DESENVOLVIMENTO DE *FIRMWARE* PARA UM INVERSOR PWM MONOFÁSICO PARA APLICAÇÕES EM QUALIDADE DE ENERGIA

REDENÇÃO
2021

SIMÃO SILVA FREITAS

DESENVOLVIMENTO DE *FIRMWARE* PARA UM INVERSOR PWM MONOFÁSICO PARA APLICAÇÕES EM QUALIDADE DE ENERGIA

Trabalho de conclusão de curso apresentado como requisito para obtenção do título de Bacharel em Engenharia de Energias da Universidade da Integração Internacional da Lusofonia Afro-Brasileira.

Orientador: Prof. Dr. Hermínio Miguel De Oliveira Filho

REDENÇÃO
2021

Universidade da Integração Internacional da Lusofonia Afro-Brasileira
Sistema de Bibliotecas da UNILAB
Catalogação de Publicação na Fonte.

Freitas, Simão Silva.

F936d

Desenvolvimento de firmware para um inversor PWM monofásico para aplicações em qualidade de energia / Simão Silva Freitas. - Redenção, 2021.
77f: il.

Monografia - Curso de Engenharia de Energias, Instituto de Engenharias e Desenvolvimento Sustentável, Universidade da Integração Internacional da Lusofonia Afro-Brasileira, Redenção, 2021.

Orientador: Prof. Dr. Hermínio Miguel de Oliveira Filho.

1. Inversores elétricos. 2. Processamento de sinais - Técnicas digitais. 3. Sistemas automáticos de aquisição de dados. I. Título

CE/UF/BSP CDD 621.38

SIMÃO SILVA FREITAS

DESENVOLVIMENTO DE *FIRMWARE* PARA UM INVERSOR PWM MONOFÁSICO PARA APLICAÇÕES EM QUALIDADE DE ENERGIA

Trabalho de conclusão de curso apresentado como requisito para obtenção do título de Bacharel em Engenharia de Energias da Universidade da Integração Internacional da Lusofonia Afro-Brasileira.

Orientador: Prof. Dr. Hermínio Miguel De Oliveira Filho

Aprovado em: 24/08/2021

BANCA EXAMINADORA

Prof. Dr. Hermínio Miguel De Oliveira Filho
Universidade da Integração Internacional da Lusofonia Afro-Brasileira (UNILAB)

Prof. Dr. Lígia Maria Carvalho Sousa Cordeiro
Universidade da Integração Internacional da Lusofonia Afro-Brasileira (UNILAB)

Eng. Gilmar Nunes dos Santos Costa
Universidade Federal do Ceará (UFC)

DEDICATÓRIA

Aos meus pais,
Edmilson e Graça.

AGRADECIMENTOS

O meu primeiro agradecimento é a Deus. Em segundo aos meus pais Edmilson e Graça pelo incentivo aos meus estudos, pela forma como me educaram e mostraram os caminhos a serem percorridos. Aos meus irmãos, Maria José, Sávio, Sales e Joélia, a minha avó materna Maria do Carmo e minha tia Cláudia que me deram grande suporte no início dessa caminhada, ao meu avô paterno, Nonato Martins, que acompanhou boa parte da minha jornada, mas infelizmente faleceu em 2019, me deixando uma enorme saudade de suas conversas e conselhos. Agradeço a todos vocês.

Agradeço a UNILAB, pela excelente estrutura física que me disponibilizaram para a elaboração e desenvolvimento deste trabalho e da minha formação como um todo.

Ao meu orientador Hermínio Miguel, pela excelente orientação e pela disponibilidade em sempre me ajudar, pela compreensão e conselhos. Além do orientador, aos demais membros da banca examinadora, Gilmar Nunes dos Santos Costa e Lígia Maria Carvalho Sousa Cordeiro. A todos os Professores da instituição, e não poderia deixar de agradecer aos meus professores do ensino médio e em especial a Gusmão e Cleiton, agradeço aos meus amigos, Kevin, Deyvid e Wagner pelos momentos de diversão e noites de estudos.

Agradeço aos programas de P&D e PEE da ANEEL e a ENEL Brasil, pelo suporte e financiamento do projeto em que este trabalho foi desenvolvido.

Por fim, agradeço à todos por todo o incentivo, paciência e dedicação que tiveram. Muito obrigado.

RESUMO

Com a expansão de plantas de produção de energia renovável, tem surgido desafios quanto a qualidade da energia e impactos na rede onde esses sistemas estão conectados. Com isso, é cada vez mais comum o desenvolvimento de trabalhos que propõem o desenvolvimento de estruturas capazes de mitigar problemas relacionados a injeção de potência em barramentos de distribuição. Para o devido funcionamento dessas estruturas é necessário a utilização e implementação de métodos de controle e sistemas que realizam a aquisição de variáveis de interesse. Portanto, este trabalho tem por objetivo o desenvolvimento de um *firmware*, que é composto por um sistema de aquisição de tensões de um inversor monofásico em ponte completa, e um algoritmo implementado no microcontrolador DSP (processador digital de sinais), para realizar o controle do conversor. O objetivo principal do inversor citado é a mitigação de oscilações de tensão em circuitos conectados à redes monofásicas de baixa tensão. Neste trabalho é apresentado o desenvolvimento de toda a estrutura de aquisição das tensões, tanto do lado de conexão com a rede, como do barramento CC, também são apresentados os detalhes sobre toda a configuração de um processador digital de sinais para realizar o controle do inversor, através do processador TMS320F28379D da *Texas Instruments*. Resultados de simulação são apresentados de modo a validar o sistema de aquisição proposto e a técnica de sincronismo com a tensão da rede.

Palavras-chave: Inversor PWM. Qualidade de Energia. Processador digital de sinais. Sistema de aquisição de dados.

ABSTRACT

With the expansion of renewable energy production plants, challenges have arisen in terms of energy quality and impacts on the network where these systems are connected. Thus, it is increasingly common to develop works that propose the development of structures capable of mitigating problems related to power injection in distribution buses. For the proper functioning of these structures it is necessary to use and implement control methods and systems that perform the acquisition of variables of interest. Therefore, this work aims to develop a firmware, with a voltage acquisition system for a single-phase DC-CA converter in full bridge, and an algorithm implemented in the DSP microcontroller (digital signal processor) to control the converter. The main purpose of the mentioned inverter is the mitigation of voltage fluctuations in circuits connected to low voltage single-phase networks. This work presents the development of the entire structure of voltage acquisition, both on the network connection side and on the DC bus, as well as details on the entire configuration of a digital signal processor to perform inverter control, through the Texas Instruments TMS320F28379D processor. Simulation results are presented in order to validate the proposed acquisition system and the synchronism technique with the mains voltage.

Keywords: PWM Inverter. PLL. Power Quality. Digital signal processor. Data acquisition system.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

A/D - Analógico/Digital

ADC - *Analog-to-Digital Converter* (Conversor analógico para digital)

ANEEL - Agência Nacional de Energia Elétrica

CA - Corrente Alternada

CC - Corrente Contínua

CPU - *Central Process Unit* (Unidade Central de Processamento)

DSP - *Digital Signal Processing* (Processador Digital de Sinais)

ePWM - *Enhanced Pulse Width Modulator* (Modulador de largura de pulso aprimorado)

FPU - *Floating Point Unit* (Unidade de Ponto Flutuante)

IDE - *Integrated Development Environment* (Ambiente de Desenvolvimento Integrado)

P&D - Pesquisa e Desenvolvimento

PAC - Ponto de Acoplamento Comum

PC - *Personal Computer* (Computador Pessoal)

PIE - *Peripheral Interrup Extension* (Expansão de Interrupção Periférica)

PLL - *Phase-Locked Loop*

PRODIST - Procedimentos de Distribuição de Energia no Sistema Elétrico Nacional

PROM - *Programmable Read Only Memory* (Memória Somente de Leitura Programável)

PWM - *Pulse Width Modulation* (Modulação por Largura de Pulso)

RAM - *Random Access Memory* (Memória de Acesso Aleatório)

SOGI - *Second Order Generalized Integrator* (Integrador Generalizado de Segunda Ordem)

SPWM - *Sinusoidal Pulse Width Modulation* (Modulação por Largura de Pulso Senoidal)

STATCOM - *Static Synchronous Compensator* (Compensador Síncrono Estático)

SVC - *Static Var Compensator* (Compensadores Estáticos de Reativos)

TI - *Texas Instruments*

TMU - *Trigonometric Math Unit* (Unidade Matemática Trigonométrica)

UC - Unidade de Controle

ULA - Unidade Lógica Aritmética

UNILAB - Universidade da Integração Internacional da Lusofonia Afro-Brasileira

USB - *Universal Serial Bus* (Porta Serial Universal)

VCO - Voltage Controlled Oscillator (Oscilador Controlado por Tensão)

LISTA DE SÍMBOLOS

$B0$ - Parâmetro do *loop filter*

$B1$ - Parâmetro do *loop filter*

C_f - Capacitância do filtro do inversor

f_{DSP} - Frequência do DSP

f_r - Frequência da rede elétrica

fsw - Frequência do PWM

G - Ganho genérico de um condicionamento

G_{LEM} - Ganho do sensor de tensão

$G_{LP}(s)$ - Função de transferência do *loop filter*

$G_{PI}(s)$ - Função de transferência do controlador PI

$G_{PLL}(s)$ - Função de transferência do PLL

$i_{entrada\_LEM}$ - Corrente de entrada do sensor de tensão

$i_{entrada\_LEM}$ - Corrente nominal de entrada do sensor de tensão

$I_{saída\_LEM}$ - Corrente de saída do sensor de tensão

K_i - Ganho integral

K_p - Ganho proporcional

L_f - Indutância do filtro do inversor

L_r - Indutância da rede de distribuição

$R_{entrada\_LEM}$ - Resistor de entrada do sensor de tensão

R_r - Resistência da rede de distribuição

$R_{saída\_LEM}$ - Resistor de saída do sensor de tensão

S_{inv} - Potência aparente do inversor

Ts - Tempo de estabilização

V_{ab} - Tensão de saída do inversor

V_{AD_CC} - Tensão do barramento CC após a entrada do A/D

V_{AD_PAC} - Tensão do PAC após a entrada do A/D

V_{AD_rede} - Tensão da rede após a entrada do A/D

V_{CC} - Tensão média no barramento CC

$V_{condicionamento\_CC}$ - Tensão do barramento CC após o condicionamento

$V_{condicionamento\_PAC}$ - Tensão do PAC após o condicionamento

$V_{condicionamento\_rede}$ - Tensão da rede após o condicionamento

$V_{controle\_PAC}$ - Tensão da entrada do sistema de condicionamento da tensão do PAC

$V_{controle\_CC}$ - Tensão da entrada do sistema de condicionamento da tensão do barramento CC

$v_{in}(t)$ - Sinal de entrada do PLL genérico

V_{of} - *Offset* genérico de um condicionamento

V_{PLL} - Tensão da rede modificada para a entrada do PLL

V_r - Tensão de pico da rede

V_{ref} - Sinal de referência do circuito de *offset*

$Vrms$ - Tensão eficaz da rede

$V_{saída\_LEM}$ - Tensão de saída do sensor de tensão

$v_{\alpha}(t)$ - Componente alfa do PLL

$v_{\beta}(t)$ - Componente beta do PLL

ζ - Coeficiente de amortecimento

θ_{PLL} - Ângulo de fase do PLL

θ_r - Ângulo de fase da tensão da rede

ω_n - Frequência natural do PLL

∂ - Banda de erro

SUMÁRIO

1 INTRODUÇÃO

É evidente a expansão de plantas de produção de energia renovável, principalmente a solar fotovoltaica. O desenvolvimento destas fontes de energia está ligado, principalmente, com crescimento da preocupação com questões ambientais, com destaque para os gases de efeito estufa. O crescimento da produção de energia por este tipo de fonte tem criado desafios para as empresas de distribuição de energia e entidades reguladoras no que se refere a qualidade da energia gerada e os impactos a serem refletidos na rede de distribuição. Este tema é de tamanha relevância, que existe um módulo especifico do PRODIST (Procedimentos de Distribuição de Energia no Sistema Elétrico Nacional) que trata da qualidade de energia, critérios de amostragem, valores tomados como referência e os aspectos que dizem respeito a qualidade do produto e do serviço (ENDERLE, 2012). Diante da importância do tema, são diversos os trabalhos que propõem estruturas ativas com o objetivo de mitigar problemas decorrentes da injeção de potência nos barramentos de distribuição e deste modo reduzir os impactos refletidos para outras unidades consumidoras conectadas no mesmo alimentador.

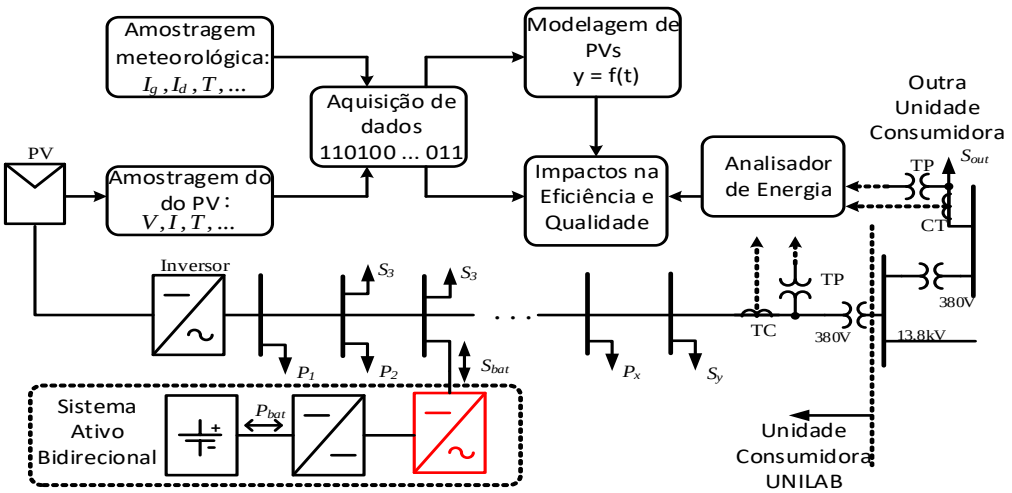
Diversos trabalhos propõem sistemas ativos que consigam operar conectados à rede elétrica convencional de forma a mitigar problemas relacionados à qualidade de energia, sejam estes harmônicos, interrupção no fornecimento ou ainda oscilação de tensão. Estes sistemas são divididos em Compensadores Estáticos de Reativos (SVC – Static Var Compensator) e Compensadores Síncronos Estáticos (STATCOM – Static Synchronous Compensator), sendo o primeiro considerado um compensador de baixa frequência e o segundo, um compensador de alta frequência. Para realizar comutação, os SVC's utilizam tiristores, sendo assim, existe a possibilidade de introduzir harmônicos de corrente de baixa frequência (MANJREKAR; VENKATARAMANAN, 1996). Um STATCOM é uma estrutura que emprega dispositivos semicondutores, que operam utilizando um determinado tipo de modulação, sendo uma estrutura que possibilita o controle da corrente reativa injetada na rede, independente da tensão, além de possibilitar uma resposta rápida e estabilidade a variações na impedância do sistema (ARNOLD, 2001). A problemática quanto a qualidade da energia também está presente em instalações com injeção de potência ativa oriunda de mini ou microgeração fotovoltaica, por exemplo, onde se faz interessante a manutenção dos barramentos e, consequentemente, conferindo melhor qualidade de energia entregue aos consumidores.

Neste sentido, o desenvolvimento de uma estrutura ativa com as características de um conversor CC-CA para garantir a interface entre um banco de baterias e uma carga elétrica,

seja a mesma uma unidade consumidora residencial ou a um circuito interno de uma unidade de maior porte, está alinhado com o escopo do projeto estratégico de P&D da chamada pública da Agência Nacional de Energia Elétrica (ANEEL) n° 001/2016 em execução na Universidade da Integração Internacional da Lusofonia Afro-Brasileira (UNILAB). O Projeto Prioritário de Eficiência Energética e Estratégico de P&D da ANEEL surge a partir da observação que gastos com energia elétrica representam uma significativa parcela do custeio das instituições públicas de ensino superior e que tais gastos poderiam ser reduzidos a partir de ações de eficiência energética e a implantação de sistemas próprios de produção de energia. O projeto desenvolvido na UNILAB trata do estudo de sistemas de monitoramento, qualidade, modelagem e mitigação de oscilações elétricas em unidades consumidoras contendo geração distribuída fotovoltaica. Neste projeto P&D é proposto um sistema ativo bidirecional para realizar a conexão de armazenadores de energia (baterias) a rede elétrica com a finalidade mitigar oscilações de tensão em circuitos estratégicos instalados na unidade acadêmica do Campus das Auroras da Universidade. O esquemático do sistema proposto como solução no projeto de P&D supracitado é apresentado na Figura 1, com destaque para o sistema ativo bidirecional.

Como pode ser observado na Figura 1, o sistema ativo é composto basicamente por dois estágios de processamento, um CC-CC e outro CC-CA, sendo o segundo estágio o objeto de interesse deste trabalho.

Figura 1 - Esquemático do sistema proposto no projeto estratégico de P&D (Chamada Pública ANEEL nº 01/2016).



Fonte: Autor.

O estágio CC-CA é realizado por um inversor monofásico operando de forma similar a um STATCOM. Os inversores, também conhecidos como conversores CC-CA, são estruturas que convertem um sinal constante de entrada, em um sinal senoidal de saída. São utilizados em várias aplicações industriais, como o acionamento de máquinas CA, aquecimento

indutivo, fontes auxiliares (CARRAH, 2010), além da integração de fontes de energias renováveis. O sinal constante de entrada pode ser qualquer tipo de fonte CC, como uma bateria, célula combustível ou uma célula solar. Para este projeto o conversor realiza a interface entre o barramento CC do sistema ativo bidirecional e o ponto de acoplamento comum (PAC) da unidade consumidora.

Quando se faz o desenvolvimento de conversores de potência, o tema controle digital está cada vez mais atrelado a eletrônica de potência. Por meio dos processadores digitais de sinais é possível realizar a implementação, em projetos de baixo custo, de técnicas de controle robustas (HOLDEFER, 2004). Os primeiros processadores digitais de sinais (DSP's) surgiram na década de 80, fabricados pela Bell Labs, mediante uma necessidade crescente por processamento de sinais. Eles são processadores com características próprias, que possuem alta capacidade para processar uma grande quantidade de cálculos. Esses dispositivos ficaram mais populares após a Texas Instruments (TI) lançar seu primeiro DSP, com objetivo de processar sinais em tempo real (ROSA; WOLTER, 2007). Os processadores atuais possuem alta performance.

Portanto o presente trabalho tem como objetivo apresentar o desenvolvimento de um *firmware* para realizar o controle de um inversor de tensão. Neste contexto, será apresentado a confecção do *firmware* no DSP, além do sistema de aquisição de dados, que realizará a interface entre o inversor de tensão e o processador de sinais.

Este trabalho está organizado em 6 capítulos. O capítulo 2 mostra uma revisão sobre o desenvolvimento de processadores digitais. O capitulo 3 mostra as partes que constituem o sistema proposto, apresentando a topologia do inversor, o sistema de aquisição de dados e o sistema de processamento de sinais. O capítulo 4 mostra um exemplo de projeto, onde é definido algumas especificações e parâmetros para realizar a simulação do sistema. O capítulo 5 apresenta os resultados de simulação para o sistema proposto. No capítulo 6 é feita uma conclusão geral e sugestões de continuidade da pesquisa, em trabalhos futuros.

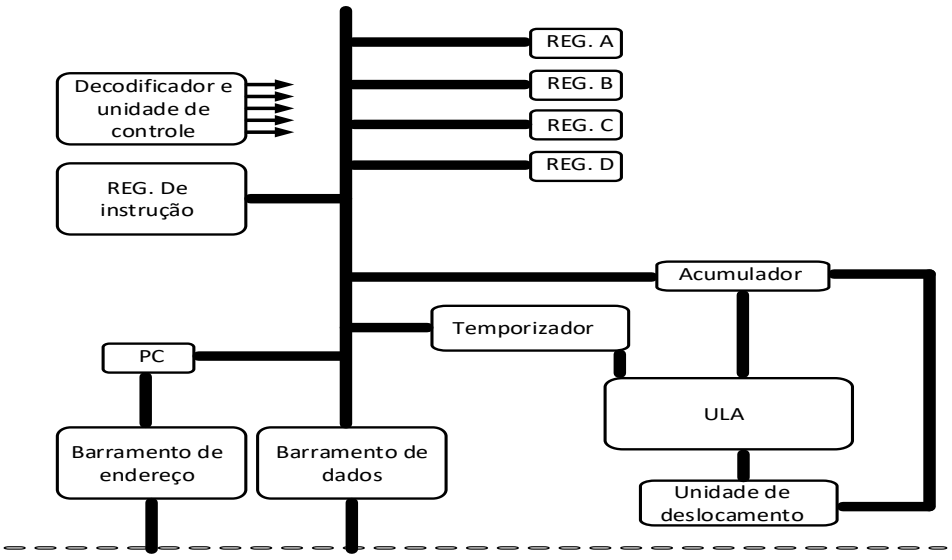
2 REVISÃO DE LITERATURA

Este capítulo mostra uma revisão sobre processamento digital de sinais, apresentando as principais ferramentas utilizadas para este fim e sua evolução no tempo. No final é mostrado o processador que será utilizado neste trabalho.

2.1 Microprocessadores

O microprocessador é um componente lógico programável, compactado em um único circuito integrado, cuja principal unidade é a central de processamento, ou a *Central Process Unit* (CPU). Foram os primeiros sistemas de controle por circuito integrado programável a serem desenvolvidos, substituindo estruturas que possuíam um número elevado de dispositivos eletrônicos (CARRAH, 2010). São usados de forma geral para processamentos complexos, pois possuem altos custos. Estas estruturas possuem ampla utilização, estando presente em computadores pessoais, aparelhos eletrônicos diversos e equipamentos biomédicos. Para configuração e utilização destes dispositivos são necessários componentes externos, como conversores A/D, pinos de entrada/saída, dentre outros. Apesar de existir diferentes fabricantes, os microprocessadores possuem vários aspectos comuns a todos, que são mostrados na Figura 2.

Figura 2 – Estrutura de um microprocessador.



Fonte: Autor.

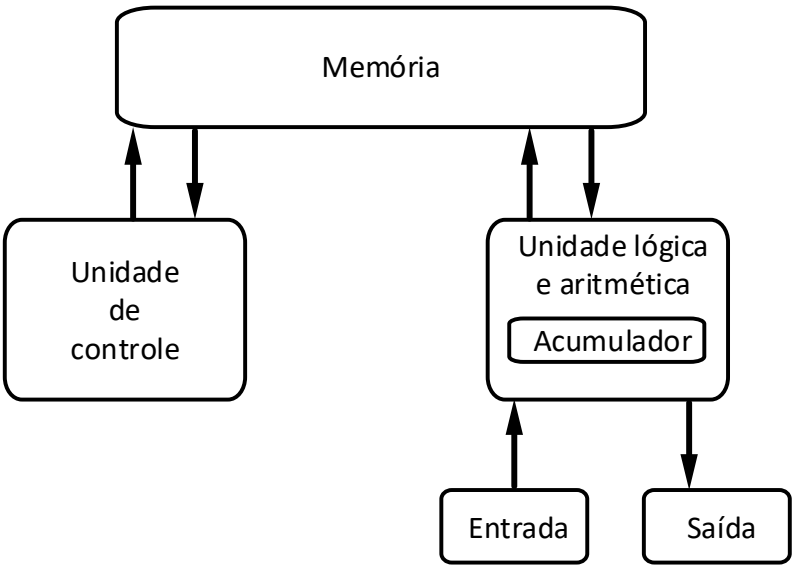
Como pode ser observado na Figura 2, um microprocessador possui duas unidades principais: A Unidade Lógica Aritmética (ULA), que tem como função a realização das

operações lógicas e aritméticas e a Unidade de Controle (UC), que realiza a decodificação e execução das instruções. Como citado anteriormente, este tipo de componente é utilizado em projetos que necessitam de um controle mais rebuscado.

2.2 Microcontroladores

Os microcontroladores de forma geral apresentam todos os componentes necessários em uma única placa, contendo: conversor analógico digital, pinos de comunicação, memórias, *timer's*, dentre outros (CARRAH, 2010). Um microcontrolador possui uma CPU que tem a finalidade de interpretar as instruções de programa, uma memória PROM onde são gravadas as instruções de programa e uma memória RAM que tem a função de armazenar as variáveis utilizadas. A Figura 3 apresenta a arquitetura de Von Neuman para um microcontrolador, e observa-se que esse tipo de arquitetura tem como principal característica a presença de uma única área de memória, armazenando as variáveis e o programa. Outra arquitetura possível é a Harvard, onde as variáveis ficam armazenadas em uma área de memória e o programa fica armazenado em outra memória.

Figura 3 – Arquitetura Von Neuman.



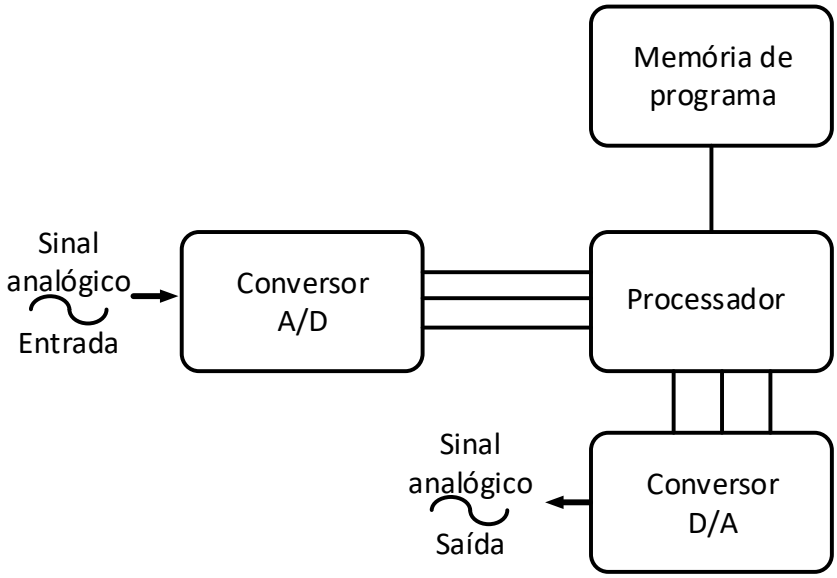
Fonte: Autor.

Os microcontroladores são amplamente utilizados em automação industrial, residencial e predial ou qualquer sistema de baixo custo que necessite de controle de dispositivos eletrônicos (CARRAH, 2010). Quando comparado a um microprocessador, possui menor desempenho, mas seu custo é relativamente baixo.

2.3 DSP´s

O Processador Digital de Sinais (*Digital Signal Processor* - DSP) é uma estrutura de controle especializada em processamento matemático, enquanto a maioria dos processadores têm como principal característica a manipulação e gerenciamento de dados. A Figura 4 mostra um diagrama genérico do funcionamento de um DSP. É um dispositivo otimizado para uso em tratamento de sinais, portanto difere dos processadores genéricos tanto em hardware, quanto em software (CARRAH, 2010). Como citado anteriormente a arquitetura Harvard tem a característica de possuir memórias diferentes para variáveis e instruções e o DSP possui esse tipo de arquitetura.

Figura 4 – Esquemático de funcionamento de um DSP.



Fonte: Autor.

Enquanto microprocessadores e microcontroladores utilizam vários ciclos de *clock* para execução de atividades mais robustas, as operações no processador digital são aprimoradas de tal forma, que a maioria das instruções são executadas em um único ciclo de operação, sendo que os dispositivos mais avançados conseguem realizar multiplicações em paralelo e operações na unidade logica aritmética (HOLDEFER, 2004). Os primeiros DSP's eram usados para aplicações militares e médicas, mas poucos anos depois seu uso se expandiu, tornando-se um processador com ampla aplicação. Os DSP's são processadores específicos para trabalhar com sinais analógicos. Devido a essa característica, as principais áreas que fazem uso deste processador supracitado são:

- Eletrônica de potência (Controle de conversores).
- Processamento de áudio (Sintetizadores, *mixers* e reconhecimento de voz).

- Processamento de imagens (aplicações biomédicas).
- Telecomunicações (filtros, multiplexação, etc).

Devido as características citadas ao longo dessa seção será utilizado um DSP para realizar o controle digital do inversor em ponte completa.

2.4 DSP utilizado

Este trabalho tem como principal objetivo apresentar o desenvolvimento de um *firmware* para um inversor de tensão, haja vista a redução dos custos dos processadores e sua alta capacidade de processamento. Com o desenvolvimento da microeletrônica, dispositivos cada vez mais compactos possuem uma alta capacidade de processamento, com a possibilidade de serem aplicados em projetos diversos.

Para realização deste projeto o processador considerado mais adequado, em relação a custos e processamento, foi o DSP. Foi escolhido o TMS320F28379D fabricado pela *Texas Instruments*, suas principais características são:

- Tipo de arquitetura:
  - Dupla CPU TMS320C28x de 32 bits;
  - 200 MHz de *clock*;
  - Unidade de ponto flutuante de precisão única IEEE 754 (FPU);
  - Unidade Matemática Trigonométrica (TMU);
  - Unidade matemática complexa (VCU-II).
- Dois Aceleradores de Lei de Controle (CLAs) programáveis:
  - 200 MHz de *clock*;
  - Instruções de ponto flutuante de precisão simples IEEE 754;
  - Executa código independentemente da CPU principal.
- Memória no chip:
  - 1 MB (512 KB) de *flash*;
  - 204 KB (102 KB) de RAM;
  - Segurança de zona dupla;
  - Número de identificação único.
- Relógio e controle do sistema:

- Dois osciladores internos de pino zero de 10 MHz;
- Oscilador de cristal on-chip;
- Módulo de cronômetro de *watchdog* em janela.

Para este trabalho o DSP tem a função de receber, via sistema de aquisição de dados, as amostras dos sinais de tensões no ponto de acoplamento do inversor, na rede e no barramento CC e realizar o processamento desses sinais através do algoritmo construído no ambiente de programação e compilação do fabricante, o CCS (*Code Composer Studio*). O acionamento das chaves do inversor será realizado por modulação SPWM, usando o módulo ePWM a partir da configuração de seus registradores apropriados. A partir do sinal de tensão da rede obtido pelo sistema de sensoriamento, será executado o código do sistema de sincronismo com a rede. Portanto todos os sinais de controle do inversor serão processados ou gerados no DSP.

2.5 Considerações finais

Neste capítulo foi apresentada uma revisão sobre o processamento digital de sinais e as principais estruturas que podem ser utilizadas. Foi mostrado que o DSP possuía características adequadas para ser utilizado neste trabalho, pois apresenta em uma única placa todos os periféricos necessários, além do seu baixo custo. Por fim, foram expostas as principais características do processador utilizado.

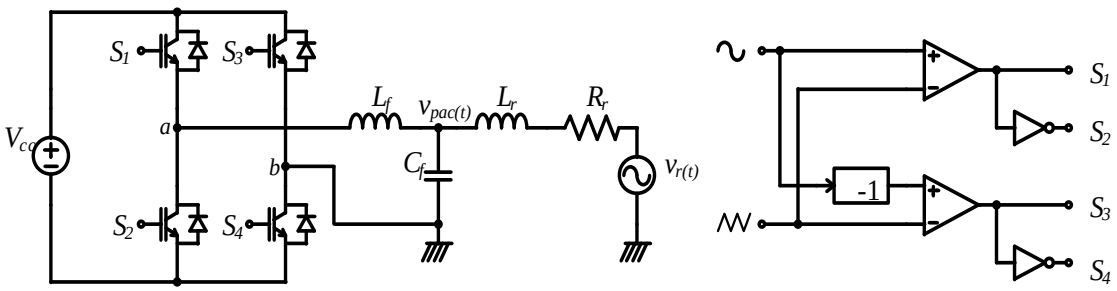
3 ANÁLISE DO SISTEMA DE AQUISIÇÃO E CONTROLE

Neste capítulo serão apresentados os elementos constituintes do sistema de aquisição e controle. Inicialmente é feita a descrição do inversor, pois o sistema supracitado irá operar sobre este conversor CC-CA. Em seguida é discutido a necessidade de sensoriamento e condicionamento para amostrar variáveis de interesse, para posteriormente serem processadas digitalmente. Os detalhes sobre o kit de DSP utilizado, também são apresentados, mostrando os principais módulos presentes e suas funções. Na última seção é descrita a metodologia utilizada no desenvolvimento do circuito de sincronismo com a rede elétrica, o PLL.

3.1 Topologia do inversor e método de modulação

A interface de conexão do *firmware* desenvolvido neste trabalho, com a rede elétrica, será realizada utilizando um inversor monofásico em ponte completa, pois essa estrutura apresenta flexibilidade quanto a técnica de modulação, sendo bastante conhecida na literatura. Uma das principais vantagens da utilização deste tipo de inversor é o fato de conseguir-se baixas taxas de distorção harmônica total, mesmo com um número reduzido de chaves e filtros pequenos (SOOMRO et al., 2016). Dessa forma, é mostrado na Figura 5 a estrutura proposta, onde é apresentado o modelo simplificado do conversor CC-CA conectado à rede elétrica. O filtro de saída (L_f e C_f) é responsável pela eliminação ou redução significativa das harmônicas. Os elementos L_r e R_r representam a impedância da rede de distribuição. Como pode ser observado na figura, as chaves do inversor são comandadas por sinais independentes, sendo assim cada chave precisa de um sinal específico, para se obter a forma de onda desejada na saída V_{ab}.

Figura 5 - Inversor conectado à rede e esquema da modulação SPWM unipolar.

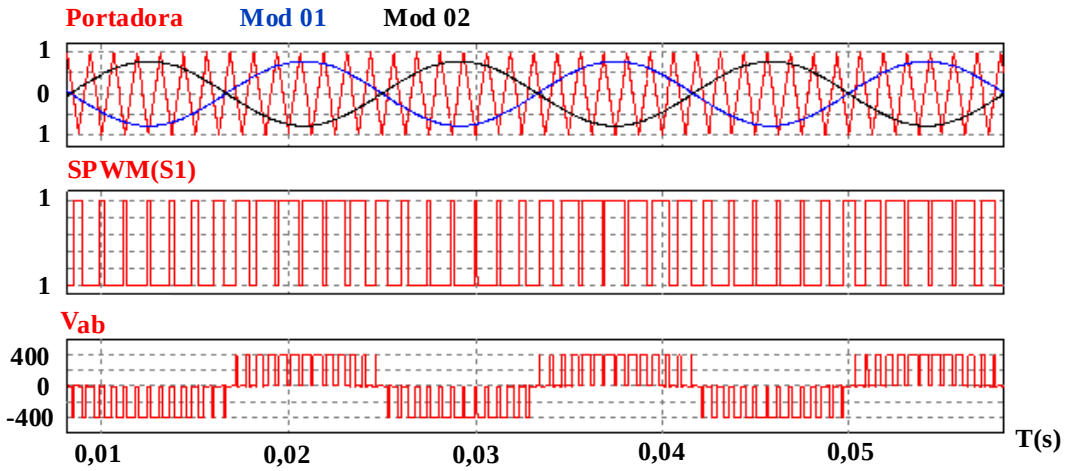


Fonte: Autor.

No esquema da modulação, do tipo PWM senoidal, ou SPWM, é possível observar que nas entradas não inversoras dos comparadores são conectadas senoides, que modulam o sinal de saída para ter a mesma frequência fundamental, que neste caso, é a frequência da rede elétrica (HART, 1997). Na entrada inversora é conectada a portadora, com um sinal no formato de triangular em alta frequência. Com a utilização deste sinal em alta frequência, é possível reduzir o volume dos elementos do filtro de saída.

Na Figura 6 são apresentadas as principais formas de onda referentes ao tipo de modulação que será utilizada neste trabalho. Como se pode observar, as moduladoras são sinais senoidais e estão defasadas, entre si, de 180º. A portadora triangular possui frequência bem mais alta em relação a moduladora.

Figura 6 – Principais formas de onda para a modulação SPWM unipolar.

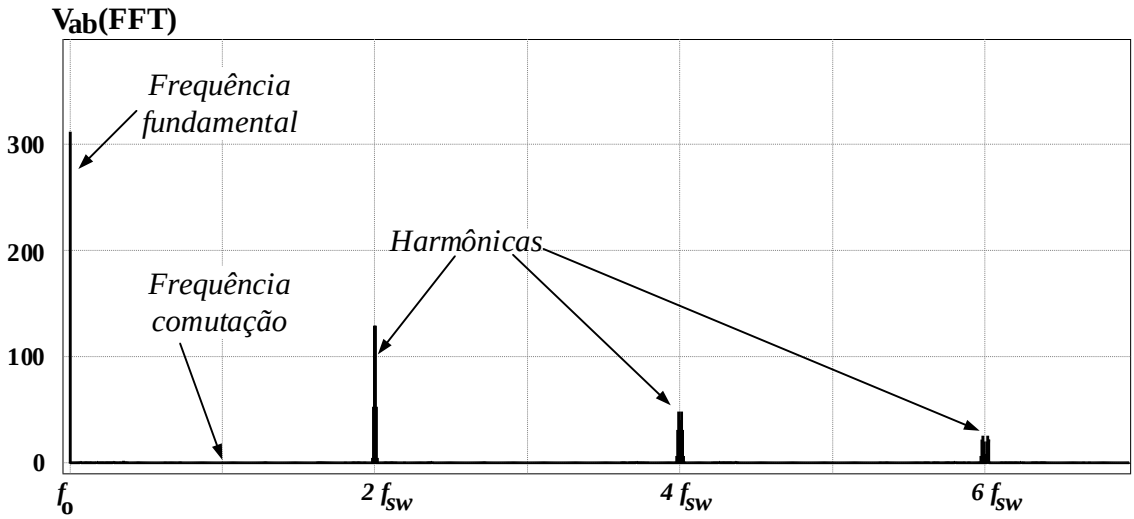


Fonte: Autor.

A partir da comparação dos sinais da moduladora e da portadora é possível sintetizar um sinal com largura de pulso variável, sendo o mesmo responsável pelo comando das chaves do inversor. Na Figura 6 este pulso de acionamento é representado pela segunda forma de onda (SPWM (S1)), retratando neste caso o sinal de comando da chave S1 do conversor, conforme a Figura 5. É possível observar que a largura do pulso do sinal de acionamento tem relação direta com a amplitude da moduladora. A tensão de saída V_{ab} é um sinal alternado em três níveis ($+V_{cc,}$ $-V_{cc}$ e 0), que é característico da modulação SPWM unipolar.

Como já foi discutido anteriormente, uma das principais vantagens da utilização da modulação SPWM unipolar, corresponde a possibilidade da utilização de filtros de magnitude, volume e tamanho relativamente reduzidos. O sinal de saída do inversor V_{ab} possui harmônicas apenas na frequência fundamental e em frequências múltiplas do dobro da frequência de chaveamento do inversor, como pode ser observado na Figura 7.

Figura 7 – Espectro em frequência da tensão V_{ab} no inversor.

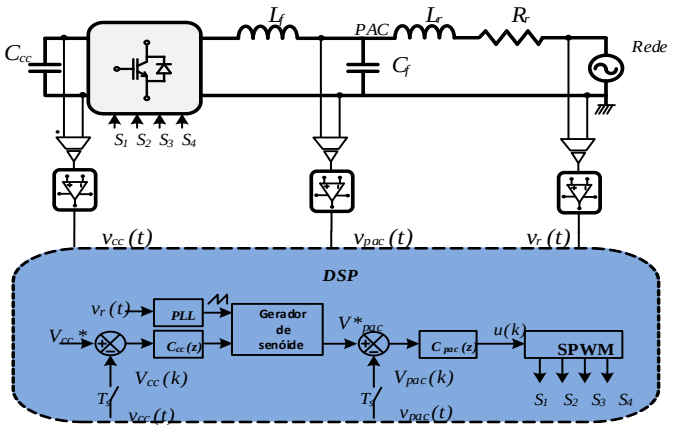


Fonte: Autor.

3.2 Sistema de controle e aquisição de sinal

A Figura 8 mostra o sistema de controle implementado no inversor citado, com o acionamento dos interruptores e o circuito PLL, toda essa estrutura foi implementada no DSP. A estrutura anterior ao processador, é composta pelo sensoriamento e condicionamento. Os sinais de acionamento dos interruptores, provenientes do DSP, são conectados ao conversor por meio de drivers.

Figura 8 – Inversor proposto com sistema de controle e acionamento.



Fonte: Autor.

3.2.1 Sistema de sensoriamento

Para realizar o processamento digital de sinais, um passo fundamental é o uso de sensores para amostrar sinais de interesse. Sensores podem ser entendidos como dispositivos que tem a capacidade de adquirir informações de grandezas físicas (KONZEN, 2019). O sistema de sensoriamento deste trabalho é composto por três sensores de tensão, um para a tensão no PAC, outro para a tensão da rede e o terceiro para o barramento CC. A etapa de sensoriamento representa o primeiro estágio de processamento dos sinais do inversor de tensão, que posteriormente, serão enviados para o sistema de condicionamento e em seguida para o DSP. Os sensores de tensão utilizados neste trabalho são fabricados pela LEM e o modelo utilizado foi o LP-20. A Tabela 1 mostra as principais caraterísticas do sensor supracitado.

Tabela 1 – Características do sensor de tensão.

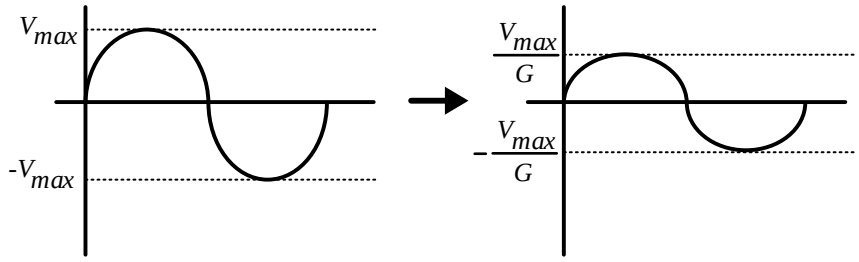
Fabricante	LEM
Modelo	LP-20
Corrente nominal de entrada ($i_{entrada\_LEM}$)	10 mA
Ganho (G_{LEM})	2500:1000

Fonte: Autor.

3.2.2 Sistema de condicionamento de sinal

Como discutido anteriormente, a medição dos sinais de tensão será realizada por meio de sensores. Porém é comum que os sinais provenientes da parte de sensoriamento possuam baixa intensidade, ruídos e outros distúrbios, devendo passar por um tratamento antes de serem analisados por um processador digital de sinais (KONZEN, 2019). Esta etapa serve de interface entre os sensores e o DSP, realizando a função de adequação do sinal para ser processado. O condicionamento de sinal possui várias etapas, que dependem das características dos sensores utilizados e as especificações do conversor A/D. Uma das etapas mais comuns, é condicionamento da amplitude, consistindo em proporcionar um ganho aos sinais, conforme mostrado Figura 9, onde o G representa o ganho. Esse processo consegue garantir que o sinal possua uma amplitude adequada a entrada do conversor A/D.

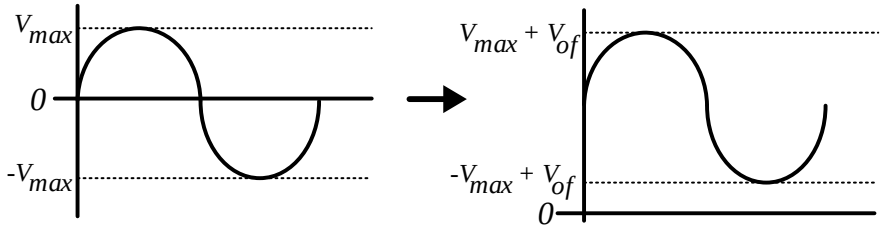
Figura 9 – Condicionamento da amplitude de um sinal.



Fonte: Autor.

Outra etapa extremamente importante diz respeito a inserção de *offset* nos sinais provenientes do sensoriamento, pois os processadores digitais de sinais não permitem que um conversor A/D receba uma tensão menor que zero. Esta etapa é ilustrada na Figura 10, onde V_{of} é valor de *offset* a que o sinal foi submetido. Para este trabalho a estrutura de condicionamento deve proporcionar níveis tensão de 0 a 3,3 V, pois o processador digital de sinais possui tais limites para seus conversores A/D.

Figura 10 – Adição de offset em um sinal.

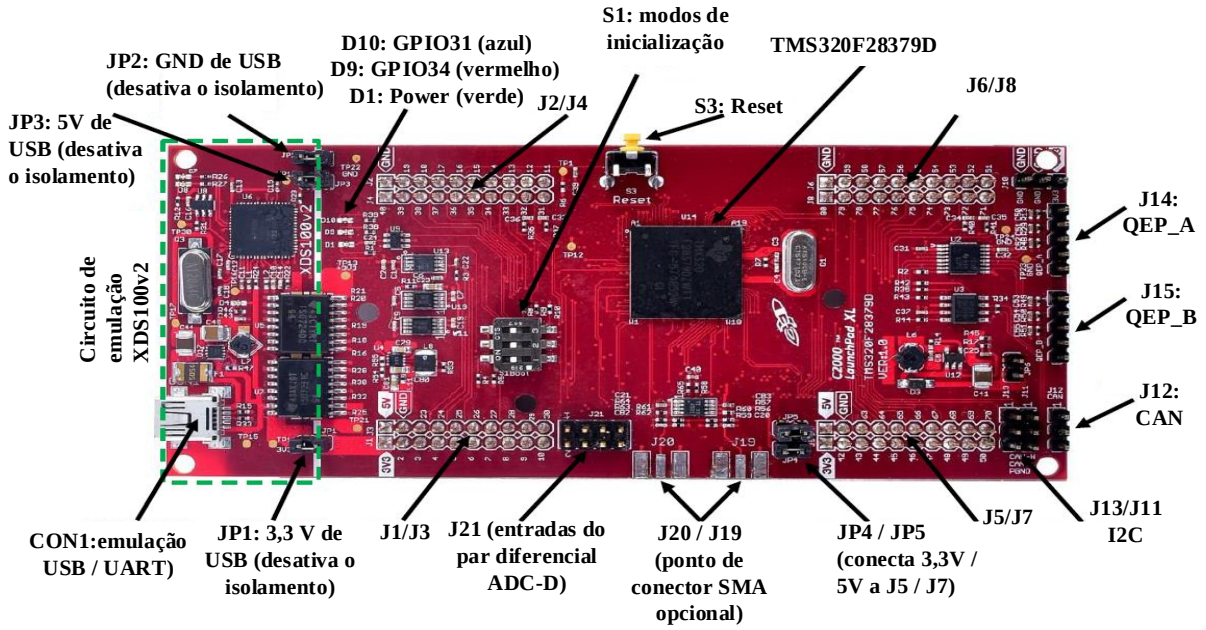


Fonte: Autor.

3.3 Estrutura do DSP

A Figura 11 mostra a estrutura de hardware do kit de desenvolvimento DSP *texas instruments* (TI) que será utilizado neste trabalho, no qual é possível observar os pinos de entrada/saída, o processador TMS320F28379D, conexões USB, botão de reset, dentre outros. A TI desenvolveu o IDE (*Integrated Development Environment)* para as principais famílias de processadores, neste ambiente de desenvolvimento é possível realizar a confecção de algoritmos, que serão carregados na placa por meio de conexão USB com um PC.

Figura 11 – Placa do DSP utilizado.

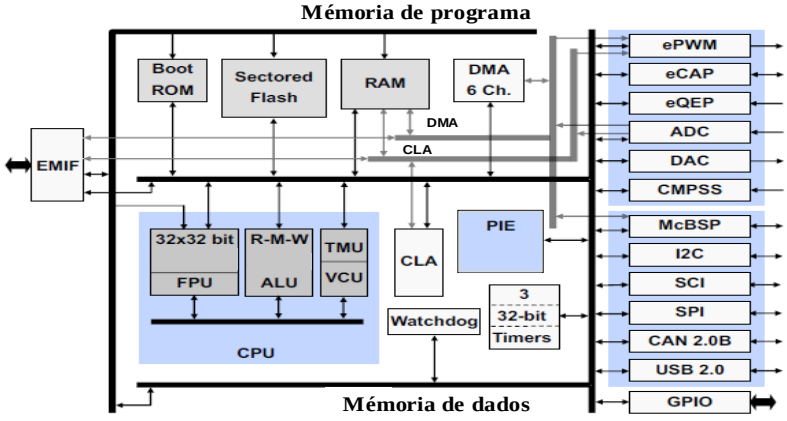


Fonte: Modificado de *Texas Instruments* (2019).

Na Figura 12 são apresentados os principais módulos presentes no processador de sinais. É possível observar que a CPU é um processador de ponto fixo de 32 bits, possuindo arquitetura Harvard modificada. Possui uma unidade de ponto flutuante (FPU) capaz de suportar operações de ponto flutuante de precisão única. A unidade trigonométrica (TMU), substitui a necessidade de integrar uma biblioteca de funções trigonométricas, além de realizar cálculos mais rápidos e com maior precisão. Os principais módulos utilizados neste projeto são:

- ePWM;
- ADC;
- PIE (Módulo de interrupções).

Figura 12 – Principais módulos presentes no DSP.

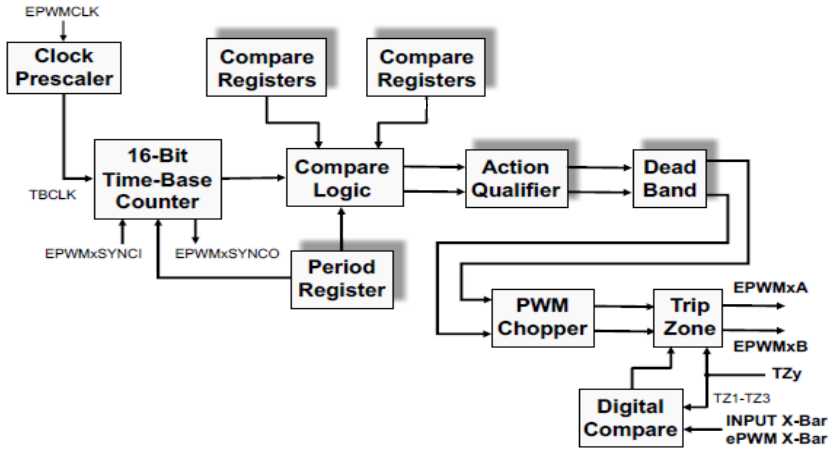


Fonte: Modificado de *Texas Instruments* (2019).

3.3.1 Módulo ePWM

A Figura 13 mostra um diagrama que representa os submódulos presentes no módulo ePWM. Este modulo é um modulador de largura de pulso aprimorado, sendo essencial para aplicações em eletrônica de potência, tendo aplicação em controle digital de máquinas elétricas, fontes de alimentação ininterrupta e conversores de energia em geral. Um DSP possui múltiplos módulos ePWM com um contador de 16 bits e saída dupla, que são sinais complementares (PWMA e PWMB).

Figura 13 – Diagrama do módulo ePWM.



Fonte: Modificado de *Texas Instruments* (2019).

A partir da configuração dos registradores presentes em cada submódulo é possível obter uma forma onda de saída de acordo com os parâmetros desejados. Os principais blocos mostrados na Figura 13 e suas funções, são:

- *Clock Prescaler*:
  - Habilita o *clock* para um canal especificamente.
- *Time Base*:
  - Configura o período do PWM com base em sua frequência;
  - Define o formato da dente-de-serra em crescente, decrescente ou triangular;
  - Sincroniza o contador dos módulos PWM.
- *Counter Compare*:
  - Configura o ciclo de trabalho;
  - Configura o momento de ocorrência dos eventos de comutação.
- *Action Qualifier*:
  - Especifica a saída do PWM (0 ou 1) quando ocorre uma comparação do contador;

- Controla o tempo morto, através de *software*;
- Configura a saída de um PWM, através de controle de *software*.

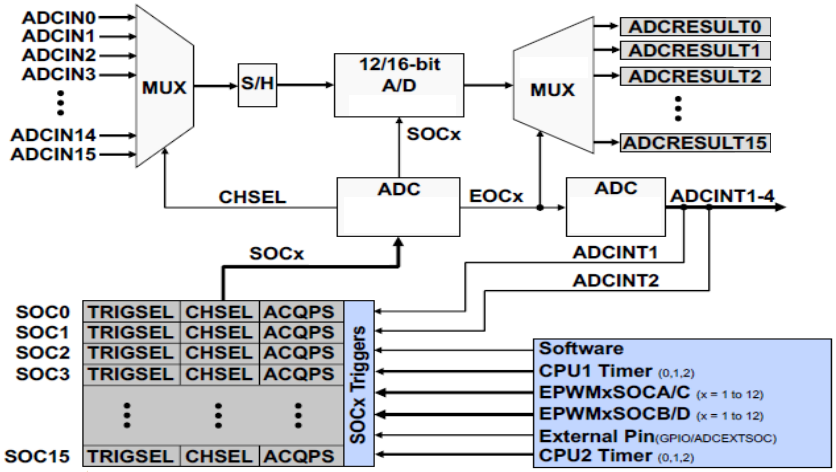
- *Dead Band*:
  - Controla o tempo morto entre interruptores superiores e inferiores;
  - Configura a borda ascendente e ascendente de saída;
  - Possibilita a mudança de fase entre o PWMB e o PWMA.
- *PWM Chopper*:
  - Cria uma frequência de corte;
  - Configura a largura de pulso do primeiro pulo no trem de pulso cortado;
  - Pode ser configurada para ignorar esse submódulo, gerando o PWM sem modificação.
- *Trip Zone*:
  - Configura o PWM para reagir a sinais de comparação digital;
  - Especifica o valor que deve ser setado na saída do PWM, na ocorrência de uma falha.
- *Event Trigger*:
  - Configura os eventos que poderão disparar uma interrupção;
  - Configura eventos que poderão iniciar uma conversão ADC.
- *Digital Compare*;
  - Configura opções de filtragem de eventos para capturar o contador.

3.3.2 Módulo ADC

Para que os sinais de interesse, provenientes do sistema de condicionamento, possam ser processados digitalmente, é necessário convertê-los em um sinal discreto no tempo, para isso é necessário a utilização de um conversor A/D. A Figura 14 mostra o diagrama do módulo ADC do processador digital de sinais. O mesmo possui resolução selecionável de 16 ou 12 bits, sendo do tipo aproximação sucessiva, onde a saída digital é comparada com a entrada analógica. O ADC pode ser dividido em duas partes principais, os circuitos analógicos responsáveis pela seleção do canal, o circuito de *Sample & Hold*, e demais circuitos de suporte analógico. A segunda parte é composta por circuitos digitais, responsáveis pela lógica para conversões, configuração de registradores de resultados e demais circuitos complementares digitais. O circuito de *Sample & Hold* é responsável pelo processo de amostragem, obtendo amostras do sinal periodicamente. Este circuito de amostragem pode ser entendido como um

chaveamento, que faz a determinação do momento em que a leitura deve ser realizada, e a um capacitor que funciona como uma memória (RAUTH; RANDAL, 2005).

Figura 14 – Diagrama do ADC.



Fonte: Modificado de *Texas Instruments* (2019).

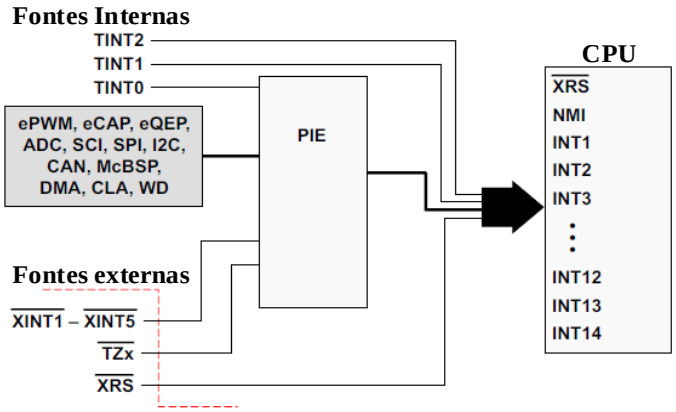
Como pode ser observado na Figura 14 a base do módulo ADC são os SOC'S (*start-of-conversion*), sendo a principal unidade de configuração para que ocorra o processo de conversão digital. O modelo de DSP utilizado neste trabalho possui 16 SOC's configuráveis. Como mostrado, é preciso configurar o evento que irá disparar o ADC, e pode ser gerenciado por software, *timer*, um módulo PWM ou por um pino externo, sendo que essa escolha é feita pelo registrador TRIGSEL. Cada unidade de conversão (SOC) possui até 16 canais possíveis para realizar a conversão e a escolha do canal que será utilizado é feita por meio do registrador CHSEL. Outro parâmetro importante é o tamanho da janela de aquisição, que corresponde a duração de tempo que o capacitor de amostragem pode ser carregado. O tamanho dessa janela deve ser ajustado de acordo com a resolução que foi escolhida para o ADC, e seu valor é configurado pelo registrador ACQPS. Após a configuração destes registradores principais, o modulo iniciará a conversão e o resultado desse processo, ou seja, o sinal de saída do ADC, é enviado para o registrador ADCRESULTx, onde x é o número do canal utilizado.

3.3.3 Módulo de interrupções (*Peripheral interrup extension* - PIE)

Uma interrupção pode ser definida como uma estrutura de *hardware* ou *software* que faz a CPU realizar uma pausa em sua atividade atual e realize outra instrução definida na

rotina de interrupção. Esse tipo de estrutura força que a CPU processe os eventos no seu momento de ocorrência, evitando a necessidade *polling* para confirmar se um evento realmente ocorreu (AVELINO, 2009). No caso do DSP utilizado, quando uma interrupção é enviada para a CPU, uma tabela de vetores com todas interrupções é acessada, sendo feita a busca pela rotina que foi enviada. Na Figura 15 é mostrado um diagrama do módulo de expansão de interrupção periférica para o DSP da *Texas Instruments*. Como pode observado, a CPU possui 14 possibilidades de interrupção, sendo que os *timers* 1 e 2 estão conectados diretamente a ela. As outras 12 estão associadas ao PIE (Módulo de Expansão de Interrupção Periférica), que faz a multiplexação em 16 outras interrupções, gerando uma tabela de interrupções. Os outros módulo presentes no DSP, assim como fontes externas também podem funcionar como fontes de interrupção.

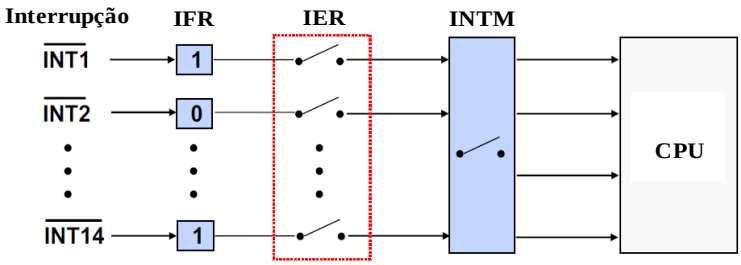
Figura 15 – Diagrama do módulo de interrupções.



Fonte: Modificado de *Texas Instruments* (2019).

A Figura 16 mostra um diagrama com os principais registradores necessários para fazer a configuração de uma interrupção. O registrador IER de 16 bits é responsável pela habilitação tanto da linha quanto da coluna da tabela, da respectiva interrupção. O registrador IFR é um flag, um sinalizador de ocorrência de um evento de interrupção. O último registrador mostrado, INTM, é uma interrupção global e pode ser utilizado para desabilitar todas as interrupções.

Figura 16 – Diagrama de processamento de uma interrupção.



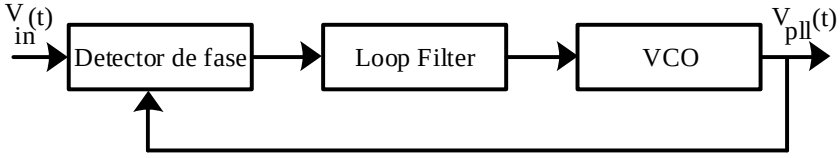
Fonte: Modificado de *Texas Instruments* (2019).

3.4 Circuito PLL

Uma estrutura de processamento de energia que está conectada à rede, tem como característica a necessidade de sincronismo entre o sinal da tensão de saída (PAC) com a rede elétrica. A detecção da passagem por zero é o método de sincronismo mais simples, pois determina a frequência e a fase da rede quando sua tensão passa de positiva para negativa e vice-versa. O principal problema deste método é sua sensibilidade ao ruído, ocasionando erro na detecção da fase (CHOI; KIM; KIM, 2006). Outra método é a detecção da fase a partir da tensão de uma das fases. A presença de ruídos ou defasagens são responsáveis por provocar erros consideráveis nos sinais de sincronismo.

Desta forma, malhas de captura de fase ou PLL (*Phase Locked Loop*) são circuitos mais sofisticados que detectam com precisão a fase do conjunto de tensões que formam as componentes de sequência positiva, sendo uma solução para o problema de ruídos. A Figura 17 apresenta um diagrama de blocos da estrutura básica de um PLL. Como pode ser observado a estrutura básica é composta por três blocos principais: um detector de fase, um compensador de fase, também chamado na literatura de *loop filter* e um oscilador controlado por tensão (VCO).

Figura 17 - Diagrama de blocos de um PLL.



Fonte: autor.

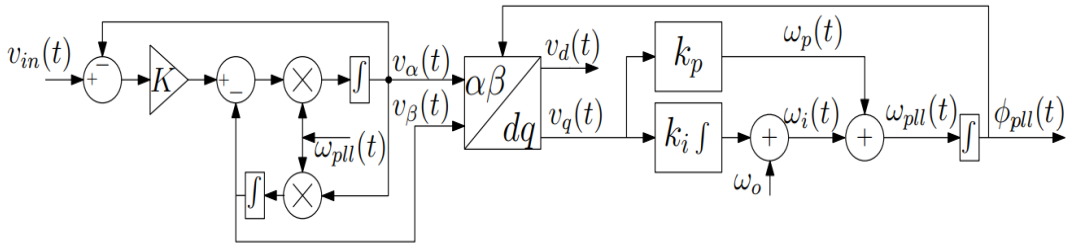
O *loop filter* pode ser entendido como um controlador proporcional-integral, resultando indiretamente em um filtro passa-baixas (BRENNAN, 1996). O detector de fase gera

um sinal de erro referente a diferença de fase entre o sinal produzido pelo VCO e componente fundamental do sinal de entrada ($V_{in}(t)$) (LESSA, 2019).

3.4.1 SOGI-PLL

A Figura 18 mostra a estrutura de um Integrador Generalizado de Segunda Ordem (SOGI-*Second Order Generalized Integrator*), aplicado a um PLL, resultando em um SOGI-PLL. Conforme observado, K representa o ganho da malha do SOGI, $v_{in}(t)$ é o sinal de entrada, o sinal $v_{\alpha}(t)$ está em sincronismo com a componente fundamental do sinal de entrada e o sinal $v_{\beta}(t)$ estar deslocado 90º em relação a $v_{\alpha}(t)$ (LESSA, 2019).

Figura 18- Diagrama de blocos de um SOGI-PLL.



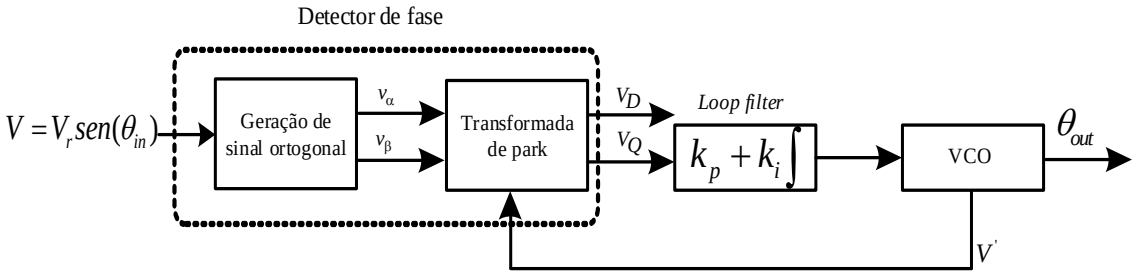
Fonte: Lessa (2019).

Portanto, como pode ser deduzido, SOGI é uma estrutura capaz de produzir dois sinais em quadratura, a partir de um sinal senoidal de entrada, sendo a estrutura mais utilizada para esse fim.

3.4.2 PLL proposto

A Figura 19 apresenta o diagrama de blocos da estrutura de PLL proposta neste trabalho. A geração de um sinal ortogonal seguido por uma transformada de Park tem como objetivo a eliminação do sinal com o dobro da frequência da rede, presente na saída da detecção de fase. O sinal ortogonal é gerado por um SOGI, já discutido na seção anterior. Após a geração do sinal ortogonal, a transformada de Park é usada para detectar os componentes *D* e *Q*.

Figura 19 – Digrama do PLL proposto.



Fonte: autor.

A tensão de entrada do PLL, pode ser escrita da seguinte forma:

$$V = V_r sen(\omega_r t + \theta_r) \tag{1}$$

Assumindo que o VCO está gerando uma onda senoidal em sincronismo com a rede, então:

$$V' = \cos(\theta_{out}) = \cos(\omega_{PLL} t + \theta_{PLL}) \tag{2}$$

O bloco referente a detecção de fase tem como função comparar a senoide de entrada como o senoide de saída do VCO, gerando um sinal de erro proporcional ao do ângulo de fase. O bloco citado faz esse cálculo de erro, a partir da multiplicação do sinal de saída de VCO pela senoide de entrada, obtendo:

$$V_Q = \frac{V_r}{2}[sen((\omega_r - \omega_{PLL})t + (\theta_r - \theta_{PLL})) + sen((\omega_r + \omega_{PLL})t + (\theta_r - \theta_{PLL}))] \tag{3}$$

Ignorando o dobro de componente de frequência da rede, tem-se:

$$\overline{V_Q} = \frac{V_r}{2} sen((\omega_r - \omega_{PLL})t + (\theta_r - \theta_{PLL})) \tag{4}$$

Considerando a operação em estado estacionário e que para pequenos ângulos sen(θ) é aproximadamente igual a θ, o sinal de erro obtido é:

$$E = \frac{V_r(\theta_r - \theta_{PLL})}{2} \tag{5}$$

A função de transferência do PLL é dada por:

$$G_{PLL}(s) = \frac{\theta_{out}(s)}{\theta_{in}(s)} = \frac{V_r(K_p s + \frac{K_p}{T_i})}{s^2 + V_r K_p s + V_r \frac{K_p}{T_i}} \tag{6}$$

A função de transferência apresentada em (6) pode ser comparada com a função de transferência de segunda ordem genérica mostrada em (7).

$$G(s) = \frac{2\xi\omega_n s + \omega^2{}_n}{s^2 + 2\xi\omega_n s + \omega^2{}_n} \tag{7}$$

A partir dessa comparação, tem-se:

$$\omega_n = \sqrt{\frac{V_r K_p}{T_i}} \tag{8}$$

$$\xi = \sqrt{\frac{V_r T_i K_p}{4}} \tag{9}$$

Considerando que o *loop filter* pode ser modelado como um dispositivo que possua a função de transferência de um filtro passa baixa de segunda ordem (BRASIL, 2013). Dessa forma:

$$G_{LP}(s) = \frac{\omega^2{}_n}{s^2 + 2\xi\omega_n s + \omega^2{}_n} \tag{10}$$

A equação (10) pode ser entendida como a resposta ao degrau para um sistema de segunda ordem, conforme modelado pela equação abaixo:

$$y(t) = 1 - ce^{-\sigma t} sen(\omega_d t + \varphi) \tag{11}$$

Como o *loop filter* pode ser modelado por um controlador PI, pode ser feita a atribuída ao mesmo a equação (12).

$$G_{LP}(z) = \frac{B0 + B1z^{-1}}{1 - z^{-1}} \tag{12}$$

A função de transferência de um controlador PI é dada por:

$$G_{PI}(s) = K_p + \frac{K_i}{s} \tag{13}$$

Considerando o método Tustin para discretização de funções no domínio da frequência:

$$s = \frac{2}{T}\left(\frac{z-1}{z+1}\right) \tag{14}$$

Substituindo (14) em (13), é obtido:

$$G(z)_{PI} = \frac{\left(\frac{2K_p + K_iT}{2}\right) - \left(\frac{2K_p - K_iT}{2}\right)z^{-1}}{1 - z^{-1}} \tag{15}$$

É comumente atribuído um PI ao *loop filter* (BRASIL, 2013). Sendo assim:

$$B0 = \frac{2K_p + K_iT}{2} \tag{16}$$

$$B1 = -\left(\frac{2K_p - K_iT}{2}\right) \tag{17}$$

A partir dos cálculos desenvolvidos nesta seção é possível realizar o projeto do filtro do algoritmo de PLL.

3.5 Considerações finais

Neste capítulo foi demonstrado os principais componentes do *firmware* proposto. Foi citado que a estrutura possui um sistema de aquisição de sinal, por meio de sensoriamento e condicionamento. Em seguida foi apresentado o DSP que foi utilizado, mostrando suas principais características. Por fim, foi desenvolvido um algoritmo de PLL que foi implementado no processador de sinais.

4 EXEMPLO DE PROJETO

Neste capítulo foi realizado um exemplo de projeto para o *firmware* proposto. Foi feita a modelagem do sistema de aquisição de dados, configuração dos módulos do DSP e cálculo do *loop filter* do algoritmo de sincronismo.

4.1 Especificações e parâmetros de projeto

Na Tabela 2 são apresentadas as especificações, e na Tabela 3 os parâmetros assumidos, que representam, respectivamente, as especificações do exemplo de projeto, assim como os parâmetros assumidos para realização da modelagem e simulação do sistema de aquisição e do sistema de sincronismo. Os parâmetros de rede utilizados são baseados em sistemas de distribuição em tensão alternada 220V/60Hz com impedância de 0,2 p.u relativa a potência nominal do barramento de distribuição genérico. Os detalhes da escolha das especificações e os cálculos dos parâmetros, podem ser consultados em Costa et al. (2020).

Tabela 2 – Especificações do inversor.

Tensão média no barramento CC V_{CC}	400 V
Tensão eficaz da rede *Vrms*	220 V
Frequência da rede elétrica f_r	60 Hz
Potência aparente do inversor S_{inv}	3,5 kVA

Fonte: Autor.

Tabela 3 – Parâmetros assumidos para o DSP.

Frequência do DSP f_{DSP}	200 MHz
Frequência do PWM *fsw*	20 kHz
Resolução do PWM	16 bits
Resolução do A/D	12 bits

Fonte: Autor.

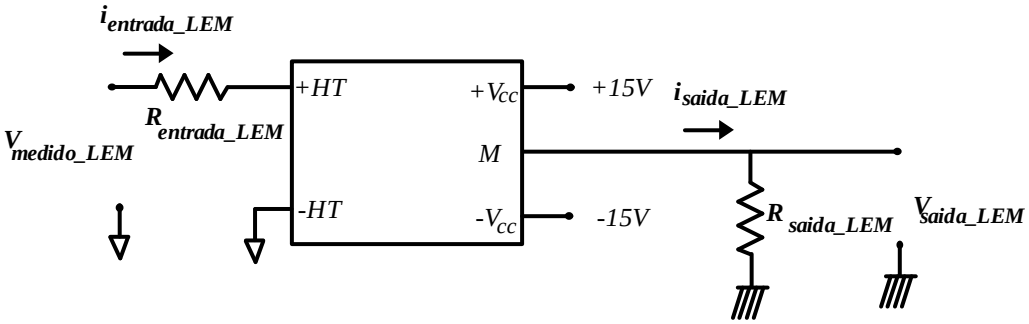
4.2 Modelagem do sistema de aquisição.

Nesta seção será apresentado o circuito do sistema de sensoriamento e condicionamento, bem como os cálculos dos componentes destes circuitos.

4.2.1 Modelagem do sistema de sensoriamento

O sensor de tensão que foi utilizado neste trabalho é detalhado na Tabela 1 e, como foi mencionado no capítulo 3, para o correto funcionamento do sensor, é necessário projetar seu circuito. A Figura 20 mostra o esquemático do circuito do sensor, onde é possível observar a necessidade de realizar o projeto de resistências na entrada e saída.

Figura 20 – Esquemático do circuito do sensor de tensão.



Fonte: Autor.

A resistência de entrada do circuito ($R_{entrada\_LEM}$), deve ser projetada de modo a garantir que a corrente ($i_{entrada\_LEM}$) de entrada do sensor não ultrapasse seu valor nominal, de 10 mA. Para a tensão de entrada (V_{medido_LEM}) é considerado que o máximo valor será, 1,4 vezes a tensão de pico da rede (OLIVEIRA, 2018). O valor comercial do resistor encontrado em (18), é 47 kΩ e de pelo menos 3 W. O valor da corrente de entrada é 9,3 mA.

$$R_{entrada\_LEM\_ca} = \frac{1,4 \cdot 311}{10 \cdot 10^{-3}} = 43,54\,\text{k}\Omega \tag{18}$$

A Tabela 1 apresentou as especificações do sensor, bem como o seu ganho. Considerando um ganho de 0,0016 para o circuito do sensor, de modo a garantir níveis adequados de tensão ao conversor A/D. Portanto, o ganho total será obtido a partir da multiplicação dessas duas parcelas citadas. Dessa forma, é possível calcular a resistência de saída de acordo com (19). Para um melhor ajuste do valor calculado será utilizado um potenciômetro de 0-200 Ω.

$$R_{saída\_LEM\_ca} = \frac{1,79}{9,3 \cdot 10^{-3} \cdot 2,5} = 76,98\,\Omega \tag{19}$$

Os cálculos realizados em (18) e (19) serão utilizados para o sensor da tensão da rede e da tensão no PAC. Para o sensor do barramento CC do inversor, será considerado que a

máxima tensão de entrada, terá um valor igual a 1,4 vezes a tensão média do barramento CC (OLIVEIRA, 2018). Portanto, a resistência de entrada é:

$$R_{entrada\_LEM\_cc} = \frac{1,4 \cdot 400}{10 \cdot 10^{-3}} = 56\,\text{k}\Omega \tag{20}$$

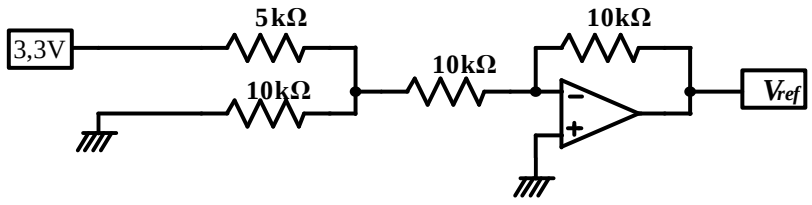
A resistência calculada em (20) é um valor comercial, e a potência para o resistor é de 6 W. Considerando um ganho de 0,00084 para o circuito do sensor, de modo a garantir níveis adequados de tensão ao conversor A/D. Portanto, o ganho total será obtido a partir da multiplicação dessa parcela pelo ganho do sensor. Dessa forma é possível calcular o resistor de saída, conforme (21). Também, Para um melhor ajuste do valor calculado será usado um potenciômetro de 0-200 Ω.

$$R_{saída\_LEM\_cc} = \frac{1,18}{10 \cdot 10^{-3} \cdot 2,5} = 47,2\,\Omega \tag{21}$$

4.2.2 Modelagem do sistema de condicionamento

O sinal proveniente do sensor de tensão deverá ter níveis adequados para a entrada do conversor A/D do DSP. Na Figura 21 é mostrado o circuito para gerar o *offset* para o circuito de condicionamento do sensor da rede e do PAC. O circuito citado é responsável por realizar o ajuste necessário no sinal de saída do sensor. O circuito amplificador do tipo inversor, possui um ganho unitário, enquanto que o sinal constante que é submetido as resistências na entrada (5 kΩ e 10 kΩ), resulta em um valor de 1,65 V. Portanto, este circuito gera um sinal de saída (V_{ref}), com valor igual a -1,65 V.

Figura 21 – Circuito para gerar o offset.

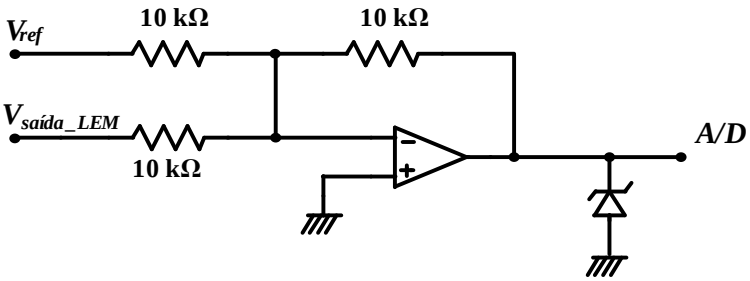


Fonte: Autor.

A saída do circuito de *offset* é enviada para o circuito de condicionamento apresentado na Figura 22, do tipo amplificador somador. Para garantir um ganho unitário deste circuito, todos os resistores possuem um valor comercial de 10 kΩ. O diodo zener na saída do circuito é incluído para assegurar que a tensão não ultrapasse valor máximo aceito pelo módulo

ADC do DSP. Para o condicionamento do sinal do sensor de tensão do barramento CC, não é necessário *offset*, portanto será utilizado apenas um amplificador operacional, em um circuito não inversor.

Figura 22 - Circuito de condicionamento.

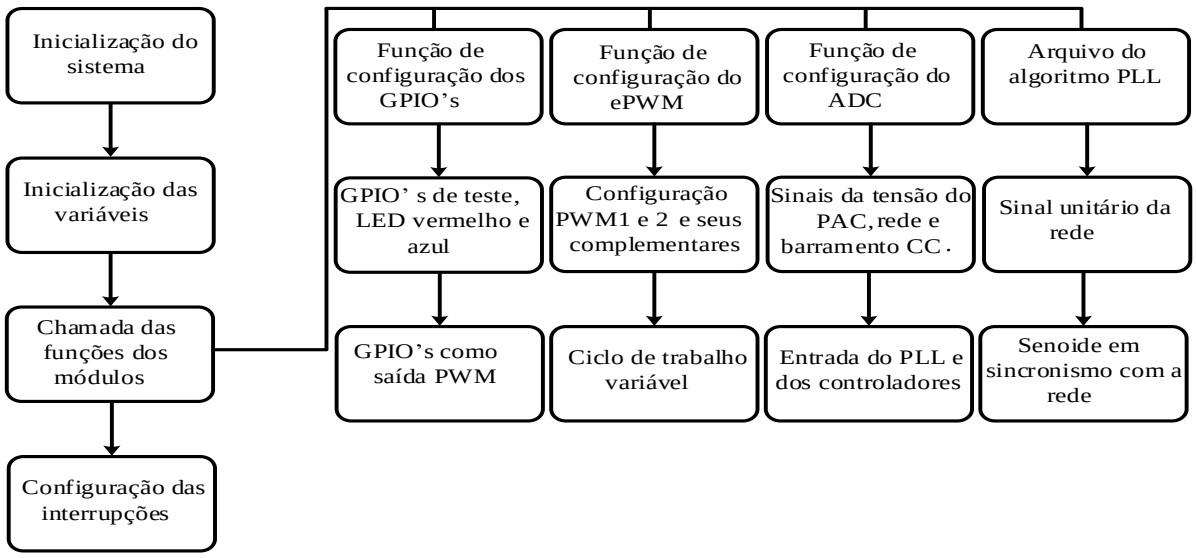


Fonte: Autor.

4.3 Algoritmo implementado no DSP

A Figura 23 apresenta o diagrama de blocos que representa o processo de elaboração do algoritmo de supervisão e controle que foi implementado no DSP. Como pode ser observado o programa é composto por quatro blocos principais. Os dois primeiros blocos representam o cabeçalho do programa, que é parte no qual é iniciada a função que realiza o controle do DSP (*InitSysCtrl*) e as funções que habilitam a tabela do módulo de interrupções, além da declaração das variáveis. Os dois últimos blocos realizam a chamada de outros arquivos do programa. Inicialmente é realizada a referência as funções que foram criadas para configurar os registradores dos módulos que serão utilizados. Em seguida é realizada a rotina que estará presente dentro da interrupção do ADC.

Figura 23 – Fluxograma do programa criado no DSP.



Fonte: Autor.

Foi dado destaque para o bloco que se refere as funções dos módulos do DSP. A primeira função diz respeito aos GPIO's, onde foi configurado os pinos referentes a dois LED's do DSP, com a função de realizar testes em rotinas temporizadas. Além disso, foi configurado dois GPIO's como saída PWM. O DSP possui um módulo ePWM, como citado no capítulo 3, que deverá ser configurado, de modo a proporcionar a modulação SPWM do inversor de tensão, para isto foi criada uma função de configuração dos registradores referentes a este módulo. Para o ADC também foi criada uma função para configuração dos seus registradores. Essa parte será responsável pelo processamento das amostras de tensão provenientes do sistema de aquisição de dados. A tensão da rede deverá ser transformada em uma senoide unitária, para ser utilizada como entrada do PLL, enquanto a tensão do PAC será amostrada para ser utilizada como moduladora e na malha de controle da tensão no PAC, assim como a tensão do barramento CC será utilizada em sua malha de controle. Por último, no diagrama, é mostrado o bloco referente ao PLL, que irá capturar a fase da tensão da rede e gerar um sinal em sincronismo. Todos os detalhes sobre o algoritmo do PLL podem ser consultados no apêndice A.

4.4 Configuração do DSP

Nesta seção é realizada a configuração dos principais registradores dos módulos do DSP, que serão usados para implementar o *firmware* de controle do inversor.

4.4.1 Configuração do módulo ePWM

Como citado anteriormente, o DSP utilizado neste trabalho, possui um módulo ePWM. Portanto, para gerar a modulação SPWM unipolar que será usada para o acionamento das chaves do inversor, é necessário fazer a adequada configuração deste módulo. Para este projeto será utilizado os módulos ePWM1 e ePWM2 e seus respectivos sinais complementares. Uma vez que a configuração de ambos seguem os mesmos procedimentos, esta seção apresentará a configuração apenas para o módulo ePWM1. Para maiores detalhes, além destes aqui apresentados, consultar o apêndice A. Os primeiros registradores a serem configurados pertencem ao submódulo *Time Base*. Iniciando pelo registrador TBPRD, que determina o período do PWM, com base na frequência de chaveamento (*fsw*) e na frequência do DSP (f_{DSP}), conforme (22).

$$EPwm1Regs.TBPRD = \frac{f_{DSP}}{4 f_{sw}} \rightarrow TBPRD = 2500 \quad (22)$$

Como a modulação utilizada neste trabalho é SPWM, o registrador referente ao ciclo de trabalho deverá ser zerado, conforme a equação (23). Seu valor irá variar de acordo com malha de tensão no PAC.

$$EPwm1Regs.CMPA.bit.CMPA = 0 \quad (23)$$

Após a configuração do período na equação (22) é possível determinar o deslocamento de fase, usando o registrador TBPHS. Quando o valor de fase for igual ao período, ocorrerá um defasamento de 180º. Como o ePWM1 é o sinal de referência, seu ângulo de fase deverá ser zero, sendo assim:

$$EPwm1Regs.TBPHS.bit.TBPHS = 0 \quad (24)$$

Depois da determinação do valor de (24), é preciso determinar que o ePWM1 será a referência para o defasamento dos demais, isso deve ser feito a partir do bit SYNCOSEL, presente no registrador TBCTL. Portanto, quando o contador do ePWM1, for igual a zero será enviando um pulso de sincronismo para os demais, dessa forma:

$$EPwm1Regs.TBCTL.bit.SYNCOSEL = TB\_CTR\_ZERO \quad (25)$$

Como o ePWM1 foi definido como sendo a referência, é necessário desabilitar o deslocamento de fase. Isso é feito por meio do bit PHSEN, do registrador TBCTL. Conforme:

$$EPwm1Regs.TBCTL.bit.PHSEN = TB\_DISABLE \quad (26)$$

Para que o PWM possa funcionar é necessário configurar o seu contador, que irá variar de 0 até o período, o registrador que faz essa configuração é o TBCTR, sendo assim:

$$EPwm1Regs.TBCTR = 0x0000 \quad (27)$$

Após configurar os primeiros parâmetros de base de tempo, é necessário escolher a forma de onda da portadora. Para este trabalho será feita a escolha para o tipo triangular. Para isto foi utilizado o bit CTRMODE, do registrador TBCTL. Logo:

$$EPwm1Regs.TBCTL.bit.CTRMODE = TB\_COUNT\_UPDOWN \tag{28}$$

Quando a frequência de chaveamento é muito baixa, o valor do período do ePWM tende a ser muito alto, ultrapassando o valor possível para aquele registrador de 16 bits. Para resolver esse problema existem dois registradores que realizam a divisão do relógio do sistema. Já que para esta aplicação não será necessário realizar essa divisão, basta configurar os dois registradores para fazer uma divisão por 1, conforme (29) e (30).

$$EPwm1Regs.TBCTL.bit.CLKDIV = TB\_DIV1 \tag{29}$$

$$EPwm1Regs.TBCTL.bit.HSPCLKDIV = TB\_DIV1 \tag{30}$$

Os próximos registradores a serem configurados pertencem ao submódulo *Counter Compare*, sendo configurado um registrador e quatro bits ao todo. O primeiro bit, SHDWAMODE, que pertence ao registrador CMPCTL, gerencia a atualização do período e ciclo de trabalho do PWM, forçando esses dois parâmetros a serem atualizados em situações específicas. Como pode ser observado na equação (31), foi feita a configuração para o PWM1A e PWM1B.

$$\begin{aligned} EPwm1Regs.CMPCTL.bit.SHDWAMODE &= CC\_SHADOW \\ EPwm1Regs.CMPCTL.bit.SHDWBMODE &= CC\_SHADOW \end{aligned} \tag{31}$$

O segundo bit a ser configurado, LOADAMODE, determina o momento em que poderá ocorrer uma atualização de período ou ciclo de trabalho. Sendo assim, o bit foi configurado de tal forma que essas atualizações ocorrerão quando o contador (CTR) for igual ao período (PRD) ou igual a zero. Com isso garante-se a atualização da razão cíclica no início do período de chaveamento. Portanto:

$$EPwm1Regs.CMPCTL.bit.LOADAMODE = CC\_CTR\_ZERO\_PRD \quad (32)$$
$$EPwm1Regs.CMPCTL.bit.LOADBMODE = CC\_CTR\_ZERO\_PRD$$

O registrador do submódulo *Action Qualifier* é o próximo a ser configurado. Por meio do mesmo será possível implementar a modulação que irá acionar as chaves do inversor. O primeiro bit a ser configurado é o bit PRD do registrador AQCTLA, que configura a saída do GPIO correspondente ao PWM1. Quando o contador (CTR) é igual ao período, neste caso não será feita nenhuma ação, sendo assim:

$$EPwm1Regs.AQCTLA.bit.PRD = AQ\_NO\_ACTION \quad (33)$$

Quando o contador é igual ao zero, nenhuma ação será realizada, dessa forma:

$$EPwm1Regs.AQCTLA.bit.ZRO = AQ\_NO\_ACTION \quad (34)$$

Quando o contador estiver crescendo e for igual ao CMPA, deverá ser setado em nível baixo, conforme destacado a seguir:

$$EPwm1Regs.AQCTLA.bit.CAU = AQ\_CLEAR \quad (35)$$

Para a situação em que o contador estiver decrescendo e for igual ao CMPA, deverá ser setado em nível alto, sendo assim:

$$EPwm1Regs.AQCTLA.bit.CAD = AQ\_SET \quad (36)$$

O *Dead Band* é o último submódulo a ser ajustado, no qual será realizada a configuração do PWM complementar. A ativação desse pulso complementar é realizada pelo bit POLSEL do registrador DBCTL, conforme a equação (37).

$$EPwm1Regs.DBCTL.bit.POLSEL = DB\_ACTV\_HIC \quad (37)$$

O bit OUT_MODE do registrador DBCTL, deve ser habilitado para que o submódulo *Dead Band* seja efetivamente ativado. Sendo assim:

$$EPwm1Regs.DBCTL.bit.OUT\_MODE = DB\_FULL\_ENABLE \quad (38)$$

Com o módulo de tempo morto habilitado, é possível configurar o tempo que levará para ligar o PWM1B, depois que o PWM1A é desligado, registrador DBFED, configura essa ação. Será utilizado o valor de 1 μs, que é aceitável para a maioria das chaves eletrônicas. Dessa forma:

$$EPwm1Regs.DBFED.bit.DBFED = 100 \quad (39)$$

Já o processo contrário, ou seja, ligar o PWM1A depois que PWM1B é desligado, é configurado registrado DBRED, do seguinte modo:

$$EPwm1Regs.DBRED.bit.DBRED = 100 \quad (40)$$

4.4.2 Configuração do módulo ADC

Como citado anteriormente para o processamento dos sinais de tensão provenientes do sistema de aquisição de dados, os mesmos deverão ser transformados em sinais discretos no tempo. Por isso será necessário realizar a configuração do módulo ADC presente no DSP. Como foi dito no capítulo 3, é possível escolher a resolução do conversor A/D em 12 ou 16 bits. Para este trabalho será escolhida uma resolução de 12 bits, conforme baixo:

$$Adca\,\mathrm{Re}\,gs.ADCCTL2.bit.RESOLUTION = ADC\_RESOLUTION\_12BIT \quad (41)$$

O tamanho da janela de aquisição (SH) está relacionada a resolução que foi escolhida em (41). Inicialmente, a equação (42) faz este cálculo para ser posteriormente configurado no registrador adequado. Para este projeto é considerado um valor de $75\ ns$ para a janela de aquisição, sendo assim:

$$acqps = \frac{SH}{1/\ f_{DSP}} - 1 \rightarrow acqps = 14 \quad (42)$$

Como será habilitada a interrupção do ADC, é necessário configurar o momento em que o pulso da interrupção será disparado. Para este caso, o disparo será no final da conversão, dessa forma:

$$Adca\,\mathrm{Re}\,gs.ADCCTL1.bit.INTPULSEPOS = 1 \tag{43}$$

Como o ADC é composto por uma série de circuitos analógicos, é necessário habilitar todos esses circuitos, de acordo com a equação abaixo:

$$Adca\,\mathrm{Re}\,gs.ADCCTL1.bit.ADCPWDNZ = 1 \tag{44}$$

O próximo passo é a escolha dos canais que farão a conversão dos sinais de tensão. Será utilizado o canal 3 para o sistema de aquisição da tensão da rede, o 4 para o sistema de aquisição do sinal do PAC e o 5 para o sistema de aquisição da tensão no barramento CC. Para isto é necessário configurar os bits CHSEL, ACQPS e TRIGSEL. As equações a seguir mostram a configuração para o canal 3, logo, para os demais canais a configuração será idêntica. Contudo, caso seja de interesse observar as configurações dos demais canais, os mesmos estão disponíveis no apêndice A.

$$adca\,\mathrm{Re}\,gs.ADCSOC0CTL.bit.CHSEL = 3 \tag{45}$$

$$adca\,\mathrm{Re}\,gs.ADCSOC0CTL.bit.ACQPS = 14 \tag{46}$$

$$adca\,\mathrm{Re}\,gs.ADCSOC0CTL.bit.TRIGSEL = TRIG\_SEL\_ePWM1\_SOCA \tag{47}$$

Como é possível observar o canal 3 utilizou o SOC0, para os demais canais deverá ser utilizado outros SOC's. O último bit configurado, TRIGSEL, é responsável por configurar a fonte de acionamento do A/D, neste caso a fonte de acionamento será o PWM1, pois a leitura dos sensores será realizada sempre no início do chaveamento. O passo seguinte é realizar a configuração do PWM para executar esse acionamento, conforme (48), (49) e (50).

$$EPwm1.ETSEL.bit.SOCAEN = 1 \tag{48}$$

$$EPwm1.ETSEL.bit.SOCASEL = ET\_CTR\_PRDZERO \tag{49}$$

$$EPwm1.ETPS.bit.SOCAPRD = ET\_1ST \tag{50}$$

A equação (48) faz a habilitação do SOCA dentro da rotina do PWM, em seguida é necessário determinar o momento do disparo da conversão. Neste trabalho, o A/D terá o dobro da frequência do PWM, portanto a conversão será disparada quando o contador do PWM passar pelo PRD e pelo zero, conforme a equação (49), sendo que o acionamento sempre irá ocorrer no primeiro evento, de acordo com a equação (50).

Para realizar uma rotina de leitura da conversão A/D é preciso configurar sua interrupção. Como foi configurado três canais, é preciso que a interrupção seja habilitada após a conversão no último canal, sendo assim:

$$adca\,\mathrm{Re}\,gs.ADCINTSEL1N2.bit.INT1SEL = 0x02 \tag{51}$$

Esta interrupção deverá ser habilitada através da equação (52) e seu flag deverá ser limpo de acordo com a equação (53):

$$adca\,\mathrm{Re}\,gs.ADCINTSEL1N2.bit.INT1E = 1 \tag{52}$$

$$adca\,\mathrm{Re}\,gs.ADCINTFLGCLR.ADCINT1 = 1 \tag{53}$$

4.4.3 Configuração do módulo de interrupções (PIE)

Como foi argumentado no capítulo 3, as interrupções forçam que a CPU realize uma rotina especificamente e, portanto, pausando suas atuais atividades em processamento. Os módulos presentes no DSP, podem em muitas situações serem fontes de interrupções. Para

configurar o módulo de interrupções, o primeiro passo é inicializar todas as interrupções, para posteriormente serem utilizadas. Inicialmente é necessário desabilitar todas as interrupções, por meio do registrador DINT, e limpar os registradores de habilitação e sinalização, conforme (54) e (55):

$$IER = 0x0000 \tag{54}$$

$$IFR = 0x0000 \tag{55}$$

Após essa inicialização, a tabela de vetores deve ser chamada na rotina por meio da função *InitPieVectTable*. A única interrupção a ser utilizada, será aquela referente ao ADCA. A rotina de interrupção deste módulo será chamada de isr_adc. Essa nomenclatura deverá ser associada a interrupção da tabela, correspondente ao ADCA.

$$PieVectTable.ADCA1\_INT = \&isr\_adc \tag{56}$$

Esta interrupção do ADC, está alocada na linha um e coluna um da tabela, sendo assim é necessário habilitar essa coluna, conforme a equação:

$$PieCtrl\,\mathrm{Re}\,gs.PIEIER1.bit.INTx1 = 1 \tag{57}$$

Para que a interrupção seja habilitada corretamente, o último passo diz respeito a habilitação de toda a linha um da tabela, dessa forma:

$$IER \mid = M\_INT1 \tag{58}$$

4.5 Projeto do *loop filter* compensador do PLL

Como citado no capítulo 3, o compensador de fase ou *loop filter* é comumente modelado como um dispositivo cuja a função de transferência é a de um filtro passa-baixa, funcionando como um controlador PI. Para resolver as equações (16) e (17), é necessário calcular os parâmetros do controlador PI supracitado. A metodologia utilizada nesta seção é

baseado em *Texas Instruments* (2017). A Tabela 4 mostra os parâmetros assumidos para realizar os cálculos pertinentes.

Tabela 4 – Parâmetros do PLL.

Tempo de estabilização *Ts*	30 ms
Banda de erro ∂	5%
Coeficiente de amortecimento ζ	0,7

Fonte: Autor.

Para encontrar o parâmetro proporcional, é necessário encontrar as variáveis ω_n e T_i da equação (8), conforme:

$$\omega_n = \frac{\zeta}{\sigma} = 158,68 \text{ rad/s} \tag{59}$$

$$T_i = \frac{2\zeta}{\omega_n} = 8,82 \text{ ms} \tag{60}$$

Dessa forma é isso é possível calcular o valor de K_P, resultando em:

$$K_p = 222,16 \text{ rad/s} \tag{61}$$

De posse do ganho proporcional, é possível calcular o ganho integral, do seguinte modo:

$$K_i = \frac{K_p}{T_i} = 25181,22 \text{ rad/s}^2 \tag{62}$$

A partir dos ganhos obtidos e considerando que a rotina do PLL estará submetida a interrupção do módulo ADC, quem tem o dobro da frequência de chaveamento, é possível calcular os parâmetros B0 e B1, a partir das equações (16) e (17). Dessa forma:

$$B0 = 222,47 \text{ rad/s} \tag{63}$$

$$B1 = -221{,}85 \text{ rad/s} \tag{64}$$

Após os cálculos dos parâmetros do *loop filter* é possível realizar a sua implementação no processador digital de sinais.

4.6 Considerações finais

Neste capítulo foi realizado um exemplo de projeto para o *firmware* proposto. Foi apresentada a modelagem do sistema de aquisição de dados, onde foi feito os cálculos referentes ao circuito dos sensores de tensão e do circuito de condicionamento. Foi apresentada a configuração dos principais registradores dos módulos do DSP. Por fim, foi feito os cálculos do *loop filter* do algoritmo de sincronismo.

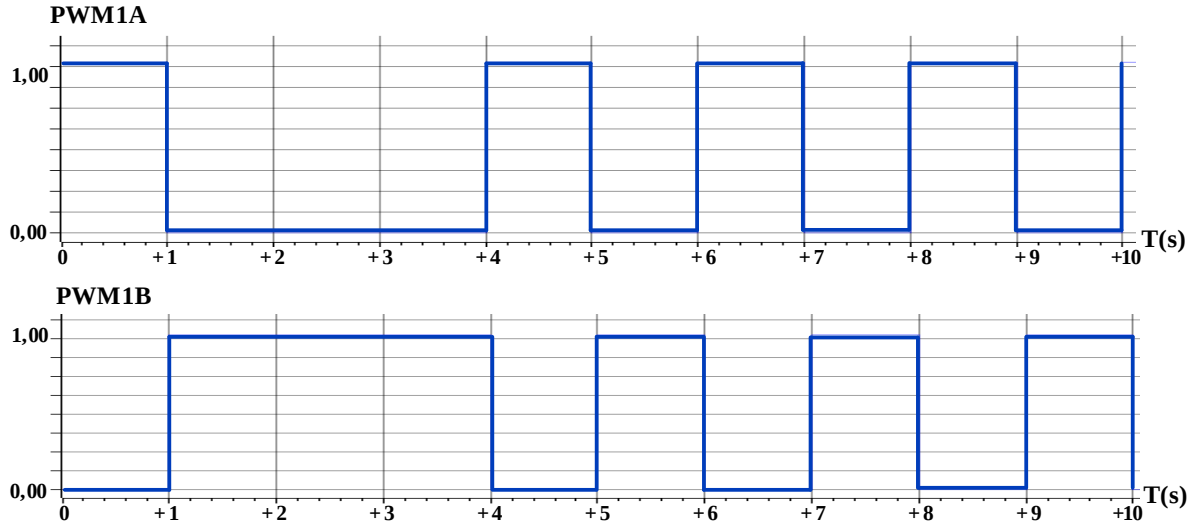
5 RESULTADOS DE SIMULAÇÃO

Neste capítulo serão apresentados os resultados de simulação do módulo Epwm, do sistema de sincronismo (PLL) e do sistema de aquisição da tensão da rede, tensão no PAC e tensão no barramento CC. No capítulo 4 foi apresentado os detalhes da elaboração do algoritmo implementado no DSP. Portanto, este capítulo apresenta os resultados necessários para validar o sistema de aquisição de dados e controle. As simulações deste capítulo se baseiam nas especificações mostradas nas Tabela 2 e Tabela 3.

5.1 Simulação para o módulo Epwm

A Figura 24 mostra a forma de onda na saída do PWM1, usando a modulação SPWM. Como a frequência de chaveamento tem um valor de 20 kHz, a visualização da variação da largura de pulso por meio do gráfico do CCS, fica um pouco prejudicada. Mesmo assim é possível perceber que o módulo ePMW1 está funcionando da forma esperada, também é mostrado que o PWM1B, é um sinal complementar, como foi dito no capítulo 4. Para o módulo ePWM2 as formas de onda observadas são as mesmas, com o sinal defasado em 180º do ePWM1.

Figura 24 – forma de onda para o módulo ePWM1.

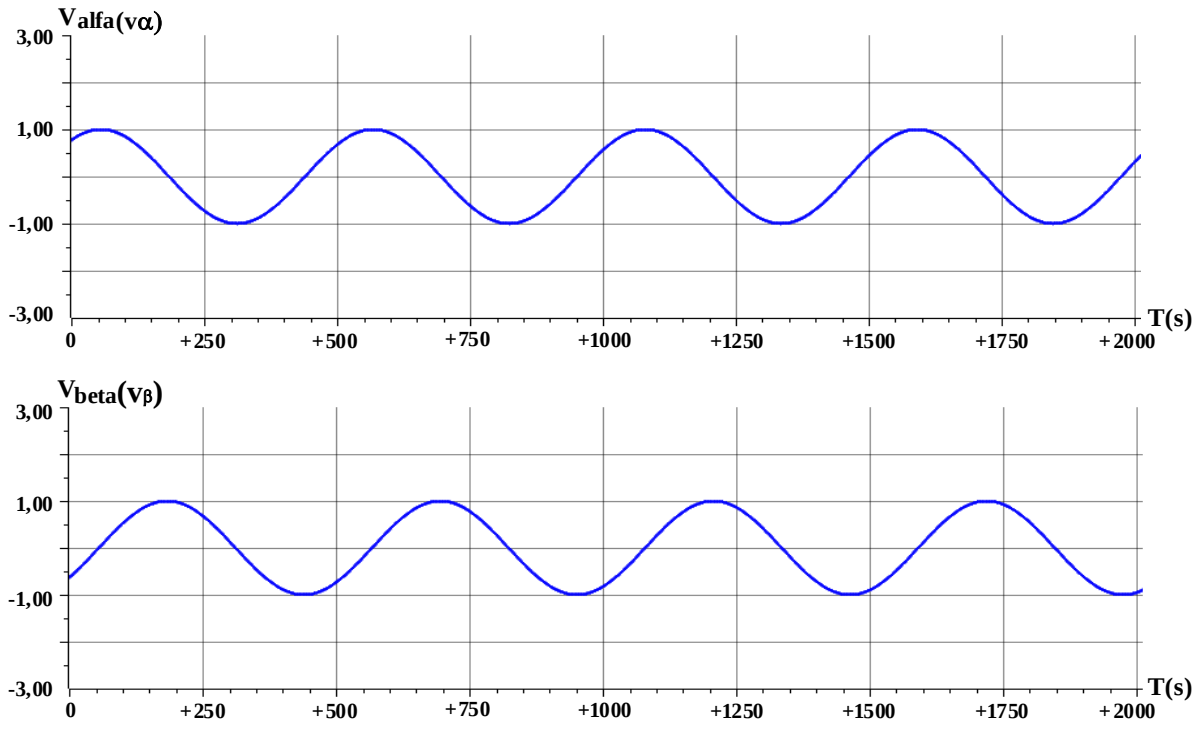


Fonte: Autor.

5.2 Simulação para o PLL

As simulações de teste do PLL foram realizadas no ambiente de desenvolvimento da *Texas Instruments*, chamado de *Code Composer Studio* (CCS). Além de ser um ambiente de edição e depuração de um algoritmo, o CCS possui ferramentas de visualização gráfica e bibliotecas com exemplos de projetos. Como foi argumentado no capítulo 3, a metodologia utilizada no circuito de sincronismo proposto neste trabalho, necessita que sua entrada seja uma senoide unitária, sendo este sinal a componente $v_\alpha(t)$. Como o circuito de sincronismo é uma estrutura monofásica, existe a necessidade de defasar em 90º a componente alfa. Este segundo sinal em quadratura, será chamado de componente $v_\beta(t)$. A Figura 25 mostra as componentes $v_\alpha(t)$ e $v_\beta(t)$ que foram geradas a partir do algoritmo de PLL que foi desenvolvido no CCS.

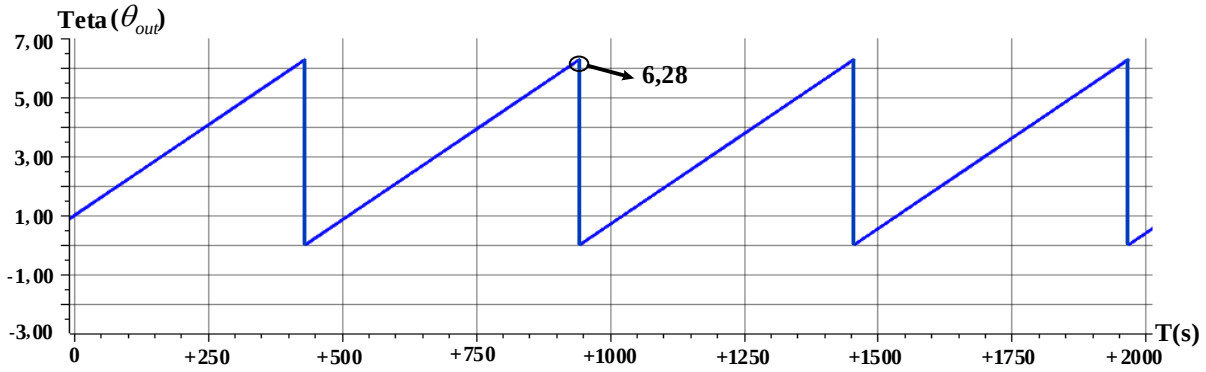
Figura 25 – Componente alfa e beta do PLL.



Fonte: Autor.

Garantindo os dois sinais de entrada do PLL, o sistema deverá ter como saída, a captura do ângulo de fase do sinal de entrada (tensão da rede). O formato da onda de saída deverá ser uma dente de serra, variando de 0 a 2π, identificando, portanto, o ciclo de rede completo. Este sinal de saída deve ser utilizado para gerar um sinal em sincronismo com a tensão de entrada. Na Figura 26 é mostrado o ângulo de fase do sinal de entrada, que foi capturado pelo PLL. Como pode ser observado, o sinal está variando de 0 a 2π.

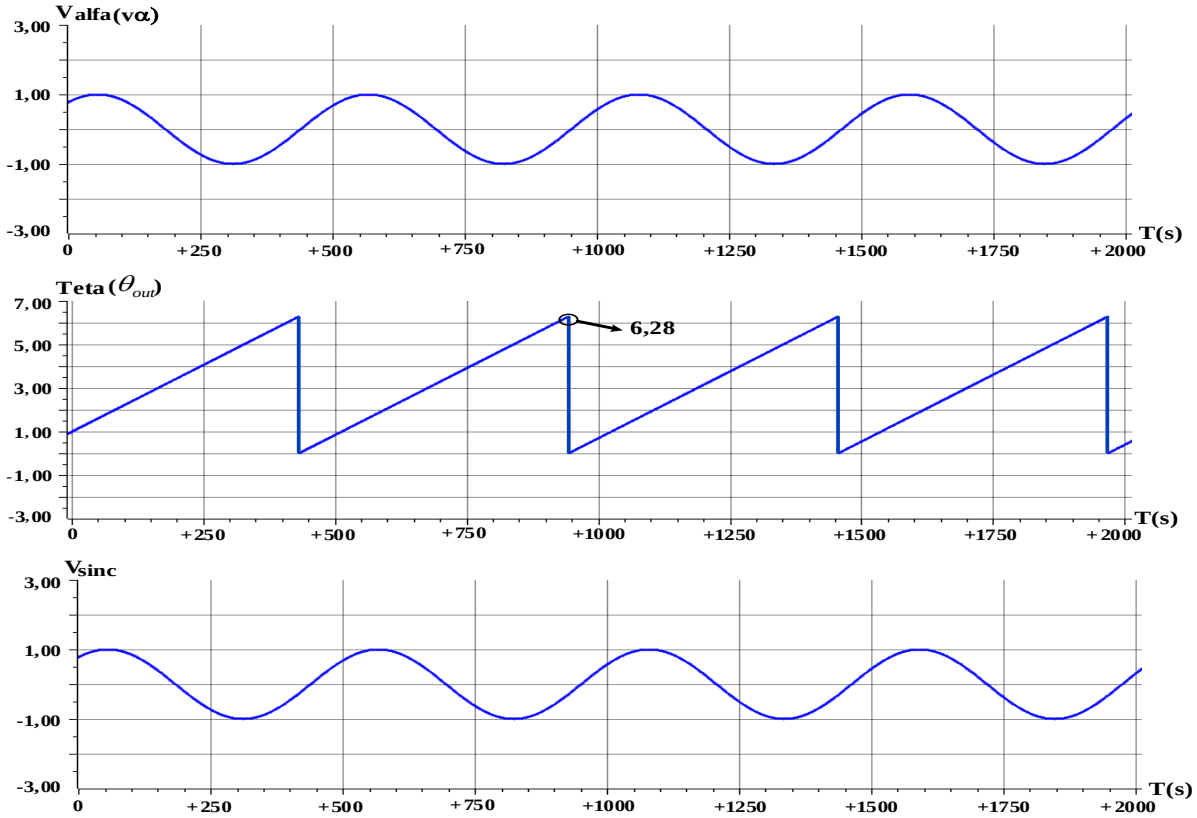
Figura 26 – Ângulo de fase capturado pelo PLL.



Fonte: Autor.

A Figura 27 mostra a forma de onda referente ao sinal unitária de saída que está em sincronismo com a tensão de entrada. Como já foi dito anteriormente, a tensão de entrada deve ser convertida em um sinal unitário, que é a componente alfa, e como pode ser observado na figura citada, os dois sinais apresentados estão em sincronismo.

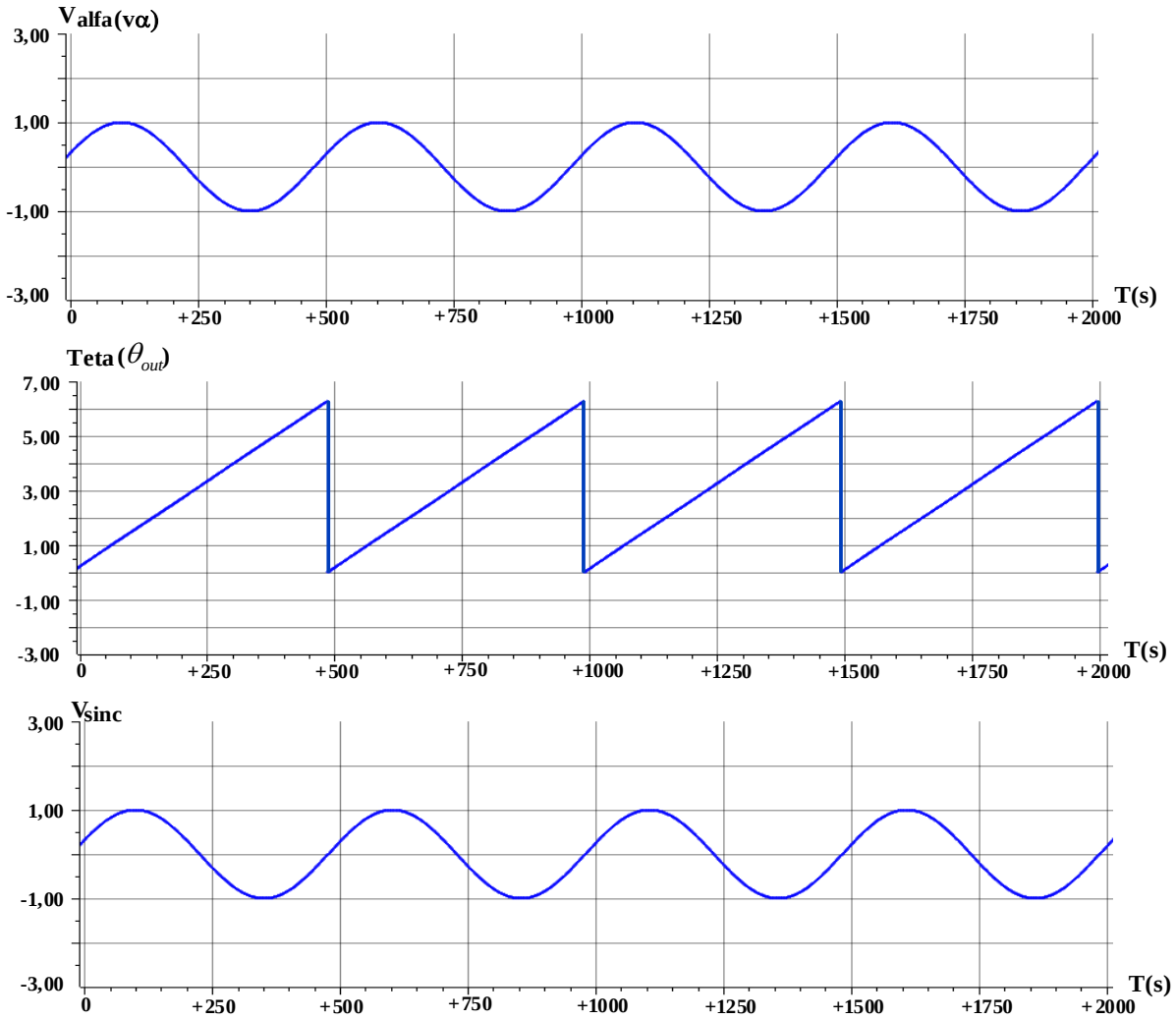
Figura 27 - Formas de onda de sincronismo para *fr* = 60 Hz.



Fonte: Autor.

O módulo 8 do PRODIST que trata da qualidade de energia, determina que em condições de distúrbios no sistema de distribuição, as fontes geradoras precisam garantir que frequência seja restituída, em um intervalo de 30 segundos, para a faixa de 59,5 Hz a 60,5 Hz. Portanto, o algoritmo de sincronismo deve suportar pequenas variações de frequência, dessa forma o sistema será submetido a variações de 1 Hz de frequência. Na Figura 28 é apresentada a simulação para frequência da rede com valor de 61 Hz. Como pode ser observado o sistema de sincronismo continuou capturando a fase do sinal da rede, e produzindo um sinal em sincronismo com a tensão de entrada.

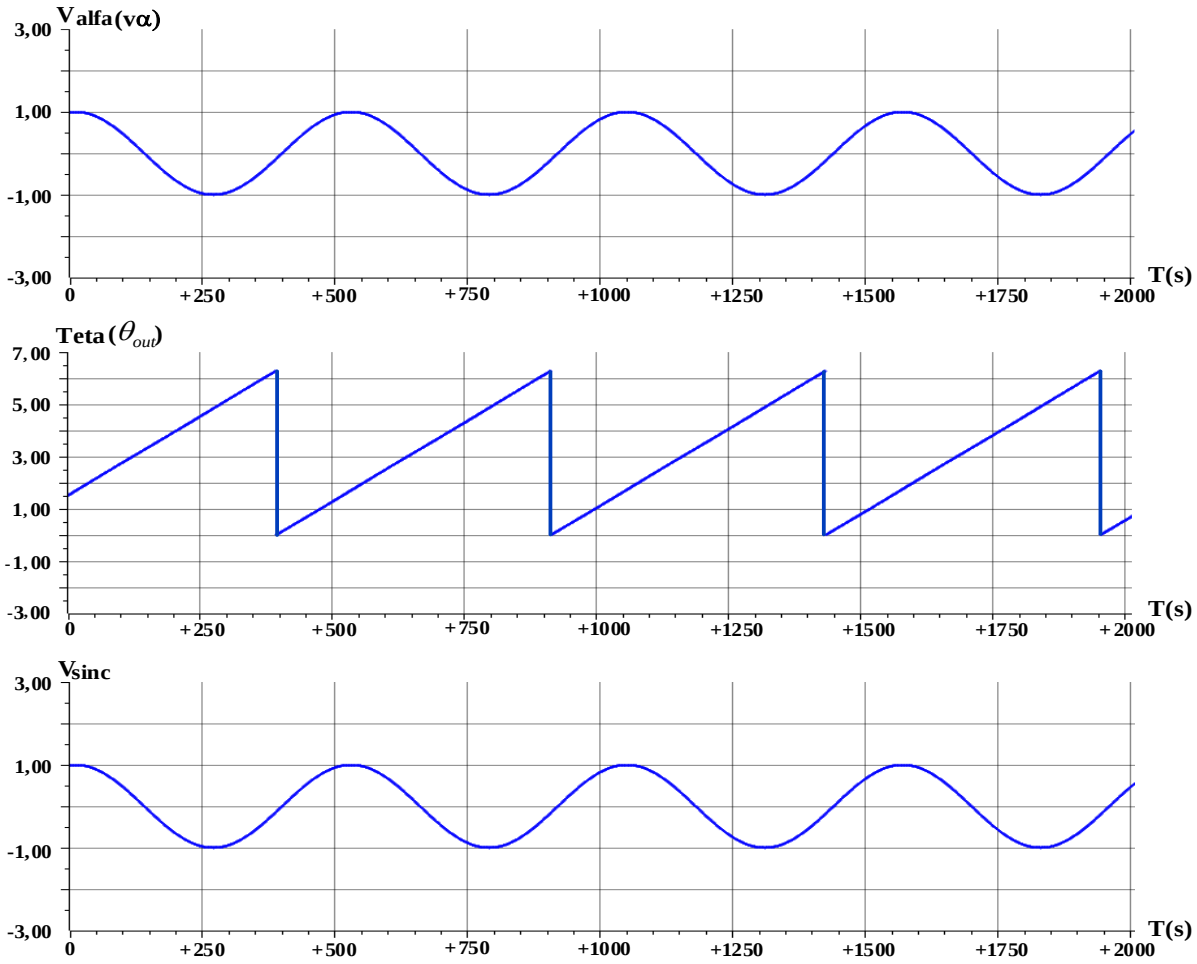
Figura 28 – Formas de onda de sincronismo para *fr* = 61 Hz.



Fonte: Autor.

A Figura 29 mostra os resultados de simulação, quando a frequência da rede varia seu valor para 59 Hz. Como pode ser observado o algoritmo de sincronismo capturou a fase do sinal da rede de forma adequada e gerou um sinal em sincronismo com a tensão de entrada.

Figura 29 - Formas de onda de sincronismo para *fr = 59* Hz.



Fonte: Autor.

5.3 Simulação do sistema de aquisição

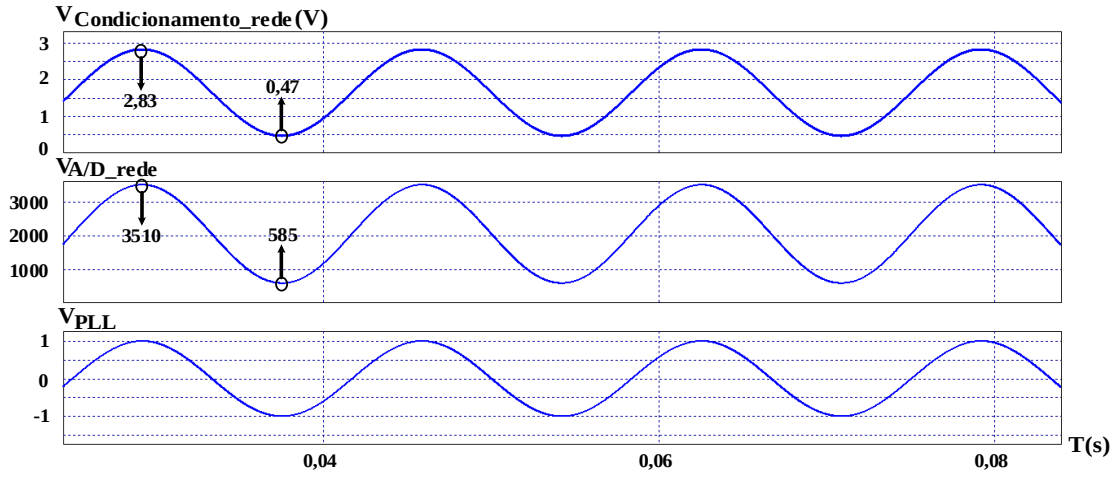
Nesta seção serão apresentados os resultados de simulação para o sistema de aquisição das tensões do inversor, que serão processadas no algoritmo de controle desenvolvido no DSP.

5.3.1 Sistema de aquisição da tensão da rede e no PAC

Como foi discutido anteriormente, o sistema de aquisição é composto pela parte de sensoriamento e condicionamento. Este sistema faz os ajustes necessários para que os sinais de tensão possuam valores adequados ao máximo do conversor A/D do DSP. As simulações apresentadas nesta seção foram realizadas no *software* PSIM, considerando os valores referentes ao sistema conectado à rede elétrica. Os primeiros resultados obtidos dizem respeito

ao sistema de aquisição da tensão no PAC e na rede. Como é possível observar na Figura 30, a tensão da rede após passar pelo sensor e pelo condicionamento ($V_{condicionamento\_rede}$), apresenta valores adequados para a entrada A/D do DSP utilizado neste projeto. O processo de conversão analógica/digital, proporciona um ganho ao sinal de entrada, a forma de onda resultante (V_{AD_rede}) é mostra na figura citada. Este sinal convertido deverá sofrer uma modificação, de modo a obter uma senoide unitária, que será a entrada do PLL (V_{PLL}).

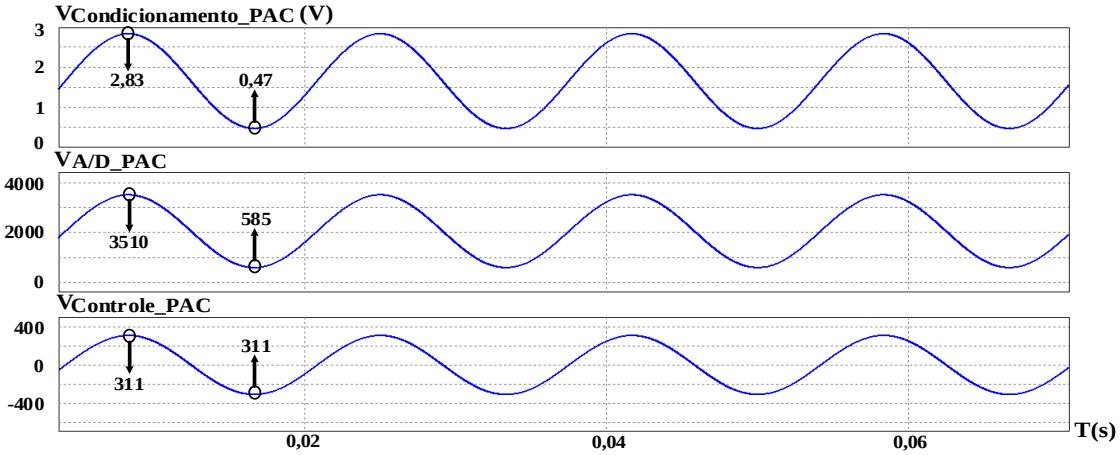
Figura 30 – Formas de onda para o sistema de aquisição da tensão da rede.



Fonte: Autor.

A Figura 31 mostra as formas de onda referentes ao sistema de aquisição para a tensão no PAC. Como pode ser observado, o sinal de entrada do A/D e o sinal com o ganho deste conversor, possuem os mesmos valores para o sistema de aquisição da tensão da rede. A única diferença diz respeito a última forma de onda, ($V_{controle\_PAC}$), que será entrada para a malha de controle de tensão, que necessita de um ganho unitário entre este sinal e a tensão do PAC.

Figura 31 - Formas de onda para o sistema de aquisição da tensão do PAC.

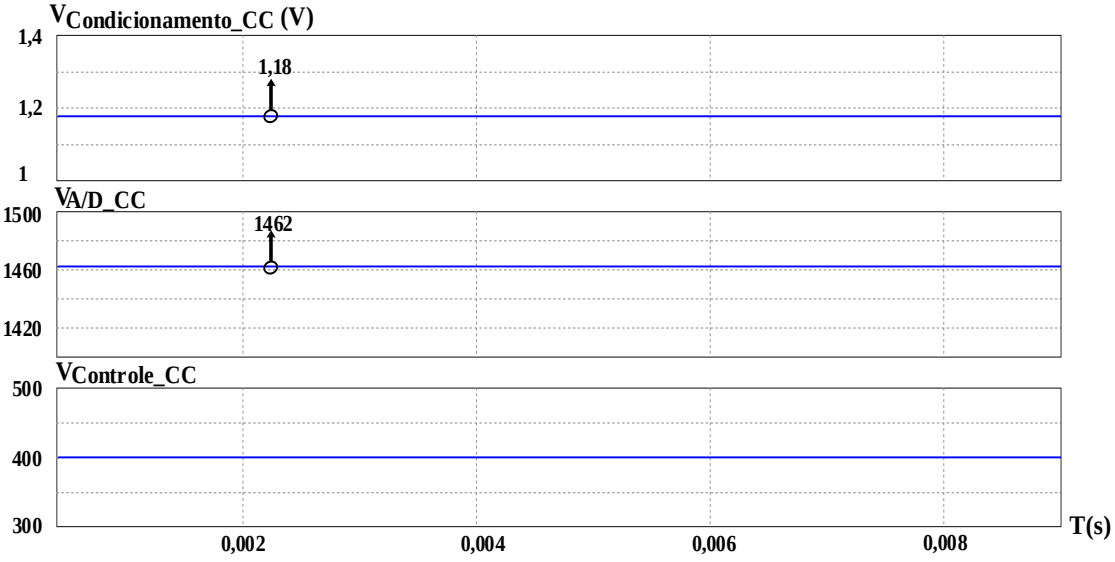


Fonte: Autor.

5.3.2 Sistema de aquisição da tensão no barramento CC.

A Figura 32 apresenta as formas de onda referentes ao sistema de aquisição da tensão no barramento CC. Como pode ser observado a tensão na saída do sistema de sensoriamento e condicionamento ($V_{condicionamento\_CC}$) possui um valor adequado a entrada do conversor A/D do DSP. O conversor A/D proporciona um ganho, resultando no sinal mostrado no segundo gráfico da figura. Para ser utilizado na malha de controle, é necessário um ganho unitário entre este sinal e tensão no barramento CC.

Figura 32 - Formas de onda para o sistema de aquisição da tensão do barramento CC.



Fonte: Autor.

5.4 Considerações finais

Os resultados de simulação obtidos para o *firmware* foram apresentados neste capítulo. Foi possível observar as principais formas de onda para o sistema de aquisição de dados. Como mostrado, o sistema de sensoriamento e condicionamento foi validado, tanto para a amostragem dos sinais do inversor, como para o envio destes sinais para o DSP, possuindo níveis adequados ao conversor A/D.

Foi mostrado que o módulo ePWM estava variando o ciclo de trabalho em sua saída. Também foram apresentadas as principais formas de onda para o sistema de sincronismo. Inicialmente o algoritmo foi submetido as condições nominais da rede, apresentando resultados satisfatórios. Em seguida o sistema foi submetido a variação na frequência da rede, indo para 61 Hz. Para essa condição o PLL conseguiu realizar a captura do ângulo de fase da rede, além

de gerar um sinal em sincronismo com a tensão de entrada. Por último a estrutura de PLL, foi submetida a um sinal da rede com frequência de 59 Hz, apresentando resultados satisfatórios para essa condição.

6 CONCLUSÕES E SUGESTÕES PARA TRABALHOS FUTUROS

Este trabalho apresentou o desenvolvimento de um *firmware* para ser aplicado a inversor de tensão monofásico. Foi feita a modelagem do sistema de aquisição de dados, composto pelo sensoriamento e pelo condicionamento. Para o sistema de sensoriamento foi feita a modelagem de seu circuito, baseado nas especificações do sensor de tensão. O sistema de condicionamento, foi projetado para garantir que os sinais amostrados no inversor de tensão, teriam valores adequados para a entrada do conversor A/D do DSP. Em seguida foi realizada a configuração dos principais módulos do DSP, além de ser realizado o projeto do algoritmo de PLL.

Foi observado por meio de simulações, que o sistema de aquisição de dados conseguiu cumprir sua função de proporcionar níveis adequados de tensão ao processador digital de sinais. Também foi mostrado que a configuração do módulo ePWM conseguiu realizar a modulação utilizada no inversor. O sistema de sincronismo mostrou resultados satisfatórios para pequenas variações de frequência, apresentando problemas apenas para variações muito bruscas.

Como sugestões para trabalhos futuros, os seguintes temas são pertinentes: 1) Tornar o PLL mais robusto, conseguindo sincronizar para grades variações da frequência do sinal de entrada; 2) Realizar a expansão do *firmware* para ser aplicado em inversor de tensão trifásico; 3) Aplicar o *firmware* em inversores com outra modulação e técnica de controle diferente da utilizada neste trabalho.

REFERÊNCIAS

ARNOLD, Ray. Solutions to the power quality problem. **Power Engineering Journal**, v. 15, n. 2, p. 65-73, 2001.

AVELINO, Álvaro Medeiros. **Processamento embarcado aplicado a um sistema de detecção de vazamentos**. 2009. 47 f. Dissertação (Mestrado) - Curso de Engenharia Elétrica e Computação, Universidade Federal do Rio Grande do Norte, Natal, 2009.

BRENNAN, P. V. **Phase-Locked Loops**: principles and practice. [S. L.]: McGraw-Hill Professional, 1996.

CARRAH, Israel Franklin Dourado. **Inversor monofásico tipo ponte completa com controle digital**. 2010. 159 f. TCC (Graduação) - Curso de Engenharia Elétrica, Universidade Federal do Ceará, Fortaleza, 2010.

CHOI, J.W.; KIM, Y.K.; KIM, H.G. Digital PLL control for single-phase photovoltaic system. **IEE Proceedings Electric Power Applications**, v. 153, n. 1, p. 40-46, fev. 2006.

COSTA, Gilmar N. S. et al. Estudo do Estágio STATCOM em um Sistema Ativo Bidirecional para Aplicação de Armazenadores de Energia na Mitigação de Oscilações em Redes de Distribuição Monofásicas de Baixa Tensão. In: CONGRESSO BRASILEIRO DE AUTOMÁTICA - 2020, Porto Alegre. **Anais do Congresso Brasileiro de Automática 2020.** [S.L.]: Sbabra, 2020. p. 1-7.

ENDERLE, Taciana Paula. **Análise, projeto e implantação de um D-STATCOM para redes de distribuição monofásica**. 2012. 117 f. Dissertação (Mestrado) - Curso de Engenharia Elétrica, Universidade Federal de Santa Maria, Santa Maria, 2012.

HART, Daniel W. **Power Electronics**. Valparaiso: Pearson Education, 1996. 494 p.

HOLDEFER, Antônio Eliseu. **Controle digitral de retificador trifásico utilizando o controlador TMS320LF2407**. 2004. 122 f. Dissertação (Mestrado) - Curso de Engenharia Elétrica, Universidade Federal de Santa Catarina, Florianópolis, 2004.

KONZEN, Joelmir Augusto. **Unidade de condicionamento de sinais como elemento de interface entre transformadores de instrumentação e dispositivo de aquisição de dados**. 2019. 93 f. Trabalho de Conclusão de Curso (Graduação) - Curso de Engenharia Elétrica, Universidade Federal de Santa Maria, Santa Maria, 2019.

LESSA, Dayane Mendonça. **Uso Combinado de Filtros Digitais com Circuitos de Sincronismo Monofásicos**. 2019. 104 f. Dissertação (Mestrado) - Curso de Engenharia Eletrônica, Universidade do Estado do Rio de Janeiro, Rio de Janeiro, 2019.

MANJREKAR, Madhav; VENKATARAMANAN, Gin. Control Strategies for a Hybrid Static Reactive Coimpensator. In: **conference on electrical and computer engineering**, v. 2, p. 834-837 v. 2, 26-19, 1996.

OLIVEIRA, Raimundo Nonato Moura de. **Conversor cc-cc pwm bidirecional trifásico com três portas e isolado em alta frequência para aplicação em sistemas fotovoltaicos**. 2018. 156 f. Dissertação (Mestrado) - Curso de Engenharia Elétrica, Universidade Federal do Ceará, Fortaleza, 2018.

RAUTH, D. A., RANDAL, V. T. **Analog-to-Digital Conversion**: Part 5 in a series of tutorials in instrumentation and measurement. 2005. IEEE Instrumentation & Measurement Magazine.

ROSA, Marcelo; WOLTER, Stefan Klaus. **Processadores de sinais digitais – DSP**. Curitiba: Universidade Federal do Paraná, 2007. (Apostila).

SOOMRO, Jahangeer; MEMON, Tayab D.; SHAH, M. A. Design and analysis of single phase voltage source inverter using Unipolar and Bipolar pulse width modulation techniques. **2016 International Conference On Advances In Electrical, Electronic And Systems Engineering (Icaees)**, [S.L.], p. 277-282, nov. 2016.

TEXAS INSTRUMENTS. **Software Phase Locked Loop Design Using C2000™ Microcontrollers for Single Phase Grid Connected Inverter**. [2017]. Disponível em: https://www.ti.com/lit/an/sprabt3a/sprabt3a.pdf. Acesso em: 10 Agosto. 2021.

TEXAS INSTRUMENTS. **Technical Reference Manual**. 2019. Disponível em: https://www.ti.com/lit/ug/spruhm8i/spruhm8i.pdf. Acesso em: 10 ago. 2021.

APÊNDICE A – CÓDIGO DO ALGORITMO IMPLEMENTADO NO DSP.

```
#include "Peripheral_Setup.h"
#include "math.h"
#include "sogi.h"
/**
 * Função principal
 */

SPLL_SOGI v_pll;
uint32_t count = 0, index = 0;
uint16_t sinetable[16];                    // Variável da tabela de senos.

uint16_t adc1 = 0;                          // A variável adc1 recebe a saida do
SOC0
uint16_t adc2 = 0;                          // A variável adc1 recebe a saida do
SOC1
uint16_t adc3 = 0;                          // A variável adc1 recebe a saida do
SOC2

uint16_t plot[16];
uint16_t plot1[16];
uint16_t *adc =  &adc1;
uint16_t *graf = &adc2;

// CRIAÇÃO DE FUNÇÕES.
__interrupt void isr_cpu_timer0(void);      // Função que foi criada para o
timer0.
__interrupt void isr_adc(void);             // Função que foi criada para o A/D.


int main(void){

    // INICIALIZA A CONFIGURAÇÃO DO DSP.
    InitSysCtrl();                          // Inicializa o sistema de controle
com alguns clock's.
    DINT;                                   // Desabilita toadas as interrupções
do DSP.
    InitPieCtrl();                          // Inicilaiza o controle dos
periféricos e seus estados.
    IER = 0x0000;                           // Desabilita o registrador de
habilitação.
    IFR = 0x0000;                           // Limpa o registrador de sinalização.
    InitPieVectTable();                     // Inicializa a tabela de
interrupções.


    // CHAMADA DAS FUNÇÕES QUE CONFIGURAMA OS MÓDULOS DO DSP
    Setup_GPIO();                           // Função responsável pela
configuração dos GPIO's.
    Setup_ePWM();                           // Função responsável pela
configuração dos PWM's.
    Setup_ADC();                            // Função responsável pela
configuração dos ADC's.


    // CONFIGURAÇÃO DAS INTERRUPÇÕES.
```

```
    EALLOW;                                     // Habilita a configuração de
registradores protegidos.
    CpuSysRegs.PCLKCR0.bit.CPUTIMER0 = 1;       // Habilitação do clock do
timer0.
    PieVectTable.TIMER0_INT = &isr_cpu_timer0;  // Associação da função que foi
criada para o timer0, com sua função no DSP.
    PieVectTable.ADCA1_INT = &isr_adc;          // Associação da função que foi
criada para o A/D A, com sua função no DSP.
    PieCtrlRegs.PIEIER1.bit.INTx7 = 1;          // Interrupção do timer0,
presente na linha 1 e coluna 7, habilita a coluna 7.
    PieCtrlRegs.PIEIER1.bit.INTx1 = 1;          // Interrupção do A/D A,
presente na linha 1 e coluna 1, habilita a coluna 1.
    EDIS;                                       // Desabilita a configuração de
regitradores protegidos.
    IER |= M_INT1;                              // Habilita a linha 1 da tabela
de interrupções.

    //CONFIGURAÇÃO DO TIMER
    InitCpuTimers();                            // Inicializa o timer.
    ConfigCpuTimer(&CpuTimer0, 200, 1000000);   // Configura o timer (Timer
utilizado, frequência do DSP, período do timer)
    CpuTimer0Regs.TCR.all = 0x4001;             // Habilita a interrupção dentro
do timer0.
    EINT;                                       // Habilita a interrupção global
    ERTM;                                       // Habilita o debug em tempo
real.

    SOGI_init(60, 25.00E-06, &v_pll);
    SOGI_coeff_update(25.00E-06, 376.99112, 0.7,&v_pll);

    //GERAÇÃO DE TABLE DE SENOS PARA VARIAR O CMPA.
    for(index = 0; index < 16 ; index++){
        sinetable[index] = (uint16_t)(500*(1.0 +
sin(6.28318531/16.0*((float)index))));
    }
    index = 0;


    GpioDataRegs.GPBDAT.bit.GPIO34 = 1;   //Seta um valor no pino, LED red fica
apagado.
    GpioDataRegs.GPADAT.bit.GPIO31 = 0;  //Seta um valor no pino, LED blue fica
apagado.

    //LOOP INFINITO PARA O PROGRAMA SER EXECUTADO DIVERSAS VEZES.
    while(1)
    {
        for(count = 0; count < 0x00FFFFFF; count++)
        {

        }
        GpioDataRegs.GPBTOGGLE.bit.GPIO34 = 1; // Alterna entre 1 e 0 O valor
setado no PWM.

    }
    return 0;
}

// FUNÇÃO DA INTERRUPÇÃO DO TIMER0 E SUA ROTINA
__interrupt void isr_cpu_timer0(void)
```

```
{
    GpioDataRegs.GPATOGGLE.bit.GPIO31 = 1;    // Alterna entre 1 e 0 O valor
setado no PWM (LED blue).
    PieCtrlRegs.PIEACK.all = PIEACK_GROUP1;   // Limpa o registrador de
sinalização após a ocorrência da interrupção .
}

__interrupt void isr_adc(void)
{
    GpioDataRegs.GPADAT.bit.GPIO14 = 1;

    adc1 =  AdcaResultRegs.ADCRESULT0;          // A variável adc1 recebe a saida
do SOC0
    adc2 =  AdcaResultRegs.ADCRESULT1;          // A variável adc1 recebe a saida
do SOC1
    adc3 =  AdcaResultRegs.ADCRESULT2;          // A variável adc1 recebe a saida
do SOC2

    index = (index == 15) ? 0 : (index+1);     // Variação da varável index da
tabela de senos.

    EPwm2Regs.CMPA.bit.CMPA = sinetable[index]; // Variação do CMPA do PWM2

    plot[index] = *adc;
    plot1[index] = *graf;

    AdcaRegs.ADCINTFLGCLR.bit.ADCINT1 = 1;      //clear INT1 flag
    PieCtrlRegs.PIEACK.all = PIEACK_GROUP1;
    GpioDataRegs.GPADAT.bit.GPIO14 = 0;
}



/*
 * Peripheral_Setup.c
 *
 *  Configuração dos módulos
 *
 */
#include "Peripheral_Setup.h"

// FUNÇÃO RESPONSÁVEL PELA CONFIGURAÇÃO DOS GPIO's.
void Setup_GPIO(void)
{
    EALLOW; // Registrador que habilita a edição de registradores protegidos.

    //CONFIGURAÇÃO DO GPIO DO LED BLUE.
    GpioCtrlRegs.GPAGMUX2.bit.GPIO31 = 0;                   // Configura o
agrupamento para usar como GPIO.
    GpioCtrlRegs.GPAMUX2.bit.GPIO31 = 0;                    // Confiura o pino como
GPIO.
    GpioCtrlRegs.GPAPUD.bit.GPIO31 = 1;                     // Desabilita o
resistor de pull-up
    GpioCtrlRegs.GPADIR.bit.GPIO31 = 1;                     // Configurado como
pino de saida
    GpioCtrlRegs.GPACSEL4.bit.GPIO31 = GPIO_MUX_CPU1;
    GpioCtrlRegs.GPACSEL4.bit.GPIO31 = GPIO_MUX_CPU1;

    //CONFIGURAÇÃO DO GPIO DO LED RED.
```

```
    GpioCtrlRegs.GPBGMUX1.bit.GPIO34 = 0;                 // Configura o
agrupamento para usar como GPIO.
    GpioCtrlRegs.GPBMUX1.bit.GPIO34 = 0;                  // Confiura o pino
como GPIO.
    GpioCtrlRegs.GPBPUD.bit.GPIO34 = 1;                   // Desabilita o
resistor de pull-up
    GpioCtrlRegs.GPBDIR.bit.GPIO34 = 1;                   // Configurado como
pino de saida
    GpioCtrlRegs.GPBCSEL1.bit.GPIO34 = GPIO_MUX_CPU1;
    GpioCtrlRegs.GPBCSEL1.bit.GPIO34 = GPIO_MUX_CPU1;


    //CONFIGURAÇÃO GPIO DO PWM 1A
    GpioCtrlRegs.GPAGMUX1.bit.GPIO0 = 0;                  // Configura o
agrupamento para usar como PWM.
    GpioCtrlRegs.GPAMUX1.bit.GPIO0 = 1;                   // Como o pino como
PWM.
    GpioCtrlRegs.GPAPUD.bit.GPIO0 = 1;                    // Desabilita o
resistor de pull-up

    //CONFIGURAÇÃO DO GIO DO PWM 1B
    GpioCtrlRegs.GPAGMUX1.bit.GPIO1 = 0;                  // Configura o
agrupamento para usar como PWM.
    GpioCtrlRegs.GPAMUX1.bit.GPIO1 = 1;                   // Como o pino como
PWM.
    GpioCtrlRegs.GPAPUD.bit.GPIO1 = 1;                    // Desabilita o
resistor de pull-up

    //CONFIGURAÇÃO GPIO DO PWM 2A
    GpioCtrlRegs.GPAGMUX1.bit.GPIO2 = 0;                  // Configura o
agrupamento para usar como PWM.
    GpioCtrlRegs.GPAMUX1.bit.GPIO2 = 1;                   // Como o pino como
PWM.
    GpioCtrlRegs.GPAPUD.bit.GPIO2 = 1;                    // Desabilita o
resistor de pull-up

    //CONFIGURAÇÃO DO GIO DO PWM 2B
    GpioCtrlRegs.GPAGMUX1.bit.GPIO3 = 0;                  // Configura o
agrupamento para usar como PWM.
    GpioCtrlRegs.GPAMUX1.bit.GPIO3 = 1;                   // Como o pino como
PWM.
    GpioCtrlRegs.GPAPUD.bit.GPIO3 = 1;                    // Desabilita o
resistor de pull-up


    //CONFIGURAÇÃO DO GPIO DO PWM 7A
    GpioCtrlRegs.GPEGMUX2.bit.GPIO157 = 0;                // Configura o
agrupamento para usar como PWM.
    GpioCtrlRegs.GPEMUX2.bit.GPIO157 = 1;                 // Como o pino como PWM.
    GpioCtrlRegs.GPEPUD.bit.GPIO157 = 1;                  // Desabilita o resistor
de pull-up

    //CONFIGURAÇÃO DO GPIO DO PWM 7B
    GpioCtrlRegs.GPEGMUX2.bit.GPIO158 = 0;                // Configurado como pino
de saida
    GpioCtrlRegs.GPEMUX2.bit.GPIO158 = 1;                 // Como o pino como PWM.
    GpioCtrlRegs.GPEPUD.bit.GPIO158 = 1;                  // Desabilita o resistor
de pull-up
```

```
    //Testa a entrada do ADC no PWM.
    GpioCtrlRegs.GPAGMUX1.bit.GPIO14 = 0;                // Configurado como pino
de saida.
    GpioCtrlRegs.GPAMUX1.bit.GPIO14 = 0;                 // Como o pino como GPIO.
    GpioCtrlRegs.GPAPUD.bit.GPIO14 = 1;                  // Desabilita o resistor
de pull-up.
    GpioCtrlRegs.GPADIR.bit.GPIO14 = 1;                  // onfigurado como pino
de saida.
    GpioCtrlRegs.GPACSEL2.bit.GPIO14 = GPIO_MUX_CPU1;

    EDIS;
}

// FUNÇÃO RESPONSÁVEL PELA CONFIGURAÇÃO DOS PWM's.
void Setup_ePWM(void)
{
    EALLOW; // Registrador que habilita a edição de registradores protegidos.

    CpuSysRegs.PCLKCR2.bit.EPWM1 = 1;                   // Habilita o clock do
PWM1.
    CpuSysRegs.PCLKCR2.bit.EPWM2 = 1;                   // Habilita o clock do
PWM2.


    CpuSysRegs.PCLKCR0.bit.TBCLKSYNC = 0;               // Desabilita o clock do
sistema para configurar oa registradores.

    EPwm1Regs.TBPRD = 2500;                             // Valor do período do PWM,
de acordo com a fórmula.
    EPwm1Regs.CMPA.bit.CMPA = 0;    // Ciclo de trabaho, neste caso é 50%

    EPwm1Regs.TBPHS.bit.TBPHS = 0;                      // Deslocamento de fase.
    EPwm1Regs.TBCTL.bit.SYNCOSEL = TB_CTR_ZERO;         // Registrador de
sincronismo, neste caso o PWM1 é a referência.
    EPwm1Regs.TBCTR = 0x0000;                           // Contador do PWM, neste
caso vai de 0 a 2000.
    EPwm1Regs.TBCTL.bit.CTRMODE = TB_COUNT_UPDOWN;      // Forma de onda dente
serra, neste caso é UP/DOWN
    EPwm1Regs.TBCTL.bit.PHSEN = TB_DISABLE;             // Desablita o deslocamento
de fase, pois o PWM1 é referência.
    EPwm1Regs.TBCTL.bit.HSPCLKDIV = TB_DIV1;            // Realiza a divisão do
clock do DSP.
    EPwm1Regs.TBCTL.bit.CLKDIV = TB_DIV1;               // Realiza a divisão do
clock do DSP.

    EPwm1Regs.CMPCTL.bit.SHDWAMODE = CC_SHADOW;         // Habilita a máscara,
PWMA.
    EPwm1Regs.CMPCTL.bit.LOADAMODE = CC_CTR_ZERO_PRD;   // O valor do período ou do
PRD, só vai ser atualizado no zero ou no período.
    EPwm1Regs.CMPCTL.bit.SHDWBMODE = CC_SHADOW;         // Habilita a máscara,
PWMB.
    EPwm1Regs.CMPCTL.bit.LOADBMODE = CC_CTR_ZERO_PRD;   // O valor do período ou do
PRD, só vai ser atualizado no zero ou no período.

    EPwm1Regs.AQCTLA.bit.PRD = AQ_NO_ACTION;            // Quando contador for
igual ao períodod, nehuma ação é relizada.
```

```
    EPwm1Regs.AQCTLA.bit.ZRO = AQ_NO_ACTION;           // Quando o contador for
igual ao zero, nehuma ação é relizada.
    EPwm1Regs.AQCTLA.bit.CAU = AQ_CLEAR;               // Quando o contador for
igual ao CMPA e estiver subindo, um valor 0 é setado no GPIO.
    EPwm1Regs.AQCTLA.bit.CAD = AQ_SET;                 // Quando o contador for
igual ao CMPA e estiver descendo, um valor 1 é setado no GPIO.

    EPwm1Regs.DBCTL.bit.POLSEL = DB_ACTV_HIC;          // Habilita o PWM
complementar invertido
    EPwm1Regs.DBCTL.bit.OUT_MODE = DB_FULL_ENABLE;     // Habilita o tempo morto.
    EPwm1Regs.DBFED.bit.DBFED = 100;                   // Fall Time, tempo entre o
comando para desligar e a ocorrência do desligamento.
    EPwm1Regs.DBRED.bit.DBRED = 100;                   // Tempo entre comando para
ligar e a ocorrência do ligamento.

    //Trigger ADC
    EPwm1Regs.ETSEL.bit.SOCAEN = 1;                    // Habilita os SOC's do
módulo A.
    EPwm1Regs.ETSEL.bit.SOCASEL = ET_CTR_PRDZERO;      // Evento que dispara o
A/D, neste caso quando o contador for igual ao período e a zero.
    EPwm1Regs.ETPS.bit.SOCAPRD = ET_1ST;               // Dispara o ADC sempre no
primeiro evento (PRD ou ZER)


    // PWM 2A e 2B
    EPwm2Regs.TBPRD = 2500;                            // Valor do período do PWM,
de acordo com a fórmula.
    EPwm2Regs.CMPA.bit.CMPA = 0;                       // Ciclo de trabaho, neste
caso é 50%
    EPwm2Regs.TBPHS.bit.TBPHS = 25000;                  // Deslocamento de fase
(180°).
    EPwm2Regs.TBCTL.bit.SYNCOSEL = TB_SYNC_IN;         // Registrador de
sincronismo, neste caso o PWM7 recebe o sincronismo do PM1.
    EPwm2Regs.TBCTR = 0x0000;                          // Contador do PWM, neste
caso vai de 0 a 2000.
    EPwm2Regs.TBCTL.bit.CTRMODE = TB_COUNT_UPDOWN;     // Forma de onda dente
serra, neste caso é UP/DOWN
    EPwm2Regs.TBCTL.bit.PHSEN = TB_ENABLE;             // Hablita o deslocamento
de fase, pois o PWM1 é referência.
    EPwm2Regs.TBCTL.bit.PHSDIR = TB_DOWN;              // A fase irá atrasar.
    EPwm2Regs.TBCTL.bit.HSPCLKDIV = TB_DIV1;           // Realiza a divisão do
clock do DSP.
    EPwm2Regs.TBCTL.bit.CLKDIV = TB_DIV1;              // Realiza a divisão do
clock do DSP.

    EPwm2Regs.CMPCTL.bit.SHDWAMODE = CC_SHADOW;        // Habilita a máscara,
PWMA.
    EPwm2Regs.CMPCTL.bit.LOADAMODE = CC_CTR_ZERO_PRD;  // O valor do período ou do
PRD, só vai ser atualizado no zero ou no período.
    EPwm2Regs.CMPCTL.bit.SHDWBMODE = CC_SHADOW;        // Habilita a máscara,
PWMB.
    EPwm2Regs.CMPCTL.bit.LOADBMODE = CC_CTR_ZERO_PRD;  // O valor do período ou do
PRD, só vai ser atualizado no zero ou no período.

    EPwm2Regs.AQCTLA.bit.PRD = AQ_NO_ACTION;           // Quando contador for
igual ao períodod, nehuma ação é relizada.
    EPwm2Regs.AQCTLA.bit.ZRO = AQ_NO_ACTION;           // Quando o contador for
igual ao zero, nehuma ação é relizada.
```

```
    EPwm2Regs.AQCTLA.bit.CAU = AQ_CLEAR;               // Quando o contador for
igual ao CMPA e estiver subindo, um valor 0 é setado no GPIO.
    EPwm2Regs.AQCTLA.bit.CAD = AQ_SET;                 // Quando o contador for
igual ao CMPA e estiver descendo, um valor 1 é setado no GPIO.

    EPwm2Regs.DBCTL.bit.POLSEL = DB_ACTV_HIC;          // Habilita o PWM
complementar invertido
    EPwm2Regs.DBCTL.bit.OUT_MODE = DB_FULL_ENABLE;     // Habilita o tempo morto.
    EPwm2Regs.DBFED.bit.DBFED = 100;                   // Fall Time, tempo entre o
comando para desligar e a ocorrência do desligamento.
    EPwm2Regs.DBRED.bit.DBRED = 100;                   // Tempo entre comando para
ligar e a ocorrência do ligamento.


    CpuSysRegs.PCLKCR0.bit.TBCLKSYNC = 1;              // Habilita o clock do
sistema.

    EDIS;   // Desabilita a configuração de regitradores protegidos.
}

// FUNÇÃO RESPONSÁVEL PELA CONFIGURAÇÃO DOS A/D's.
void Setup_ADC(void)
{
    Uint16 acqps;                          // Variável da janela de amostragem.

    if(ADC_RESOLUTION_12BIT == AdcaRegs.ADCCTL2.bit.RESOLUTION)  // Valor da
janela de amostregem de acordo com a resolução do A/D.
        acqps = 14;                        // Janela de amostragem: tempo que o
circuto do A/D fica ligado para capturar o sinal. (75ns)
    else
        acqps = 63;                        // Janela de amostragem: tempo que o
circuto do A/D fica ligado para capturar o sinal.

    EALLOW;                               // Registrador que habilita a edição de
registradores protegidos.

    CpuSysRegs.PCLKCR13.bit.ADC_A = 1;                                  // Habilita
o clock do ADCA.
    AdcaRegs.ADCCTL2.bit.PRESCALE = 6;                                  // Número
de pulsos do clock do A/D.
    AdcSetMode(ADC_ADCA, ADC_RESOLUTION_12BIT, ADC_SIGNALMODE_SINGLE); // Função
que habilita o módulo (ADCA), a resolução (12 bits) e o tipo de amostragem
(single)
    AdcaRegs.ADCCTL1.bit.INTPULSEPOS = 1;                               // A
geração de pulso de interrupção ocorre no final da conversão.
    AdcaRegs.ADCCTL1.bit.ADCPWDNZ = 1;                                  // Todos os
circuitos analógicos dentro do núcleo estão ligados.
    DELAY_US(1000);

    AdcaRegs.ADCSOC0CTL.bit.CHSEL = 3;                                  // Escolha
do canal ADINA3 para o SOC0.
    AdcaRegs.ADCSOC0CTL.bit.ACQPS = acqps;                              // Tamanho
da janela de amostragem.
    AdcaRegs.ADCSOC0CTL.bit.TRIGSEL = 0x05;                             // Evento
que vai disparar o ADC (TRIG_SEL_ePWM1_SOCA).

    AdcaRegs.ADCSOC1CTL.bit.CHSEL = 4;                                  // Escolha
do canal ADINA3 para o SOC1.
```

```
    AdcaRegs.ADCSOC1CTL.bit.ACQPS = acqps;                          // Tamanho
da janela de amostragem.
    AdcaRegs.ADCSOC1CTL.bit.TRIGSEL = 0x05;                         // Evento
que vai disparar o ADC (TRIG_SEL_ePWM1_SOCA).

    AdcaRegs.ADCSOC2CTL.bit.CHSEL = 5;                              // Escolha
do canal ADINA3 para o SOC2.
    AdcaRegs.ADCSOC2CTL.bit.ACQPS = acqps;                          // Tamanho
da janela de amostragem.
    AdcaRegs.ADCSOC2CTL.bit.TRIGSEL = 0x05;                         // Evento
que vai disparar o ADC (TRIG_SEL_ePWM1_SOCA).

    AdcaRegs.ADCINTSEL1N2.bit.INT1SEL = 0x02;                       // Momento
de dipsaro da interrupção do ADC, aós a conversão do SOC2
    AdcaRegs.ADCINTSEL1N2.bit.INT1E = 1;                            // Habilita
a interrupção.
    AdcaRegs.ADCINTFLGCLR.bit.ADCINT1 = 1;                          // Limpa o
flag.

    EDIS;                                   // Desabilita a configuração de
regitradores protegidos.
}



/*
 * sogi.h
 *
 *
 *
 */

#ifndef SOGI_H_
#define SOGI_H_


typedef struct
{
      float  osg_k;
      float  osg_x;
      float  osg_y;
      float  osg_b0;
      float  osg_b2;
      float  osg_a1;
      float  osg_a2;
      float  osg_qb0;
      float  osg_qb1;
      float  osg_qb2;
}SPLL_SOGI_OSG_COEFF;

typedef struct
{
      float  B1_lf;
      float  B0_lf;
      float  A1_lf;
}SPLL_SOGI_LPF_COEFF;

typedef struct
{
      float u[3];                // em 1ph AC Sinal medido e normalizado.
```

```
	float osg_u[3];			// Tensão de rede estimada
	float osg_qu[3];	// Tensão de rede ortogonal estimada
	float u_Q[2];			// fora do componente do eixo Q da rede estimada
	float u_D[2];			// fora do componente do eixo D da rede estimada
	float ylf[2];
	float fo;			// Saída PLL de frequência de rede instantânea
	float fn;			// Frequência nominal da rede
	float theta[2];			// Ângulo de fase da rede
	float cos_;			// out Cos(Ângulo de fase da rede)
	float sin_;			// Sin(Ângulo de fase da rede)
	float delta_T;			// 1/Frequency of calling the PLL routine
	float delta_fn;
	SPLL_SOGI_OSG_COEFF osg_coeff;
	SPLL_SOGI_LPF_COEFF lpf_coeff;
}SPLL_SOGI;

void SOGI_init(float Grid_freq, float DELTA_T, SPLL_SOGI *spll_obj);
void SOGI_coeff_update(float delta_T, float wn, float k, SPLL_SOGI *spll);
void SPLL_SOGI_CALC(SPLL_SOGI *spll_obj);
void SOGI_CALC(SPLL_SOGI *obj);
//SPLL_SOGI  sogi_pll, sogi_vS, sogi_iL, sogi_1h, sogi_3h, sogi_5h, sogi_7h,
sogi_9h;

#endif /* NANOGRID_BGIC_SOGI_H_ */



/*
 * sogi.c
 *
 *
 *
 */
#include "sogi.h"
#include "math.h"
void SOGI_init(float Grid_freq, float DELTA_T, SPLL_SOGI *spll_obj)
{
	spll_obj->u[0] = 0.0;
	spll_obj->u[1] = 0.0;
	spll_obj->u[2] = 0.0;

	spll_obj->osg_u[0] = 0.0;
	spll_obj->osg_u[1] = 0.0;
	spll_obj->osg_u[2] = 0.0;

	spll_obj->osg_qu[0] = 0.0;
	spll_obj->osg_qu[1] = 0.0;
	spll_obj->osg_qu[2] = 0.0;

	spll_obj->u_Q[0] = 0.0;
	spll_obj->u_Q[1] = 0.0;

	spll_obj->u_D[0] = 0.0;
	spll_obj->u_D[1] = 0.0;

	spll_obj->ylf[0] = 0.0;
	spll_obj->ylf[1] = 0.0;

	spll_obj->fo = 0;
```

```
	spll_obj->fn = Grid_freq;

	spll_obj->theta[0] = 0.0;
	spll_obj->theta[1] = 0.0;

	spll_obj->sin_ = 0.0;
	spll_obj->cos_ = 0.0;
	spll_obj->delta_fn = 0.0;

	// loop filter coefficients for 30720Hz
	spll_obj->lpf_coeff.B0_lf = 222.47;
	spll_obj->lpf_coeff.B1_lf = -221.85;
	spll_obj->lpf_coeff.A1_lf = -1.0;

	spll_obj->delta_T = DELTA_T;
}

void SOGI_coeff_update(float delta_T, float wn, float k, SPLL_SOGI *spll)
{
	float osgx,osgy,temp;
	spll->osg_coeff.osg_k = k;
	osgx=(2.0*0.5*wn*delta_T);
	spll->osg_coeff.osg_x = osgx;
	osgy = wn*delta_T*wn*delta_T;
	spll->osg_coeff.osg_y = osgy;
	temp = 1.0/(osgx+osgy+4.0);
	spll->osg_coeff.osg_b0 = osgx*temp;
	spll->osg_coeff.osg_b2 = (-1.0)*spll->osg_coeff.osg_b0;
	spll->osg_coeff.osg_a1 = (2.0*(4.0-osgy))*temp;
	spll->osg_coeff.osg_a2 = (osgx-osgy-4)*temp;
	spll->osg_coeff.osg_qb0 = (0.5*osgy)*temp;
	spll->osg_coeff.osg_qb1 = spll->osg_coeff.osg_qb0*(2.0);
	spll->osg_coeff.osg_qb2 = spll->osg_coeff.osg_qb0;
}

void SPLL_SOGI_CALC(SPLL_SOGI *spll_obj)
{
	// Update the spll_obj->u[0] with the grid value before calling this routine
	// Orthogonal Signal Generator   //
	spll_obj->osg_u[0] = (spll_obj->osg_coeff.osg_b0*(spll_obj->u[0]-spll_obj-
>u[2])) + (spll_obj->osg_coeff.osg_a1*spll_obj->osg_u[1]) + (spll_obj-
>osg_coeff.osg_a2*spll_obj->osg_u[2]);
	spll_obj->osg_u[2] = spll_obj->osg_u[1];
	spll_obj->osg_u[1] = spll_obj->osg_u[0];
	spll_obj->osg_qu[0] = (spll_obj->osg_coeff.osg_qb0*spll_obj->u[0]) +
(spll_obj->osg_coeff.osg_qb1*spll_obj->u[1]) + (spll_obj-
>osg_coeff.osg_qb2*spll_obj->u[2]) + (spll_obj->osg_coeff.osg_a1*spll_obj-
>osg_qu[1]) + (spll_obj->osg_coeff.osg_a2*spll_obj->osg_qu[2]);
	spll_obj->osg_qu[2] = spll_obj->osg_qu[1];
	spll_obj->osg_qu[1] = spll_obj->osg_qu[0];
	spll_obj->u[2] = spll_obj->u[1];
	spll_obj->u[1] = spll_obj->u[0];
	// Park Transform from alpha beta to d-q axis                    //
	spll_obj->u_Q[0] = (spll_obj->cos_*spll_obj->osg_u[0]) + (spll_obj-
>sin_*spll_obj->osg_qu[0]);
	spll_obj->u_D[0] = (spll_obj->cos_*spll_obj->osg_qu[0]) - (spll_obj-
>sin_*spll_obj->osg_u[0]);
	// Loop Filter                    //
```

```
	spll_obj->ylf[0]=spll_obj->ylf[1] + (spll_obj->lpf_coeff.B0_lf*spll_obj-
>u_Q[0]) + (spll_obj->lpf_coeff.B1_lf*spll_obj->u_Q[1]);
	spll_obj->ylf[1]=spll_obj->ylf[0];
	spll_obj->u_Q[1]=spll_obj->u_Q[0];

	spll_obj->fo=spll_obj->fn+spll_obj->ylf[0];
	spll_obj->theta[0]=spll_obj->theta[1] + (spll_obj->fo*spll_obj-
>delta_T)*6.2831853072;

	if(spll_obj->theta[0] > 6.2831853072)
		spll_obj->theta[0] -= 6.2831853072;
	else if(spll_obj->theta[0] < -6.2831853072)
		spll_obj->theta[0] += 6.2831853072;

	spll_obj->theta[1] = spll_obj->theta[0];
	//spll_obj->sin_ = __sin(spll_obj->theta[0]);
	//spll_obj->cos_ = __cos(spll_obj->theta[0]);

	spll_obj->sin_ = sin(spll_obj->theta[0]);
	spll_obj->cos_ = cos(spll_obj->theta[0]);

}

void SOGI_CALC(SPLL_SOGI *obj)
{
	// Geração de sinal ortogonal.
	obj->osg_u[0] = (obj->osg_coeff.osg_b0*(obj->u[0]-obj->u[2])) + (obj-
>osg_coeff.osg_a1*obj->osg_u[1]) + (obj->osg_coeff.osg_a2*obj->osg_u[2]);
	obj->osg_u[2] = obj->osg_u[1];
	obj->osg_u[1] = obj->osg_u[0];

	obj->osg_qu[0] = (obj->osg_coeff.osg_qb0*obj->u[0]) + (obj-
>osg_coeff.osg_qb1*obj->u[1]) + (obj->osg_coeff.osg_qb2*obj->u[2]) + (obj-
>osg_coeff.osg_a1*obj->osg_qu[1]) + (obj->osg_coeff.osg_a2*obj->osg_qu[2]);
	obj->osg_qu[2] = obj->osg_qu[1];
	obj->osg_qu[1] = obj->osg_qu[0];

	obj->u[2] = obj->u[1];
	obj->u[1] = obj->u[0];

	// Park Transform from alpha beta to d-q axis                    //
	obj->u_Q[0] = (obj->cos_*obj->osg_u[0]) + (obj->sin_*obj->osg_qu[0]);
	obj->u_D[0] = (obj->cos_*obj->osg_qu[0]) - (obj->sin_*obj->osg_u[0]);
}

////////////////////////////////////////////////////SOGI PLL
```