TRABALHO DE GRADUAÇÃO

IMPLEMENTAÇÃO DE UM ALGORITMO PLL PARA SINCRONIZAÇÃO DE CONVERSORES DE ENERGIA COM A REDE ELÉTRICA

Thalles Martins Feitosa Cid

Brasília, julho de 2016

UNIVERSIDADE DE BRASÍLIA

UNIVERSIDADE DE BRASÍLIA
Faculdade de Tecnologia
Curso de Graduação em Engenharia de Controle e Automação

TRABALHO DE GRADUAÇÃO

IMPLEMENTAÇÃO DE UM ALGORITMO PLL PARA SINCRONIZAÇÃO DE CONVERSORES DE ENERGIA COM A REDE ELÉTRICA

Thalles Martins Feitosa Cid

Trabalho de Graduação submetido ao Departamento de Engenharia Elétrica da Universidade de Brasília como parte dos requisitos para a obtenção do grau de Engenheiro de Controle e Automação

Banca Examinadora

Prof. Lélio Ribeiro Soares Júnior, ENE/UnB _____
Orientador

Prof. Francisco Damasceno Freitas, ENE/UnB _____
Examinador

Prof. Gerson Henrique Pfitscher, ENE/UnB _____
Examinador

FICHA CATALOGRÁFICA

CID, THALLES MARTINS FEITOSA

IMPLEMENTAÇÃO DE UM ALGORITMO PLL PARA SINCRONIZAÇÃO DE CONVERSORES DE ENERGIA COM A REDE ELÉTRICA [Distrito Federal] 2016.

xvi, 46 p., 210 x 297 mm (ENE/FT/UnB, Engenheiro, Engenharia de Controle e Automação, 2016).

Trabalho de Graduação - Universidade de Brasília, Faculdade de Tecnologia.

Departamento de Engenharia Elétrica

1. PLL
2. Componentes simétricas
3. Sincronização com a rede
4. Fontes renováveis

I. Mecatrônica/FT/UnB

II. Título (série)

REFERÊNCIA BIBLIOGRÁFICA

CID, T. M. F. (2016). *IMPLEMENTAÇÃO DE UM ALGORITMO PLL PARA SINCRONIZAÇÃO DE CONVERSORES DE ENERGIA COM A REDE ELÉTRICA*. Trabalho de Graduação em Engenharia de Controle e Automação, Publicação FT TG n.º 12, Faculdade de Tecnologia, Universidade de Brasília, Brasília, DF, 46 p.

CESSÃO DE DIREITOS

Thalles Martins Feitosa Cid

Quadra 10 Conjunto K Casa 3 - Setor Sul

Gama - DF - Brasil

CEP: 72415-511

Agradecimentos

Agradeço a toda a minha família, especialmente aos meus pais e meus irmãos, os quais, diariamente, desde pequeno, me mostraram o quanto desejavam a minha felicidade e sucesso, ao me motivar a ser a melhor versão de mim mesmo. Meu muito obrigado aos meus amigos César Guimarães, Ravena Borges, Paulo Bahia e Emanuel Neto, que mantiveram minha sanidade nesses anos de graduação com conversas quase nunca sérias e conselhos quase sempre sábios. Sou extremamente grato ao meu orientador, prof. Lélio Ribeiro Soares Júnior, um dos melhores professores que tive o prazer de conhecer. Não poderia ter escolhido melhor orientador. E, finalmente, agradeço a todo o pessoal do Laboratório de Qualidade de Energia Elétrica, especialmente ao João Pedro, ao Marcos Diego e ao prof. Anésio Filho. Vocês me ajudaram demais.

Thalles Martins Feitosa Cid

RESUMO

A conexão de plantas de energia renovável à rede elétrica é mediada por conversores de energia, que alteram a tensão gerada para compatibilizá-la com as tensões da rede. Esses equipamentos são controlados por sinais produzidos por um processador de sinais digitais (DSP). O ângulo de fase da tensão fundamental da rede é uma das informações que o DSP deve possuir para produzir os sinais de controle adequados. A obtenção do ângulo de fase da tensão é geralmente realizada por um PLL. Os PLLs modernos devem ser capazes de obter corretamente o ângulo de fase da rede mesmo na presença de fenômenos que afetam a qualidade da energia, como desequilíbrios e distorções. Um tipo de PLL com essas características exposto na literatura pertinente é o DSOGI-PLL, que é descrito, simulado e implementado neste trabalho.

ABSTRACT

The connection of renewable energy plants to the grid is mediated by power converters, which change the generated voltage to adjust it with the voltage of the grid. These devices are controlled by signals produced by a DSP (Digital Signal Processor). The phase angle of the fundamental voltage grid is an information that the DSP must possess for producing appropriate control signals. The obtaining of the phase angle is usually performed by a PLL (Phase-Locked Loop). The modern PLLs should be able to get correctly the phase angle of the grid even in the presence of phenomena that affect the quality of power, such as unbalances and distortions. One type of PLL with those features found in the literature is the DSOGI-PLL, which is described, simulated, and implemented in this work.

SUMÁRIO

LISTA DE FIGURAS

LISTA DE SÍMBOLOS

Símbolos latinos

v_x	Tensão da fase x de uma rede polifásica	[V]
qv_r	Tensão em quadratura com a tensão v_r	[V]
$T_{\alpha\beta}$	Matriz da transformação de Clarke	
T_{dq}	Matriz da transformação de Park	
k_p	Ganho da ação proporcional de um controlador PI	$[\mathrm{V}^{-1} \cdot \mathrm{s}^{-1}]$
k_i	Ganho da ação integral de um controlador PI	$[\mathrm{V}^{-1} \cdot \mathrm{s}^{-2}]$
K	Ganho de um SOGI	
det $\mathbf{M}$	Determinante da matriz $\mathbf{M}$	
adj $\mathbf{M}$	Matriz adjunta da matriz $\mathbf{M}$	
a	Operador de Fortescue no domínio do tempo	
T	Período de amostragem	[s]
$\hat{U}$	Fasor U	

Símbolos gregos

α	Operador de Fortescue no domínio da frequência	
ω	Frequência da rede elétrica	[Hz]
θ	Ângulo de fase da rede elétrica	[rad]

Sobrescritos

$+$	Indica a componente simétrica da sequência positiva de certa grandeza
$'$	Indica uma grandeza estimada

Siglas

DSOGI	Dual Second Order Generalized Integrator
FLL	Frequency Locked Loop
PLL	Phase Locked Loop
PSC	Positive Sequence Calculator
QSG	Quadrature Signal Generator
SRF	Synchronous Reference Frame

1 INTRODUÇÃO

1.1 MOTIVAÇÃO E OBJETIVO DO TRABALHO

Nos últimos anos, tem havido um significante crescimento da inserção de fontes de energia renovável na rede elétrica. Sistemas eólicos e fotovoltaicos já participam de forma considerável no suprimento de potência na rede elétrica de algumas nações, como a Dinamarca. Nesse país, 28 % da energia elétrica consumida provém do vento em contraposição ao Brasil, no qual essa taxa é de apenas 1 % [1]. Estima-se que, em comparação ao crescimento da hulha ou do linhito, as energias renováveis experimentaram um crescimento de 30 % ao ano. O objetivo da comunidade europeia é que, até 2020, 20 % da energia elétrica gerada nos países-membros provenha de fontes renováveis [2].

Os Estados Unidos, que detém 22 % da produção mundial de energia, adotou políticas similares às da Europa, devido, principalmente, à opinião pública e para superar uma crise econômica nos anos de 2008 a 2010. Apesar de os países europeus e os EUA estarem fomentando políticas ambientais inovadoras com relação à eficiência energética, os países emergentes da Ásia e do Pacífico, como a Índia e a China, constituem uma preocupação maior, pois, a cada ano, a demanda por energia desses países aumenta. Na China, por exemplo, desde o ano 2000, a demanda por energia cresceu um ponto percentual a cada ano [3].

No Brasil, a matriz energética ainda é predominantemente hidrelétrica, com 64,33 % do total de energia gerada. Porém, a legislação ambiental no país tem exigido, cada vez mais, que se evite a construção de usinas com reservatório, devido, principalmente, ao impacto ambiental e social que elas provocam [4]. Além da fonte hídrica, o Brasil possui termelétricas, que são fontes altamente poluidoras e também a energia nuclear, da qual decorre, invariavelmente, a produção de resíduos tóxicos radioativos. Nesse cenário, as fontes renováveis de energia surgem como uma alternativa viável. Boa parte dos investimentos em projetos de sistemas de geração de energia no Brasil é destinada a projetos de plantas de geração fotovoltaicas e eólicas. O objetivo é que, até 2040, a participação das energias fotovoltaica, eólica e de biomassa passe de seus atuais 14 % para 51 % no Brasil [5].

A tensão gerada através de sistemas eólicos e fotovoltaicos não pode ser introduzida na rede elétrica diretamente. Antes, é preciso que ela seja sincronizada com a tensão da rede principal, isto é, a tensão gerada deve ter amplitude, frequência e fase adaptadas para tornar-se com-

patível com as tensões da rede. A adaptação da tensão gerada é feita por conversores de energia, que operam controlados por sinais provenientes de um DSP (*Digital Signal Processor*).

A Fig. 1.1 ilustra, de forma simplificada, a conexão de um aerogerador à rede elétrica. O gerador da turbina eólica produz tensão alternada. Para compatibilizar a tensão gerada com as tensões da rede, deve-se adaptar amplitude, fase e frequência da tensão produzida. Isso se consegue inserindo-se a tensão gerada num conversor de energia, o qual, neste caso, é formado por um retificador e um inversor em sequência. O sucesso das conversões realizadas pelo retificador e inversor depende do chaveamento adequado desses equipamentos. As chaves são controladas por sinais oriundos de um DSP, o qual, para produzir sinais de controle corretos, precisa obter continuamente informações da tensão da planta de geração e da tensão da rede elétrica. Uma das informações mais importantes é a fase das tensões de interesse. Para obter a fase, usualmente é utilizado um sistema PLL (*Phase-Locked Loop*).

Por meio de um computador, formaliza-se, numa linguagem de programação compreendida pelo DSP, o algoritmo que descreve o PLL e carrega-se o código gerado nesse dispositivo. Uma vez inserido nele, o código passa a ser executado ininterruptamente, realizando a coleta de tensões da rede a intervalos de tempo regulares e estimando os respectivos ângulos de fase.

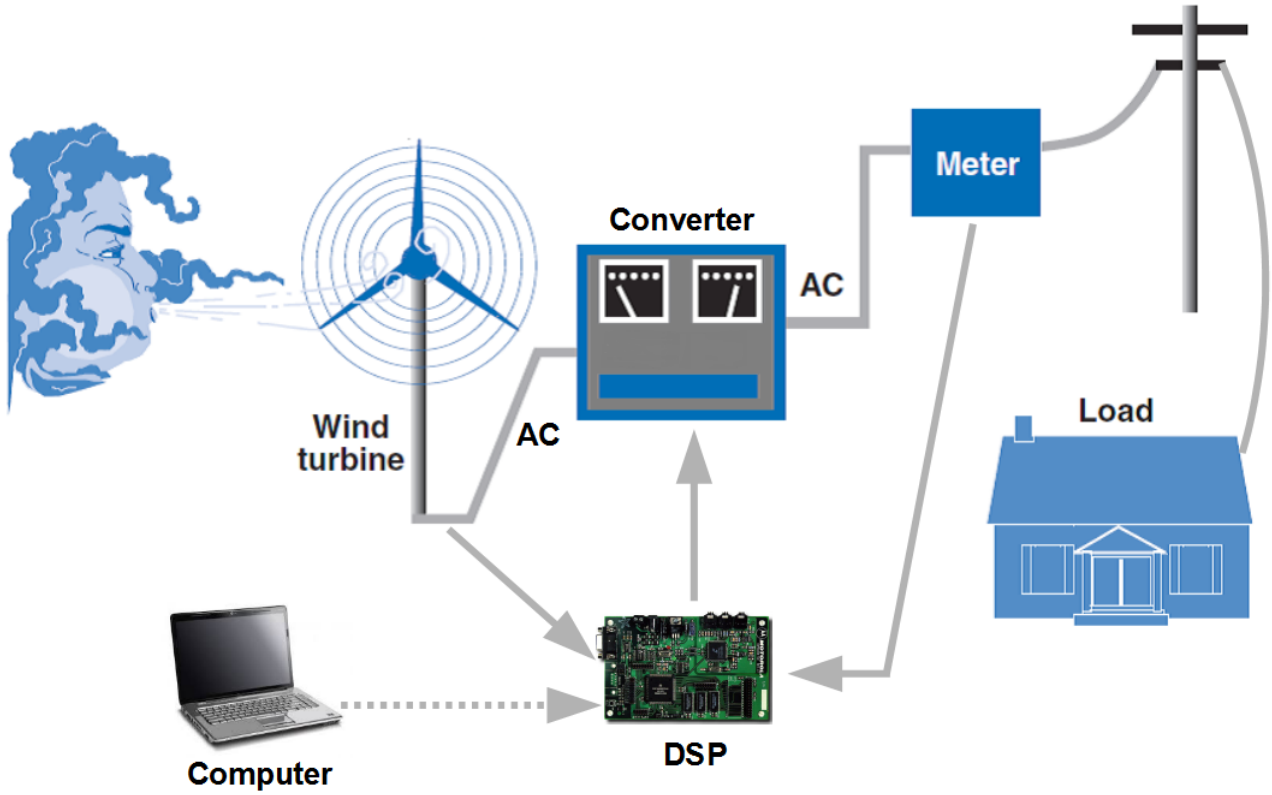


Figura 1.1 – Esquema simplificado da conexão de uma turbina eólica à rede elétrica

Devido ao aumento da penetração de fontes renováveis, principalmente eólica e fotovoltaica, na rede elétrica, muitos países tem criado e adotado normas mais rigorosas para a inserção na rede da energia gerada. Em consonância com essas normas, os modernos conversores de energia devem manter sua sincronização com a rede elétrica mesmo sob condições de falha dela, como desequilíbrios, afundamentos e elevações de tensão, variações de frequência e contaminação por harmônicas. Nesse contexto, os sistemas PLL devem possuir considerável robustez, driblando eventuais falhas para detectar corretamente o ângulo de fase das tensões da rede e da planta de geração. O objetivo deste trabalho é descrever, simular e testar um sistema PLL específico, chamado DSOGI-PLL, que detecta o ângulo de fase da tensão fundamental de uma rede mesmo em condições de falha dela.

O trabalho está dividido em quatro capítulos, sendo este o primeiro. O capítulo 2 apresenta a teoria que fundamenta o funcionamento do DSOGI-PLL. No capítulo 3, o sistema em estudo é apresentado e os módulos que o constituem são descritos. No capítulo 4, mostra-se como foi feita a discretização do DSOGI-PLL, as simulações do sistema discretizado e os resultados dos testes realizados com a rede elétrica. Finalmente, no capítulo 5, é exposta a conclusão do trabalho.

2 FUNDAMENTAÇÃO TEÓRICA

Neste capítulo, é apresentada sucintamente a teoria necessária para a compreensão do funcionamento do sistema PLL implementado. Serão explicadas a teoria das componentes simétricas, as transformações de Clarke e de Park e a dinâmica de um SRF-PLL.

2.1 COMPONENTES SIMÉTRICAS

De acordo com a teoria das componentes simétricas de Fortescue [6], um sistema trifásico desequilibrado, representado por um vetor de fasores $\mathbf{V}_{\text{abc}}$, pode ser decomposto em três sistemas trifásicos equilibrados $\mathbf{V}^+$, $\mathbf{V}^-$ e $\mathbf{V}^0$, chamados, respectivamente, sequência positiva, sequência negativa e sequência zero, de modo que

$$\mathbf{V}_{abc} = \begin{bmatrix} \hat{V}_a \\ \hat{V}_b \\ \hat{V}_c \end{bmatrix} = \mathbf{V}^+ + \mathbf{V}^- + \mathbf{V}^0 = \begin{bmatrix} \hat{V}_a^+ \\ \hat{V}_b^+ \\ \hat{V}_c^+ \end{bmatrix} + \begin{bmatrix} \hat{V}_a^- \\ \hat{V}_b^- \\ \hat{V}_c^- \end{bmatrix} + \begin{bmatrix} \hat{V}_a^0 \\ \hat{V}_b^0 \\ \hat{V}_c^0 \end{bmatrix} \tag{2.1}$$

Dito de outra forma, o método das componentes simétricas garante que qualquer uma das fases pode ser visualizada, equivalentemente, como a soma fasorial de suas componentes correspondentes em cada sequência. Por exemplo, para a fase b, tem-se que $\hat{V}_b = \hat{V}_b^+ + \hat{V}_b^- + \hat{V}_b^0$.

A Fig. 2.1 ilustra um sistema trifásico de tensões desequilibrado. A Fig. 2.2 apresenta as sequências positiva, negativa e zero do sistema da Fig. 2.1.

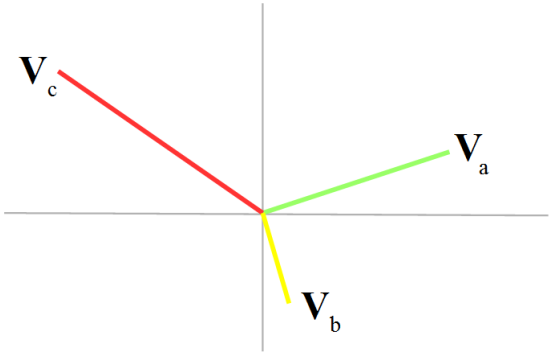
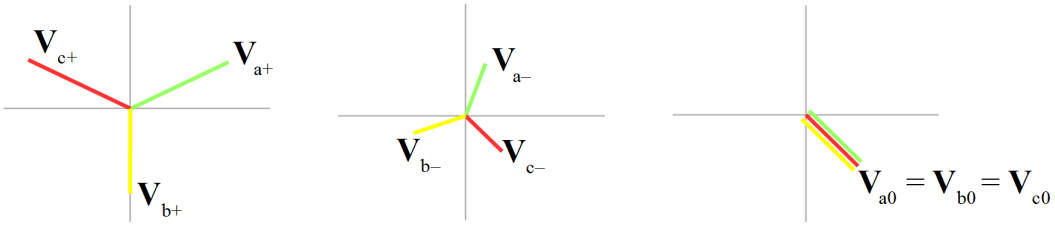


Figura 2.1 – Fasores de uma rede trifásica desequilibrada

Os fasores da sequência positiva possuem a mesma magnitude, estão defasados $\frac{2\pi}{3}$ rad



(a) Fasores da sequência positiva

(b) Fasores da sequência negativa

(c) Fasores da sequência zero

Figura 2.2 – Sequências positiva, negativa e zero do sistema da Fig. 2.1

entre si e têm a mesma sequência de fase dos fasores originais. Os fasores da sequência negativa possuem magnitudes iguais, estão defasados $\frac{2\pi}{3}$ rad entre si e estão dispostos na sequência de fase oposta à dos fasores originais. Os fasores da sequência zero são iguais entre si. Considerando o operador de Fortescue $\alpha = \mathrm{e}^{-\frac{2\pi}{3}\mathrm{j}}$, o qual defasa em $\frac{2\pi}{3}$ rad o fasor pelo qual é multiplicado, pode-se afirmar que

$$\begin{aligned} \hat{V}_b^+ &= \alpha \hat{V}_a^+ \\ \hat{V}_c^+ &= \alpha^2 \hat{V}_a^+ \end{aligned}$$

$$\begin{aligned} \hat{V}_b^- &= \alpha^2 \hat{V}_a^- \\ \hat{V}_c^- &= \alpha \hat{V}_a^+ \end{aligned} \tag{2.2}$$

$$\hat{V}_a^0 = \hat{V}_b^0 = \hat{V}_c^0$$

A partir de (2.2), tendo como referência a fase a, é possível re-escrever (2.1) como

$$\begin{bmatrix} \hat{V}_a \\ \hat{V}_b \\ \hat{V}_c \end{bmatrix} = \begin{bmatrix} \hat{V}_a^+ \\ \alpha \hat{V}_a^+ \\ \alpha^2 \hat{V}_a^+ \end{bmatrix} + \begin{bmatrix} \hat{V}_a^- \\ \alpha^2 \hat{V}_a^- \\ \alpha \hat{V}_a^- \end{bmatrix} + \begin{bmatrix} \hat{V}_a^0 \\ \hat{V}_a^0 \\ \hat{V}_a^0 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 \\ \alpha & \alpha^2 & 1 \\ \alpha^2 & \alpha & 1 \end{bmatrix} \begin{bmatrix} \hat{V}_a^+ \\ \hat{V}_a^- \\ \hat{V}_a^0 \end{bmatrix} = \mathbf{T}\,\mathbf{V}_a^{+-0}, \tag{2.3}$$

em que $\mathbf{T}$ é a matriz de transformação de coordenadas. Sabendo que o determinante de $\mathbf{T}$ é $3\alpha(\alpha-1)$, $\alpha^{-1} = \alpha^2$, $\alpha^3 = 1$ e $\alpha^4 = \alpha$, conclui-se que $\mathbf{T}$ admite inversa $\mathbf{T}^{-1}$ dada por

$$
\begin{aligned}
\mathbf{T}^{-1} &= \frac{1}{\det \mathbf{T}} \operatorname{adj} \mathbf{T} \\
&= \frac{1}{3\alpha(\alpha-1)} \begin{bmatrix} \alpha(\alpha-1) & \alpha-1 & 1-\alpha^2 \\ \alpha(\alpha-1) & 1-\alpha^2 & \alpha-1 \\ \alpha(\alpha-1) & \alpha(\alpha-1) & \alpha(\alpha-1) \end{bmatrix} \\
&= \begin{bmatrix} \frac{1}{3} & \frac{1}{3\alpha} & \frac{\alpha}{3} \\ \frac{1}{3} & \frac{\alpha}{3} & \frac{1}{3\alpha} \\ \frac{1}{3} & \frac{1}{3} & \frac{1}{3} \end{bmatrix} \\
&= \frac{1}{3} \begin{bmatrix} 1 & \alpha^2 & \alpha \\ 1 & \alpha & \alpha^2 \\ 1 & 1 & 1 \end{bmatrix}
\end{aligned}
\tag{2.4}
$$

Com a matriz $\mathbf{T}^{-1}$, é possível obter $\hat{V}^+$, $\hat{V}^-$ e $\hat{V}^0$ de qualquer das fases a partir de $\hat{V}_a$, $\hat{V}_b$ e $\hat{V}_c$, como exemplificado em (2.5) para a fase a.

$$
\begin{bmatrix} \hat{V}_a^+ \\ \hat{V}_a^- \\ \hat{V}_a^0 \end{bmatrix} = \frac{1}{3} \begin{bmatrix} 1 & \alpha^2 & \alpha \\ 1 & \alpha & \alpha^2 \\ 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} \hat{V}_a \\ \hat{V}_b \\ \hat{V}_c \end{bmatrix} \tag{2.5}
$$

A partir de (2.2) e (2.5), é possível determinar as componentes simétricas das sequências positiva e negativa em função das componentes do sistema original do seguinte modo:

$$
\begin{bmatrix} \hat{V}_a^+ \\ \hat{V}_b^+ \\ \hat{V}_c^+ \end{bmatrix} = \frac{1}{3} \begin{bmatrix} 1 & \alpha^2 & \alpha \\ \alpha & 1 & \alpha^2 \\ \alpha^2 & \alpha & 1 \end{bmatrix} \begin{bmatrix} \hat{V}_a \\ \hat{V}_b \\ \hat{V}_c \end{bmatrix} \tag{2.6}
$$

$$
\begin{bmatrix} \hat{V}_a^- \\ \hat{V}_b^- \\ \hat{V}_c^- \end{bmatrix} = \frac{1}{3} \begin{bmatrix} 1 & \alpha & \alpha^2 \\ \alpha^2 & 1 & \alpha \\ \alpha & \alpha^2 & 1 \end{bmatrix} \begin{bmatrix} \hat{V}_a \\ \hat{V}_b \\ \hat{V}_c \end{bmatrix} \tag{2.7}
$$

O método das componentes simétricas de Fortescue era restrito ao domínio da frequência, porém, em 1937, Lyon estendeu o método para o domínio do tempo [7]. No domínio do tempo, os fasores $\hat{\mathrm{V}}$, representativos das três tensões da rede, são substituídos por variáveis temporais senoidais $v(t)$ e o operador α de Fortescue é trocado pelo seu equivalente no domínio do tempo,

o operador a. Assim, (2.6) e (2.7) tornam-se:

$$\mathbf{v}^{+}_{abc} = \begin{bmatrix} v^{+}_{a} \\ v^{+}_{b} \\ v^{+}_{c} \end{bmatrix} = \mathbf{T}_{+}\,\mathbf{v}_{abc} = \frac{1}{3}\begin{bmatrix} 1 & a^2 & a \\ a & 1 & a^2 \\ a^2 & a & 1 \end{bmatrix}\begin{bmatrix} v_a \\ v_b \\ v_c \end{bmatrix} \tag{2.8}$$

$$\mathbf{v}^{-}_{abc} = \begin{bmatrix} v^{-}_{a} \\ v^{-}_{b} \\ v^{-}_{c} \end{bmatrix} = \mathbf{T}_{-}\,\mathbf{v}_{abc} = \frac{1}{3}\begin{bmatrix} 1 & a & a^2 \\ a^2 & 1 & a \\ a & a^2 & 1 \end{bmatrix}\begin{bmatrix} v_a \\ v_b \\ v_c \end{bmatrix} \tag{2.9}$$

2.2 TRANSFORMAÇÃO DE CLARKE

Num sistema trifásico equilibrado, a transformação de Clarke [8] (também conhecida como transformação $\alpha\beta$) leva valores de grandezas do referencial natural *abc* para o referencial estacionário $\alpha\beta$.

A Fig. 2.3 mostra parte das senoides de um sistema trifásico equilibrado e um segmento de reta tracejado destacando um instante de tempo genérico. A Fig. 2.4 mostra o referencial natural *abc* cuja origem coincide com a origem do referencial $\alpha\beta$. No instante de tempo genérico indicado pela reta tracejada da Fig. 2.3, os valores das fases são a magnitude dos vetores verdes da Fig. 2.4. Sendo assim, pode-se pensar a transformação de Clarke como a projeção das componentes *abc* sobre os eixos $\alpha\beta$.

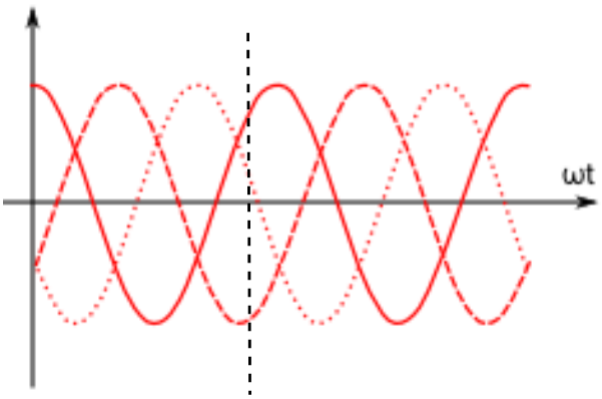


Figura 2.3 – Rede trifásica equilibrada com destaque para um instante de tempo

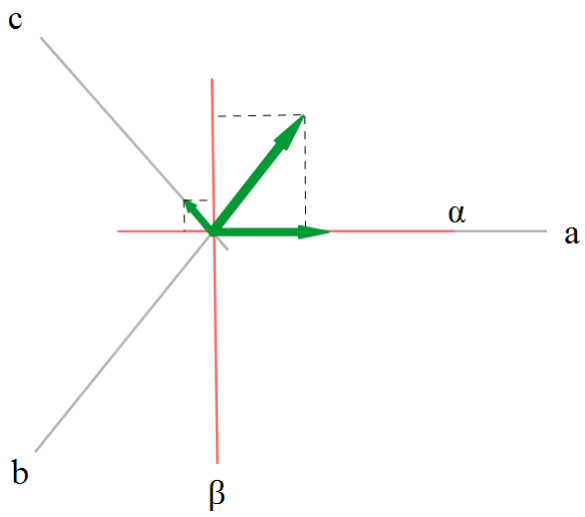


Figura 2.4 – Mudança de coordenadas pela transformação de Clarke

Da Fig. 2.4, é possível deduzir equações que fornecem v_α e v_β em função de v_a, v_b e v_c:

$$\begin{aligned} v_\alpha &= v_\mathrm{a} - v_\mathrm{b}\cos\frac{\pi}{3} - v_\mathrm{c}\cos\frac{\pi}{3} \\ &= v_\mathrm{a} - \frac{1}{2}v_\mathrm{b} - \frac{1}{2}v_\mathrm{c} \end{aligned} \tag{2.10a}$$

$$\begin{aligned} v_\beta &= -v_\mathrm{b}\cos\frac{\pi}{6} + v_\mathrm{c}\cos\frac{\pi}{6} \\ &= -\frac{\sqrt{3}}{2}v_\mathrm{b} + \frac{\sqrt{3}}{2}v_\mathrm{c} \end{aligned} \tag{2.10b}$$

A (2.10) pode ser condensada na seguinte equação matricial

$$\mathbf{v}_{\alpha\beta} = \begin{bmatrix} v_\alpha \\ v_\beta \end{bmatrix} = \mathbf{T}\,\mathbf{v}_{abc} = \begin{bmatrix} 1 & -\frac{1}{2} & -\frac{1}{2} \\ 0 & \frac{\sqrt{3}}{2} & -\frac{\sqrt{3}}{2} \end{bmatrix} \begin{bmatrix} v_a \\ v_b \\ v_c \end{bmatrix} \tag{2.11}$$

Sendo $\mathbf{v}_{\mathrm{abc}}$ um sistema trifásico equilibrado e v_a, v_b e v_c tensões senoidais no tempo, pode-se escrever que

$$
\begin{aligned}
v_{\mathrm{a}} &= V \operatorname{sen} \omega t \\
v_{\mathrm{b}} &= V \operatorname{sen} \left(\omega t - \frac{2\pi}{3} \right) \\
v_{\mathrm{c}} &= V \operatorname{sen} \left(\omega t + \frac{2\pi}{3} \right)
\end{aligned}
\tag{2.12}
$$

Substituindo (2.12) em (2.10), obtêm-se

$$
\begin{aligned}
v_\alpha &= V \operatorname{sen}(\omega t) - \frac{V}{2} \operatorname{sen} \left(\omega t - \frac{2\pi}{3} \right) - \frac{V}{2} \operatorname{sen} \left(\omega t + \frac{2\pi}{3} \right) \\
&= \frac{3}{2} V \operatorname{sen} \omega t \\
v_\beta &= -\frac{\sqrt{3}}{2} V \operatorname{sen} \left(\omega t - \frac{2\pi}{3} \right) + \frac{\sqrt{3}}{2} V \operatorname{sen} \left(\omega t + \frac{2\pi}{3} \right) \\
&= -\frac{3}{2} V \cos \omega t
\end{aligned}
\tag{2.13}
$$

De acordo com (2.13), a rede trifásica original $\mathbf{v}_{abc}$ pode ser representada por um sitema bifásico $\mathbf{v}_{\alpha\beta}$ de senoides defasadas em $\frac{\pi}{2}$ rad e amplitude 1,5 vezes a amplitude das senoides originais. Para que as amplitudes de v_α e v_β sejam iguais às de v_a, v_b e v_c, a matriz $\mathbf{T}$ é multiplicada por $\frac{2}{3}$, gerando a matriz $\mathbf{T}_{\alpha\beta}$, chamada transformação de Clarke invariante em amplitude, definida por 2.14.

$$
\mathbf{v}_{\alpha\beta} = \begin{bmatrix} v_\alpha \\ v_\beta \end{bmatrix} = \mathbf{T}_{\alpha\beta}\, \mathbf{v}_{abc} = \frac{2}{3} \begin{bmatrix} 1 & -\frac{1}{2} & -\frac{1}{2} \\ 0 & \frac{\sqrt{3}}{2} & -\frac{\sqrt{3}}{2} \end{bmatrix} \begin{bmatrix} v_a \\ v_b \\ v_c \end{bmatrix}
\tag{2.14}
$$

A (2.14) define, na verdade, um caso particular da transformação de Clarke. No caso geral, a matriz da transformação de Clarke é definida como em (2.15), possuindo uma terceira linha referente à componente γ.

$$
\mathbf{T}_{\alpha\beta\gamma} = \frac{2}{3} \begin{bmatrix} 1 & -\frac{1}{2} & -\frac{1}{2} \\ 0 & \frac{\sqrt{3}}{2} & -\frac{\sqrt{3}}{2} \\ \frac{1}{2} & \frac{1}{2} & \frac{1}{2} \end{bmatrix}
\tag{2.15}
$$

Entretanto, em sistemas trifásicos equilibrados, isto é, sistemas em que, por exemplo, $v_a + v_b + v_c = 0$, a componente γ da variável de interesse é nula, $v_\gamma = 0$, o que justifica a exclusão da terceira linha da matriz $\mathbf{T}_{\alpha\beta\gamma}$ nesses casos. A matriz em (2.15) tem determinante não-nulo, logo é inversível e sua inversa $\mathbf{T}^{-1}_{\alpha\beta\gamma}$ é dada por

$$\begin{aligned}\mathbf{T}^{-1}_{\alpha\beta\gamma} &= \frac{1}{\det(\mathbf{T}_{\alpha\beta\gamma})}\text{adj}(\mathbf{T}_{\alpha\beta\gamma}) \\ &= \frac{9}{2\sqrt{3}}\begin{bmatrix} \frac{2\sqrt{3}}{9} & 0 & \frac{2\sqrt{3}}{9} \\ -\frac{\sqrt{3}}{9} & \frac{3}{9} & \frac{2\sqrt{3}}{9} \\ -\frac{\sqrt{3}}{9} & -\frac{3}{9} & \frac{2\sqrt{3}}{9} \end{bmatrix} \\ &= \begin{bmatrix} 1 & 0 & 1 \\ -\frac{1}{2} & \frac{\sqrt{3}}{2} & 1 \\ -\frac{1}{2} & -\frac{\sqrt{3}}{2} & 1 \end{bmatrix}\end{aligned} \tag{2.16}$$

2.3 TRANSFORMAÇÃO DE PARK

A transformação de Park leva tensões do referencial estacionário $\alpha\beta$ para o referencial rotativo dq [9]. A Fig. 2.5 ilustra um sistema de tuas tensões resultantes da transformação de Clarke. Nessa figura, um segmento de reta tracejado marca um instante de tempo genérico em que as tensões v_α e v_β assumem valores proporcionais à magnitude dos vetores da Fig. 2.6.

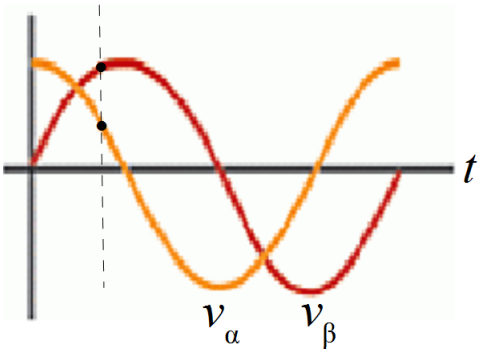


Figura 2.5 – Tensões resultantes da transformação de Clarke com destaque para instante de tempo genérico

Pela Fig. 2.6, é possível deduzir as equações que relacionam as componentes direta v_d e em quadratura v_q com as componentes v_α e v_β:

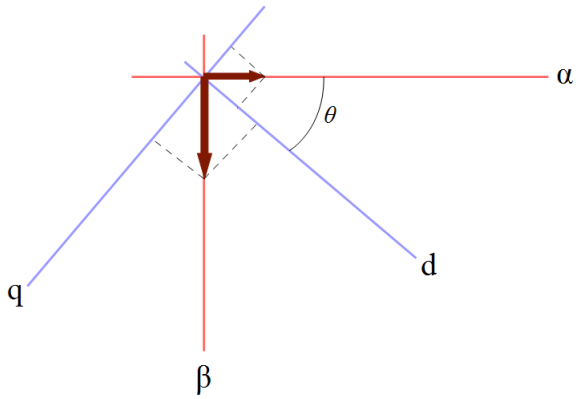


Figura 2.6 – Mudança de coordenadas pela transformação de Park

$$\begin{aligned} v_d &= v_\alpha \cos\theta + v_\beta \operatorname{sen}\left(\frac{\pi}{2} - \theta\right) \\ &= v_\alpha \cos\theta + v_\beta \operatorname{sen}\theta \end{aligned} \tag{2.17}$$

$$\begin{aligned} v_q &= -v_\alpha \cos\left(\frac{\pi}{2} - \theta\right) + v_\beta \cos\theta \\ &= -v_\alpha \operatorname{sen}\theta + v_\beta \cos\theta \end{aligned}$$

Em (2.17), θ é o ângulo mostrado na Fig. 2.6, isto é, o ângulo compreendido entre o semi-eixo positivo de α e o semi-eixo positivo de d.

A (2.17) pode ser re-escrita na forma matricial da seguinte maneira

$$\mathbf{v}_{\mathrm{dq}} = \begin{bmatrix} v_{\mathrm{d}} \\ v_{\mathrm{q}} \end{bmatrix} = \mathbf{T}_{\mathrm{dq}} \mathbf{v}_{\alpha\beta} = \begin{bmatrix} \cos\theta & \operatorname{sen}\theta \\ -\operatorname{sen}\theta & \cos\theta \end{bmatrix} \begin{bmatrix} v_\alpha \\ v_\beta \end{bmatrix} \tag{2.18}$$

Para um sistema trifásico representado por $v_\alpha = V \operatorname{sen}\omega t$ e $v_\beta = -V\cos\omega t$, verifica-se que

$$
\begin{aligned}
v_d &= v_\alpha \cos\theta + v_\beta \operatorname{sen}\theta \\
&= V \operatorname{sen}\omega t \cos\theta - V \cos\omega t \operatorname{sen}\theta \\
&= V \operatorname{sen}(\omega t - \theta)
\end{aligned}
\tag{2.19}
$$

$$
\begin{aligned}
v_q &= -v_\alpha \operatorname{sen}\theta + v_\beta \cos\theta \\
&= -V \operatorname{sen}\omega t \operatorname{sen}\theta - V \cos\omega t \cos\theta \\
&= -V \cos(\omega t - \theta)
\end{aligned}
$$

De acordo com (2.19), caso $\omega t = \theta$, $v_\mathrm{d} = 0$ e $v_\mathrm{q} = -V$, isto é, quando a frequência da rede e a velocidade angular do referencial *dq* se igualam, as tensões resultantes da transformação de Park tornam-se constantes no tempo.

2.4 SRF-PLL

Um aspecto importante na sincronização de conversores com a rede é a correta detecção do ângulo de fase. Há, pelo menos, dois tipos de métodos para estimar esse ângulo: métodos de malha aberta e de malha fechada. Os métodos de malha fechada são, genericamente, conhecidos como PLL (*Phase-Locked Loop*) [10]. Para redes trifásicas, o PLL mais usado é o SRF-PLL, cujo diagrama é mostrado na Fig. 2.7.

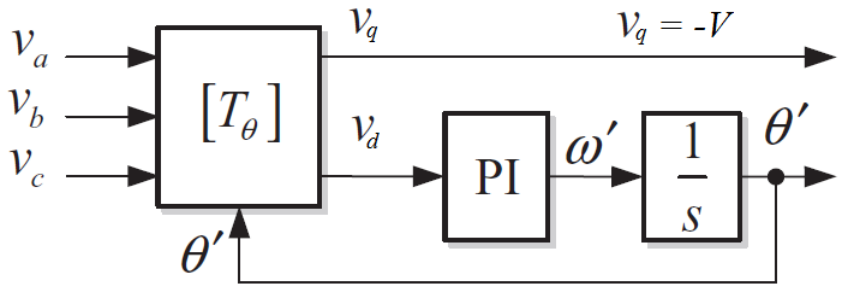


Figura 2.7 – Diagrama de blocos do SRF-PLL [3]

O primeiro estágio do SRF-PLL, denotado pelo bloco $[T_\theta]$ na figura, efetua as transformações de Clarke e de Park, nessa ordem, sobre as amostras de tensão v_a, v_b e v_c provenientes da rede, transferindo-as do referencial *abc* para o referencial *dq*. Conforme mostrado na Seção 2.3, quando a fase estimada pelo PLL, θ', iguala-se à fase da rede ωt, a tensão v_d torna-se nula

e a tensão v_q iguala-se a $-V$, o oposto da amplitude da senoide v_a. O bloco denotado por PI representa um controlador proporcional-integral, responsável por regular para zero o valor de v_d, garantindo que o ângulo de fase estimado e o ângulo de fase da rede tornem-se iguais em regime permanente (vide (2.19)). O bloco denotado por $\frac{1}{\mathrm{s}}$ é um integrador que, integrando a frequência estimada ω' em sua entrada, produz o ângulo de fase estimado θ', usado como argumento pela transformação de Park do bloco $[T_\theta]$.

Para atender atuais normas de inserção de energia renovável na rede, os sistemas PLL modernos devem possuir certo grau de imunidade a desequilíbrios e distorções, fazendo com que eventuais fenômenos que afetam a qualidade da energia não intefiram na detecção correta do ângulo de fase das tensões da rede e da planta de geração. O DSOGI-PLL, objeto de estudo deste trabalho, é um tipo de PLL que resulta da adaptação do SRF-PLL a esse novo contexto.

3 DSOGI-PLL

Neste capítulo, será apresentado o sistema de detecção de sequência positiva a ser implementado. Serão analisados os subsistemas que o compõe e como eles advêm da teoria desenvolvida no Cap. 2.

3.1 OBTENÇÃO DA SEQUÊNCIA POSITIVA NO REFERENCIAL $\alpha\beta$

O DSOGI-PLL realiza a obtenção do ângulo de fase da tensão da fase *a* de uma rede trifásica por meio da detecção das componentes simétricas da sequência positiva das tensões da rede em questão.

De acordo com (2.8), a sequência positiva $\mathbf{v}^+_{abc}$ de uma rede trifásica desequilibrada $\mathbf{v}_{abc}$ pode ser obtida a partir das tensões que compõem o sistema original por meio da transformação $\mathbf{T}_+$. E, conforme (2.14), pode-se passar as coordenadas de um sistema trifásico do referencial natural *abc* para o referencial $\alpha\beta$ por meio da transformação de Clarke $\mathbf{T}_{\alpha\beta}$. Combinando as transformaçãos $\mathbf{T}_+$ e $\mathbf{T}_{\alpha\beta}$ numa só, é possível derivar uma nova transformação que permite a obtenção da sequência positiva no referencial $\alpha\beta$, $\mathbf{v}^+_{\alpha\beta}$, a partir do sistema desequilibrado original $\mathbf{v}_{abc}$:

$$\begin{aligned}\mathbf{v}^+_{\alpha\beta} &= \mathbf{T}_{\alpha\beta}\,\mathbf{v}^+_{abc}\\ &= \mathbf{T}_{\alpha\beta}\,\mathbf{T}_+\mathbf{v}_{abc}\\ &= \mathbf{T}_{\alpha\beta}\,\mathbf{T}_+\,\mathbf{T}^{-1}_{\alpha\beta}\,\mathbf{v}_{\alpha\beta}\\ &= \mathbf{Q}\,\mathbf{v}_{\alpha\beta}\end{aligned} \tag{3.1}$$

A matriz de transformação $\mathbf{Q}$ permite obter as componentes v^+_α e v^+_β a partir das componentes v_α e v_β, que representam o sistema original. A fim de obter uma expressão para a matriz $\mathbf{Q}$, deve-se lembrar que $a = \mathrm{e}^{-\mathrm{j}\frac{2\pi}{3}}$ e, portanto, $a^2 + a = -1$ e $a^2 - a = -\sqrt{3}\mathrm{j}$. Com isso, pode-se escrever que

$$
\begin{aligned}
\mathbf{Q} &= \mathbf{T}_{\alpha\beta}\,\mathbf{T}_{+}\,\mathbf{T}_{\alpha\beta}^{-1} \\
&= \frac{2}{3}\begin{bmatrix} 1 & -\frac{1}{2} & -\frac{1}{2} \\ 0 & \frac{\sqrt{3}}{2} & -\frac{\sqrt{3}}{2} \end{bmatrix} \frac{1}{3}\begin{bmatrix} 1 & a^2 & a \\ a & 1 & a^2 \\ a^2 & a & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ -\frac{1}{2} & \frac{\sqrt{3}}{2} \\ -\frac{1}{2} & -\frac{\sqrt{3}}{2} \end{bmatrix} \\
&= \begin{bmatrix} \frac{1}{2} & \frac{\mathrm{j}}{2} \\ \frac{-\mathrm{j}}{2} & \frac{1}{2} \end{bmatrix} = \frac{1}{2}\begin{bmatrix} 1 & -q \\ q & 1 \end{bmatrix}
\end{aligned}
\tag{3.2}
$$

Em (3.2), $q = \mathrm{e}^{-\mathrm{j}\frac{\pi}{2}}$, um operador que defasa em $\frac{\pi}{2}$ rad o sinal pelo qual é multiplicado. Substituindo (3.2) em (3.1) e separando as componentes do vetor de tensões, são obtidas

$$
\begin{aligned}
v_{\alpha}^{+} &= \frac{1}{2}(v_{\alpha} - qv_{\beta}) \\
v_{\beta}^{+} &= \frac{1}{2}(qv_{\beta} + v_{\alpha})
\end{aligned}
\tag{3.3}
$$

A partir de (3.3), conclui-se que, para o cálculo da sequência positiva, são necessárias as tensões do sistema original e suas correspondentes em quadratura.

3.2 GERADOR DE SINAIS EM QUADRATURA

Para gerar os sinais em quadratura, necessários para o cálculo da sequência positiva, a referência [11] propõe o uso de um SOGI (*Second Order Generalized Integrator*), mostrado na Fig. 3.1.

Da Fig. 3.1, constata-se que o SOGI possui uma entrada, o sinal *v*, e duas saídas, o sinal *v*', resultante da filtragem do sinal *v*, e o sinal *qv*', correspondente ao sinal filtrado e em quadratura com *v*'. Também é possível, por meio da citada figura, deduzir a função de transferência *D(s)*, que relaciona a entrada *v* com a saída *v*', e a função de transferência *Q(s)*, que relaciona a entrada *v* com a saída *qv*':

$$
D(s) = \frac{v'}{v}(s) = \frac{k\omega' s}{s^2 + k\omega' s + \omega'^2} \tag{3.4a}
$$

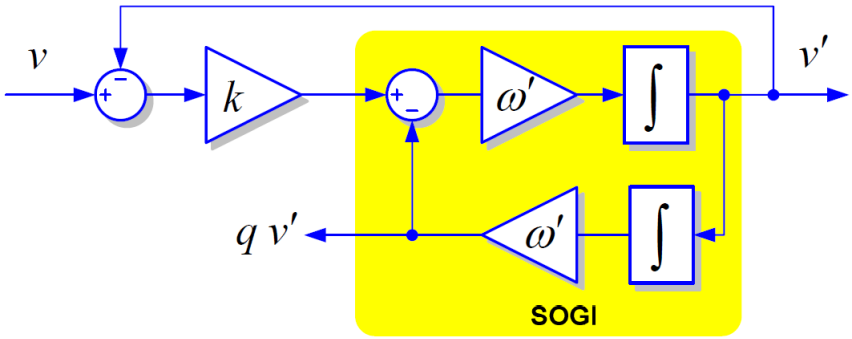


Figura 3.1 – Integrador generalizado de segunda ordem (SOGI)

$$Q(s) = \frac{qv'}{v}(s) = \frac{k\omega'^2}{s^2 + k\omega' s + \omega'^2} \tag{3.4b}$$

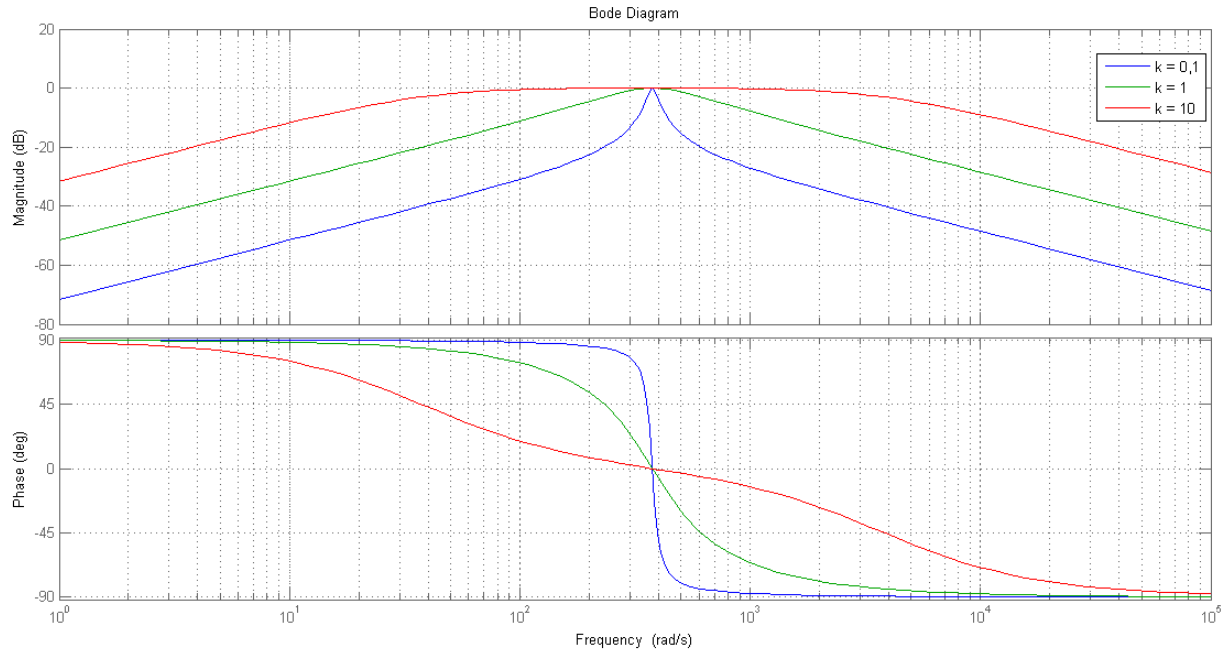
Em (3.4), ω' é a frequência estimada da rede, em rad/s, e k é um ganho adimensional que ajusta a banda passante do SOGI.

O comportamento do sistema pode ser observado na Fig. 3.2, que mostra os diagramas de Bode de D(s) e Q(s), revelando que D(s) comporta-se como um filtro passa-faixa e Q(s), como um passa-baixas, sendo a frequência de corte de ambos a frequência estimada da rede ω', que, para a construção dos diagramas, foi assumida como 120π rad/s.

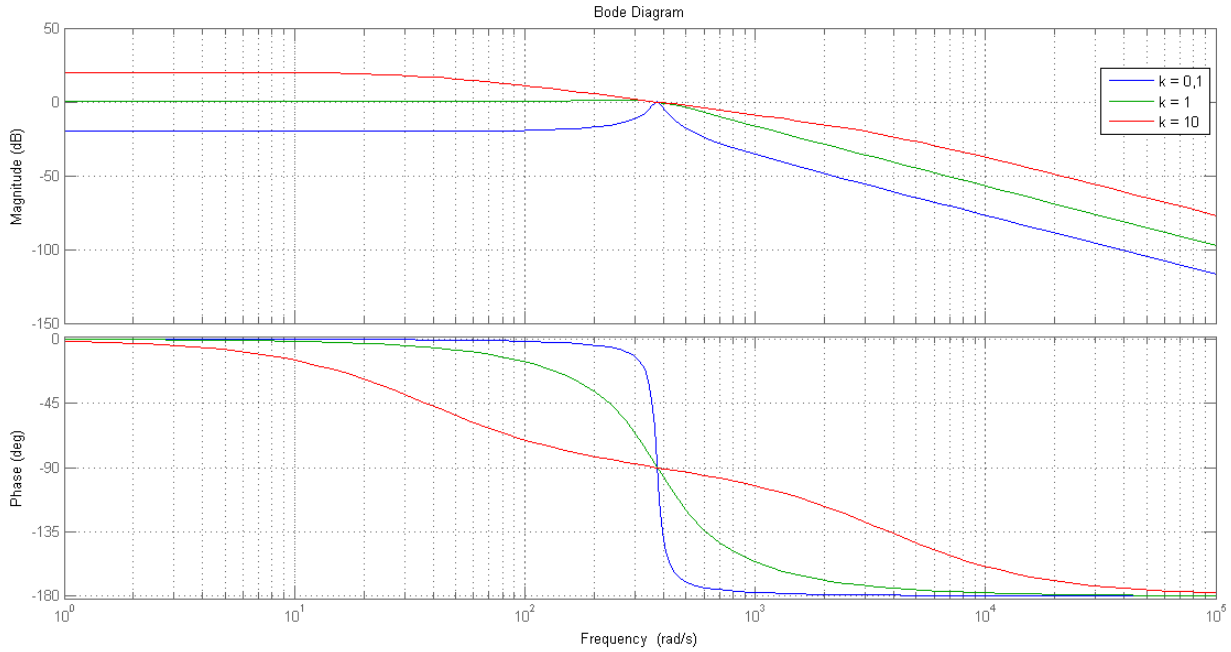
Da Fig. 3.2, também se vê que quanto menor o valor de do ganho k menor a banda passante dos filtros, isto é, um menor valor de k implica uma maior capacidade do sistema de atenuar harmônicas de ordem mais alta. É importante salientar que, via de regra, uma melhora na filtragem de harmônicas repercute numa piora de aspectos de regime transitório, como elevação do tempo de assentamento e do sobressinal.

Para provar que o SOGI produz os sinais com as características desejadas, é preciso verificar o módulo e a fase de D(s) e Q(s). Para tanto, faz-se $s = j\omega$ e

$$\begin{aligned} |D(j\omega)| &= \frac{|k\omega'\omega j|}{|(\omega j)^2 + k\omega'\omega j + \omega'^2|} = \frac{k\omega'\omega}{\sqrt{(\omega'^2 - \omega^2)^2 + (k\omega'\omega)^2}} \\ \angle D(j\omega) &= \frac{\pi}{2} - \tan^{-1}\left(\frac{k\omega'\omega}{\omega'^2 - \omega^2}\right) \end{aligned} \tag{3.5}$$



(a) Diagramas de Bode de D(s)



(b) Diagramas de Bode de Q(s)

Figura 3.2 – Diagramas de Bode das funções D(s) e Q(s)

$$
\begin{aligned}
|Q(j\omega)| &= \frac{|k\omega'^2|}{|(\omega j)^2 + k\omega'\omega j + \omega'^2|} = \frac{k\omega'^2}{\sqrt{(\omega'^2-\omega^2)^2 + (k\omega'\omega)^2}} = \frac{\omega'}{\omega}|D(j\omega)| \\
\underline{/Q(j\omega)} &= -\tan^{-1}\left(\frac{k\omega'\omega}{\omega'^2-\omega^2}\right) = \underline{/Dj\omega} - \frac{\pi}{2}
\end{aligned}
\tag{3.6}
$$

(3.5) e (3.6) demonstram que, independentemente dos valores de ω' e k, qv' estará defasado $\frac{\pi}{2}$ em relação a v', no entanto, somente quando $\omega' = \omega$, isto é, apenas quando a frequência de corte e a frequência da rede forem iguais, as amplitudes dos sinais v' e qv' serão iguais.

Como devem ser gerados os sinais em quadratura de dois sinais de tensão, v_α e v_β, são precisos dois SOGIs ou um DSOGI (*Dual* SOGI) e é crucial que ambos gerem sinais de saída cuja amplitude seja igual à dos sinais de entrada e a fase seja deslocada precisamente em $\frac{\pi}{2}$ rad.

3.3 CÁLCULO DA SEQUÊNCIA POSITIVA

O cálculo das componentes da sequência positiva de tensões é feito implementando-se (3.3). Para isso, são utilizados dois somadores que, somando convenientemente as saídas do DSOGI, efetuam a obtenção da sequência positiva de tensões no referencial $\alpha\beta$, como mostra a Fig. 3.3.

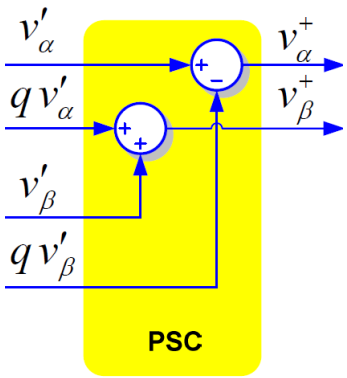


Figura 3.3 – Diagrama de blocos do módulo PSC

Os módulos DSOGI e PSC compõem o móduo DSOGI-PSC, mostrado na Fig. 3.4. Para voltar ao referencial *abc*, efetua-se a transformação de Clarke inversa sobre o vetor de tensões $\mathbf{v}^+_{\alpha\beta}$, obtendo o vetor de tensões $\mathbf{v}^+_{\text{abc}}$ (Fig. 3.4), que é o conjunto trifásico de tensões em sequência positiva correspondentes ao vetor de tensões original $\mathbf{v}_{\text{abc}}$.

3.4 DSOGI-PLL

Para descrever o funcionamento do subsistema DSOGI-PSC nas seções anteriores, a frequência da rede foi suposta constante e igual à frequência de corte dos filtros. Todavia, em aplicações realistas, deve-se considerar que a frequência da rede pode sofrer variações, o que geraria erros

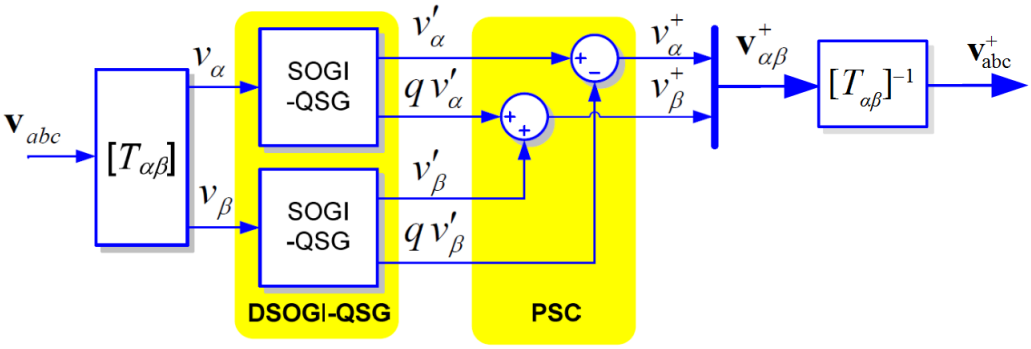


Figura 3.4 – Diagrama de blocos do módulo DSOGI-PSC

na obtenção da sequência positiva, como já demonstrado. Para garantir que o DSOGI-PSC seja adaptável à frequência, ou seja, para que ele sempre utilize a frequência nominal da rede, ignorando eventuais variações, é utilizado um SRF-PLL, cujo funcionamento foi explanado no Cap. 2. O SRF-PLL é o último estágio do sistema detector de sequência positiva estudado. As entrada dos SRF-PLL são as saídas do DSOGI-PSC. A Fig. 3.5 ilustra o sistema completo, com destaque para três dos módulos básicos que o compõe.

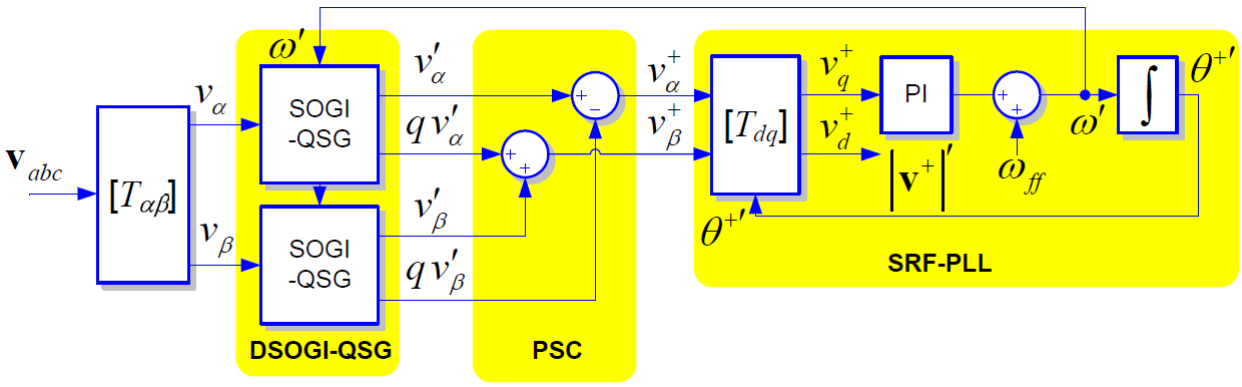


Figura 3.5 – Diagrama de blocos do DSOGI-PLL

Para ilustrar o funcionamento do sistema, suponha, por exemplo, que o DSOGI-PLL recebe como entrada um vetor de tensões desequilibradas e distorcidas $\mathbf{v}_{\text{abc}}$. Inicialmente, ocorre a passagem dessas tensões para o referencial $\alpha\beta$, bloco $[T_{\alpha\beta}]$ na Fig. 3.5. Em seguida, cada uma das duas tensões resultantes, v_α e v_β, passa por um bloco SOGI-QSG, no qual são filtradas, gerando v'_α e v'_β, e seu sinal em quadratura filtrado correspondente é gerado, qv'_α e qv'_β. As quatro tensões do módulo anterior são, então, somadas de uma forma específica, $v'_\alpha - qv'_\beta$ e $v'_\beta + qv'_\alpha$, e os dois resultados dessas somas formam a sequência positiva no referencial $\alpha\beta$, v^+_α e v^+_β. Para que eventuais variações de frequência da rede não gerem erros na detecção, um SRF-PLL é usado para detectar a frequência da rede e realimentar os blocos SOGI-QSG com o valor correto de

frequência.

4 IMPLEMENTAÇÃO E TESTES DO DSOGI-PLL

Para avaliar o desempenho do sistema estudado, é preciso inicialmente simulá-lo e, posteriormente, implementá-lo numa plataforma apropriada, realizar testes e verificar se os resultados produzidos concordam com o que teoricamente se espera. O Arduino DUE foi escolhido para testar o sistema em apreço. Para implementar o DSOGI-PLL, é necessário formalizá-lo numa linguagem de programação reconhecida pelo DUE e, para isso, é preciso antes obter seu modelo matemático em tempo discreto.

4.1 DISCRETIZAÇÃO DO SISTEMA

O modelo matemático em tempo contínuo do SOGI são as funções de transferência expressas em (4.1), em que K é o ganho do SOGI e ω', a frequência da rede, em rad/s, estimada pelo SRF-PLL.

$$\begin{aligned} \frac{v'(s)}{v(s)} &= \frac{K\omega' s}{s^2 + K\omega' s + \omega'^2} \\ \frac{qv'(s)}{v(s)} &= \frac{K\omega'^2}{s^2 + K\omega' s + \omega'^2} \end{aligned} \tag{4.1}$$

Para discretizar o SOGI, foi usada a transformação bilinear (também conhecida como método de Tustin ou transformação trapezoidal), que consiste em substituir, na função de transferência em tempo contínuo, a variável s pela expressão $\frac{2}{T}\frac{z-1}{z+1}$, em que T é o período de amostragem, em segundos. Aplicando-se a transformação bilinear em (4.1) e rearrajando os termos resultantes, é obtida (4.2), que constitue o modelo matemático dos SOGIs em tempo discreto.

$$\frac{v'(z)}{v(z)} = \frac{\frac{2K\omega'}{T}(z^2-1)}{(\frac{2K\omega'}{T}+\frac{4}{T^2}+\omega'^2)z^2+(2\omega'^2-\frac{8}{T^2})z+\frac{4}{T^2}+\omega'^2-\frac{2K\omega'}{T}}$$

$$\frac{qv'(z)}{v(z)} = \frac{K\omega'^2(z^2+2z+1)}{(\frac{2K\omega'}{T}+\frac{4}{T^2}+\omega'^2)z^2+(2\omega'^2-\frac{8}{T^2})z+\frac{4}{T^2}+\omega'^2-\frac{2K\omega'}{T}} \tag{4.2}$$

Aplicando a transformada z inversa sobre (4.2) e isolando os termos v'(k+2) e qv'(k+2) resultantes, é obtida (4.3), que constitue um método de cálculo de amostras de tensão e suas correspondentes em quadratura.

$$v'(k+2) = \frac{(\frac{8}{T^2}-2\omega'^2)v'(k+1)+(\frac{2K\omega'}{T}-\frac{4}{T^2}-\omega'^2)v'(k)+\frac{2K\omega'}{T}[v(k+2)-v(k)]}{\frac{2K\omega'}{T}+\frac{4}{T^2}+\omega'^2} \tag{4.3a}$$

$$qv'(k+2) = \frac{(\frac{8}{T^2}-2\omega'^2)qv'(k+1)+(\frac{2K\omega'}{T}-\frac{4}{T^2}-\omega'^2)qv'(k)+K\omega'^2[v(k+2)+2v(k+1)+v(k)]}{\frac{2K\omega'}{T}+\frac{4}{T^2}+\omega'^2} \tag{4.3b}$$

Como mostrado no Cap. 3, para obter a estimativa da sequência positiva no referencial $\alpha\beta$, as amostras estimadas de tensão v' e qv' devem ser somadas conforme (4.4). Caso se queira a sequência positiva no referencial natural *abc* em lugar do referencial $\alpha\beta$, pode-se aplicar a transformação de Clarke inversa no vetor cujas componentes $\alpha\beta$ resultam de (4.4).

$$v^+_\alpha(k) = v'_\alpha(k) - qv'_\beta(k) \tag{4.4a}$$

$$v^+_\beta(k) = qv'_\alpha(k) + v'_\beta(k) \tag{4.4b}$$

A discretização do SRF-PLL resume-se na discretização de seu controlador PI e de seu integrador. Sendo assim, considere a função de transferência em tempo contínuo do controlador PI do SRF-PLL, mostrada na (4.5), em que a entrada é a tensão $v^+_q(t)$, em V, a saída é a frequência $x(t)$, em rad/s, sendo k_p o ganho da ação proporcional, em rad/s·V, e k_i o ganho da ação integral, em rad^2/s^2·V.

$$\frac{X(s)}{V_q^+(s)} = k_p + \frac{k_i}{s} \tag{4.5}$$

Aplicando-se a transformação bilinear em (4.5), obtém-se a função de transferência mostrada em (4.6), que é o modelo em tempo discreto do controlador PI.

$$\frac{X(z)}{V_q^+(z)} = \frac{(2k_p + k_i T)z + k_i T - 2k_p}{2(z-1)} \tag{4.6}$$

Aplicando-se a transformada z inversa em (4.6) e isolando o termo $x(k+1)$ resultante, tem-se (4.7), que estabelece um método para encontrar determinada amostra da saída do controlador PI.

$$x(k+1) = \left(\frac{k_i T}{2} + k_p\right) v_q^+(k+1) + \left(\frac{k_i T}{2} - k_p\right) v_q^+(k) + x(k) \tag{4.7}$$

Para discretizar o integrador do SRF-PLL , procede-se de modo análogo. A função de transferência em tempo contínuo do integrador do SRF-PLL é mostrada na (4.8), a entrada $\omega'(t)$ é a frequência estimada , em rad/s, e a saída $\theta'(t)$, a fase estimada, em rad.

$$\frac{\Theta'(s)}{\Omega'(s)} = \frac{1}{s} \tag{4.8}$$

Aplicando a transformação bilinear em (4.8), obtém-se a função de transferência em tempo discreto do integrador do SRF-PLL, mostrada em (4.9).

$$\frac{\Theta'(z)}{\Omega'(z)} = \frac{T}{2}\frac{z+1}{z-1} \tag{4.9}$$

Aplicando a transformada z inversa em (4.9) e isolando o termo $\theta'(k+1)$ resultante, obtém-se (4.10), com a qual é possível calcular o valor da amostra de fase atual $\theta'(k+1)$ com base na amostra imeditamente anterior da fase $\theta'(k)$ e nas amostras atual e imediatamente anterior de frequência, $\omega'(k+1)$ e $\omega'(k)$, respectivamente.

$$\theta'(k+1) = \frac{T}{2}[\omega'(k+1) + \omega'(k)] + \theta'(k) \tag{4.10}$$

4.2 SIMULAÇÃO DO SISTEMA

Para verificar a correção do modelo em tempo discreto obtido, o sistema foi simulado por meio de um *script* de MATLAB, apresentado no Apêndice 1. Inicialmente, foram feitos testes no SOGI e SRF-PLL separados e, posteriormente, nos dois juntos.

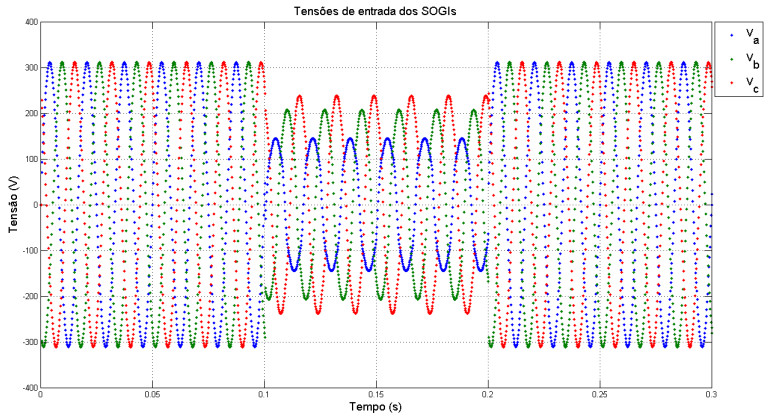
Para a simulação do SOGI, foi escolhido arbitrariamente T = 0,2 ms, que dá um total de 83 amostras por ciclo de rede, $K = \sqrt{2}$, que corresponde ao caso criticamente amortecido, e ω' = 120π rad/s, que é a frequência da rede elétrica. O resultado da simulação, para alguns sinais de interesse, são mostrados na Fig. 4.1.

As tensões da rede, mostradas na Fig. 4.1(a), até o instante 0,1 s, constituem um sistema trifásico equilibrado e não distorcido, o caso ideal. No instante 0,1 s, a rede passa a conter tanto desequilíbrio quanto distorções, harmônicas de ordens 3, 5 e 7. No instante 0,2 s, o sistema volta a tornar-se equilibrado e livre de harmônicas. Todos esses eventos se sucedem sem haver alteração de frequência, mantida em 120π rad/s.

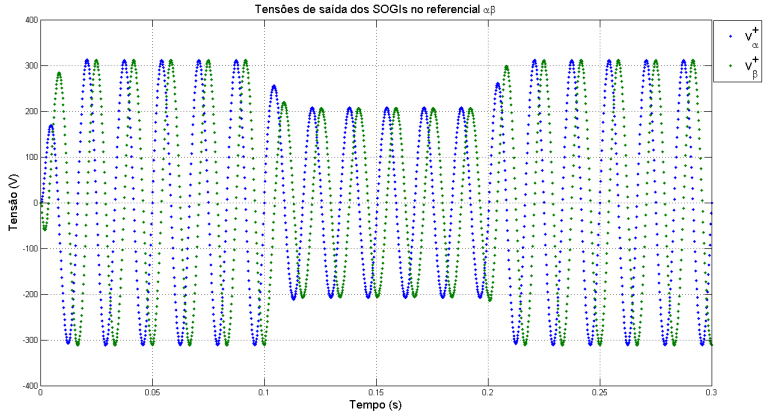
Como dito no Cap. 3, os SOGIs possuem duas funções principais: gerar o sinal em quadratura e atenuar harmônicas de alta frequência, enquanto que o PSC deve obter, no referencial $\alpha\beta$, a sequência positiva do sistema numa dada frequência. Observando a Fig. 4.1(b), que mostra as tensões produzidas pelo SOGI, nota-se que os sinais gerados pelo conjunto SOGI-PSC, as componentes α e β, estão defasadas em $\pi/2$ rad e tem a mesma amplitude, como esperado teoricamente, e, visualmente, nota-se que as harmônicas foram consideravelmente atenuadas. Na Figura 4.1(c), são mostrados os três sinais de tensão que formam a sequência positiva calculada pelo sistema, obtida pela aplicação da transformação de Clarke inversa sobre as tensões da Fig. 4.1(b).

Para a simulação do SRF-PLL, os parâmetros k_p e k_i do controlador PI foram determinados conforme a referência [12], supondo fator de amortecimento ξ = $\sqrt{2}$ e frequência de corte ω_c = 25π rad/s, o período de amostragem T = 0,2 ms e a frequência de alimentação direta ω_{ff} = 120π rad/s. Os resultados da simulação para alguns sinais de interesse são mostrados na Figura 4.2.

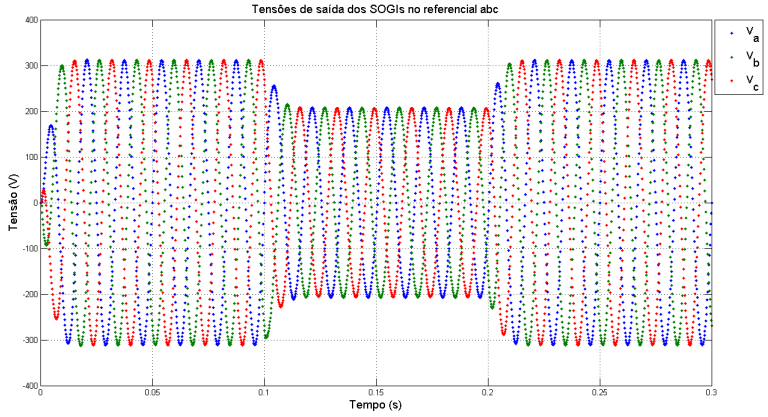
As tensões mostradas na Fig. 4.2(a) servem de entrada para o PLL. São formas de onda de tensão sem harmônicas nem desequilíbrio. Até o instante 0,15 s, as formas de onda têm frequência 120π rad/s. No instante 0,15 s, a frequência sofre uma redução de 10 %, indo para 108π rad/s. Observando a Fig. 4.2(b), vê-se que, inicialmente, o SRF-PLL sintoniza a frequência inicial de 120π rad/s. Ao ocorrer a alteração brusca de frequência, o PLL leva cerca de 50 ms para sintonizar a nova frequência de 108π rad/s. Na Fig. 4.2(c), são mostradas as tensões v_d e



(a) Entrada do sistema



(b) Saídas do PSC

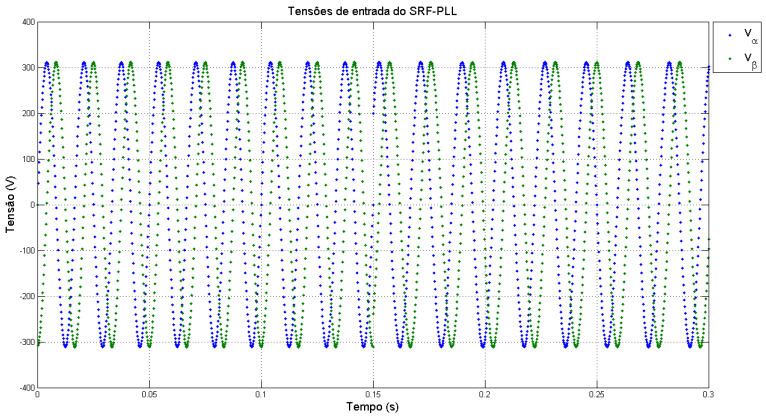


(c) Sequência positiva estimada

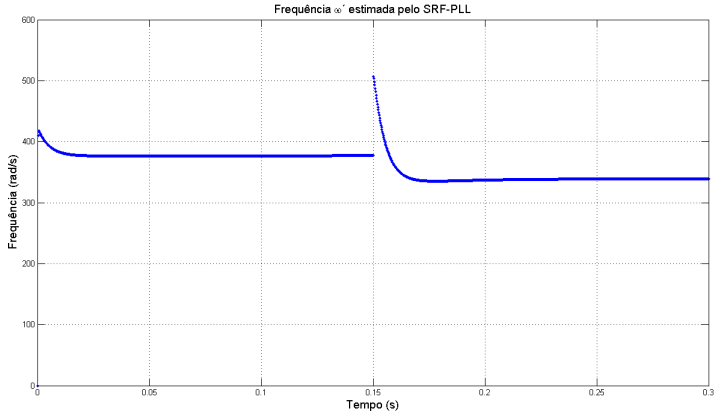
Figura 4.1 – Simulações do subsistema DSOGI-PSC

v_q. Pode-se ver na figura que, como esperado, em regime permanente, v_d tende a tornar-se nula enquanto v_q tende a assumir o oposto do valor da amplitude da tensão da fase a.

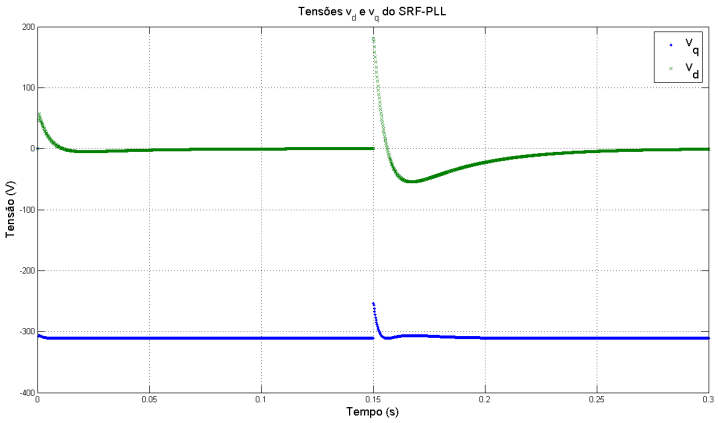
Para a simulação de todo o sistema, foram usados $T = 0{,}2$ ms, $K = \sqrt{2}$, $\xi = \sqrt{2}$ e ω_c



(a) Tensões de entrada do SRF-PLL



(b) Frequência estimada pelo SRF-PLL

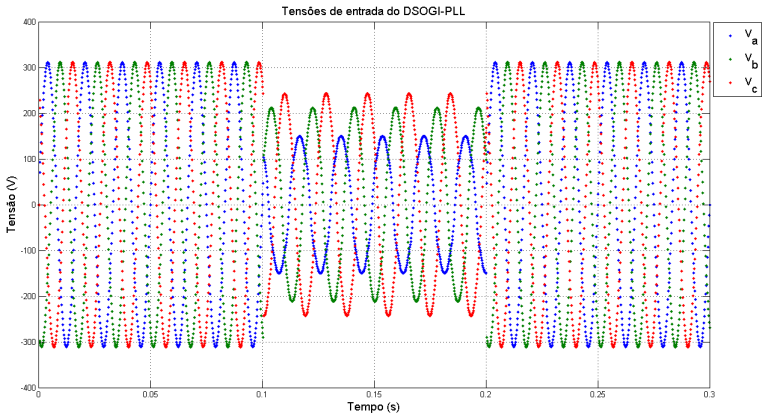


(c) Tensões v_d e v_q

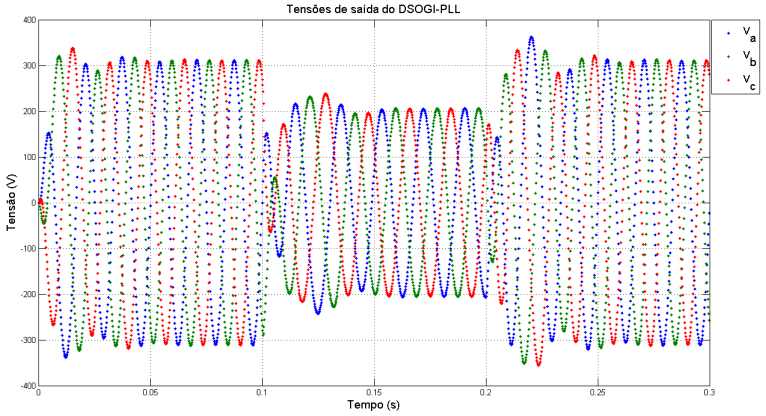
Figura 4.2 – Sinais da simulação do SRF-PLL

= 25π, os mesmos valores usados nas simulações anteriores. As formas de onda resultantes da simulação para alguns sinais de interesse são mostrada na Fig. 4.3.

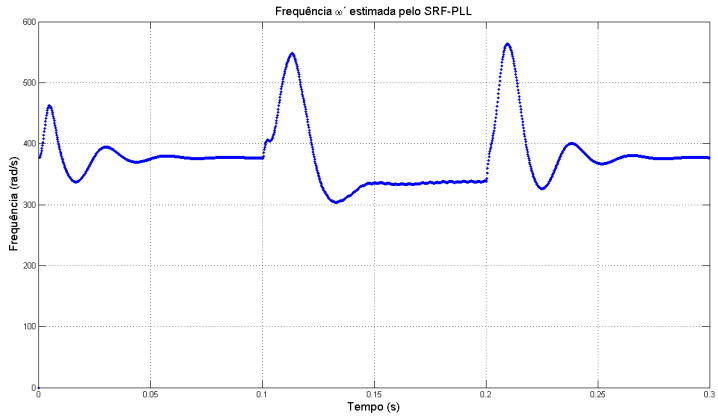
O sinal de entrada do sistema é mostrado na Fig. 4.3(a), na qual se vê que do instante 0 ao



(a) Entrada do sistema



(b) Saídas do DSOGI-PLL



(c) Frequência estimada pelo SRF-PLL

Figura 4.3 – Alguns dos sinais resultantes da simulação do DSOGI-PLL

instante 0,1 s, as tensões de entrada do sistema constituem um sistema trifásico equilibrado e sem distorções. No instante 0,1 s, ocorre uma falha na rede, que faz com que a frequência do sistema se altere, passando de 120π rad/s para 108π rad/s, além disso, a rede passa a ser desequilibrada e

a conter harmônicas de ordens 3, 5 e 7. Passada a falha, a partir do instante 0,2 s, a rede volta a ser equilibrada e livre de harmônicas na frequência original de 120π rad/s.

Pela Fig. 4.3(b), que mostra a resposta do DSOGI-PLL, é possível ver que, na configuração em que está, ele leva cerca de 50 ms (3 ciclos de rede) para, saindo de condições iniciais nulas, atingir o regime permanente. No instante 100 ms, quando ocorre a falha na rede, o sistema leva novamente cerca de 50 ms para atingir o regime, situação em que as formas de onda de saída, em comparação com as de entrada, estão visivelmente menos distorcidas e formando um sistema equilibrado na nova frequência, como esperado. Do instante 200 ms em diante, o DSOGI-PLL gasta o mesmo tempo de 3 ciclos de rede para atingir novamente o estado estacionário, agora nas condições anteriores à falha.

A Fig. 4.3(c), apresenta o comportamento, ao longo do tempo, da frequência da rede estimada pelo SRF-PLL. Nos instantes em que a frequência sofre variação brusca, o PLL leva cerca de 50 ms para atingir o estado estacionário. Entre os instantes 100 ms e 200 ms é ainda possível observar um ruído presente no sinal, o qual se deve às harmônicas, que não foram completamente atenuadas.

4.3 DESCRIÇÃO DO HARDWARE USADO PARA A IMPLEMENTAÇÃO FÍSICA

Para realizar os ensaios em tempo real, nos quais a tensão trifásica de entrada provém da rede elétrica, é preciso, antes, formalizar o algoritmo que descreve o modelo discreto do DSOGI-PLL numa linguagem de programação apropriada, gerando um código que será carregado para uma plataforma de processamento de dados. Em aplicações reais, o PLL é, em geral, carregado num DSP (*Digital Signal Processor*), mas, como esse tipo de dispositivo é bastante complexo no que se refere a programação e conexão com periféricos, optou-se pelo Arduino DUE, mostrado na Fig. 4.4.

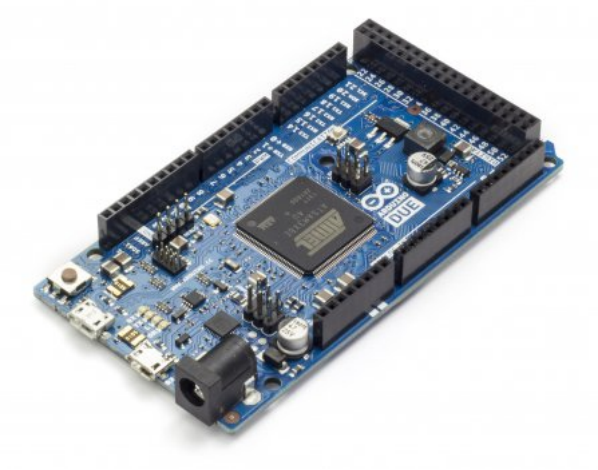


Figura 4.4 – Placa Arduino DUE

O DUE foi escolhido porque, assim como os demais Arduinos, possui um ambiente de desenvolvimento integrado (IDE), o qual facilita consideravelmente a codificação, o carregamento de programas e a depuração. Além disso, dentre as placas Arduino disponíveis comercialmente, o DUE é a que possui maior capacidade de processamento (maior memória e maior frequência de clock).

O DUE é uma placa baseada no microcontrolador Atmel SAM3X8E, com arquitetura ARM Cortex-M3 de 32 bits, capaz de realizar operações de 4 *bytes* num ciclo de *clock*. O DUE conta com 54 pinos de entrada e de saída digitais, dos quais 12 podem ser usados como saída PWM. Conta também com 12 pinos de entrada analógica, 2 pinos de saída analógica (DAC0 e DAC1), 512 kB de memória para programas e *clock* de 84 MHz. Diferentemente da maioria das placas da série Arduino, que trabalham com tensão de operação de 5 V, a tensão de operação do Arduino DUE é de 3,3 V. Os 12 pinos de entrada e os 2 de saída analógicas do DUE podem

comportar até 12 bits de resolução, totalizando 4096 níveis distintos de tensão no intervalo de 0 a 3,3 V, o que gera 0,8 mV de resolução. Como o Arduino DUE possui apenas duas portas de saída analógica, só é possível visualizar num osciloscópio dois sinais de saída por vez.

Os dois pinos de saída analógica do DUE, DAC0 e DAC1, supostamente, deveriam fornecer tensões de 0 a 3,3 V. Entretanto, testes feitos antes de carregar e executar o DSOGI-PLL mostraram que a tensão mínima fornecida nas saídas analógicas era de, aproximadamente, 0,5 V e a tensão máxima, 2,7 V.

Como as entradas analógicas do DUE admitem tensões de, no máximo, 3,3 V, as três tensões da rede, que, em condições ideais de operação, variam senoidalmente de -220$\sqrt{2}$ a 220$\sqrt{2}$ V, foram transformadas em três sinais senoidais, de mesma frequência, que variavam de 0 a 3 V. Essa transformação foi efetuada por meio do sistema mostrado na Fig. 4.5.

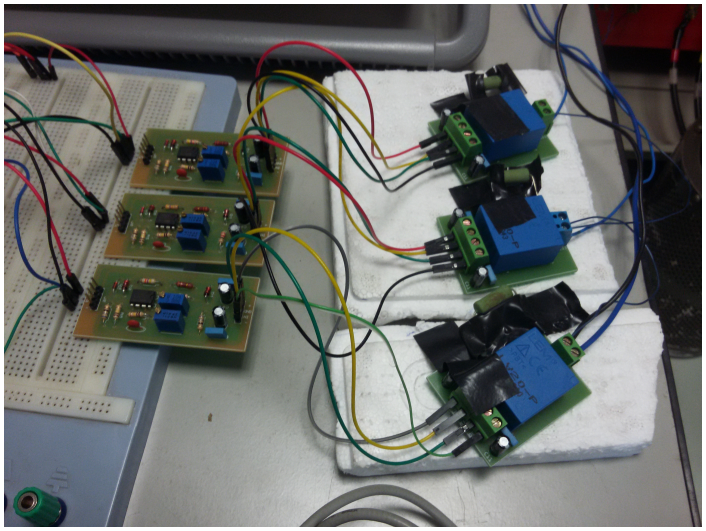


Figura 4.5 – Sistema usado para transformar a tensão da rede

Na Fig. 4.5, cada uma das três placas da direita é um circuito de medição que, usando um sensor de efeito Hall, mensura a tensão das fases *a*, *b* e *c* da rede. As três placas da direita, por sua vez, são circuitos de condicionamento que, além de transferirem as tensões medidas pelo circuito anterior para a faixa de 0 a 3 V, filtram os sinais e permitem o ajuste da amplitude e do *offset* de cada senoide. As senoides de 0 a 3 V servem de entrada para o Arduino. Assim, teoricamente, quando, em certo instante de tempo, a tensão da rede for, por exemplo, 220$\sqrt{2}$ V, o Arduino enxergará 3 V na entrada analógica correspondente e, quando a tensão instantânea for nula, momento em que a senoide cruza o eixo do tempo, o DUE verá 1,5 V.

4.4 ENSAIOS COM A REDE ELÉTRICA

O *script* de Arduino apresentado no Apêndice 2 é o código elaborado para os testes do sistema com a rede elétrica.

Os sinais mostradas na Fig 4.6 são as tensões da rede transformadas para a faixa de 0 a 3 V. Observando as medidas de tensão média, máxima e frequência feitas pelo osciloscópio, nota-se que condizem com o que se espera da senoide na faixa de 0 a 3 V, isto é, tensão média de 1,5 V, tensão máxima de 3 V e frequência de 60 Hz.

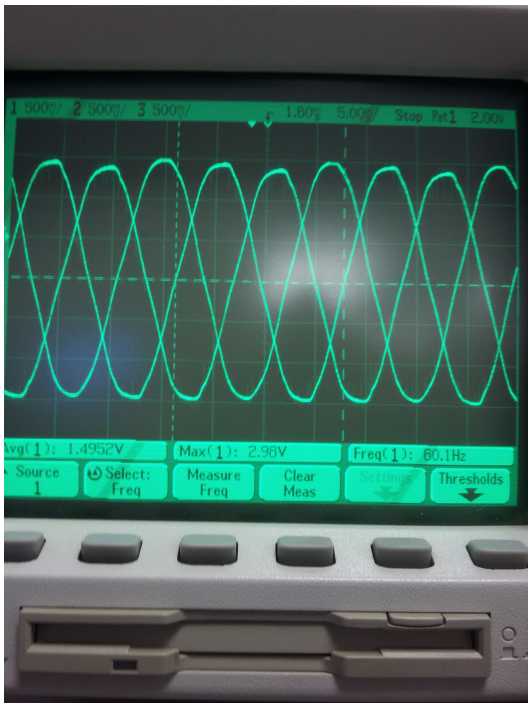


Figura 4.6 – Tensões representativas de v_a, v_b e v_c

O algoritmo implementado deve utilizar os valores instatâneos reais da tensão da rede, portanto, dentro do código, as tensões de 0 a 3 V são matematicamente trazidas de volta para a faixa -220$\sqrt{2}$ a 220$\sqrt{2}$ V, e os cálculos são feitos com esses valores. Caso se deseje visualizar certo sinal, deve-se proceder a passagem dele para a faixa de 0 a 3 V, dentro do próprio código, aplicar o sinal desejado num dos pinos DAC e conectar o referido pino a um canal de osciloscópio.

Na Fig. 4.7, são mostradas as saídas do sistema, isto é, as tensões v_α^+ e v_β^+. As medições apresentadas pelo osciloscópio mostram que a frequência de ambas é de, aproximadamente, 60 Hz e estão defasadas π/2 rad entre si. No tempo, a defasagem de π/2 rad equivale, teoricamente, a um atraso de, aproximadamente, 4,17 ms. Para estimar esse atraso, basta observar que na Fig. 4.7 cada quadradinho corresponde a 5 ms e a distância entre os picos das senoides é pouco menor

que o comprimento de um quadradinho.

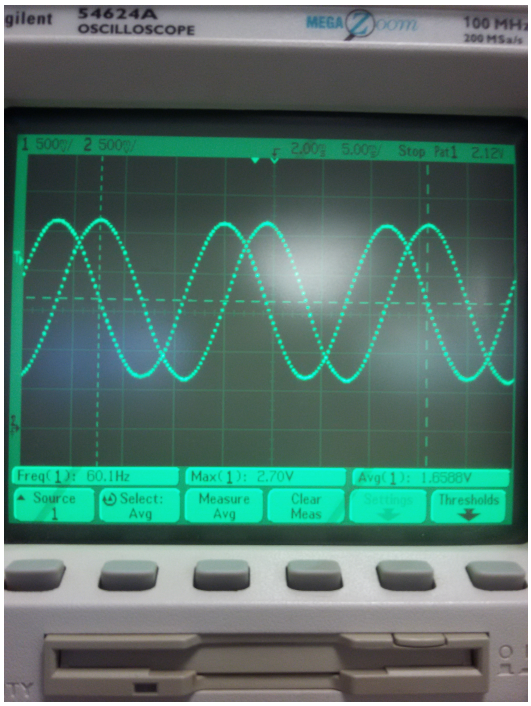


Figura 4.7 – Tensões v_{α}^{+} e v_{β}^{+}

Os sinais mostrados na Fig. 4.7 servem de entrada para o PLL, cuja saída, a fase $\theta^{+'}$, é mostrada na Fig. 4.8 juntamento com a tensão v_{α}^{+}. Nessa figura, é possível ver que a frequência dos dois sinais da Fig. 4.8 é de, aproximadamente, 60 Hz, como esperado. Percebe-se também que a forma de onda dente de serra está sincronizada com a tensão v_{α}^{+}, uma vez que o ciclo de ambas se inicia e termina juntamente. Isso comprova que a fase captada pelo PLL é a da tensão de entrada e, portanto, a frequência estimada é a da rede.

Na Fig. 4.9, são mostradas as tensões v_{d}^{+}, curva inferior, e v_{q}^{+}, curva superior. No Cap. 3, mostrou-se que, teoricamente, a tensão v_{q}^{+}, em regime permanente, se iguala à menos a amplitude da tensão de entrada e a tensão v_{d}^{+} se iguala a zero. No caso das entradas na faixa dos 3 V, isso significa que v_{q}^{+} deverá tender ao valor máximo de v_{α}^{+}, ou seja, 2,56 V e v_{d}^{+} ao valor médio de v_{α}^{+}, que é 1,616 V, fato que é comprovado ao se comparar a Fig. 4.7 com a Fig. 4.9.

É recomendável que o tempo de execução do DSOGI-PLL seja o menor possível, pois, em aplicações reais, o DSP é encarregado de executar outros algoritmos além do PLL, os quais somarão certo tempo ao processamento caso não sejam executados paralelamente. A diminuição do tempo de processamento foi materializada no código, por exemplo, reduzindo-se, sempre que possível, a quantidade de multiplicações, divisões, radiciações e potenciações, que são operações que consomem tempo considerável de processamento. Optou-se também pela substituição

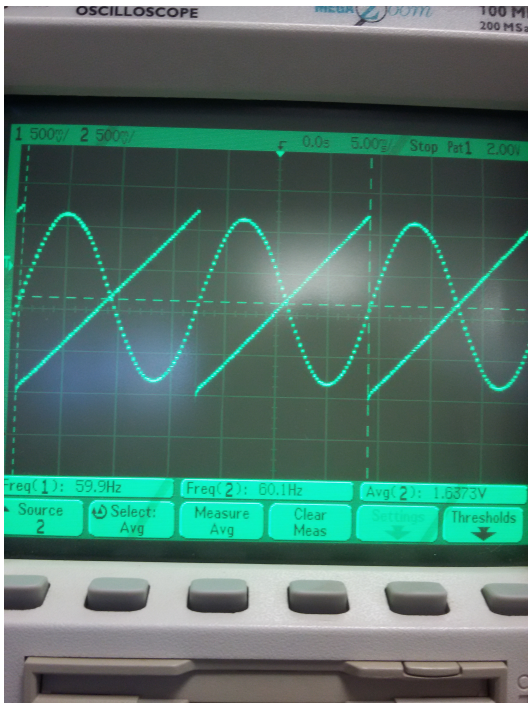


Figura 4.8 – Tensões v_{α}^{+} e $\theta^{+'}$

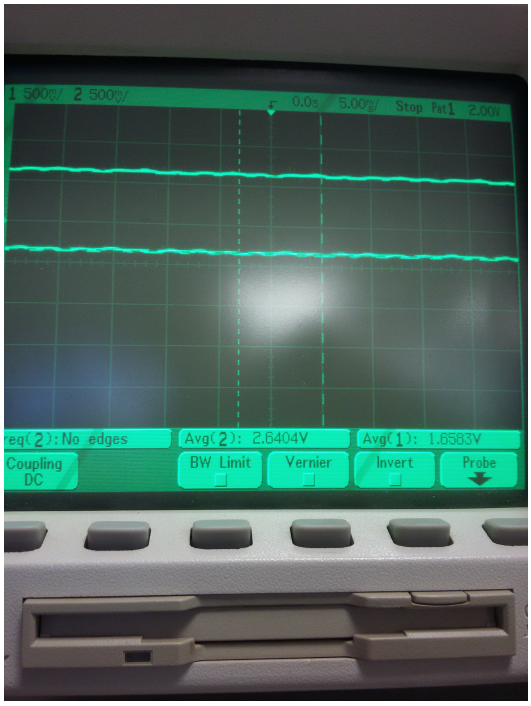


Figura 4.9 – Tensões v_{d}^{+} e v_{q}^{+}

das constantes resultantes de operações pelos resultados finais correspondentes. E a propriedade distributiva foi amplamente usada para permutar expressões do tipo *a*p + b*p* por *p*(a + b)*, devido à clara conveniência de se realizar apenas uma multiplicação em vez de duas. Também foram usadas tabelas de aproximações de senos e cossenos, visto que o cálculo dessas funções

pelo Arduino é computacionalmente dispendioso. Como resultado, o tempo de processamento de cada iteração do *loop* principal do código ficou em torno de 223 us, como mostrado na Fig 4.10, na qual um pulso é usado para estimar o tempo de processamento do *loop* principal. Como o tempo de cada iteração foi estimado em 223 us, o período de amostragem escolhido foi um valor próximo, 230 us.

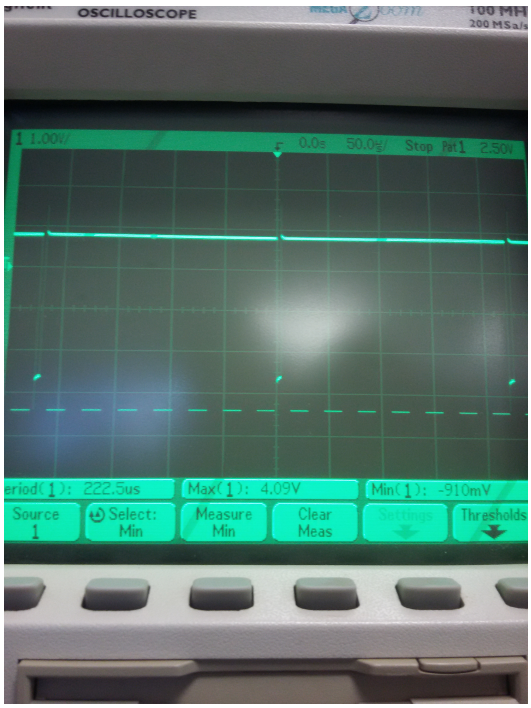


Figura 4.10 – Pulso usado para estimar o tempo de execução de uma iteração do laço *loop* do código para Arduino

Como o processamento do código leva certo tempo, é natural que surja um atraso entre os sinais produzidos pelo programa e os sinais de entrada provenientes da rede. Esse atraso pode ser visualizado na Fig. 4.11, na qual se vê duas formas de onda. A forma de onda com linha contínua é a tensão da fase *a* da rede e a forma de onda com linha descontínua é a tensão da fase *a* da rede estimada pelo algoritmo.

Teoricamente, o atraso entre as tensões deve ser da mesma ordem de grandeza do tempo de execução de uma iteração do bloco *loop* do Arduino. Pela Fig. 4.11, é possível estimar esse atraso. Sabendo que, de acordo com o osciloscópio, cada quadradinho da figura corresponde a 2 ms e, ao longo da reta tracejada, a distância entre as duas formas de onda é de aproximadamente 1/4 de um quadradinho, o atraso é de aproximadamente 500 us.

As formas de onda da Fig. 4.11 deveriam ter amplitudes iguais, o que não se observa. Isso se deve ao fato de que, como mencionado anteriormente, os pinos DAC do Arduino DUE

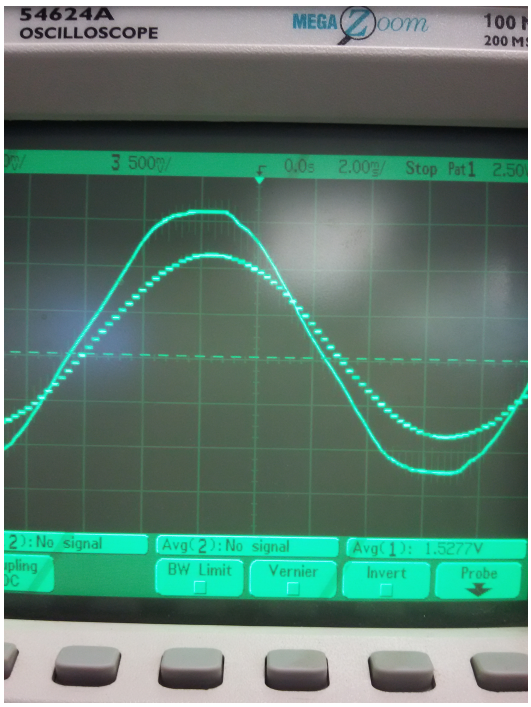


Figura 4.11 – Atraso entre as tensões v_a e v_α^+

não fornecem tensão máxima de 3,3 V nem tensão mínima de 0 V, mas sim 2,7 V e 0,5 V, respectivamente.

5 CONCLUSÃO

A detecção correta da sequência positiva de tensões da rede elétrica é um aspecto essencial para o adequado funcionamento dos atuais conversores de potência que conectam plantas de energia renovável à rede. O DSOGI-PLL é um sistema que se presta a esse propósito. Neste trabalho, ele foi descrito, simulado e testado, apresentando desempenho condizente com o que se esperava.

Na referência [11], em que o DSOGI-PLL é apresentado, os autores simularam o sistema em tempo contínuo para uma rede elétrica fictícia de 70,71 Vrms e 50 Hz. Nesse sentido, a contribuição deste trabalho reside no fato de que, como desde o início sempre houve a pretensão de se implementar o sistema num processador e testá-lo com a rede, as simulações do sistema foram feitas em tempo discreto e as tensões trifásicas usadas nas simulações procuraram refletir a realidade da rede elétrica brasileira, portanto nelas foram usadas como entrada tensões trifásicas de 220 Vrms e 60 Hz. Por outro lado, para testar a robustez do algoritmo, a magnitude dos fenômenos que afetam a qualidade da energia foi ajustada para valores irrealisticamente elevados nas simulações, a exemplo de variações de frequência de 10 % e 3 % de harmônicas de 3.ª ordem.

A referência mencionada não elenca dados de uma implementação em hardware do sistema. Neste trabalho, no entanto, o DSOGI-PLL foi implementado numa plataforma de prototipagem eletrônica, o Arduino DUE. Apesar de não ser a plataforma mais adequada, um DSP seria mais apropriado, o DUE se mostrou suficientemente adequado para constatar a correção do algoritmo e estimar tempos de processamento. Não foi possível, entretanto, avaliar o desempenho do método frente a condições adversas da rede.

Alguns aspectos secundários foram deixados de fora deste trabalho, tornando-o suscetível a aperfeiçoamentos posteriores. Poder-se-ia, por exemplo, alterar a programação de ponto flutuante para ponto fixo, o que tornaria a execução do código do Arduino, a cada iteração, ainda mais rápida. Sugere-se também fazer ensaios nos quais o DSOGI-PLL é submetido a redes trifásicas desequilibradas, distorcidas ou com variação de frequência, usando, para produzir as tensões, uma fonte de potência programável.

Em trabalhos futuros, pode-se aperfeiçoar o DSOGI-PLL para que ele forneça outras informações além da sequência positiva de tensões como, por exemplo, a sequência negativa e, com base nisso, calcular o fator de desequilíbrio da rede usando um método conhecido. Na litera-

tura pertinente é ainda possível encontrar outros métodos de obtenção da sequência positiva, a exemplo do DSOGI-FLL, que, como o nome sugere, utiliza um FLL (*Frequency Locked Loop*) em lugar de um PLL para obter a frequência da rede elétrica. Sugere-se ainda que, implementados e testados ambos os métodos, se faça uma comparação de desempenho funcional entre eles, evidenciando vantagens e desvantagens de cada um.

REFERÊNCIAS BIBLIOGRÁFICAS

1 DACHERY, J. M. http://energiarenovavel.org/index.php?option=com_content&task=view&id=725&itemid=182. *Consulta realizada em 23/2/2016*.

2 http://www.portal-energia.com/uniao-europeia-comprometida-com-metas-nas-energias-renovaveis/. *Consulta realizada em 9/5/2016*.

3 TEODORESCU, R.; LISERRE, M.; RODRíGUEZ, P. Grid converters for photovoltaic and wind power systems. *John Wiley & Sons, Ltd.*, 2011.

4 http://g1.globo.com/globo-news/noticia/2013/06/producao-de-energia-no-brasil-segue-concentrada-nas-hidreletricas.html. *Consulta realizada em 23/2/2016*.

5 BARBOSA, V. http://exame.abril.com.br/economia/noticias/a-nova-era-da-energia-renovavel-ja-comecou-no-brasil. *Consulta realizada em 23/2/2016*.

6 FORTESCUE, C. L. Method of symmetrical coordinates applied to the solution of polyphase networks. *Transactions of the AIEE*, 1918.

7 LYON, W. V. Application of the method of symmetrical components. *New York: McGraw-Hill*, 1937.

8 CLARKE, E. Circuit analysis of ac power systems. *New York: John Wiley and Sons, Inc.*, v. 1, 1950.

9 PARK, R. H. Two reaction theory of synchronus machines. generalized method of analysis - part 1. *In Proceedings of the Winter Convetion of the AIEE*, 1929.

10 FILHO, R. M. dos S.; SEIXAS, P. F.; CORTIZO, P. C. A comparative study of three-phase and single-phase pll algorithms for grid-connected systems. *In Proc. INDSUCON Conf. Rec*, Recife, Brasil 2006.

11 RODRíGUEZ, P.; TEODORESCU, R.; CANDELA, I.; TIMBUS, A. V.; LISERRE, M.; BLAABJERG, F. New positive-sequence voltage detector for grid synchronization of power converters under faulty grid conditions. *in IEEE ANNUAL POWER ELECTRONICS SPECIALISTS CONFERENCE*, Montreal, Canadá, 2006.

12 CHUNG, S. A phase tracking system for three phase utility interface inverters. *IEEE Transactions on Power Electronics, VOL. 15, NO. 3, p. 431-438*, May 2000.

APÊNDICES

I.1 SCRIPT DE MATLAB USADO NA SIMULAÇÃO

Abaixo, tem-se o *script* de MATLAB usado na simulação do DSOGI-PLL

```
T = 0.0002; % período de amostragem
K = sqrt(2); % ganho dos SOGIs
limite = 3000; % número de amostras desejados
ksi = sqrt(2); % fator de amortecimento
omega_c = 25.0*pi; % frequência de corte
Em = 220.0*sqrt(2); % amplitude da senoide
clarke = [2/3 -1/3 -1/3; 0 sqrt(3)/3 -sqrt(3)/3]; % matriz de transformação de Clarke
clarke_inversa = [1 0; -1/2 sqrt(3)/2; -1/2 -sqrt(3)/2]; % matriz de transformação inversa de Clarke

v_alfa_beta(:, 1) = 0.0; %
v_alfa_beta(:, 2) = 0.0; %
v_alfa_linha(2) = 0.0; %
v_alfa_linha(1) = 0.0; %
v_beta_linha(2) = 0.0; %
v_beta_linha(1) = 0.0; %
qv_alfa_linha(2) = 0.0; %
qv_alfa_linha(1) = 0.0; % Valores iniciais
qv_beta_linha(2) = 0.0; %
qv_beta_linha(1) = 0.0; %
teta_mais_linha(1) = 0.0; %
omega_linha(1) = 0.0; %
x(1) = 0.0; %
v_d(1) = 0.0; %
v_q(1) = 0.0; %

omega_ff = 120.0*pi; % frequência de alimentação direta
k_p = 2.0*ksi*omega_c/Em; % ganho da ação proporcional
k_i = omega_cÂ²/Em; % ganho da ação integradora

for k = 1:1:limite-2;

% Rede equilibrada e sem distorções (caso ideal)
% vabc = [100.0*cos(100.0*pi*k*T); 100.0*cos(100.0*pi*k*T - 2.0*pi/3); 100.0*cos(100.0*pi*k*T + 2.0*pi/3)];
% vabc(:, k+2) = [220.0*sqrt(2)*cos(120.0*pi*(k+2)*T); 220.0*sqrt(2)*cos(120.0*pi*(k+2)*T - 2.0*pi/3); 220.0*sqrt(2)*cos(120.0*pi*(k+2)*T
+ 2.0*pi/3)];

% Rede com alteração de frequência
% if k < limite/3
% vabc(:, k+2) = [220.0*sqrt(2)*cos(120.0*pi*(k+2)*T); 220.0*sqrt(2)*cos(120.0*pi*(k+2)*T - 2.0*pi/3); 220.0*sqrt(2)*cos(120.0*pi*(k+2)*T
+ 2.0*pi/3)];
% elseif k > limite/3 && k < (2/3)*limite
% vabc(:, k+2) = [220.0*sqrt(2)*cos(100.0*pi*(k+2)*T); 220.0*sqrt(2)*cos(100.0*pi*(k+2)*T - 2.0*pi/3); 220.0*sqrt(2)*cos(100.0*pi*(k+2)*T
+ 2.0*pi/3)];
% else
% vabc(:, k+2) = [220.0*sqrt(2)*cos(120.0*pi*(k+2)*T); 220.0*sqrt(2)*cos(120.0*pi*(k+2)*T - 2.0*pi/3); 220.0*sqrt(2)*cos(120.0*pi*(k+2)*T
+ 2.0*pi/3)];
% end
% Rede desequilibrada e não distorcida
% if k < limite/3
% vabc(1, k+2) = 220.0*sqrt(2.0)*cos(120.0*pi*(k+2)*T);
% vabc(2, k+2) = 220.0*sqrt(2.0)*cos(120.0*pi*(k+2)*T - 2.0*pi/3);
% vabc(3, k+2) = 220.0*sqrt(2.0)*cos(120.0*pi*(k+2)*T + 2.0*pi/3);
% elseif k > limite/3 && k < (2/3)*limite
% vabc(1, k+2) = 0.5*220.0*sqrt(2.0)*cos(120.0*pi*(k+2)*T - 20.0*pi/180.0);
% vabc(2, k+2) = 0.7*220.0*sqrt(2.0)*cos(120.0*pi*(k+2)*T - 134.0*pi/180.0);
% vabc(3, k+2) = 0.8*220.0*sqrt(2.0)*cos(120.0*pi*(k+2)*T + 110.0*pi/180.0);
% else
% vabc(1, k+2) = 220.0*sqrt(2.0)*cos(120.0*pi*(k+2)*T);
```

```
% vabc(2, k+2) = 220.0*sqrt(2.0)*cos(120.0*pi*(k+2)*T - 2.0*pi/3);
% vabc(3, k+2) = 220.0*sqrt(2.0)*cos(120.0*pi*(k+2)*T + 2.0*pi/3);
% end

% Rede equilibrada com harmônicas de ordens 3, 5 e 7
% if k < limite/3
% vabc(1, k+2) = 220.0*sqrt(2.0)*cos(120.0*pi*(k+2)*T);
% vabc(2, k+2) = 220.0*sqrt(2.0)*cos(120.0*pi*(k+2)*T - 2.0*pi/3);
% vabc(3, k+2) = 220.0*sqrt(2.0)*cos(120.0*pi*(k+2)*T + 2.0*pi/3);
% elseif k > limite/3 && k < (2/3)*limite
% vabc(1, k+2) = 220.0*sqrt(2.0)*cos(120.0*pi*(k+2)*T) + 0.1*220.0*sqrt(2.0)*cos(3*(120.0*pi*(k+2)*T)) + 0.075*220.0*sqrt(2.0)*cos(5*(120.0*pi*(k+2)
+ 0.05*220.0*sqrt(2.0)*cos(7*(120.0*pi*(k+2)*T));
% vabc(2, k+2) = 220.0*sqrt(2.0)*cos(120.0*pi*(k+2)*T - 120.0*pi/180.0) + 0.1*220.0*sqrt(2.0)*cos(3*(120.0*pi*(k+2)*T - 2.0*pi/3))
+ 0.075*220.0*sqrt(2.0)*cos(5*(120.0*pi*(k+2)*T - 2*pi/3)) + 0.05*220.0*sqrt(2.0)*cos(7*(120.0*pi*(k+2)*T - 120.0*pi/180.0));
% vabc(3, k+2) = 220.0*sqrt(2.0)*cos(120.0*pi*(k+2)*T + 120.0*pi/180.0) + 0.1*220.0*sqrt(2.0)*cos(3*(120.0*pi*(k+2)*T + 2.0*pi/3))
+ 0.075*220.0*sqrt(2.0)*cos(5*(120.0*pi*(k+2)*T + 2*pi/3)) + 0.05*220.0*sqrt(2.0)*cos(7*(120.0*pi*(k+2)*T + 120.0*pi/180.0));
% else
% vabc(1, k+2) = 220.0*sqrt(2.0)*cos(120.0*pi*(k+2)*T);
% vabc(2, k+2) = 220.0*sqrt(2.0)*cos(120.0*pi*(k+2)*T - 2.0*pi/3);
% vabc(3, k+2) = 220.0*sqrt(2.0)*cos(120.0*pi*(k+2)*T + 2.0*pi/3);
% end

% Rede desequilibrada, com harmônicas de ordens 3, 5 e 7 e alteração
% de frequência
if k < limite/3
vabc(1, k+2) = 220.0*sqrt(2.0)*sin(120.0*pi*(k+2)*T);
vabc(2, k+2) = 220.0*sqrt(2.0)*sin(120.0*pi*(k+2)*T - 2.0*pi/3);
vabc(3, k+2) = 220.0*sqrt(2.0)*sin(120.0*pi*(k+2)*T + 2.0*pi/3);
elseif k > limite/3 && k < (2/3)*limite
vabc(1, k+2) = 0.5*220.0*sqrt(2.0)*sin(108.0*pi*(k+2)*T - 20.0*pi/180.0) + 0.03*220.0*sqrt(2.0)*sin(3*(108.0*pi*(k+2)*T - 20.0*pi/180.0))
+ 0.02*220.0*sqrt(2.0)*cos(5*(108.0*pi*(k+2)*T - 20.0*pi/180.0)) + 0.01*220.0*sqrt(2.0)*cos(7*(108.0*pi*(k+2)*T - 20.0*pi/180.0));
vabc(2, k+2) = 0.7*220.0*sqrt(2.0)*sin(108.0*pi*(k+2)*T - 134.0*pi/180.0) + 0.03*220.0*sqrt(2.0)*sin(3*(108.0*pi*(k+2)*T - 134.0*pi/180.0))
+ 0.02*220.0*sqrt(2.0)*cos(5*(108.0*pi*(k+2)*T - 134.0*pi/180.0)) + 0.01*220.0*sqrt(2.0)*cos(7*(108.0*pi*(k+2)*T - 134.0*pi/180.0));
vabc(3, k+2) = 0.8*220.0*sqrt(2.0)*sin(108.0*pi*(k+2)*T + 110.0*pi/180.0) + 0.03*220.0*sqrt(2.0)*sin(3*(108.0*pi*(k+2)*T + 110.0*pi/180.0))
+ 0.02*220.0*sqrt(2.0)*cos(5*(108.0*pi*(k+2)*T + 110.0*pi/180.0)) + 0.01*220.0*sqrt(2.0)*cos(7*(108.0*pi*(k+2)*T + 110.0*pi/180.0));
else
vabc(1, k+2) = 220.0*sqrt(2.0)*sin(120.0*pi*(k+2)*T);
vabc(2, k+2) = 220.0*sqrt(2.0)*sin(120.0*pi*(k+2)*T - 2.0*pi/3);
vabc(3, k+2) = 220.0*sqrt(2.0)*sin(120.0*pi*(k+2)*T + 2.0*pi/3);
end

v_alfa_beta(1, k+2) = 0.5 * clarke(1, :)  * [vabc(1, k+2); vabc(2, k+2); vabc(3, k+2)];
v_alfa_beta(2, k+2) = 0.5 * clarke(2, :)  * [vabc(1, k+2); vabc(2, k+2); vabc(3, k+2)];

v_alfa_linha(k+2) = ((8/T^2 - 2*omega_linha(k)^2)*v_alfa_linha(k+1) + (2*K*omega_linha(k)/T - 4/T^2 - omega_linha(k)^2)*v_alfa_linha(k)
+ (2*K*omega_linha(k)/T)*v_alfa_beta(1, k+2) - (2*K*omega_linha(k)/T)*v_alfa_beta(1, k))/(2*K*omega_linha(k)/T + 4/T^2 + omega_linha(k)^2);
v_beta_linha(k+2) = ((8/T^2 - 2*omega_linha(k)^2)*v_beta_linha(k+1) + (2*K*omega_linha(k)/T - 4/T^2 - omega_linha(k)^2)*v_beta_linha(k)
+ (2*K*omega_linha(k)/T)*v_alfa_beta(2, k+2) - (2*K*omega_linha(k)/T)*v_alfa_beta(2, k))/(2*K*omega_linha(k)/T + 4/T^2 + omega_linha(k)^2);
qv_alfa_linha(k+2) = ((8/T^2 - 2*omega_linha(k)^2)*qv_alfa_linha(k+1) + (2*K*omega_linha(k)/T - 4/T^2 - omega_linha(k)^2)*qv_alfa_linha(k)
+ (K*omega_linha(k)^2)*v_alfa_beta(1, k+2) + (2*K*omega_linha(k)^2)*v_alfa_beta(1, k+1) + (K*omega_linha(k)^2)*v_alfa_beta(1, k))/(2*K*omega_linha(k)/T
+ 4/T^2 + omega_linha(k)^2);
qv_beta_linha(k+2) = ((8/T^2 - 2*omega_linha(k)^2)*qv_beta_linha(k+1) + (2*K*omega_linha(k)/T - 4/T^2 - omega_linha(k)^2)*qv_beta_linha(k)
+ (K*omega_linha(k)^2)*v_alfa_beta(2, k+2) + (2*K*omega_linha(k)^2)*v_alfa_beta(2, k+1) + (K*omega_linha(k)^2)*v_alfa_beta(2, k))/(2*K*omega_linha(k)/T
+ 4/T^2 + omega_linha(k)^2);

v_alfa_mais(k+1) = v_alfa_linha(k+1) - qv_beta_linha(k+1);
v_beta_mais(k+1) = qv_alfa_linha(k+1) + v_beta_linha(k+1);

v_d(k+1) = v_alfa_mais(k+1)*cos(teta_mais_linha(k)) + v_beta_mais(k+1)*sin(teta_mais_linha(k));
v_q(k+1) = v_beta_mais(k+1)*cos(teta_mais_linha(k)) - v_alfa_mais(k+1)*sin(teta_mais_linha(k));
x(k+1) = ((2*k_p + k_i*T)*v_d(k+1)+(k_i*T - 2*k_p)*v_d(k))/2 + x(k);
omega_linha(k+1) = x(k+1) + omega_ff;
teta_mais_linha(k+1) = (T*omega_linha(k+1) + T*omega_linha(k))/2 + teta_mais_linha(k);

if teta_mais_linha(k+1) > 2*pi
```

```
teta_mais_linha(k+1) = teta_mais_linha(k+1) - 2*pi;
end

vabc_mais(:, k+1) = clarke_inversa * [v_alfa_mais(k+1); v_beta_mais(k+1)];

end

k = 1:1:limite;
% plot(k*T, vabc, '.');
% title('Tensões de entrada do DSOGI-PLL', 'FontSize', 14);
% xlabel('Tempo (s)', 'FontSize', 14);
% ylabel('Tensão (V)', 'FontSize', 14);
% legend('v_a','v_b', 'v_c', 'FontSize', 14);

k = 1:1:limite-1;
% plot(k*T, vabc_mais, '.');
% title('Tensões de saída do DSOGI-PLL', 'FontSize', 14);
% xlabel('Tempo (s)', 'FontSize', 14);
% ylabel('Tensão (V)', 'FontSize', 14);
% legend('v_a','v_b', 'v_c', 'FontSize', 14);

% plot(k*T, teta_mais_linha, '.');
% title('Fase teta^ estimada pelo SRF-PLL', 'FontSize', 14);
% xlabel('Tempo (s)', 'FontSize', 14);
% ylabel('Fase estimada (rad)', 'FontSize', 14);
% plot(k*T, omega_linha, '.');

% title('Frequência omega estimada pelo SRF-PLL', 'FontSize', 14);
% xlabel('Tempo (s)', 'FontSize', 14);
% ylabel('Frequência (rad/s)', 'FontSize', 14);
% plot(k*T, v_q, 'x', k*T, v_d, '.');

% plot (k*T, x, 'x');
% plot(k*T, v_alpha_beta_mais);
% title('Tensões de entrada do SRF-PLL', 'FontSize', 14);
% xlabel('Tempo (s)', 'FontSize', 14);
% ylabel('Tensão (V)', 'FontSize', 14);
% legend('v_alpha','v_beta', 'FontSize', 14);

grid on;

clear;

clc;
```

I.2 SKETCH DE ARDUINO USADO NOS TESTES DO SISTEMA

Abaixo, tem-se o *sketch* de Arduino usado nos testes do DSOGI-PLL com a rede elétrica.

```
#define T 0.00028 // Período de amostragem (s)
#define omega_ff 376.991118 // Frequência de alimentação direta (rad/s)
#define ksi 1.414214 // Fator de amortecimento
#define omega_c 78.539816 // Frequência de corte = 25*pi rad/s
#define Em 311.126984 // Amplitude da tensão de entrada (V)
#define npoints 201
#define incr 0.0314159265 // incremento = pi/100
#define K 1.414214 // Ganho dos SOGIs = sqrt(2)

float cosseno[202] = {
1.0000, 0.9995, 0.9980, 0.9956, 0.9921, 0.9877, 0.9823, 0.9759, 0.9686, 0.9603,
0.9511, 0.9409, 0.9298, 0.9178, 0.9048, 0.8910, 0.8763, 0.8607, 0.8443, 0.8271,
0.8090, 0.7902, 0.7705, 0.7501, 0.7290, 0.7071, 0.6845, 0.6613, 0.6374, 0.6129,
0.5878, 0.5621, 0.5358, 0.5090, 0.4818, 0.4540, 0.4258, 0.3971, 0.3681, 0.3387,
0.3090, 0.2790, 0.2487, 0.2181, 0.1874, 0.1564, 0.1253, 0.0941, 0.0628, 0.0314,
0.0000, -0.0314, -0.0628, -0.0941, -0.1253, -0.1564, -0.1874, -0.2181, -0.2487, -0.2790,
-0.3090, -0.3387, -0.3681, -0.3971, -0.4258, -0.4540, -0.4818, -0.5090, -0.5358, -0.5621,
-0.5878, -0.6129, -0.6374, -0.6613, -0.6845, -0.7071, -0.7290, -0.7501, -0.7705, -0.7902,
-0.8090, -0.8271, -0.8443, -0.8607, -0.8763, -0.8910, -0.9048, -0.9178, -0.9298, -0.9409,
-0.9511, -0.9603, -0.9686, -0.9759, -0.9823, -0.9877, -0.9921, -0.9956, -0.9980, -0.9995,
-1.0000, -0.9995, -0.9980, -0.9956, -0.9921, -0.9877, -0.9823, -0.9759, -0.9686, -0.9603,
-0.9511, -0.9409, -0.9298, -0.9178, -0.9048, -0.8910, -0.8763, -0.8607, -0.8443, -0.8271,
-0.8090, -0.7902, -0.7705, -0.7501, -0.7290, -0.7071, -0.6845, -0.6613, -0.6374, -0.6129,
-0.5878, -0.5621, -0.5358, -0.5090, -0.4818, -0.4540, -0.4258, -0.3971, -0.3681, -0.3387,
-0.3090, -0.2790, -0.2487, -0.2181, -0.1874, -0.1564, -0.1253, -0.0941, -0.0628, -0.0314,
-0.0000, 0.0314, 0.0628, 0.0941, 0.1253, 0.1564, 0.1874, 0.2181, 0.2487, 0.2790,
0.3090, 0.3387, 0.3681, 0.3971, 0.4258, 0.4540, 0.4818, 0.5090, 0.5358, 0.5621,
0.5878, 0.6129, 0.6374, 0.6613, 0.6845, 0.7071, 0.7290, 0.7501, 0.7705, 0.7902,
0.8090, 0.8271, 0.8443, 0.8607, 0.8763, 0.8910, 0.9048, 0.9178, 0.9298, 0.9409,
0.9511, 0.9603, 0.9686, 0.9759, 0.9823, 0.9877, 0.9921, 0.9956, 0.9980, 0.9995,
1.0000, 0.9995
};

float seno[202] = {
0.0000, 0.0314, 0.0628, 0.0941, 0.1253, 0.1564, 0.1874, 0.2181, 0.2487, 0.2790,
0.3090, 0.3387, 0.3681, 0.3971, 0.4258, 0.4540, 0.4818, 0.5090, 0.5358, 0.5621,
0.5878, 0.6129, 0.6374, 0.6613, 0.6845, 0.7071, 0.7290, 0.7501, 0.7705, 0.7902,
0.8090, 0.8271, 0.8443, 0.8607, 0.8763, 0.8910, 0.9048, 0.9178, 0.9298, 0.9409,
0.9511, 0.9603, 0.9686, 0.9759, 0.9823, 0.9877, 0.9921, 0.9956, 0.9980, 0.9995,
1.0000, 0.9995, 0.9980, 0.9956, 0.9921, 0.9877, 0.9823, 0.9759, 0.9686, 0.9603,
0.9511, 0.9409, 0.9298, 0.9178, 0.9048, 0.8910, 0.8763, 0.8607, 0.8443, 0.8271,
0.8090, 0.7902, 0.7705, 0.7501, 0.7290, 0.7071, 0.6845, 0.6613, 0.6374, 0.6129,
0.5878, 0.5621, 0.5358, 0.5090, 0.4818, 0.4540, 0.4258, 0.3971, 0.3681, 0.3387,
0.3090, 0.2790, 0.2487, 0.2181, 0.1874, 0.1564, 0.1253, 0.0941, 0.0628, 0.0314,
0.0000, -0.0314, -0.0628, -0.0941, -0.1253, -0.1564, -0.1874, -0.2181, -0.2487, -0.2790,
-0.3090, -0.3387, -0.3681, -0.3971, -0.4258, -0.4540, -0.4818, -0.5090, -0.5358, -0.5621,
-0.5878, -0.6129, -0.6374, -0.6613, -0.6845, -0.7071, -0.7290, -0.7501, -0.7705, -0.7902,
-0.8090, -0.8271, -0.8443, -0.8607, -0.8763, -0.8910, -0.9048, -0.9178, -0.9298, -0.9409,
-0.9511, -0.9603, -0.9686, -0.9759, -0.9823, -0.9877, -0.9921, -0.9956, -0.9980, -0.9995,
-1.0000, -0.9995, -0.9980, -0.9956, -0.9921, -0.9877, -0.9823, -0.9759, -0.9686, -0.9603,
-0.9511, -0.9409, -0.9298, -0.9178, -0.9048, -0.8910, -0.8763, -0.8607, -0.8443, -0.8271,
-0.8090, -0.7902, -0.7705, -0.7501, -0.7290, -0.7071, -0.6845, -0.6613, -0.6374, -0.6129,
-0.5878, -0.5621, -0.5358, -0.5090, -0.4818, -0.4540, -0.4258, -0.3971, -0.3681, -0.3387,
-0.3090, -0.2790, -0.2487, -0.2181, -0.1874, -0.1564, -0.1253, -0.0941, -0.0628, -0.0314,
0.0000, 0.0314
};

int teta_bit = 0, vd_1_bit = 0, vq_1_bit = 0, CosTeta_bit = 0, SenTeta_bit = 0, v_alfa_bit = 0, v_beta_bit = 0;
```

```
int r = 0, v_alfa_mais_1_bit = 0, v_beta_mais_1_bit = 0, omega_linha_bit = 0, x_1_bit = 0;
float va220 = 0.0, vb220 = 0.0, vc220 = 0.0;

float vd = 0.0, vd_1 = 0.0, vq = 0.0, vq_1 = 0.0, x = 0.0, x_1 = 0.0, omega_linha = 0.0, omega_linha_1 = 0.0, teta_mais_linha = 0.0;
float teta_mais_linha_1 = 0.0, v_alfa_mais_1 = 0.0, v_beta_mais_1 = 0.0, CosTeta = 0.0, SenTeta = 0.0;
float piR = 0.0, fracR = 0.0, R = 0.0;

float const k_p = 2.0*ksi*omega_c/Em; // Ganho da ação proporcional
float const k_i = omega_c*omega_c/Em; // Ganho da ação integral

int va = 0, vb = 0, vc = 0, v_alfa_mais_bit = 0, v_beta_mais_bit = 0, v_alfa_linha_2_bit = 0, v_beta_linha_2_bit = 0;
int qv_alfa_linha_2_bit = 0, qv_beta_linha_2_bit = 0;
float v_alfa_linha = 0.0, v_alfa_linha_1 = 0.0, v_alfa_linha_2 = 0.0;
float v_beta_linha = 0.0, v_beta_linha_1 = 0.0, v_beta_linha_2 = 0.0;
float qv_alfa_linha = 0.0, qv_alfa_linha_1 = 0.0, qv_alfa_linha_2 = 0.0;
float qv_beta_linha = 0.0, qv_beta_linha_1 = 0.0, qv_beta_linha_2 = 0.0;
float v_alfa_mais = 0.0, v_beta_mais = 0.0;
float v_alfa = 0.0, v_beta = 0.0, v_alfa_1 = 0.0, v_beta_1 = 0.0, v_alfa_2 = 0.0, v_beta_2 = 0.0;
float A = 0.0, B = 0.0, C = 0.0, D = 0.0, E = 0.0;

void setup()
{
Serial.begin(9600);
analogReadResolution(12);
analogWriteResolution(12);
pinMode(DAC0, OUTPUT);
pinMode(DAC1, OUTPUT);
pinMode(2, OUTPUT);
}

void loop()
{
// digitalWrite(2, HIGH);

// Tensões da rede em bits (0x000 a 0x7ff) 12 bits 0 V a 3 V
va = analogRead(A0);
//analogWrite(DAC0, va);
vb = analogRead(A2);
//analogWrite(DAC1, vb);
vc = analogRead(A1);
//analogWrite(DAC1, vc);

// Tensões da rede (-220sqrt2 a 220sqrt2 V)
va220 = 0.151955*va - 311.126984;
vb220 = 0.151955*vb - 311.126984;
vc220 = 0.151955*vc - 311.126984;

// Transformação de Clarke
//v_alfa_2 = 0.3333*va220 - 0.16665*(vb220 + vc220);
v_alfa_2 = 0.6666*va220 - 0.3333*(vb220 + vc220);
//v_alfa_bit = 6.580914*(v_alfa + 311.126984);
//analogWrite(DAC0, v_alfa_bit);
//v_beta_2 = 0.28865*(vb220 - vc220);
v_beta_2 = 0.5773*(vb220 - vc220);
//v_beta_bit = 6.580914*(v_beta + 311.126984);
//analogWrite(DAC1, v_beta_bit);

A = 8.0/(T*T) - 2.0*omega_linha*omega_linha;
B = 2.0*K*omega_linha/T - 4.0/(T*T) - omega_linha*omega_linha;
C = 2.0*K*omega_linha/T;
D = 2.0*K*omega_linha/T + 4.0/(T*T) + omega_linha*omega_linha;
E = K*omega_linha*omega_linha;

// DSOGI-QSG
v_alfa_linha_2 = (A*v_alfa_linha_1 + B*v_alfa_linha + C*(v_alfa_2 - v_alfa))/D;
```

```
//v_alfa_linha_2_bit = 6.580914*(v_alfa_linha_2 + 311.126984); //6.580914
//analogWrite(DAC0, v_alfa_linha_2_bit);
v_beta_linha_2 = (A*v_beta_linha_1 + B*v_beta_linha + C*(v_beta_2 - v_beta))/D;
//v_beta_linha_2_bit = 6.580914*(v_beta_linha_2 + 311.126984); //6.580914
//analogWrite(DAC0, v_beta_linha_2_bit);
qv_alfa_linha_2 = (A*qv_alfa_linha_1 + B*qv_alfa_linha + E*(v_alfa_2 + 2.0*v_alfa_1 + v_alfa))/D;
//qv_alfa_linha_2_bit = 6.580914*(qv_alfa_linha_2 + 311.126984); //6.580914
//analogWrite(DAC1, qv_alfa_linha_2_bit);
qv_beta_linha_2 = (A*qv_beta_linha_1 + B*qv_beta_linha + E*(v_beta_2 + 2.0*v_beta_1 + v_beta))/D;
//qv_beta_linha_2_bit = 6.580914*(qv_beta_linha_2 + 311.126984); //6.580914
//analogWrite(DAC1, qv_beta_linha_2_bit);

// PSC
v_alfa_mais_1 = v_alfa_linha_1 - qv_beta_linha_1;
// v_alfa_mais_bit = 6.580914*(v_alfa_mais_1 + 311.126984);
//Serial.println(v_alfa_mais_bit);
//v_alfa_mais_bit = 0.82261425*(v_alfa_mais + 311.126984);
// analogWrite(DAC0, v_alfa_mais_bit);
v_beta_mais_1 = qv_alfa_linha_1 + v_beta_linha_1;
//v_beta_mais_bit = 6.580914*(v_beta_mais + 311.126984);
//v_beta_mais_bit = 0.82261425*(v_beta_mais + 311.126984);
//analogWrite(DAC1, v_beta_mais_bit);

// Atualização de valores
v_alfa_linha = v_alfa_linha_1;
v_alfa_linha_1 = v_alfa_linha_2;

v_alfa = v_alfa_1;
v_alfa_1 = v_alfa_2;

v_beta_linha = v_beta_linha_1;
v_beta_linha_1 = v_beta_linha_2;

v_beta = v_beta_1;
v_beta_1 = v_beta_2;

// Aproximação do seno e cosseno por meio das tabelas
R = (npoints*teta_mais_linha)/6.283185;
piR = floor(R);
fracR = R - piR;
r = (int)piR;
if(fracR < 0.00001)
{
CosTeta = cosseno[r];
SenTeta = seno[r];
}
else
{
CosTeta = ((cosseno[r+1] - cosseno[r])/incr)*(teta_mais_linha - incr*r) + cosseno[r];
SenTeta = ((seno[r+1] - seno[r])/incr)*(teta_mais_linha - incr*r) + seno[r];
}

// Transformação de Park
vd_1 = v_alfa_mais_1*CosTeta + v_beta_mais_1*SenTeta;
//vd_1_bit = 6.580914*(vd_1 + 311.126984);
//analogWrite(DAC0, vd_1_bit);
vq_1 = v_beta_mais_1*CosTeta - v_alfa_mais_1*SenTeta;
//vq_1_bit = 6.580914*(vq_1 + 311.126984);
//analogWrite(DAC1, vq_1_bit);

// Controlador PI
x_1 = (k_i*T/2.0 + k_p)*vd_1 + (k_i*T/2.0 - k_p)*vd + x;
omega_linha_1 = x_1 + omega_ff;
teta_mais_linha_1 = (omega_linha_1 + omega_linha)*(T/2.0) + teta_mais_linha;
```

```
// Forma de onda dente de serra
if (teta_mais_linha_1 > 6.283185)
{
teta_mais_linha_1 = teta_mais_linha_1 - 6.283185;
}

x = x_1;
vd = vd_1;
vq = vq_1;
omega_linha = omega_linha_1;
teta_mais_linha = teta_mais_linha_1;
//teta_bit = 651.739491*teta_mais_linha;
//analogWrite(DAC1, teta_bit);
//Serial.println(teta_mais_linha_1);

//digitalWrite(2, LOW);
}
}
```