# Filtragem de Ruídos Utilizando FastICA Implementado por Blocos Funcionais Foundation Fieldbus

Gaudio Vinicius Lopes Besch, Isabele Morais Costa, Adrião Duarte Dória Neto, Jorge Dantas de Melo
Departamento de Engenharia de Computação e Automação
Universidade Federal do Rio Grande do Norte
Natal - Brasil
E-mail: gaudio@dca.ufrn.br, isabele@dca.ufrn.br, adriao@dca.ufrn.br, jdmelo@dca.ufrn.br

***Resumo*— Este artigo apresenta a descrição e o funcionamento de um sistema composto por um algoritmo inteligente, capaz de separar informação e ruídos provenientes de fontes independentes, implementado em blocos funcionais de uma rede Foundation Fieldbus. A técnica utilizada nesse processo de separação cega de fontes (Blind Source Separation - BSS) foi ICA (Independent Component of Analysis), a qual explora a possibilidade de separar uma mistura de sinais baseando-se no fato de que eles são estatisticamente independentes. O algoritmo e sua implementação nos blocos funcionais serão apresentados, bem como os resultados obtidos.**

***Palavras-Chave*— ICA, Redes Industriais, Blocos Funcionais**

## I. Introdução

No processo de extração da informação gerada por um sensor instalado em uma rede de campo é muito comum a adição de ruído. O ruído e a informação, misturados, podem apresentar-se na mesma faixa de freqüência [1], o que dificulta a utilização de filtragem seletiva em freqüência.

Em certos casos, a filtragem pode ser modelada como um problema de Separação Cega de Sinais (*Blind Source Separation - BSS*). Nesse modelo, os sinais observados são constituídos por misturas do sinal de interesse com um ruído. O termo cega indica que as fontes e a maneira como estas foram combinadas são desconhecidas [2]. O objetivo é recuperar as componentes originais, isto é, o sinal e o ruído, a partir dos sinais observados usando-se uma transformação linear encontrada por um algoritmo que considera as propriedades estatísticas de sinais independentes.

Hoje, o BSS é uma das principais áreas de pesquisa em processamento digital de sinais pela sua aplicabilidade a diversos problemas reais. Algoritmos baseados em Análise de Componentes Independentes (*Independent Component of Analysis - ICA*) vêm fornecendo soluções aos problemas BSS [3] e sendo aplicados com sucesso em diversos campos, tais como: processamento de imagens, processamento de sinais biomédicos, sistemas de telecomunicação, entre outros [4].

Neste trabalho, propõe-se a aplicação da técnica de Análise de Componentes Independentes no ambiente de redes industriais Fieldbus para a filtragem de ruídos. A abordagem, emprega blocos funcionais Foundation Fieldbus (FF) para executar o algoritmo ICA localmente nos sensores de campo, visando uma forma robusta de filtragem de ruído, mantendo o padrão de interoperabilidade do sistema.

O objetivo, além de testar o algoritmo utilizado, é o desenvolvimento de novas funcionalidades para dispositivos de campos inteligentes [5], conectados através de redes FF.

O artigo está organizado da seguinte forma: na Seção 2, o protocolo Foundation Fieldbus é descrito resumidamente com ênfase na camada de aplicação do usuário. Essa camada implementa os objetos de maior interesse nesse artigo, que são os blocos funcionais, objetos básicos para construção da aplicação da técnica proposta. A seguir, na Seção 3, é apresentado o algoritmo FastICA, baseado no método de Análise de Componentes Independentes, incluindo seu treinamento e operação. Já na Seção 4 é explanada a implementação do algoritmo FastICA por meio de blocos funcionais FF, que constitui o objetivo principal do trabalho. Nessa parte, é detalhada a disposição, interligação e configuração dos blocos funcionais, situados nos instrumentos de uma rede FF laboratorial, de forma a realizar as operações necessárias para a execução do algoritmo. Os resultados obtidos são mostrados na Seção 5 e as conclusões na Seção 6.

## II. Redes de Dispositivos de Campo FOUNDATION FIELDBUS

Uma das principais tendências na evolução de redes industriais vem sendo o incremento da inteligência integrada aos instrumentos de campo [6]. O processamento das funções de controle, que normalmente era efetuado por um equipamento centralizado, agora pode ser realizado por processadores incorporados a dispositivos, tais como sensores e atuadores localizados no chamado "chão de fábrica". Esta concepção aumenta a confiabilidade do sistema de controle, pois falhas localizadas em determinados setores de uma planta podem ser isoladas e os equipamentos que funcionam corretamente podem adotar medidas para contornar os problemas gerados por outros instrumentos.

A tecnologia FF é atualmente considerada uma das melhores soluções dentro desta abordagem. As normas FF padronizam não apenas a arquitetura dos protocolos de comunicação entre os instrumentos de campo, como também descreve componentes de software de alto nível, que permitem a construção

de aplicações de controle pelo usuário. Os instrumentos FF possuem uma razoável capacidade interna de processamento que lhes permite o processamento de várias funções de controle baseadas em variáveis contínuas ou discretas. Esta capacidade também viabiliza a implementação de uma pilha de protocolos complexos que possibilitam a comunicação digital *full duplex* entre os vários equipamentos de campo ligados por um barramento FF.

A arquitetura FF é baseada no modelo ISO/OSI, em que são definidas diversas camadas de software em diferentes níveis. Cada nível provê serviços à camada acima e serve-se de serviços disponibilizados pela camada imediatamente abaixo. A correspondência entre a pilha de protocolos FF e o modelo OSI é ilustrado pela Figura 1. No escopo deste artigo, o interesse é apenas na camada de aplicação do usuário. Detalhes da pilha de comunicações podem ser obtidos em [7]

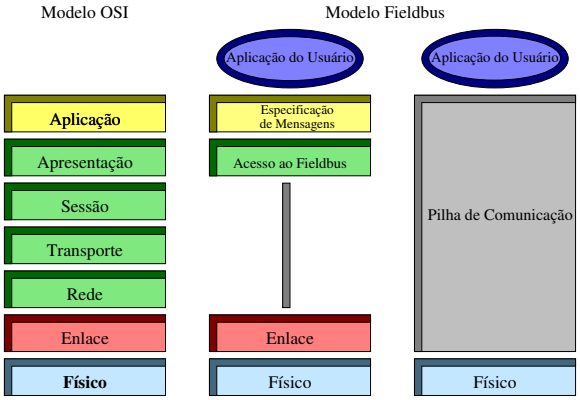


Fig. 1. Modelo OSI

Observa-se via Figura 1 que a pilha de protocolos FF inclui uma camada adicional no nível mais alto, chamada camada de aplicação do usuário, destinada a padronizar alguns componentes de software úteis em sistemas de controle de processos.

Estes componentes, denominados genericamente de Blocos Funcionais (BF) [8], são objetos implementados por estruturas de software que realizam as funções comuns ao ambiente de controle de processo. São padronizados blocos funcionais FF tanto para funções simples, como entrada e saída, como para algumas funções de maior complexidade, como controladores. Lançando-se mão desses recursos, é possível se construírem aplicações de controle com certa facilidade já que os componentes básicos, representados pelos BF, já estão instalados nos equipamentos de campo e basta, apenas, que sejam instanciados e conectados adequadamente através de suas interfaces padronizadas.

Uma das aplicações mais comuns e ilustrativas em processos são os laços de controle. Normalmente, um laço de controle simples, implementado pelos componentes da tecnologia FF, é formado por um bloco de entrada, um bloco de controle e um bloco de saída. O bloco de entrada recebe e condiciona o valor lido pelo sensor. O bloco controlador calcula o valor de atuação e o bloco de saída ajusta aquele valor para as condições do atuador em uso.

Na Figura 2, um laço de controle típico é apresentado.

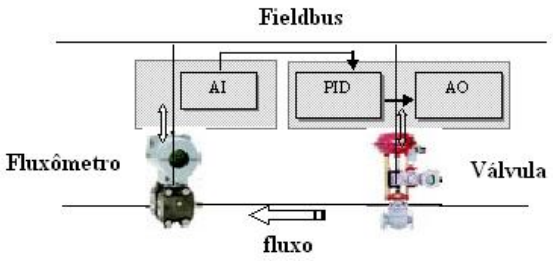


Fig. 2. Laço de controle típico em FF

Nesse laço, foram usados o bloco AI (*Analog Input*), integrado ao sensor de fluxo e os blocos PID (*Proporcional Integrativo Derivativo*) e AO (*Analog Output*), alojados na válvula. Em conjunto e comunicando-se pelo barramento FF, eles implementam um laço de controle para o fluxo na tubulação que passa pelo fluxômetro e é regulado pela válvula. O valor de fluxo lido pelo sensor é captado pelo bloco AI instanciado naquele instrumento. Nesse bloco, este valor é colocado em unidades de engenharia convenientes e limitado de acordo com os fatores de segurança requeridos. A saída do AI é transmitida periodicamente ao bloco PID situado na válvula, por meio de mensagens padronizados do protocolo FF. O bloco PID recebe a saída do AI através de um de seus parâmetros de entrada, executa o algoritmo próprio do bloco PID, previamente configurado pelo usuário, e envia sua saída no bloco AO, situado também na válvula. No AO, o valor de atuação, recebido do PID, é condicionado em escala e unidades adequadas para as particularidades da válvula em uso. No instante seguinte, o fluxo afetado pela atuação da válvula é novamente lido pelo fluxômetro e o ciclo é reiniciado.

As interfaces dos blocos funcionais são formadas por parâmetros cujos valores podem transitar pela rede FF sempre que um bloco alojado em um determinado dispositivo precisar de valores gerados por blocos residentes em um outro dispositivo. A padronização das interfaces permite que BFs implementados por diferentes fornecedores em diferentes equipamentos FF possam ser interligados sem problemas de compatibilidade. O usuário precisa apenas definir quais blocos deseja usar na sua aplicação, instanciá-los nos equipamentos que considere mais adequados, configurá-los e conectá-los convenientemente. A tarefa de conectar os blocos é normalmente realizada com auxílio de um software gráfico.

Vários blocos funcionais foram padronizados pela FF-890 Part 2 [9], mas na FF-890 Part 3 [10] são padronizados blocos de funções mais avançadas, incluindo o Bloco Aritmético, que foi utilizado na construção do algoritmo ICA de interesse neste artigo. Esse bloco e o bloco AI são detalhados na seção Implementação de algoritmos FastICA por meio de blocos funcionais.

## III. Algoritmo FastICA

O algoritmo escolhido para a implementação deste trabalho foi o FastICA [11], [12], especificamente rápido e simples, baseado na técnica de Análise de Componentes Independentes, descrita a seguir.

### A. Análise de Componentes Independentes

Seja um vetor de observação aleatório,

$$\mathbf{X} = [X_1, X_2, ..., X_n]^t \tag{1}$$

composto por n componentes provenientes da combinação linear de um vetor fonte aleatório com componentes estatisticamente independentes,

$$\mathbf{U} = [U_1, U_2, ..., U_n]^t \tag{2}$$

e uma matriz de mistura $A_{nxn}$, não singular, onde $n$ é a quantidade de amostras utilizadas.

$$\mathbf{X} = A \cdot \mathbf{U} \tag{3}$$

Sabendo-se que $a_{ij}$ é um elemento da matriz A e $u_j$ uma variável independente, temos que:

$$x_i = a_{i1} \cdot u_1 + a_i 2 \cdot u_2 + ... + a_i n \cdot u_n, \forall i \tag{4}$$

Assume-se que as variáveis observadas e as componentes independentes possuem média igual a zero [13]. Se isso não for verdadeiro, subtrai-se do vetor de misturas a média e garante-se, assim, que possui média zero. A separação cega de fontes consiste então em encontrar uma matriz W tal que o vetor original possa ser recuperado. Combinando a matriz de separação com as variáveis misturadas obtemos o sinal de saída Y, que é uma aproximação de U.

$$\mathbf{Y} = \mathbf{W} \cdot \mathbf{X} \tag{5}$$

onde,

$$\mathbf{Y} = [Y_1, Y_2, ..., Y_n]^t \tag{6}$$

Na Figura 3 podemos observar um diagrama de blocos que representa o problema da separação cega de fontes e o uso da técnica de Análise de Componentes Independentes para solucioná-lo.

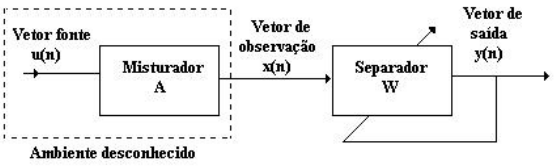


Fig. 3. Diagrama de blocos da técnica ICA

O ICA possui algumas restrições [12] que devem ser consideradas para que forneça bons resultados. A primeira restrição é que as componentes devem ser estatisticamente independentes, ou seja, a informação sobre o valor de uma componente não deve depender da informação sobre o valor de outra componente. Nesta técnica, a não-correlacionalidade não é suficiente para separar as componentes independentes. O fato de duas variáveis, x e y, serem estatisticamente independentes, implica que sua correlação é igual a zero, mas o contrário nem sempre é verdadeiro. Por isso, algumas técnicas, como o PCA (*Principal Component of Analysis*) que garantem apenas a não-correlacionalidade, não são capazes de separar os sinais.

A segunda restrição requer que as componentes independentes não sejam gaussianas, uma vez que, se isso acontecer, as variáveis observadas também serão gaussianas. A função de densidade de probabilidade de variáveis gaussianas é simétrica não contendo nenhuma informação sobre a direção das colunas da matriz A, e por esta razão a matriz A não pode ser estimada utilizando este método [11].

Os algoritmos ICA são baseados na medida de não-gaussianidade das componentes independentes e, além de descorrelacionarem os sinais (estatística de segunda ordem), também reduzem a dependência de estatísticas de ordens superiores.

### B. Treinamento do algoritmo FastICA

O FastICA tem como objetivo encontrar uma matriz W de separação em que, $\mathbf{W}^T\mathbf{X}$ maximiza a não-gaussianidade das componentes resultantes.

Neste trabalho, a medida de não-gaussianidade utilizada foi a negentropia, que é baseada na quantidade de informação de uma variável. De fato, devido à complexidade de implementação da definição matemática rigorosa da negentropia, foi tomada uma aproximação dessa medida.

O algoritmo tanto pode se basear no esquema de iteração de ponto fixo como também na iteração de Newton. A regra de aprendizagem do FastICA [12], baseada no modelo neural [14] representado pelo vetor de pesos W, é descrita da seguinte forma:

1) Obtém-se, aleatoriamente, um vetor W inicial;
2) $\mathbf{W}^+ = E\{\mathbf{X} \cdot g(\mathbf{W}^t \cdot \mathbf{X})\} - E\{g'(\mathbf{W}^t \cdot \mathbf{X})\} \cdot \mathbf{W}$;
3) $\mathbf{W} = \mathbf{W}^+ / ||\mathbf{W}^+||$;
4) Repetir passo 2 até que haja convergência.

Neste caso, convergir significa que o valor atual e o anterior de W, tendem para a mesma direção, ou seja, o produto interno entre eles é aproximadamente igual a 1 [15].

A função não quadrática utilizada no algoritmo e sua derivada foram, respectivamente:

$$G(u) = 1/a_1 \cdot \log\cosh(a_{1u}) \tag{7}$$

$$g(u) = \tanh(a_{1u}) \tag{8}$$

Antes de iniciar o treinamento do FastICA foi efetuado um pré-processamento que consistiu na aplicação de duas técnicas: a centralização e o branqueamento. O objetivo foi a simplificação do algoritmo de treinamento para acelerar a sua convergência. A centralização é uma técnica, aplicada às variáveis observadas, que simplesmente subtrai a sua média. Se for necessário, este valor pode ser devolvido às variáveis de saída mais tarde. Já o branqueamento, aplicado após a centralização, transforma o vetor inicial em um vetor branco onde suas componentes são descorrelacionadas e suas variâncias iguais a 1.

Este processamento modifica os valores de entrada do algoritmo de treinamento, mas não altera as características relevantes da saídas, porque são transformações lineares que,

portanto, não provocam mudanças nas componentes independentes.

## IV. Experimento

### A. Simulação de sinais de entrada

Os sinais de entrada para a filtragem foram simulados. Arquivos de dados formatados, contendo amostras de uma senóide e um ruído uniformemente distribuído, ilustrados na Figura 4, foram lidos e transformados em sinais de tensão com auxílio do software LABVIEW e uma placa de aquisição de dados.

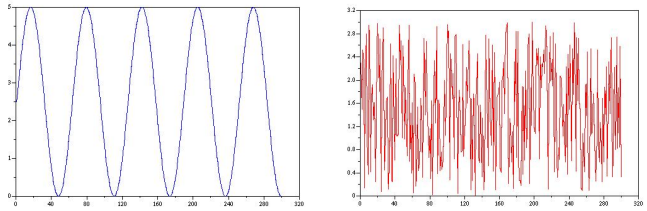

Fig. 4. Sinais Originais.

A placa de aquisição está ligada à rede Fieldbus através de uma interface capaz de converter níveis de tensão em níveis de corrente 4 a 20mA, implementada com o XTR 115U da Texas Instruments INC, como mostrado na Figura 5, e instrumentos de campo IF-302 que convertem níveis de corrente 4 a 20mA em sinais Foundation Fieldbus.

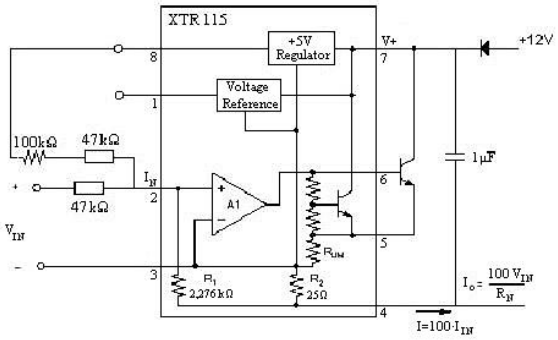


Fig. 5. Conversor tensão/corrente

Os sinais simulados foram misturados, aleatoriamente, como visto na Figura 6, através da matriz A, dada abaixo.

$$A = \begin{bmatrix} -0.815345 & -0.91101 \\ 0.452653 & -0.43908 \end{bmatrix} \quad (9)$$

Na etapa seguinte foi feito o treinamento off-line do algoritmo FastICA, como já detalhado na subseção "Treinamento do algoritmo FastICA", obtendo-se, após a convergência do algoritmo, a seguinte matriz de separação que foi utilizada para o experimento nos blocos funcionais:

$$W = \begin{bmatrix} 0.560568 & -0.603342 \\ 0.62634 & 0.58525 \end{bmatrix} \quad (10)$$

Com a matriz de separação, gerou-se o vetor de saída Y, constatando-se que o processo de separação dos dados simulados foi realizado com sucesso, Figura 7.

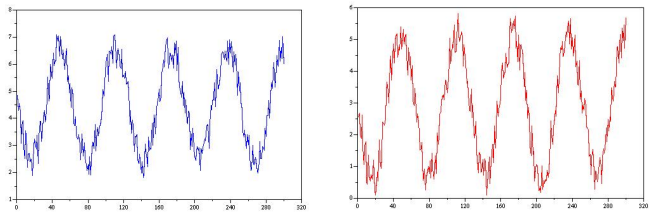

Fig. 6. Sinais Misturados.

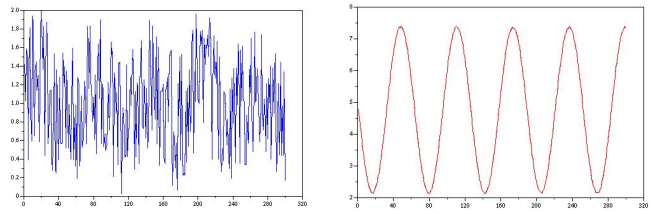

Fig. 7. Sinais Separados.

### B. Implementação do algoritmo FastICA por meio de blocos funcionais

Como citado anteriormente o trabalho propõe a utilização de um algoritmo, baseado na técnica de análise de componentes independentes, e sua implementação através de blocos funcionais. Esses blocos funcionais foram instanciados em instrumentos de campo do tipo conversores corrente/*Fieldbus* (IF-302), fabricado pela SMAR. Os blocos empregados foram o AI (Analog Input) e o ARITH (Aritmético).

Por ser um algoritmo neural, a arquitetura do FastICA, utilizada neste trabalho, foi formada por dois neurônios na camada de entrada, dois neurônios na camada de saída e função de ativação linear. Cada neurônio da camada de saída foi implementado por um bloco funcional do tipo ARITH, enquanto que os neurônios da camada de entrada foram implementados por blocos AI.

Os blocos AI convertem as entradas simuladas, provenientes da placa de aquisição de dados e lidas pelos transdutores, para as representações digitais usadas pelos blocos funcionais FF. Estes blocos fornecem também um estado correspondente a este valor, contendo uma informação sobre a sua confiabilidade. Adicionalmente, através da configuração do bloco AI, o usuário pode limitar a sua saída para uma faixa de valores considerada segura, filtrando valores fora desta faixa e colocando-os em escala conveniente para o processamento pelo bloco seguinte. A Figura 8 mostra o esquema geral do bloco AI e os seus principais parâmetros configuráveis.

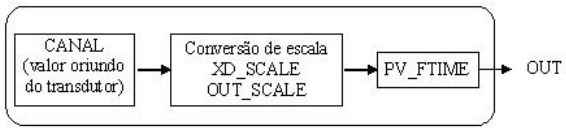


Fig. 8. Bloco AI

O Bloco ARITH pode facilmente implementar um neurônio artificial cuja função de ativação seja linear. Este objeto FF

possui três entradas e uma saída. A função a ser realizada pode ser escolhida entre diversas opções, tais como a média aritmética, raiz quadrada do produto das entradas, o quociente entre duas delas ou outras funções mais específicas do ambiente de controle. Neste trabalho, escolheu-se a soma tradicional, que efetua a soma do produto das entradas por coeficientes configurados pelo usuário. Os coeficientes, que são neste caso os elementos da matriz W, exercem o papel dos pesos sinápticos do neurônio. A Figura 9 mostra os principais detalhes do bloco aritmético usados nesta aplicação.

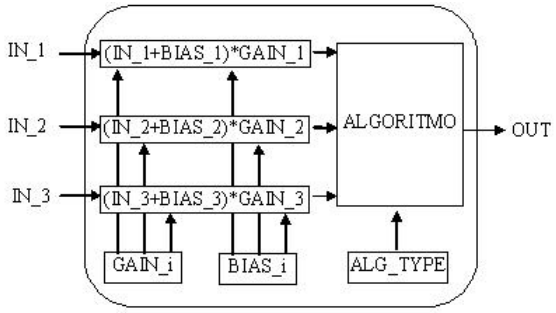


Fig. 9. Bloco ARITH

O algoritmo neural foi implementado conforme a Figura 10. Os blocos aritméticos recebem dois sinais de entrada provenientes de cada bloco AI e fornecem as saídas do algoritmo FastICA. A terceira entrada de cada ARITH foi desprezada. A operação realizada pelos dois neurônios, corresponde à multiplicação da matriz W, obtida do treinamento off-line, pelo vetor de entrada.

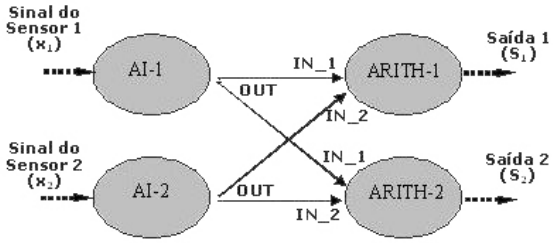


Fig. 10. Modelo ICA através de blocos funcionais

Cada Bloco ARITH realiza as operações de multiplicação e soma correspondentes ao somatório dos produtos dos elementos da linha "i" da matriz W pelos elementos do vetor de sinais de entrada, resultando no elemento da linha "i" do vetor dos sinais de saída. Os dois blocos ARITH foram instanciados em um dos IF-302 empregados no experimento, enquanto os blocos AI foram instanciados um em cada IF para possibilitar a entrada simulada proveniente das placas de interface com o LABVIEW.

A Figura 11 mostra as ligações dos equipamentos utilizados no experimento:

## V. Resultados

As entradas descritas na subseção "Simulação de sinais de entrada" e tratadas pela implementação do algoritmo nos

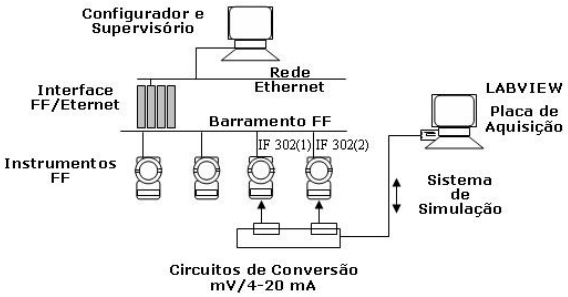


Fig. 11. Arranjo Experimental

instrumento FF, de acordo com a subseção "Implementação do algoritmo FastICA por meio de blocos funcionais", produziram as saídas que foram capturadas por um supervisório instalado nas estações de trabalho da rede, e podem ser visualizadas na Figura 12.

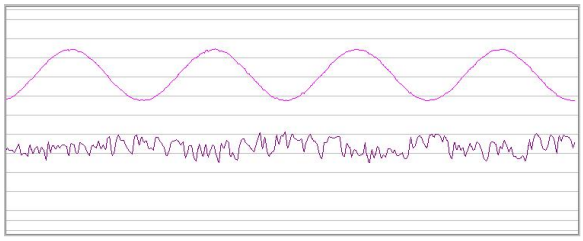

Fig. 12. Sinais Separados

Percebe-se que o ruído, adicionado às senoidais na entrada quase desaparece totalmente nas curvas de saída, verificando-se que o sistema apresentou bons resultados completando com êxito o processo de separação cega dos sinais.

É importante realizar uma avaliação do desempenho do separador e isso pode ser feito através de uma medida quantitativa do ruído usando um índice de rejeição global definido por [2]. Com esta medida é possível avaliar, por exemplo, quando um sensor, que teve seu sinal filtrado, necessita de calibração.

A implementação do algoritmo FastICA em blocos funcionais traz a grande vantagem da robustez, uma vez que ao ser implementado em um dispositivo interno da rede, BF, o sistema está menos propício a adição de ruídos no processo.

Porém, a interface no ambiente FF é feita através da camada de aplicação do usuário, utilizando os BF que são objetos pré-definidos. Com isso, a estrutura fixa desses blocos não permite a implementação de algoritmos mais complexos nos dispositivos de campos o que favorece à implementação da técnica em hardware dedicado inserido na rede FF [16]. Assim, a implementação de aplicações ou algoritmos mais complexos, como por exemplo um treinamento on-line do FastICA, tornaria-se possível neste ambiente.

Após os testes e resultados alcançados, os quais validaram experimentalmente o algoritmo proposto e a sua implementação no ambiente FF, propõe-se uma nova etapa de testes com sinais reais coletados de sensores da rede FF.

## VI. Conclusões

A descrição e o funcionamento de um sistema capaz de realizar separação cega de sinais utilizando um algoritmo inteligente baseado na técnica de Análise de Componentes Independentes, bem como sua implementação, interagindo diretamente com dispositivos de redes de campo Foundation Fieldbus, através de blocos funcionais, foi apresentado neste trabalho.

Os resultados obtidos foram satisfatórios e condizentes com o objetivo, uma vez que sinais ruidosos provenientes de uma planta simulada, foram perfeitamente filtrados, separados e interagiram com a rede de campo.

A técnica de Análise de Componentes Independentes mostrou-se eficiente na separação cega de fontes e quando implementada em redes de campos apresentou bons resultados. A capacidade de extração de ruído apresentada por essa técnica viabiliza a implementação em redes de sensores sujeitos a ruídos.

## Referências


[1] P. Smaragdis, *Blind Separation of Convolved Mixtures in The Frequency Domain*, International Workshop on Independence Artificial Neural Networks University of La Laguna, 1998.

[2] S. Amari, *Superefficiency in Blind Source Separation*, IEEE Transaction on Signal Processing, 1998.

[3] S. Amari; T. Chen and A. Cichocki, *Stability Analysis of Adaptive Blind Source Separation*, IEEE Transactions on Neural Networks, 1997.

[4] J. Cardoso, *The Invariant Approach to Source Separation*, NOLTA'95, 1995.

[5] G. Y. Tian; Z. X. Zhao and R. W. Baines, *A Fieldbus-Based Intelligent Sensor*, In: mechatronics, 1999.

[6] J. Berge, *Fieldbuses for Process Control: Engineering, Operation and Maintenance, The Instrumentation, Systems and Automation*, Society-ISA, 2001.

[7] FIELDBUS FOUNDATION FF581F12, *Foundation Specification - System Architecture*, Austin, Texas, 2001.

[8] FIELDBUS FOUNDATION FF-890F15, *Foundation Specification - Function Block Application Process - Part 1*, Austin, Texas, 2001.

[9] FIELDBUS FOUNDATION FF-891F15, *Foundation Specification - Function Block Application Process - Part 2*, Austin, Texas, 2001.

[10] FIELDBUS FOUNDATION FF-892F15, *Foundation Specification - Function Block Application Process - Part 3* Austin, Texas, 2001.

[11] A. Hywarinen, *Fast and Robust Fixed Point Algorithms for Independent Component Analysis*, IEEE Transactions on Neural Network, 1999.

[12] A. Hywarinen and E. Oja, *Independent Component of Analysis: Algorithms and aplications*, 1999.

[13] S. Haykin, *Redes Neurais: Princípios e Prática*, Bookman, 2001.

[14] J. Karhunen, *Neural Approaches to Independent Component Analysis And Source Separation*, 4th European Symposium on Artificial Neural Networks, 1996.

[15] E. Oja and O. Taipale, *Applications Off-Learning and Intelligent Systems*, Int. Conf. on Artificial Neural Networks, 1995.

[16] I. Costa, A. Duarte, J. Melo, J. Oliveira, *Embedded FastICA Algorithm Applied to the Sensor Noise Extraction Problem of Foundation Fieldbus Network*, Int. Joint Conference on Neural Network, 2005.