UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
ESCOLA DE ENGENHARIA
DEPARTAMENTO DE ENGENHARIA QUÍMICA
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA QUÍMICA

DESENVOLVIMENTO DE UM SISTEMA INTEGRADO DE AUDITORIA E COMPRESSÃO DE DADOS PARA PLACAS DE BAIXO CUSTO

DISSERTAÇÃO DE MESTRADO

Wilson Roberto Barbutti Filho

Porto Alegre

2014

UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL

ESCOLA DE ENGENHARIA

DEPARTAMENTO DE ENGENHARIA QUÍMICA

PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA QUÍMICA

DESENVOLVIMENTO DE UM SISTEMA INTEGRADO DE AUDITORIA E COMPRESSÃO DE DADOS PARA PLACAS DE BAIXO CUSTO

Wilson Roberto Barbutti Filho

Dissertação de Mestrado apresentada como requisito parcial para obtenção do título de Mestre em Engenharia

Área de concentração: Desenvolvimento de Processos

Linha de Pesquisa: Engenharia de Sistemas – Projeto, Modelagem, Controle e Otimização de Processos.

Orientadores:
Prof. Dr.Pedro Rafael Bolognese Fernandes
Prof. Dr.Jorge Otávio Trierweiler

Porto Alegre

2014

UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL

ESCOLA DE ENGENHARIA

DEPARTAMENTO DE ENGENHARIA QUÍMICA

PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA QUÍMICA

A Comissão Examinadora, abaixo assinada, aprova a *Desenvolvimento de um Sistema Integrado de Auditoria e Compressão de Dados para Placas de Baixo Custo*, elaborada por *Wilson Roberto Barbutti Filho*, como requisito parcial para obtenção do Grau de Mestre em Engenharia.

Comissão Examinadora:

Prof. Dr. Alexandre Arruda - Unipampa

Prof. Dr. André Rodrigues Muniz - DEQUI/UFRGS

Profª. Drª. Luciane Ferreira Trierweiler – DEQUI/UFRGS

Prof. Dr. Luiz Muniz – UCS e UpControl

Resumo

Um dos problemas para a realização de produções descentralizadas está nos equipamentos necessários para as plantas. Grandes sistemas necessitam um investimento muito alto além de pessoal especializado para operá-los. Com a finalidade de facilitar o uso de sistemas de pequeno porte, os algoritmos devem ser modificados para possibilitar a operação dos mesmos com poder de processamento limitado. Outro fator é a transmissão de dados, como ha possibilidade de sistemas em locais afastados, a compressão dos dados se torna muito importante. O objetivo deste trabalho é o de desenvolver um sistema integrado de compressão e auditoria para a utilização em uma placa processadora de baixo custo. O algoritmo tem base no Swinging Door Trending, o qual gera os parâmetros para a auditoria. Para teste de desempenho foi utilizado um Computador PC e o algoritmo em uma placa Raspberry Pi. A avaliação foi feita com dados históricos de planta e simulados, contando com compactação, auditoria, estabilidade do algoritmo e *outliers*. Os dados históricos compreendem diversos tipos de sensores com diversas dinâmicas.

Abstract

One of the problems to the realization of decentralized productions lies on the required plant equipment. Large systems need a very high investment beyond having specialized personnel to operate them. With the aim of facilitating usage on small scale systems, the algorithms must be modified to enable the operations of them on limited processing power. Another important issue is data transfer, since there is the possibility of systems on remote locations, data compression becomes very important. The objective of this work is the development of a joint compression and audition to usage on a low cost processor board. The algorithm is based on the Swinging Door Trending, which generates the parameters to the audition. To the performance test a PC Computer was used and the algorithm was implemented in a Raspberry Pi board. The evaluation was made with historical and simulated plant data, comprising compression, auditing, and outliers. The historical data comprises diverse sensor types and multiple dynamics.

“The voice of the intellect is a soft one,
but it does not rest until it has gained a hearing.”
- Sigmund Freud

Agradecimentos

Agradeço à minha família, por toda a compreensão que tiveram durante esses duros anos. À minha mãe pelo apoio que me ajudou a seguir em frente. A meu pai que sempre acreditou que eu tivesse capacidade para seguir até o fim. A meu irmão pelos finais de semana que me acolheu em sua casa e me ajudou em muitas coisas.

Aos meus orientadores, Prof. Dr.Pedro Rafael Bolognese Fernandes e Prof. Dr.Jorge Otávio Trierweiler que me deram o rumo que eu precisava para poder concluir o mestrado.

À Universidade Federal do Rio Grande do Sul que me acolheu por dias longos durante esses anos e por propiciar tudo que precisei para dar andamento aos meus trabalhos.

Aos meus amigos deixo meus agradecimentos por me compreenderem às muitas vezes que não pude me fazer presente.

Por fim agradeço a todos que de uma forma ou outra estiveram presentes durante a realização deste trabalho.

Lista de Figuras

Lista de Tabelas

Abreviações

ADC	*Analog to Digital Converter*
ALICE	*A Large Ion Collider Experiment*
BC	*Box-Car*
BS	*Backward Slope*
BWT	*Burrows–Wheeler Transform*
CAN	*Controller Area Network*
CMTU	*Charge Time Measurement Unit*
CPU	*Central Processing Unit*
DMA	*Direct Memory Access*
DGA	*Dissolved Gas Analysis*
EEG	Eletro-Encefalo-Grafia
EEPROM	*Electrically-Erasable Programmable Read-Only Memory*
EPROM	*Erasable Programmable Read-Only Memory*
FDI	*Failure Detection and Identification*
GPIO	*General Purpose Input/Output*
GPU	*Graphics Processing Unit*
HDMI	*High-Definition Multimedia Interface*
HLT	*High Level Trigger*
HMI	*Human Machine Interface*
I/O	*Input/output*
I²C	*Inter-Integrated Circuit*
I²S	*Integrated Interchip Sound*
JPEG	*Joint Photographic Experts Group*
JTAG	*Joint Test Action Group*
KPCA	*Kernel Principal Component Analysis*
LHC	*Large Hadron-Collider*
LZMA	*Lempel-Ziv-Markov Algorithm*
MDCT	*Modified Discrete Cosine Transform*
MIPS	*Million Instructions Per Second*
MKPCA	*Multiway Kernel Principal Component Analysis*
MSE	*Minimum Squared Error*
OTG	*On-The-Go*
PCA	*Principal Component Analysis*
PIC	*Programmable Intelligent Computer*
PID	*Proportional Integral Derivative*
PLC	*Programmable Logic Controller*
PLOT	*Piecewise Linear Online Trending*

PWM	*Pulse Width Modulation*
RAM	*Random Access Memory*
RCA	*Radio Corporation of America*
RISC	*Reduced Instruction Set Computer*
ROM	*Read-Only Memory*
RTC	*Real-Time Clock*
RTU	*Remote Terminal Unit*
SCADA	*Supervisory Control and Data Acquisition*
SD	*Secure Digital*
SDT	*Swinging Door Trending*
SLIM	*Straight Line Interpolative Method*
SPE	*Squared Prediction Error*
SPI	*Serial Peripheral Interface*
SRAM	*Static Random Access Memory*
SVD	*Singular Value Decomposition*
TKST	*Two-sample Kolmogorov-Smirnov Test*
TRS	*Tip-Ring-Sleeve*
UART	*Universal Asynchronous Receiver/Transmitter*
USB	*Universal Serial Bus*
VA	*Volt-Ampere*
VAC	*Voltage Alterned Current*
VDC	*Voltage Direct Current*

Notação e Simbologia

α	Ângulo gerado pelo algoritmo SDT
α_{S-1}	Ângulo SDT de um sensor anterior da linha
α_S	Ângulo SDT de um sensor de uma linha
α_{S+1}	Ângulo SDT de um sensor posterior da linha
Dir	Direção de ação do PID
n	Intervalo de analise de *outlier*
i	Intervalo de tolerância
M_{PID}	Multiplicador atualização de parâmetros PID
M_{LIM}	Multiplicador para tolerância máxima de falha
x	Ponto atual do processo
s_p	Set-Point
t	Tempo
T_{SDT}	Tolerância do algoritmo SDT

Sumário

Capítulo 1 - Introdução

Atualmente, diversos meios de produção operam através de um modelo centralizado, no qual plantas de grande capacidade são responsáveis pelo processamento da matéria-prima. Este sistema de produção tem como principal vantagem, a diminuição dos investimentos para a implantação, assim como o custo em automação e manutenção de equipamentos. Contudo este modelo requer um grande esforço em termos de estrutura, tanto para o recebimento da matéria-prima quanto para o escoamento do produto final. Com os custos relacionados a essa logística aumentando cada dia, por causa do aumento de acidentes viários bem como degradação das rodovias, dentre outros, o produto final é encarecido. Além disso, outro agravante para empresas com matéria-prima de origem vegetal, está no processamento, que deve ser feito logo após a colheita, onde o dimensionamento vinculado ao pico de produção leva a uma ociosidade na entressafra.

Há algum tempo, certas empresas têm destinado esforços para transformar seus meios produtivos no modelo descentralizado visando o corte de custos e a proximidade com seus mercados. A Toyota, por exemplo, tem adotado esse sistema no mundo. A fábrica brasileira, em São Bernardo do Campo, produz partes de veículos Corolla, mas também temos unidades produzindo partes do Corolla no exterior, como no Canadá e até Tailândia (*TOYOTA, 2012*). A produção descentralizada tem a vantagem de conferir tanto aos mercados quanto às empresas agilidade no recebimento e distribuição dos produtos. Em casos como o de discos rígidos de computadores, que possuem um modelo centralizado, requerem que você envie os produtos para a fábrica e muitas vezes os encargos desse envio ficarão sobre responsabilidade do comprador.

Para que um modelo descentralizado seja viável economicamente, cada unidade deve ser capaz de operar autonomamente, contudo manter todas as funções de controle e manutenção em todas as unidades pode elevar muito o custo de

operação para uma mesma escala de produção. Com isso devem-se projetar estratégias de operação para essas unidades. Nesse tipo de produção deseja-se manter maior parte das funções essenciais e ao mesmo tempo mantendo o gerenciamento fora da unidade com a finalidade de reduzir os custos, uma dificuldade para esse sistema é o mesmo requerer que mão-de-obra especializada seja mantida no local, por isso os custos podem não compensar. Para o caso da indústria de etanol, que possui uma produção de matérias geograficamente diversificada, se faz necessário um nível elevado de automação, de forma que os custos com pessoal em áreas remotas possam ser reduzidos, sem redução da qualidade e produtividade, ao mesmo tempo permitindo que pessoas sem um nível elevado de instrução possam operar a planta.

Com a finalidade de manter unidades funcionando em locais afastados, deve-se considerar que qualquer tomada de decisão da unidade central de monitoramento pode ter um elevado atraso de tempo até chegar às unidades controladas. Com isso, se quisermos manter unidades autônomas, precisamos de dispositivos automáticos para atuação na planta, porém os mesmo devem ser capazes de sofrer intervenções externas, por exemplo, para alteração de parâmetros operacionais. Durante um evento crítico o sistema também deve ser capaz de tomar ações que permitam a segurança da operação da planta, bem como sua desativação se necessário.

Uma das maneiras de se fazer isso é através da utilização de estratégias de controle baseados em sistemas computacionais de baixo custo, um dos empecilhos para esses sistemas está no fato de que possuem uma limitada capacidade computacional, sendo assim não podem realizar o mesmo numero de operações que uma unidade centralizada faria. Sendo assim, o desenvolvimento de novas técnicas com foco a permitir que esse trabalho possa ser feito nos sistemas descentralizados.

Outro fator é que plantas descentralizadas devem utilizar sistemas de envio de dados para uma unidade central, os quais muitas vezes tem uma capacidade de transmissão insatisfatória. Deste modo, é necessário o emprego de métodos para a redução da utilização da largura dessa banda de dados. Para tanto, indústrias tradicionais utilizam-se normalmente de sistemas de compressão de dados a fim de reduzir a necessidade de transmissão, processamento e armazenamento dos mesmos. Tais métodos são normalmente baseados em algoritmos com perda de informação, como no caso do *Swinging Door Trending* (SDT) utilizado, por exemplo, em sistemas industriais PI da OSISoft (*OSISOFT, 2003*).

Ao longo deste trabalho, será avaliada a criação de um sistema integrado de compactação e auditoria para dados de planta, o qual será baseado nos parâmetros

de um algoritmo SDT. Para isso será feita uma análise dos algoritmos, dispositivos e técnicas de auditoria disponíveis para integração. Esse algoritmo, criado na linguagem de programação Python, será testado em um Computador PC para permitir uma avaliação mais precisa da estabilidade e desempenho computacional, e em uma placa Raspberry Pi para avaliar o funcionamento do algoritmo. A avaliação conta com dados reais e simulados de plantas, contando com sensores de diversas dinâmicas. Os testes compreendem desempenho e estabilidade computacional, fidelidade de compactação e auditoria finalizando com análise de *outliers*.

1.1 Estrutura da Dissertação

No Capítulo 2, é apresentada a revisão bibliográfica a respeito das tecnologias de compressão de dados existentes, juntamente com suas vantagens e desvantagens. Também são apresentados os sistemas de detecção de falhas, assim como o hardware de controle atualmente disponível, que pode terminar por limitar a complexidade dos algoritmos que são implementados.

No Capítulo 3, é apresentada as escolhas dos algoritmos e equipamentos, juntamente com os conceitos da auditoria de falhas e também a forma pela qual o sistema e compressão SDT pode ser usado como base para a tomada de ações da planta. O objetivo é a utilização em um sistema de controle de baixo custo e alta robustez de forma a operar uma pequena usina de forma autônoma, finalizando com a apresentação do funcionamento do algoritmo.

No Capítulo 4 serão apresentados os resultados obtidos com o algoritmo proposto através de dados históricos reais de planta durante operação normal. Para a análise de estabilidade e desempenho são simulados quatro meses de operação da planta para observar o desempenho, assim como estabilidade em termos de memória, uso de processador e gravação de dados. Na compressão do SDT foram utilizados parâmetros para obter cerca de 90% de compactação, neste as curvas geradas são avaliadas juntamente com as originais utilizando o algoritmo TKST. Além disso, será avaliada a detecção de *outliers* da auditoria e capacidade de atuação autônoma.

Finalizando, no Capítulo 5 se faz a avaliação da viabilidade do sistema proposto, bem como se apresentada sugestões para alterações futuras do sistema, a fim de aperfeiçoar o sistema de auditoria.

No Apêndice A se demonstra a planta piloto na qual o algoritmo será utilizado para testes em situações reais, atendendo às proposições do trabalho.

No Apêndice B se mostra uma proposta para trabalho futuro utilizando um algoritmo PID modificado os parâmetros da compressão SDT.

Capítulo 2 - Revisão Bibliográfica

Para desenvolvermos um método de auditoria devemos fornecer dados e parâmetros. Como o sistema proposto deve operar em uma placa controladora de baixo custo, o uso dos dados de um algoritmo de compressão pode ajudar a reduzir o processamento necessário. Uma vez que a compactação dos dados seria necessária para a comunicação com um escritório central.

2.1 Compressão de Dados

O estudo dos métodos de redução de dados é relativamente antigo. Pode-se dizer que o primeiro a estudar matematicamente a compressão de dados foi Claude Elwood Shannon que em 1948 publicou "A mathematical theory of communication" (*SHANNON, 1948*).

A compressão de dados consiste na redução do volume de dados através de algoritmos. Essa redução pode ser classificada em dois tipos: a redução reversível, na qual todos os dados podem ser refeitos, e também a compressão com perda de informações na qual não é possível refazer integralmente os dados (*SAID, 1996*).

2.1.1 Algoritmos sem Perda de Dados

A compressão sem perda consiste em criar um modelo que represente exatamente a fonte original, usualmente este tipo de técnica é usado onde se requer total integridade dos dados fornecidos, tais como na compressão de textos, uma vez que a falta de caracteres pode mudar totalmente o texto (*FOWLER, 1994*). Um dos métodos mais clássicos é o método de codificação entrópica baseada em contexto, uma vez que, normalmente os dados possuem correlações. Deste modo, reúnem-se amostras que podem ser agrupadas de forma estatisticamente homogênea através de contextos espaciais ou espectrais, com isso o modelo final terá uma entropia reduzida, permitindo que o texto sofra uma maior compressão. Esse método pode ainda ser aplicado em situações onde possa ser admitida uma

compressão quase sem perda (*MAGLI, 2004*). Como o objetivo desse tipo de compressão é reduzir pacote de dados ao número mínimo de bits para representar o objeto, a entropia deve ser reduzida. Contudo, para isto, é necessária a execução de tarefas dispendiosas, pois geralmente empregam-se algoritmos de otimização com base nas equações de Wiener-Hopf. Uma alternativa seria o uso de algoritmos baseados em agrupamentos, em que cada bloco seria classificado em um número finito de classes, de modo a haver preditores específicos para cada espécie de elemento. Assim, para a determinação de um elemento X usamos os elementos vizinhos para a predição (*GOLCHIN, 1997*), conforme a Figura 2.1.

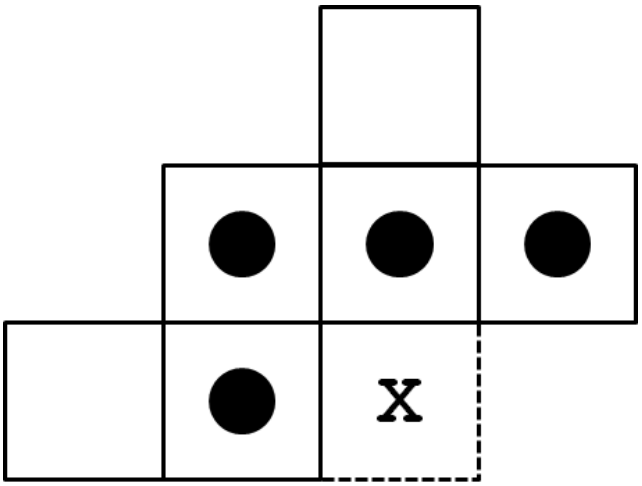

Figura 2.1 – Vizinhança Causal do Pixel X, onde os círculos marcam os pixels usados para predição do X. Adaptado de (*GOLCHIN, 1997*).

Essas técnicas têm sido empregadas para a compressão de imagens, em situações nas quais não se deseja a perda de definição, tais como em imagens multiespectrais como aquelas do projeto LANDSAT, que possuem somente sete bandas de espectro. Geralmente essas espécies de compressão reduzem um a dois bits de informação, porém podem chegar a até cerca de cinco bits em cada pixel (*MEMON, 1994*). No entanto, a redução do volume de informações pela compressão deve ser capaz de fornecer as informações suficientes para alimentar algoritmos de cálculo matemáticos quando os utilizamos, por exemplo, em uma aquisição de um EEG. Para isso, normalmente são usados variantes da técnica Lempel-Ziv para a geração de compressão sem perda de dados (*MEMON, 1999*). Uma variante desse modelo é muito usada em aplicativos computacionais, o LZMA presente em softwares tais como o WinZip® desde sua versão 12(*WINZIP, 2012*). O open-source 7zip conta com esse algoritmo desde sua versão inicial (*7-ZIP, 2012*). Esse algoritmo é uma variante do algoritmo Lempel-Ziv de 1978, que se utiliza da criação de um dicionário. Esse dicionário é feito através de dados prévios encontrados no arquivo conforme o mesmo é varrido. Após isso, as sequências são substituídas pelas palavras desse dicionário, que é mantido junto ao arquivo, por

serem menores que as sequências originais, essas palavras conseguem reduzir o volume de dados a serem escritos e a reconstrução é feita pelo processo inverso (*ZIV, 1978*). Outro método possível para realizar a compressão sem perda de informações é o BWT, o qual é baseado em permutações das palavras criando blocos que sejam de fácil compressão. A Tabela 2.1 mostra como se comporta a permutação.

Tabela 2.1 - Permutações feitas pelo Algoritmo BWT, onde M é a palavra base e M' a palavra permutada. Adaptado de (*BURROWS, 1994*)

Linha	M	M'
0	aabrac	caabra
1	abraca	aabrac
2	acaabr	racaab
3	bracaa	abraca
4	caabra	acaabr
5	racaab	bracaa

A eficiência do método está no fato de que a mesma produz diversas sequências de caracteres repetidos (*BURROWS, 1994*).

Com isso, ao se tomar uma sequência como "BANANA", tem-se primeiramente que definir os caracteres de início e parada como, por exemplo, "^" e "|" fazendo o primeiro sequenciamento temos:

Tabela 2.2 - Sequencia de permutações do algoritmo BWT para a palavra BANANA

Permutação	String
0	^BANANA|
1	BANANA|^
2	ANANA|^B
3	NANA|^BA
4	ANA|^BAN
5	NA|^BANA
6	A|^BANAN
7	|^BANANA

Após isso, se faz o ordenamento em relação às palavras e definem-se os últimos caracteres como a nova sequência, conforme mostrado na Tabela 2.3:

Tabela 2.3 - Resultado final do Sequenciamento da palavra

Permutação	String
2	ANANA|^B
4	ANA|^BAN
6	A|^BANAN
1	BANANA|^
3	NANA|^BA
5	NA|^BANA
0	^BANANA|
7	|^BANANA

Ao fim temos a sequencia "BNN^AA|A" com os caracteres N e A repetidos em sequência tornando-a mais fácil de ser comprimida. Para realizar-se a descompressão realizam-se adições da palavra e ordenamentos chegando-se à palavra original, o processo é demonstrado na Tabela 2.4 onde os passos são mostrados da direita para a esquerda, seguindo as sequências das linhas.

Tabela 2.4 - Passos para reconstrução da palavra original do algoritmo.

Adição	Sequenciamento	Adição	Sequenciamento
B	A	BA	AN
N	A	NA	AN
N	A	NA	A|
^	B	^B	BA
A	N	AN	NA
A	N	AN	NA
|	^	|^	^B
A	|	A|	|^
BAN	ANA	BANA	ANAN
NAN	ANA	NANA	ANA|
NA|	A|^	NA|^	A|^B
^BA	BAN	^BAN	BANA
ANA	NAN	ANAN	NANA
ANA	NA|	ANA|	NA|^
|^B	^BA	|^BA	^BAN
A|^	|^B	A|^B	|^BA
BANAN	ANANA	BANANA	ANANA|
NANA|	ANA|^	NANA|^	ANA|^B
NA|^B	A|^BA	NA|^BA	A|^BAN
^BANA	BANAN	^BANAN	BANANA
ANANA	NANA|	ANANA|	NANA|^
ANA|^	NA|^B	ANA|^B	NA|^BA
|^BAN	^BANA	|^BANA	^BANAN
A|^BA	|^BAN	A|^BAN	|^BANA
BANANA|	ANANA|^	BANANA|^	ANANA|^B
NANA|^B	ANA|^BA	NANA|^BA	ANA|^BAN
NA|^BAN	A|^BANA	NA|^BANA	A|^BANAN
^BANANA	BANANA|	^BANANA|	BANANA|^
ANANA|^	NANA|^B	ANANA|^B	NANA|^BA
ANA|^BA	NA|^BAN	ANA|^BAN	NA|^BANA
|^BANAN	^BANANA	|^BANANA	^BANANA|
A|^BANA	|^BANAN	A|^BANAN	|^BANANA

Após o último passo a palavra começando com o caractere de início é tomada, neste caso será "^BANANA|" onde será interpretada como BANANA.

Um dos problemas da compressão sem perdas está na dificuldade em se conseguir realizar o processo em tempo real. Métodos de compressão em palavra tais como o BWT não é possível em tempo real, uma vez que. Para se montar um bloco de dados seria necessário adicionar uma latência muito grande nas ações, assim para isso devem-se usar métodos estatísticos como o Lempel-Ziv que se utiliza de preditores para definir a probabilidade do próximo ponto, é possível usar técnicas especiais para métodos baseados em dicionário, contudo isso requer um processamento computacional adicional (*DAOUD, 2009*).

Quando a replicabilidade dos dados não é crucial pode-se criar técnicas que apesar de não armazenarem todos os dados, podem ser bem rápidas e com alta compressão, como no caso do JPEG (*WALLACE, 1991*).

Métodos sem perda de dados só conseguem trabalhar com dados estáticos sem o uso de tendências, para o fornecimento de parâmetros tais como tendências e rampas de processo, seria necessário programar um método de captura para cada bloco compactado.

2.1.2 Algoritmos com Perda de Dados

Algoritmos empregados industrialmente utilizam métodos diretos, por serem passíveis de utilização em tempo real. Esses algoritmos se baseiam em interpolações entre pontos distantes podendo descartar os pontos intermediários sem perda significativa de informação.

Um dos algoritmos amplamente utilizados na indústria é o *Swinging Door Trending* (SDT). Neste, tomando um ponto inicial, geram-se dois pontos base a uma distância determinada D, e a cada novo ponto serão geradas duas retas de exceção, cujos pontos iniciais serão os pontos base e os finais serão gerados a partir do ponto atual e uma distância que aumenta conforme o número de pontos intermediários, criando uma região de tolerância e acumulando novos pontos intermediários a cada nova iteração. Caso pelo menos um ponto fique fora da região de tolerância, o ponto real será gravado (*BRISTOL, 1990*) conforme a Figura 2.2.

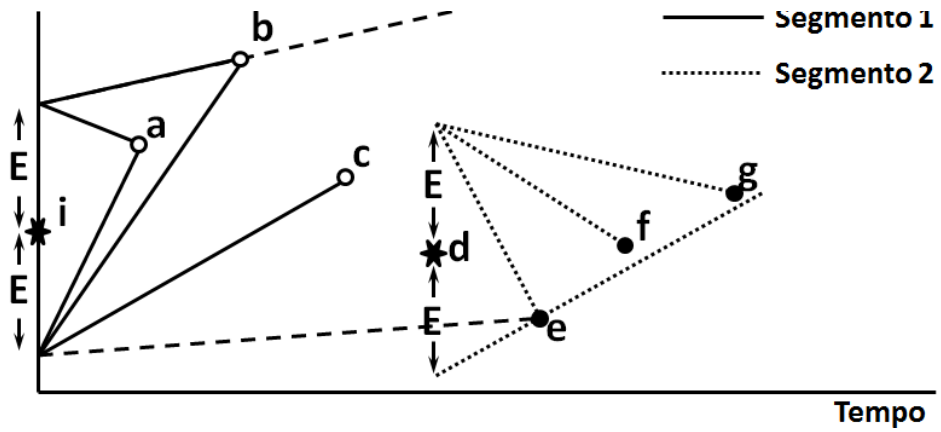

Figura 2.2 – Violações dos critérios para o SDT, no primeiro segmento, o ponto e faz com que a porta inferior viole o critério; (○) pontos não gravados, (∗) pontos gravados e (●) próximos a serem amostrados. Adaptado de (*MAH, 1995*)

Uma adaptação comercial muito utilizada baseia-se na geração de um paralelogramo, cujo comprimento x é o espaço formado entre o primeiro ponto e o último ponto, utilizando uma altura de duas vezes a tolerância, centrado nos pontos base. Dessa forma, quando um ponto intermediário encontrar-se fora da região de confiança o ponto atual será gravado e o processo é novamente iniciado até que uma nova exceção ocorra. Além disso, é estipulado um número mínimo de pontos intermediários entre um ponto gravado e outro, de forma que não haja um período muito longo sem registros gravados (*OSISOFT, 2003*). Na Figura 2.3 pode ser visto uma aplicação real de um algoritmo SDT.

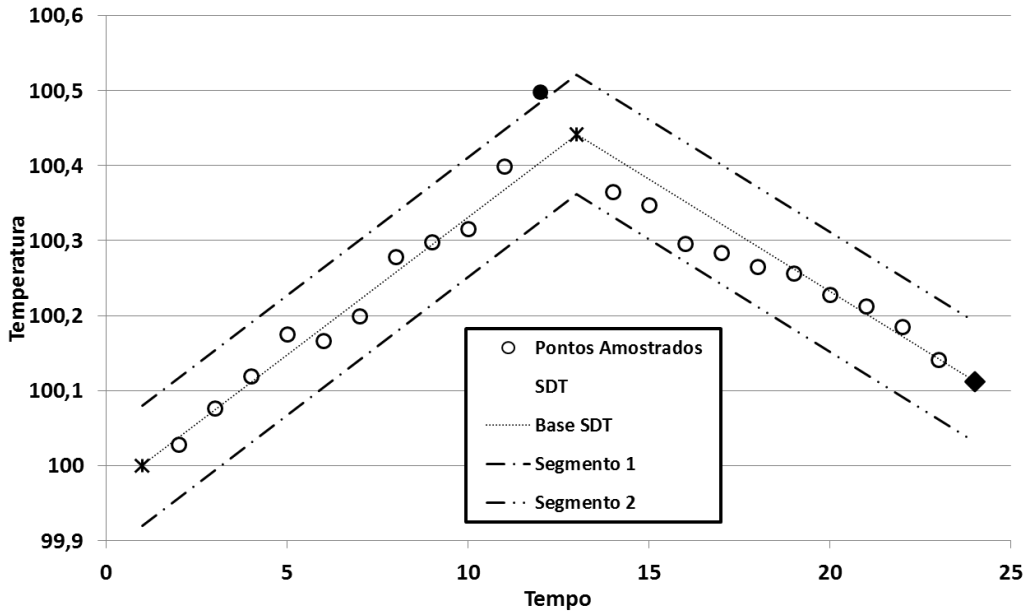

Figura 2.3 – Funcionamento do algoritmo SDT , ponto atual (◆), pontos amostrados não gravados (○), pontos gravados (∗) e ponto de exceção (●)

Similarmente ao SDT, tem-se o Box-Car (BC); nesse algoritmo, a gravação de um ponto ocorre quando o ponto atual difere do último ponto gravado por um valor acima de uma tolerância especificada. Para isso não se calculam ângulos, assim a região é sempre fixa, no entanto, todos os pontos intermediários são descartados não entrando no cálculo ponto atual (*HALE, 1981*), conforme demonstrado na Figura 2.4A. Esse algoritmo é muito útil quando temos processos que trabalham por um período considerável em estado estacionário. Para processos com transições bruscas, os autores propõem algoritmo Backward Slope (BS), neste os dois últimos pontos gravados são usados para definir um ângulo que será usado para definir a região de tolerância e o funcionamento da exceção comporta-se da mesma forma que o BS (*HALE, 1981*), conforme demonstrado na Figura 2.4B.

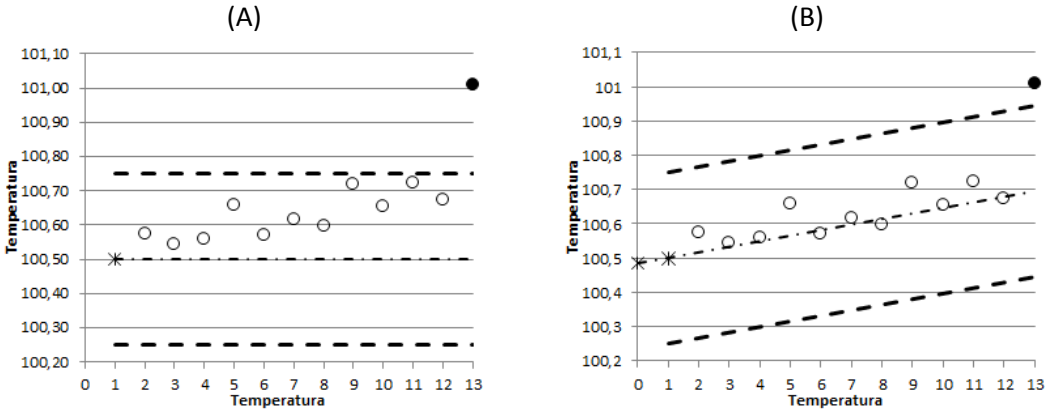

Figura 2.4 – Funcionamento dos algoritmos BC (A) e BS (B), onde (◯) pontos amostrados não gravados, (✶) pontos gravados e (●) ponto de exceção

Atualmente existem algoritmos que combinam ambos, BC e BS, através de um algoritmo único, no qual somente se a exceção ocorrer nas duas circunstâncias ao mesmo tempo o ponto será gravado. Esse algoritmo é utilizado, por exemplo, na biblioteca *PIMS InfoPlus* da Aspentech® (*ASPENTECH, 2001*).

Outro algoritmo muito utilizado é o *Straight Line Interpolative Method* (SLIM), que também faz o uso de bandas de tolerância e utiliza uma forma diferente de construção de limites de tolerância para determinar a exceção. Se tivermos dois pontos iniciais com respectivos valores de tempo e posição, são construídas duas retas partindo-se do ponto inicial e do ponto final adicionado da tolerância positiva e negativa, criando uma região triangular. Faz-se o mesmo procedimento para o ponto seguinte, se a banda do mesmo encontrar-se dentro da banda anterior, o ponto pode ser descartado, como as angulações só podem diminuir depois de um ponto ser amostrado o processo reinicia com o próximo ponto sendo o inicial (*JAMES, 1995*), como demonstrado na Figura 2.5.

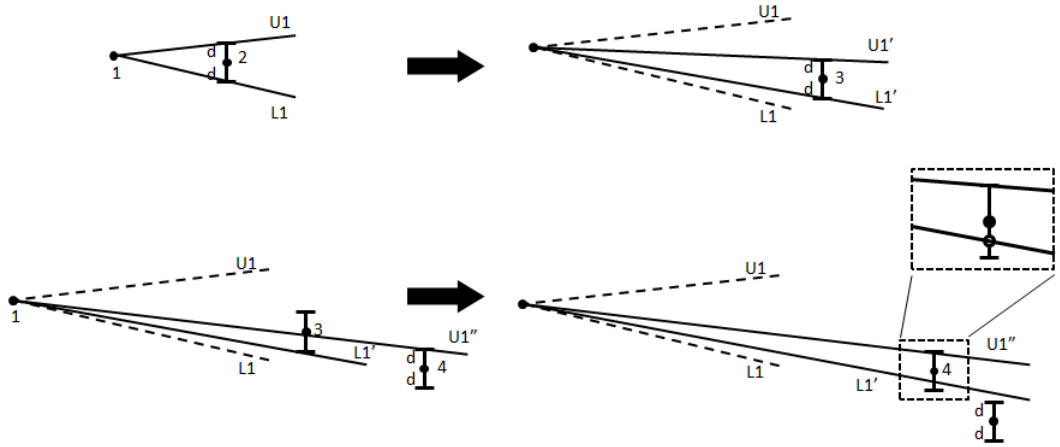

Figura 2.5 - Etapas do algoritmo SLIM, uma gravação ocorre quando a banda de tolerância do valor atual não for coberta pelos limites superior e inferior. Adaptado de (*JAMES, 1995*)

No primeiro passo dados os pontos 1 e 2, são construídas duas retas inclinadas com o parâmetro de máximo desvio. A seguir, com o ponto 3 os limites U1 e L1 são redefinidos para U1' e L1' de acordo com a tolerância fechando o cone anterior. Com o ponto 4 somente o U1' será modificado, pois a alteração do L1' abriria o cone e isso não é permitido. No ponto 5 apesar de estar dentro dos limites U1 e L1 ele está fora do cone gerado até o ponto 4, com isso a exceção é acionada e o ponto gravado.

Existem modificações desse método, conhecidas como SLIM2 e SLIM3, ambos utilizam o ponto anterior ao amostrado; no primeiro, se a banda do ponto anterior estiver fora da região de tolerância, ele fará a gravação, no segundo, o ponto é gravado se o anterior estiver fora da região de tolerância.

Um método mais elaborado para a compressão é o *Piecewise Linear Online Trending* (PLOT). Nele devemos admitir que as medidas, com comportamento discreto e em intervalos constantes de tempo, de acordo com a Equação 2.1:

$$y_t \equiv \mu_t + \varepsilon_t \, , t = 1,2,3 \ldots \qquad (2.1)$$

Onde μ_t é a média da variável na amostragem t e ε_t é o erro de medição no mesmo instante. Pode-se considerar que estes são independentes e também seguem uma distribuição normal, não necessariamente estacionária. A tendência pode ser aproximada por uma função linear dada por:

$$\mu_t = \mu_{t_{j-1}} + \delta_{j-1} + \beta_j (t - t_{j-1}) \, , j = 1,2,3 \ldots \qquad (2.2)$$

Onde δ representa um degrau na função no tempo e β a inclinação de um intervalo j, assim o objetivo do algoritmo é estimar a função linear representativa do intervalo através da detecção mudança de tendências que serão os pontos iniciais para os novos intervalos. Além disso, é possível detectar *outliers* na

tendência real. Se os eventos extrapolarem um padrão definido pode-se também fazer testes para distinguir elementos (*SINGHAL, 2005*).

2.1.3 Métodos com Base em Tendência

Há também métodos que se utilizam de transformadas, baseados em transformadas de Laplace e Fourier, através da representação do sinal no domínio da frequência (*BAKSHI, 1996*) e também em ambas, frequência e tempo, para o caso das chamadas *wavelets* (*WATSON, 1995*). Quando utilizamos as transformadas de Laplace e Fourier observamos o problema da não informação do local da ocorrência das frequências, porém, nas *wavelets*, as mesmas podem ser reproduzidas simultaneamente em escala, em função do tempo ou frequência. O funcionamento da técnica se dá através de expansões como demonstrado na Equação 2.3:

$$f(t) = \sum_{j,k=-\infty}^{\infty} c_{j,k}\, \psi_{j,k}(t) \qquad (2.3)$$

Onde $\psi_{j,k}(t)$ corresponde à função base ortonormal, provenientes de uma função definida como *wavelet mãe*, $\psi(t)$, sendo $c_{j,k}$ coeficientes dados através da Equação 2.4:

$$c_{j,k} = 2^{j/2} \sum_{l=-\infty}^{\infty} f(l)\psi(2^j j - k) \qquad (2.4)$$

Sendo $f(l)$ o l-ésimo elemento na sequência de dados discretos, j corresponde ao fator escala e k o fator translação (*MALLAT, 1989*). Na Figura 2.6 é demonstrada uma análise de *wavelets* utilizando-se de diversas escalas.

(A) (B) (C) (D)

Figura 2.6 – Análise de multirresolução de um espaço bidimensional, (A) amostra bidimensional, (B) representação em escala 1, (C) representação em escala 2 e (D) representação em escala 3. (*SHEIKHOLESLAMI, 1998*)

Outro método possível para compressão de dados é o da compressão através da quantização. Nesse método é feito uma generalização dos componentes, de modo que as distorções tenham a tendência de se tornar insignificantes. Para isso deve-se reduzir a redundância de sinais através de princípios de ortogonalidade usando, por exemplo, uma transformada do tipo MDCT (*Modified Discrete Cosine Transform*) que é dada pelas seguintes equações (*JAYANT, 1993*):

$$F(u) = \sum_{n=0}^{2N-1} h(n)x(n)\cos\left[\frac{\pi}{2N}(2u+1)(2n+1+N)\right] \quad (2.5)$$

$$u = 0{,}1 \ldots 2N-1 \quad (2.6)$$

$$h^2(N-1-n) + h^2(n) = 2 \quad (2.7)$$

$$h^2(N+n) + h^2(2N-1-n)^2 = 2 \quad (2.8)$$

$$0 \leq n < N \quad (2.9)$$

Onde $h(n)$ é a função janela que pode assumir formas como; $sen\left[\left(n+\frac{1}{2}\right)\frac{\pi}{2N}\right]$ sendo $x(n)$ os valores a serem transformados e N números reais.

Exemplo: $h(n) = \pm\sqrt{2}\, sen\left[\left(n+\frac{1}{2}\right)\frac{\pi}{2N}\right]$ quando $N = 2$ (2.10)

Nesse método os números são transformados em valores inteiros reduzindo o número de bits necessários para representar uma curva.

Existem ainda outros métodos possíveis de compressão de dados tais como o uso de *Splines* (*VEDAM, 1998*) ou o método HLT (*High Level Trigger*) usado no sensor ALICE do LHC (*RICHTER, 2012*) dentre outros mais dispendiosos computacionalmente.

Para contornar problemas relacionados a ruídos de processos ainda existem técnicas compostas onde os dados passam por um refinamento através de técnicas de suavização de curvas, como *splines*. Estas são funções aproximam sinais com ruídos, através de uma curva suave em direção aos mínimos quadrados. Uma dessas funções é a *spline* de Hermite, que é do tipo interpoladora cúbica e definida por:

$$P(t) = h_{00}(x)p_k + h_{10}(x)m_k + h_{01}(x)p_{k+1} + h_{11}(x)m_{k+1} \quad (2.11)$$

Nessa equação $x = [0,1]$, p_k e p_{k+1} correspondem aos nós e m_k e m_{k+1} são tangentes de controle. Ainda temos a função que é representada por:

$$h_{ij}(x)\begin{cases} h_{00}(x) = 2x^3 - 3x^2 + 1 \\ h_{10}(x) = x^3 - 2x^2 + x \\ h_{01}(x) = -2x^3 + 3x^2 \\ h_{11}(x) = x^3 - x^2 \end{cases} \quad (2.12)$$

As tangentes podem ser dadas por:

$$m_k = \frac{d_{min}}{\left(w_1\frac{d_{k-1}}{d_{max}} + w_2\frac{d_k}{d_{max}}\right)} \quad (2.13)$$

Onde:

$$d_k = (y_{k-1} - y_k)/h_k \ , \ d_{min} = \min(|d_{k-1}|, |d_k|) \ ,$$
$$d_{max} = \max(|d_{k-1}|, |d_k|) \ , \ w_1 = \frac{h_{k-1}+h_s}{3h_s}, w_1 = \frac{h_k+h_s}{3h_s} \qquad (2.14)$$

Utilizando:

$$h_s = h_{k-1} + h_k \ , h_k = t_{k-1} - t_k \qquad (2.15)$$

Além disso, as próprias técnicas de *splines* podem ser usadas como compressão, utilizando-se os nós como pontos da curva através de um método evolutivo. Como demonstrado na Figura 2.7. (*SILVEIRA, 2012*)

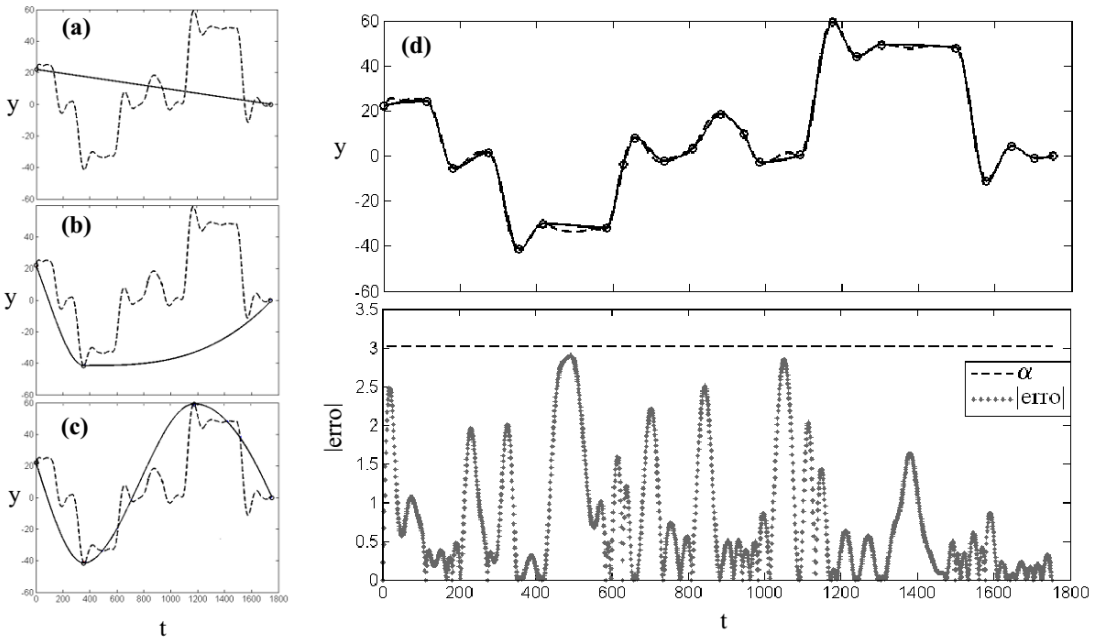

Figura 2.7 – Funcionamento do algoritmo no método Evolutivo, exemplificado pelas etapas (a), (b) e (c), e o resultado final da compressão em (d), com os erros absolutos. Nos gráficos que representam a janela de dados, a curva tracejada é o sinal suavizado, os círculos são dados gravados e a curva preta contínua, os dados reconstruídos. (*SILVEIRA, 2012*)

2.2 Algoritmos de Autodiagnostico de Plantas

Desde que se começou a falar em controle de processos muito já foi feito, desde o controle de simples abertura e fechamento de válvulas no início, até controle de plantas inteiras nos dias de hoje, graças aos adventos das metodologias e aumento da capacidade dos computadores, mas ainda assim a parte mais importante, a gestão de processos, ainda recai em grande parte nos operadores humanos. Contudo, com o incremento da dificuldade de se fazer os processos de gestão, deixar essa tarefa nas mãos dos trabalhadores se tornou menos eficiente, uma vez que o controle de uma válvula é simples, porém quando temos cem válvulas se

torna complexo. Em plantas modernas os números de variáveis medidas podem passar de 1.500, sendo seus valores atualizados em questão de segundos (*BAILEY, 1984*).

Estatisticamente sabe-se que em torno de 70% dos acidentes das indústrias tem um componente humano envolvido, sendo alguns dos maiores em plantas químicas, o da Union Carbide's Bhopal na Índia e a Occidental Petroleum's Piper Alpha no Reino Unido em que juntos responderam pela morte de cerca de 4.000 pessoas (*MANNAN, 2005*) e o maior acidente tendo ocorrido na usina nuclear de Chernobyl na Ucrânia durante o teste de um sistema de segurança. Estima-se que em 2005 cerca de cinco milhões de pessoas tivessem alguma contaminação decorrente do acidente e cerca de 4.000 mortes por câncer foram confirmados como decorrentes da radiação (*IAEA, 2005*). Apesar de desastres serem raros, acidentes menores respondem por perdas de centenas de bilhões de dólares a cada ano (*NSC, 2011*).

Para se reduzir esses problemas, diversas estratégias de controle podem ser empregadas para detecção de falhas a fim de reduzir os riscos. Para podermos definir tais estratégias, temos que delimitar o que é uma falha. Usualmente, falha pode ser definida como uma alteração da variável controlada fora do aceitável (*HIMMELBLAU, 1979*). Para se classificar as técnicas de controle, podemos usar dois parâmetros; tipo de conhecimento da variável e estratégia de auditoria. Quanto ao tipo de conhecimento podemos classificar em base em dados históricos e baseados em modelos, quanto aos métodos de busca podemos classificar de acordo com a Figura 2.8 (*RASMUSSEN, 1986*).

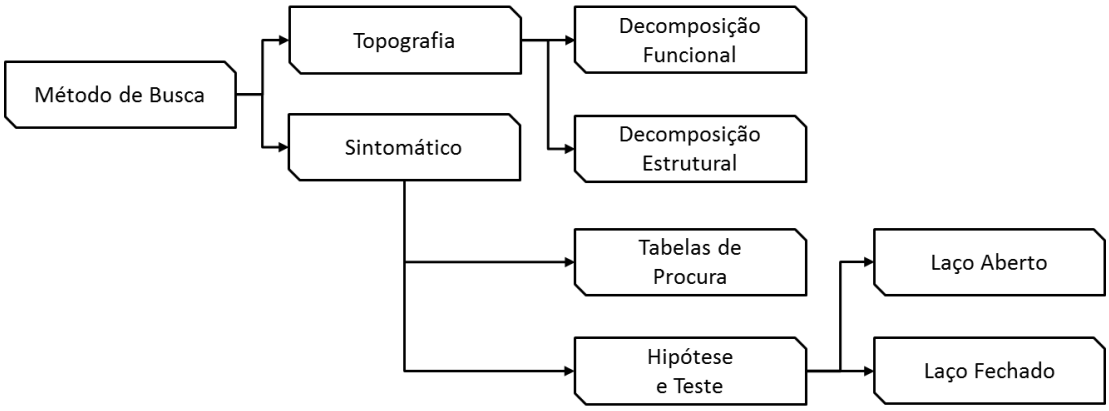

Figura 2.8 – Métodos de Busca para Sistemas de Auditoria.

2.2.1 Métodos Qualitativos

Esses métodos foram, as primeiras tentativas de realizar uma auditoria do sistema em uma maneira mais formal. Seu funcionamento se dá através de uma simulação dos processos cognitivos humanos através de uma grande gama de

regras do tipo "*se*" para definir os estados do sistema. Um problema desse tipo de técnica é o da dificuldade de entendimento físico, uma vez que falta, nas regras, a representação em alto nível. Para a construção desse tipo de sistema de auditoria são necessários três níveis de abstração, a metodologia de solução de problemas, a formalização ou modelamento da metodologia e por fim a implementação dessa metodologia (*VENKATASUBRAMANIAN, 1988*), como mostrado na Tabela 2.5.

Tabela 2.5 – Etapas do processo de abstração do processo qualitativo. Adaptado de (*VENKATASUBRAMANIAN, 1988*).

Nível	Questões
METODOLOGIA	Metodologia do Diagnóstico de Solução de Problema Conhecimento de Princípios Primários contra Compilados Abordagem Qualitativa contra Quantitativa
FORMALIZAÇÃO	Questões de Modelamento Estruturação da Base de Conhecimento Representação do Conhecimento (Regras, Quadros, Objetos, Procedimentos ou Híbrido) Formalização de Inferências (Encadeamento Frontal, Encadeamento Traseiro, Métodos Fracos, etc.)
IMPLEMENTAÇÃO	Software (Linguagem de Programação, *Shell*, etc.) Hardware (PC, Estações 32-bit, Máquinas Lisp, etc.)

Um exemplo dessa metodologia foi apresentado por Lin (*LIN, 1993*). Para a criação do sistema de diagnóstico é criado primeiro a metodologia, constando os estados possíveis do sistema como, por exemplo, na classificação de falhas em transformadores elétricos onde os gases gerados podem indicar as possíveis falhas através de uma análise de DGA. Nesta pode ser observado que valores elevados de CH_4 em relação à H_2 podem significar falha térmica por baixa temperatura, o que pode ser visto na Tabela 2.6 juntamente com outros tipos de falhas.

Tabela 2.6 – Tabela de Classificação de Gases. Adaptado de (*LIN, 1993*).

$\frac{C_2H_2}{C_2H_4}$	$\frac{CH_4}{H_2}$	$\frac{C_2H_4}{C_2H_6}$	Variância da razão de gás (Volume/Volume)
0	1	0	Menor que 0,1
1	0	0	0,1 – 1
1	2	1	1 – 3
2	2	2	Maior que 3
			Característica da Falha
0	0	0	Decaimento Normal
*	1	0	Descargas Parciais de Baixa Densidade Energética
1	1	0	Descargas Parciais de Alta Densidade Energética
1 - 2	0	1 – 2	Descargas de Baixa Energia (Faísca Contínua)
1	0	2	Descargas de Alta Energia (Com Atravessamento de Energia)
0	0	1	Falha de Temperatura, Menos de 150°C
0	2	0	Falha de Temperatura, Entre de 150 – 300°C
0	2	1	Falha de Temperatura, Entre de 300 – 700°C
0	2	2	Falha de Temperatura, Mais de 700°C

*Insignificante

Na proposição do algoritmo de diagnóstico, rotina da manutenção foi pensada baseada na técnica *Dissolved Gas Analysis* (DGA) e pode ser vista na Figura 2.9.

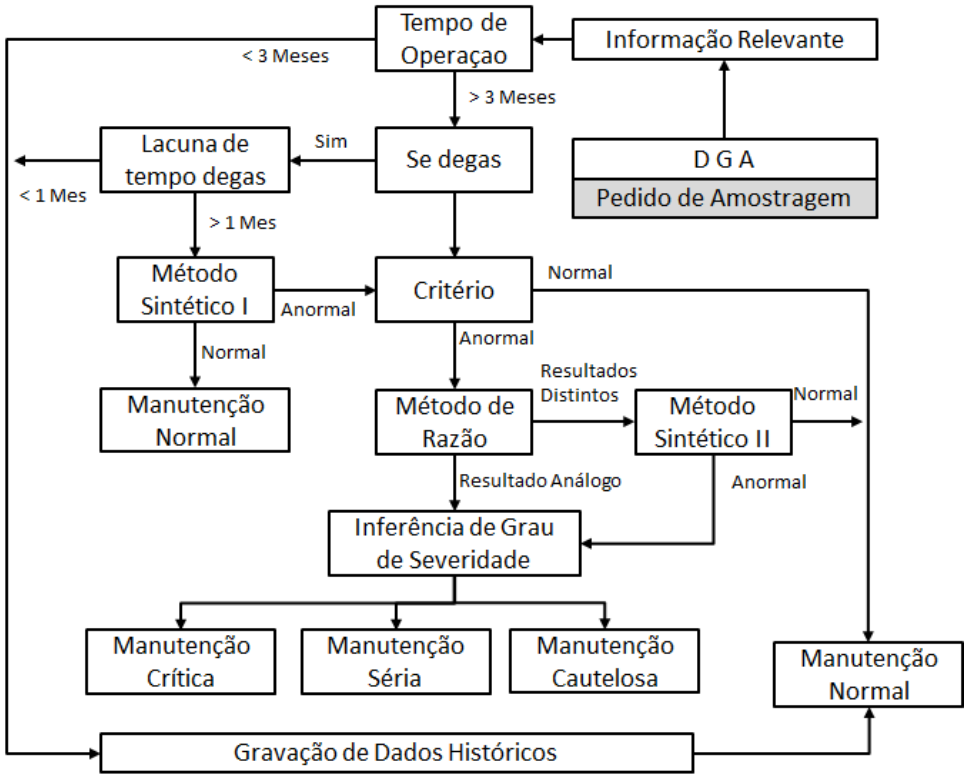

Figura 2.9 – Diagrama da Formalização do Sistema de Diagnóstico. Adaptado de (*LIN, 1993*)

Na última etapa, faz-se a programação do método, como demonstrado na Figura 2.10.

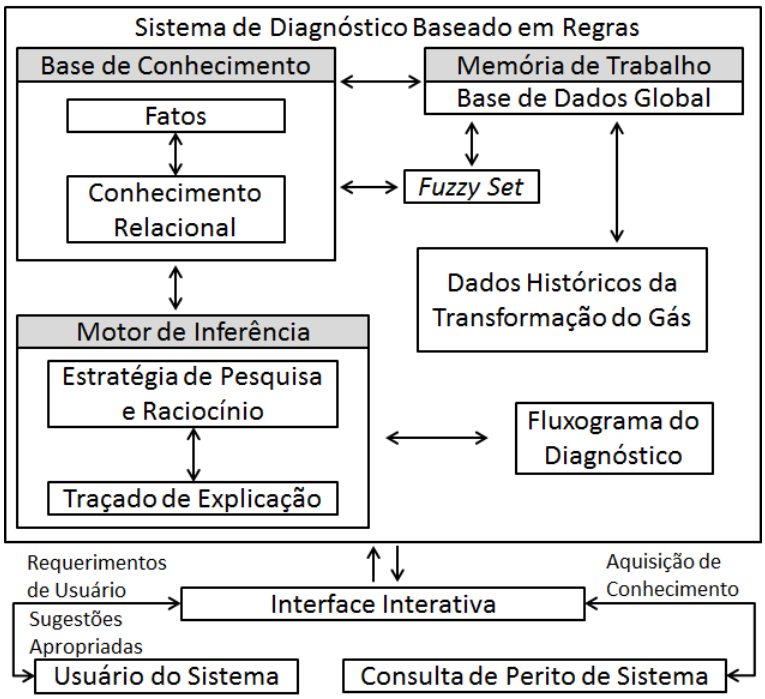

Figura 2.10 – Diagrama da Implementação do Sistema. Adaptado de (*LIN, 1993*).

Os métodos originais são baseados em sistemas especialistas, contudo esse método é uma descrição incompleta das relações e não expressa as causalidades entre elas. No entanto, podemos gerar as condições de igualdade através de equações matemáticas, dessa forma a parte qualitativa serve somente como restrições, como por exemplo, uma pressão máxima de segurança e na estrutura do sistema desde restrições algébricas até restrições baseadas em funções de transferência para diversos níveis de inter-relacionamento (*SHAO, 2012*).

2.2.2 Métodos Quantitativos (Baseados em Modelo)

Para que se obtenham bases mais precisas para a detecção de falhas em um sistema complexo, onde enumerar todas as possíveis falhas pode se tornar problemático podemos utilizar os modelos matemáticos do processo para prever os estados do mesmo.

Os sistemas baseados em modelos têm por base análises ou modelos qualitativos, dessa forma eles mostram uma relação muito forte com os métodos de controle modernos. No entanto a funcionalidade desses sistemas depende amplamente de outros sistemas computacionais interligados e da capacidade dos sistemas utilizados (*FRANK, 2000*).

Um dos sistemas baseados em modelos muito utilizado é o *Failure Detection and Identification* (FDI). O processo ocorre em duas etapas, sendo a primeira consistindo na geração de um resíduo a partir da diferença entre o valor medido e o valor predito para o sistema. Assim se, não houver alterações significativas, o sistema estaria sem falhas, que ocorrendo iriam desviar o valor observado do valor previsto. O método de geração dos resíduos pode assumir diversas complexidades dependendo do modelo adotado. Para a segunda etapa, temos o processo de decisão em si, que consiste em um teste de limiar para verificar as médias dos resíduos, o qual pode ser feito de diversas formas, como a razão de probabilidade sequencial ou até através teorias de decisão estatísticas (*CHOW, 1984*).

Para os resíduos pode-se usar a teoria da observabilidade, simplificadamente apresentado da seguinte forma, segundo Hammouri (*HAMMOURI,1999)*:

$$\begin{cases} \dot{x} = f(x) + \sum_{i=1}^{m} g_i(x)u_i \\ y = h(x) = \left(h_1(x), \dots, h_p(x)\right) \end{cases} \quad (2.16)$$

Onde $x(t) \in \mathbb{R}^n, u(t) = \left(u_1(t), \dots, u_m(t)\right)^T \in U \subset \mathbb{R}^m, u(t) \in \mathbb{R}^p$ que são respectivamente o estado, entrada e saída do sistema dinâmico.

Assim o sistema em questão será observável somente se para cada par de estados iniciais, $(x, \bar{x}), x \neq \bar{x}$, existe um controle $u: [0,T] \rightarrow U$ com um instante de tempo $t \in [0,T]$ em que $y(x,u,t) \neq y(\bar{x},u,t)$ onde $y(x,u,t) = h\left(x_u(t)\right)$, $x_u(t)$ então será a trajetória única do sistema dinâmico de forma que $x_u(0) = x$. Caso exista a entrada u que distingue $(x, \bar{x})$, a entrada $u: [0,T] \rightarrow U$ distinguiria todos $(x, \bar{x}), x \neq \bar{x}$ é universal em $[0,T]$. O sistema então será observável se para cada $T > 0$, todos os controles admissíveis de $u: [0,T] \rightarrow U$ é uma entrada universal em $[0,T]$ (*HAMMOURI, 1999*).

Caso tenhamos um sistema de dois atuadores que em um determinado tempo sofre uma falha total, a mesma seria evidenciada da seguinte forma, como demonstrado na Figura 2.11:

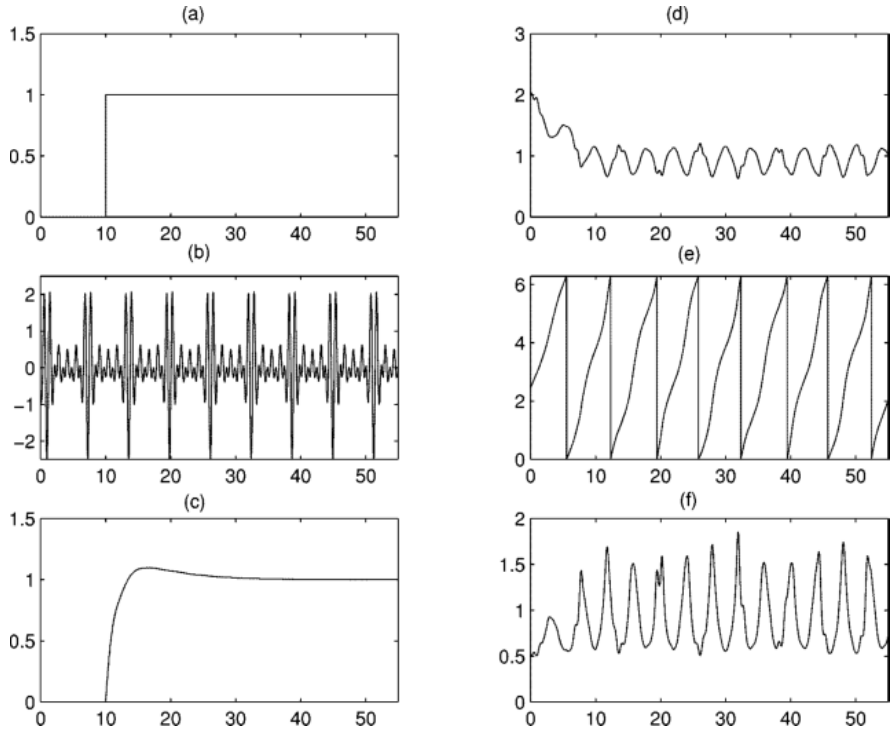

Figura 2.11 – Detecção, isolamento e identificação de uma falha de atuador em um modelo de satélite ponto-massa (*DE PERSIS, 2001*).

Nesse caso a Figura 2.11 mostra em (a) a falha total de um atuador, em (b) temos os distúrbios no sinal, em (c) a falha total em conjunto com a saída do gerador de resíduo e nos itens (d), (e) e (f) não sendo possível identificar a falha (*DE PERSIS, 2001*).

Outra abordagem possível para fornecer dados em um algoritmo de diagnóstico é o *Principal Component Analysis* (PCA). Esse método consiste na redução da dimensão de um problema. A base do PCA encontra-se na ortogonalidade de vetores, os quais podem ser ordenados pelo tamanho de sua variância. Se possuirmos n observações de conjuntos de variáveis alocadas em uma matriz $\boldsymbol{X}$, podemos calcular os vetores de carga através de um problema de otimização (*RUSELL, 2000*) do tipo:

$$\max_{v \neq 0} \frac{v^T \boldsymbol{X}^T \boldsymbol{X} v}{v^T v} \qquad (2.17)$$

Onde $v \in \mathbb{R}^m$. Assim, pontos de estados estacionários podem ser computados através de uma análise de *Singular Value Decomposition* (SVD) dada por:

$$\frac{1}{\sqrt{n-1}} \boldsymbol{X} = \boldsymbol{U} \boldsymbol{\Sigma} \boldsymbol{V}^T \qquad (2.18)$$

Então $\boldsymbol{U} \in \mathbb{R}^{n \times m}$ com $\boldsymbol{V} \in \mathbb{R}^{m \times m}$ sendo a matriz unitária e a matriz de valores singulares dada por $\boldsymbol{\Sigma} \in \mathbb{R}^{n \times m}$ que terão magnitude decrescente na forma $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_{\min(m,n)} \geq 0$. Com isso, os vetores de carga serão colunas

ortonormais na matriz $\boldsymbol{V}$ onde o conjunto de treino é projetado na *e*nésima coluna da matriz que é igual à σ_i^2 (*RUSELL, 2000*).

Uma das variações do método PCA é o MKPCA (*Multiway Kernel Principal Component Analysis*), este é uma união entre duas outras modificações do PCA, o KPCA que se utiliza de um conjunto de amostras colocadas em um espaço de forma que possa se computar a matriz de covariância na forma:

$$C^F = \frac{1}{N}\sum_{j=1}^{N} \Phi(x_j)\Phi(x_j)^T \quad (2.19)$$

Neste caso $\Phi(\cdot)$ será uma função não linear de mapeamento, assumindo-se que x_j para $k = 1, \dots, N$ é a variância escalonada e centrada na média. Dessa forma para solucionar o problema de PCA então podemos usar um problema de autovalores

$$C^F v = \lambda v \quad (2.20)$$

$$C^F v = \frac{1}{N}\sum_{j=1}^{N} \langle \Phi(x_j), v\rangle \Phi(x_j) \quad (2.21)$$

Assim os autovalores $v \geq 0$ e $v \in F\backslash\{0\}$ da Equação 2.20 substituída na 2.19 gera a Equação 2.21 em que $\langle x, y\rangle$ é o produto escalar entre x e y e implica que todas as soluções de v em que $\lambda \neq 0$ deve estar no intervalo de $\Phi(x_1), \dots, \Phi(x_N)$ com isso $C^F v = \lambda v$ será equivalente à

$$\langle \Phi(x_k), C^F v\rangle = \lambda\langle \Phi(x_k), v\rangle, k = 1, \dots, N \quad (2.22)$$

Se os coeficientes $\alpha_i (k = 1, \dots, N)$ em que

$$v = \sum_{j=1}^{N} \alpha_i \Phi(x_j) \quad (2.23)$$

Combinando então as equações (e (chegamos à:

$$\lambda \sum_{j=1}^{N} \alpha_i \langle \Phi(x_k), \Phi(x_i)\rangle = \frac{1}{N}\sum_{j=1}^{N} \alpha_i \langle \Phi(x_k), \sum_{j=1}^{N} \Phi(x_j)\rangle \langle \Phi(x_j), \Phi(x_i)\rangle \quad (2.24)$$

Quando colocamos $\langle \Phi(x_k), \Phi(x_i)\rangle$ como uma matriz quadrada de dimensão N obteremos ao final (*SCHÖLKOPF, 1998*).

$$N\lambda\alpha = K\alpha \quad (2.25)$$

Fazendo o PCA para um conjunto t de um vetor de teste x, obtida pela projeção de $\Phi(x)$ em autovalores v_k em F em que $k = 1, \dots, p$

$$tk = \langle v_k, \Phi(x)\rangle = \sum_{j=1}^{N} \alpha_i^k \langle \Phi(x_i), \Phi(x)\rangle \quad (2.26)$$

Assim o problema do PCA pode ser resolvido por uma função do tipo *kernel* que tem a seguinte forma:

$$k(x, y) = \langle \Phi(x), \Phi(y) \rangle \quad (2.27)$$

A modificação *Multiway* consiste no desdobramento de dados em trajetórias, como demonstrado pela Figura 2.12

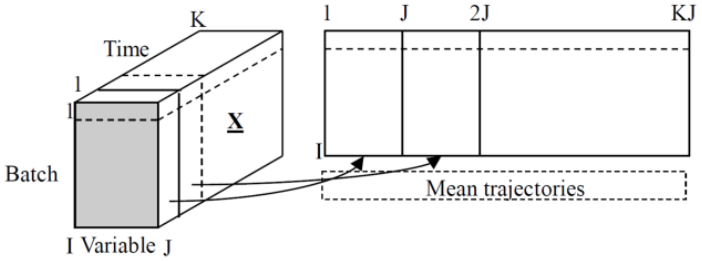

Figura 2.12 – Método de desdobramento do MPCA. (*LEE, 2004*)

Com isso podemos usar como preditor o SPE (*Squared Prediction Error*) dado pela seguinte equação:

$$SPE = \left\| \Phi(x) - \hat{\Phi}_p(x) \right\|^2 \quad (2.28)$$

Para o monitoramento ficaríamos com

$$SPE \sim g X^2_{h,\alpha} \quad (2.29)$$

Onde g corresponde à $\frac{\nu}{2m}$ e h correspondendo à $\frac{2m^2}{\nu}$ (*LEE, 2004*).

Técnicas mais recentes fazem o uso de transformadas, como as *wavelets* para a geração de funções base em tempo-frequência. Esse sistema faz uso de uma função *mãe* $\Psi(t)$ que possui tanto o domínio da frequência como no tempo, então através do uso de dilatação e translação, cria-se uma família de novas *wavelets* que tem a mesma forma da original e com tamanhos e posições diferentes, com a seguinte propriedade:

$$\psi \in L^2(\mathbb{R}) \left| \int_{-\infty}^{+\infty} \psi(t) dt = 0 \right. \quad (2.30)$$

Na qual $L^2(\mathbb{R})$ é o espaço de vetores das funções quadradas integráveis no conjunto dos reais, e normalizado por $\|\psi\| = 1$ é centrado em $t = 0$. Assim a família de *wavelets* em relação ao domínio do tempo e dilatado no domínio de escala é dada por:

$$\psi_{a,b}(t) = \frac{1}{\sqrt{a}} \psi\left(\frac{t-b}{a}\right) \quad (2.31)$$

Nesse caso, a é responsável pela dilatação e b responde pela translação. ($\|\psi_{a,b}\| = 1$). Com isso, a transformada *wavelet* de uma $f(t) \in L^2(\mathbb{R})$ a uma escala a e uma posição b correlacionando a função com a *wavelet*.

$$Wf(a,b) = \int_{-\infty}^{+\infty} f(t) \frac{1}{\sqrt{a}} \psi^* \left(\frac{t-b}{a}\right) dt \qquad (2.32)$$

Assim, qualquer função $f(t)$ em $L^2(\mathbb{R})$ pode se caracteriza pela decomposição da família de *wavelets* relacionadas à equação 2.30 aplicando-se o teorema de Parseval-Fourier no fim obtém-se:

$$\omega_c = \frac{1}{2\pi E} \int_0^{\infty} \omega \left|\hat{\psi}(\omega)\right|^2 d\omega \qquad (2.33)$$

Em que $\frac{1}{2\pi}\int_0^{\infty}\left|\hat{\psi}(\omega)\right|^2 d\omega$ e a banda energética de $\hat{\psi}(\omega)$ dado por σ_ω onde:

$$\sigma_\omega^2 = \frac{1}{2\pi E} \int_0^{\infty} (\omega - \omega_c)^2 \left|\hat{\psi}(\omega)\right|^2 d\omega \qquad (2.34)$$

Dessa forma podemos então classificar os erros em máquinas rotativas comparando-se os picos nas frequências características, demonstrado na Tabela 2.7 (*CHOW, 2004*).

Tabela 2.7 – Frequências características contra defeitos. Adaptado de (*CHOW, 2004*).

Frequência Característica	
1xf_r	Desbalanceamento, Caminho Excêntrico
2xf_r	Afrouxamento Mecânico
1x,2x,3x,4xf_r	Desalinhamento
$N[1 + (d/e)\cos\alpha]\frac{f_r}{2}$	Falha no Anel Externo do Rolamento
$N[1 - (d/e)\cos\alpha]\frac{f_r}{2}$	Falha no Anel Interno do Rolamento
$[1 - (d/e)\cos\alpha]\frac{f_r}{2}$	Desbalanceamento da Gaiola
$(e/d)[1 + (d/e)^2\cos^2\alpha]f_r$ N = Número de esferas d = diâmetro da esfera, pol. e = diâmetro do passo, pol. α = ângulo de contato, graus	Esferas do Rolamento
1x,2x(f_r ×num. de dentes)	Engrenagens

Nesse caso a frequência característica é avaliada com respeito à velocidade de rotação do eixo f_r dado em Hertz.

Ainda existem métodos mais avançados que usam conjuntos de técnicas, como o BEMD (Bi-Dimensional Empirical Mode Decomposition) consistindo de funções de modo que podem ser analisadas pelo espectro de Hilbert, que juntamente com GDA (General Discriminant Analysis) e RMV (Relevance Vector Machine) são técnicas de determinação de fatores comuns e de *machine learning* respectivamente. O uso dessas técnicas permite uma visualização melhorada em sistemas de diagnóstico

por imagens térmicas (*TRAN, 2013*) que poderiam também ser utilizados para que se possam avaliar equipamentos como destiladores sem necessitar de sensores internos.

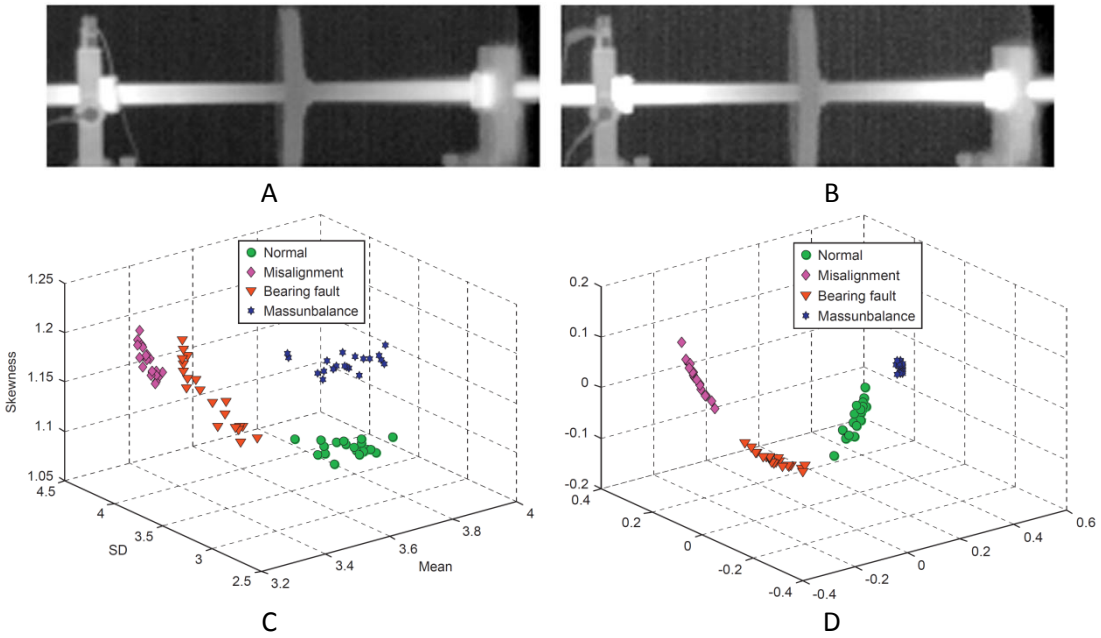

Figura 2.13 – Tratamento de uma imagem pelo método BEMD onde (A) corresponde à imagem original, (B) a imagem melhorada, (C) histograma da imagem melhorada, (D) histograma da imagem melhorada usando GDA. Adaptado de (*TRAN, 2013*).

2.3 Equipamentos para Controle de Plantas

Nos dispositivos de operação manual, como válvulas, um operador é responsável pela abertura e fechamento. No entanto para dispositivos acionados eletronicamente usamos controladores para acioná-los, havendo diversos tipos de equipamentos de controle.

2.3.1 Programmable Intelligent Computer (PIC)

O PIC e o PICmicro são microcontroladores fabricados pela empresa Microchip e tiveram inicio em 1975, tendo sido desenvolvidos para uso com a CPU, CP1600 da General Instrument, que apesar de ser 16-bit era limitado em operações I/O. Sua versão inicial era baseada em um sistema 8-bit com um código de execução mantido em uma ROM. Em 1985 o PIC sofreu uma modificação recebendo uma EPROM para que se tornasse possível, a programação customizada do mesmo. Contudo, somente com o PIC16x84 a memória EEPROM passou a ser incorporada no próprio chip, tornando menos dispendioso que as EPROM (*WILLIAMS, 1997*).

Os PICs são baseados na arquitetura Havard que foi elaborada em 1940, para uso no Computador Mark I. Essa arquitetura separa a memória de dados da memória de instruções.

O conjunto pode assumir diversas formas 12, 14, 16-bit ou outras, porém a memória de dados trabalha com conjunto 8-bit (*MAURER, 2005*). Na Figura 2.14, podemos ver como funciona a arquitetura Havard, na Figura 2.15 são mostrados dados de alguns modelos de PIC da Microchip em termos de Memória de Instruções e Dados.

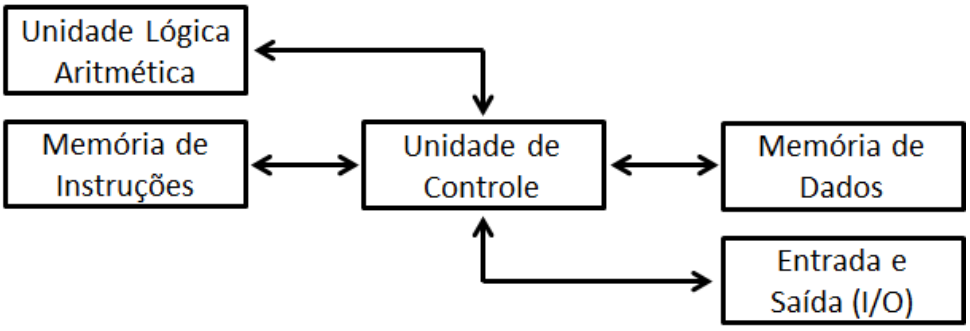

Figura 2.14 – Diagrama do funcionamento da arquitetura Havard.

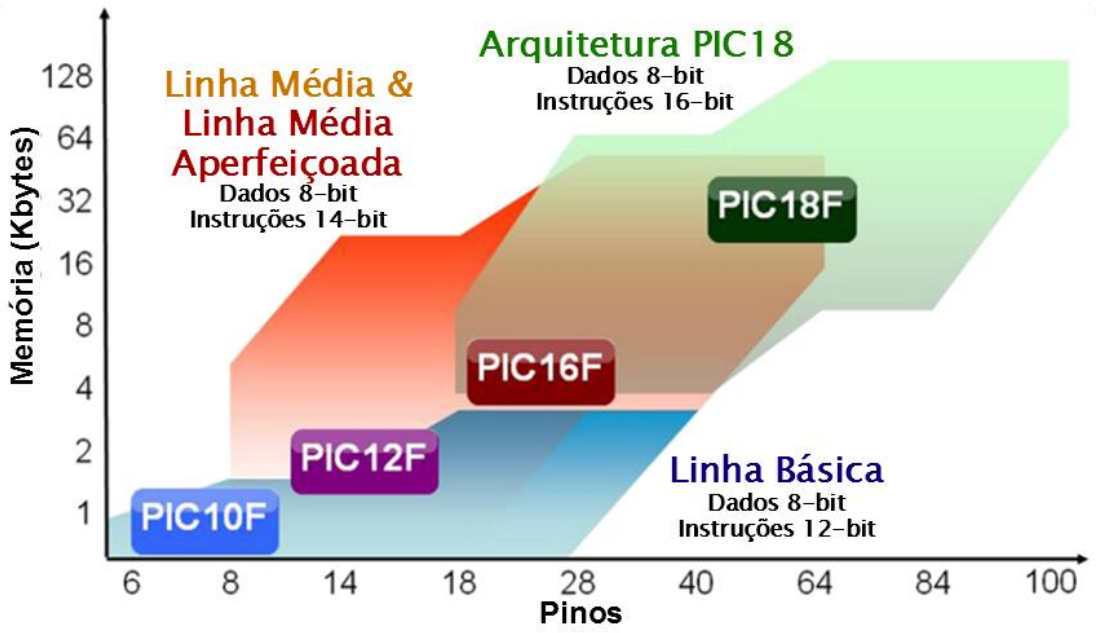

Figura 2.15 – Arquitetura de PICs fabricados pela Microchip. Adaptado de (*MICROCHIP[1], 2013*)

Atualmente a Microchip oferece uma gama muito grande de PICs, na Tabela 2.8 será apresentada uma tabela contendo alguns dos modelos fabricados.

Tabela 2.8 – Linha de PICs 8-Bit da Microchip. (*MICROCHIP[1], 2013*)

	Linha Básica	Linha Média	Média Aperfeiçoada	PIC18
Contagem de Pinos	6-40	8-64	8-64	18-100
Interrupções	Nenhum	Simples	Simples com Contexto de Hardware	Múltiplo com Contexto de Hardware
Desempenho	5 MIPS	5 MIPS	8 MIPS	16 MIPS
Instruções	33, 12-bit	35,14-bit	49,14-bit	83,16-bit
Memória de Programa	3 KB	14 KB	28 KB	128 KB
Memória de Dados	138 B	368 B	1500 B	4000 B
Camadas de Hardware	2 níveis	8 níveis	16 níveis	32 níveis
Características	Comparador 8-bit ADC Memória de Dados Oscilador Interno	Baseline + SPI/I²C™ UART PWMs LCD 10-bit ADC Op Amp	Mid-Range + Periféricos Múltiplos de Comunicação Espaço de Programação Linear PWM com Base Temporal Independente	Enhanced + Multiplicador Hardware 8x8 CMTU CAN USB Ethernet 12-bit ADC
Dispositivos	16	58	29	193

Os mais avançados PICs são os modelos PIC32MX1 e PIC32MX2, neles a arquitetura é a Von Neumann comum a grande maioria dos sistemas hoje fabricados, onde não há separação entre memória de programa e memória de dados. Os PIC32 possuem um microprocessador MIPS M4K de até 80 MHz com canais de dispositivos de cinco estágios e traçado de instruções, a memória conjunta é de 512 KB de memória Flash e 128KB em memória SRAM e separação entre o Bus de dados e instruções. Estes microcontroladores são ainda, compatíveis com USB Mestre/Dispositivo/OTG e ethernet 10/100, possuem duas portas CAN 2.0B, 6 UART, 5 I²C, 4 SPI e CMTU possuem também até 8 canais de DMA Gerais (*MICROCHIP[2], 2013*).

2.3.2 Programmable Logic Controller (PLC)

O conceito de controlador lógico programável foi introduzido em 1968 por Richard E. Morley e Bedford Associates. Esse tipo de controlador foi pensado como uma forma de reduzir o tempo de implementação de um sistema de comandos que até então se baseava em dispositivos de lógica-relé. Contudo, a primeira implementação microcontrolada ocorreu somente em 1977. Atualmente os PLCs

são normatizados através da norma IEC 61131-3 estabelecida em 2001 (*THIELE, 2009*).

Esses controladores são baseados em microcomputadores com memória programável para implementação de lógica, sequenciamento, temporizador, contagem, aritmética, usando entradas e saídas digitais e analógicas. O PLC é reprogramável, capaz de operar em ambientes adversos, aceita entrada de energia de 120 VAC e saída compatível com acionamento contínuo de dispositivos como relés, motores e outros que requeiram até 2 ampères. Esses dispositivos trabalham com a linguagem Ladder, esta é baseada no conceito de diagramas similar às implementações de contactoras e sistemas elétricos, com isso é possível reproduzir os acionamentos de forma clássica. (*MINDEK, 2008*)

Hoje um grande número de empresas fabricam PLCs como, por exemplo, a Allen Bradley, Siemens e Novus que são muito comuns em indústrias. Além disso, o sistema varia muito entre Marcas e Séries, o que pode dificultar o processo de escolha do equipamento. Na Tabela 2.9, serão apresentados alguns modelos de PLCs comerciais:

Tabela 2.9 – PLCs comercializado pela empresa Novus. (*NOVUS, 2013*)

	Novus		
	DVP-AS	DVP-SX	DVP-SV
Programação	8k Passos	8k Passos	16k Passos
Entradas e Saídas	236	230	512
Entradas Digitais	08x24Vdc	04x24Vdc	16x24Vdc
Entradas Analógicas	-	02 x ±20mA (11bit) / ±10Vdc (12bit)	-
Saídas Digitais	04 x Transistor (0,3A) ou 04 x Relé (1,5A)	02 x Transistor (0,3A) ou 02 x Relé (1,5A)	12 x Transistor (0,3A) ou 12 x Relé (1,5A)
Saídas Analógicas	-	02 x I/U	-
Comunicação	RS-232 e RS-485	RS-232 e RS-485	RS-232 e RS-485
Protocolo	ModBus RTU / ASCII	ModBus RTU / ASCII	ModBus RTU / ASCII

Tabela 2.10 – PLCs comercializados pela empresa Schneider Electric. (*SCHNEIDER ELECTRIC, 2013*)

	Schneider Electric		
	TM238LDD24DT	TM238LFDC24DT	TM258LF42DT
Número de E/S digital	24	24	42
Número de entradas digitais	8 fast input conforming to EN/IEC 61131-2 type 1	8 fast input conforming to EN/IEC 61131-2 type 1	4 for regular input
	6 input conforming to EN/IEC 61131-2 type 1	6 input conforming to EN/IEC 61131-2 type 1	12 for input
			10 for fast input
Tensão de entrada digital	24 V	24 V	24 V
Tipo de tensão de entrada digital	CC	CC	CC
Número de saída digital	6 output	6 output	12 for output
	4 fast output	4 fast output	4 for fast output
Tensão de saída digital	24 V CC	24 V CC	24 V CC
Número de módulo de expansão E/S	7	7	-
Tensão nominal de fornecimento [Us]	24 V CC	24 V CC	24 V CC
Descrição da memória	RAM interna 500 kB	RAM interna 1.000 kB	RAM interna 64 MB Flash 128 MB

Uma das limitações dos PLCs, comum aos PICs é a limitada capacidade de armazenamento, o que impede um armazenamento direto dos dados.

2.3.3 Dataloggers

Os dataloggers são dispositivos voltados para o armazenamento dos dados provenientes de sensores, atualmente utilizados em diversas funções. Um exemplo muito comum desses dispositivos são as “Caixas Pretas” presentes nos aviões. Estes dispositivos são responsáveis por manter os dados de voo e também, atualmente, podem inclusive ser usados para detectar inconsistências de medições (*ALLES, 2003*).

Esses sistemas podem ainda ser usados para realizar tarefas de controle tais como controle PID, contudo requerem implementações adicionais, as quais permitem também envio de dados por diversos protocolos (*SEHGAL, 2008*). Um dos problemas dos dataloggers está na dificuldade de seu emprego no controle de processos, uma vez que foram desenhados para trabalhar principalmente como registradores de dados. A tabela a seguir mostra alguns dos modelos disponíveis.

Tabela 2.11 – Dataloggers comerciais disponíveis no mercado.

	Abus Technologies	OMEGA Engineering	NOVUS
	Log Box-DA	OM-320	Log Box-AA
Entrada	Canal 1: NPN, PNP, Pulso, Contato Seco Canal 2: 0 ~ 20mA: 120Ω, 4 ~ 20mA: 120Ω, 0 ~ 50mV: 10MΩ, 0 ~ 10V DC: 120 KΩ	Térmicos: J. K, T, E, R, S Tensão: ±20 mV, ±40 mV, ±50 mV, ±70 mV, ±100 mV, ±200 mV, ±1 V, ±2 V, ±5 V, ±10 V, ±30 V Corrente: ±400 uA, ±1.2 mA, ±2.5 mA, ±11 mA, ±22 mA	Térmicos: Pt100, Termopar J, K, T, E, N, R, S, B Tensão: 0 a 50 mV 0 a 10 V Corrente 0 a 20 mA 4 a 20 mA
Capacidade	32000 (32K) 64000 (64K)	Amostras: 30000 Interno 330000 PCMCIA	Amostras: 32000 Canais Simples 16000 Canais Duplos
Intervalo de Amostragem	10s – 18hs	Mínimo 1s	1s – 18hs
Energia	Bateria 3,6V	Bateria de Lítio (1 Ano)	Bateria 3,6V

Os dataloggers são úteis para armazenamento de dados, contudo eles são limitados em termos de velocidade de aquisição, em alguns casos o mínimo chega a um dado a cada 10 segundos.

2.3.4 Fieldloggers e RTUs

Os RTUs e Fieldloggers são usados geralmente para a comunicação com softwares SCADA. Os sistemas RTU são sistemas mais elaborados, tendo um sistema comparável aos microcomputadores. Esses dispositivos podem inclusive conter unidades de armazenamento, capazes de manter um banco de dados para ser utilizado no sistema (*HENG, 1996*).

Por se tratarem de sistemas computacionais mais complexos, os dispositivos RTU são mais simples de serem programados e abertos além de terem códigos com maior facilidade de implementação. Muitos podem ter programação feita em diversas linguagens tais como C/C++ e Python, tornando-os muito flexíveis, além de ter suporte a uma maior gama de dispositivos, incluindo modems para comunicação remota (*CAZAN, 2008*).

Tabela 2.12 – Modelos de Fieldloggers e RTUs disponíveis comercialmente.

	Novus	B+B Thermo-Technik GmbH	FF Automation	MJK Automation
	FieldLogger	FieldLogger 512K	AutoLog GSM-20 RTU	Connect RTU
Tensão de Operação	100-240 VAC, 19-30 VDC	100-240 VAC / 20VA	115-240VAC, 12-24 VDC 10-18 VAC	
Medição	Temperatura: Termopares e Pt100, Pt1000 Tensão: Volts e millivolts Amperagem: Milliamperes Extra: Conversão A/D 24 bits	Temperatura: Todos Tensão: 0-20 mV, 0-50 mV, 0-60 mV, (-20)-20 mV, 0-5 V, 0-10 V Corrente: 0-20 mA, 4-20 mA Lógico: '0': 0-0,8 VDC; '1' 3-30 VDC	Analógico: 8-32 Entrada 2-34 Saída Digital: 8-72 Entrada 8-72 Saída Relé: 0-64	Analógico: 6 Entrada 1 Saída Digital: 16 Entrada 8 Saída
Aquisição	1000 amostragens por segundo	1000 por segundo dividido entre canais (Máx. 100 Canais)		
Memória	512 k Valores internos Expansível via SD Card	512 k Valores internos Expansível via SD Card	1x43000 ou 8x13000 interno Gravação FTP	30000 valores com data e hora
Driver	24 VDC/4-20mA	230 VAC – 3 A; 30 VDC – 3 A 24V VDC - >4mA	115-240VAC, 12-24 VDC 10-18 VAC	
Conexões	Ethernet, RS485	10 TCP; 10 UDP em Ethernet 10/100	Wireless, RS-485, Serial	GPRS, PSTN, Radio, Ethernet, RS-485, RS-232

Um dos pontos negativos do uso de fieldloggers está no seu alto custo, que para empresas pequenas pode ser inviável.

2.3.5 Dispositivos Open-Source

Com o crescimento da automação doméstica, diversas empresas começaram a fabricar alternativas de baixo custo. Devido às diferentes necessidades existem inúmeros modelos no mercado.

Um dos mais conhecidos e difundidos no mercado é o Arduino, criado em 2005 como um projeto para criar uma interação para sistemas de protótipos escolares, porém com um custo menor que os sistemas de prototipagem até então existentes. (*LAHART, 2009*). Inicialmente desenvolvido utilizando um processador Atmel de 8-bit, o software consiste em um chamado "boot loader" que carrega os comandos necessários para realizar o carregamento do sistema operacional no

microcontrolador. Hoje existem versões do Arduino que possuem processadores Atmel de 32-bit (*ARDUINO, 2013*).

Figura 2.16 – Placa Arduino UNO Rev.3 com Atmel ATmega328 8-bit. (*ARDUINO, 2013*)

Na Tabela 2.13 são mostrados alguns modelos de Arduino comercializados.

Tabela 2.13 – Modelos de Arduino comercialmente disponíveis em 2013.

	Arduino UNO Rev.3	Arduino Mega 2560 Rev.3	Arduino DUE
Microcontrolador	ATmega328 (8-bit)	ATmega2560 (8-bit)	AT91SAM3X8E (32-bit)
Voltagem de Operação	5V	5V	3,3V
Voltagem de Entrada (Recomendado)	7-12V	7-12V	7-12V
Voltagem de Entrada (Limite)	6-20V	6-20V	6-20V
Pinos de I/O Digital	14 (dos quais 6 podem ser PWM)	54 (dos quais 15 podem ser PWM)	54 (dos quais 12 podem ser PWM)
Pinos de Entrada Analógica	6	16	12
Corrente Contínua por Pino I/O	40 mA	40 mA	Total < 130 mA
Corrente Contínua por Pino	50 mA 3,3 V	50 mA 3,3 V	800 mA 3,3 V / 5 V
Memória Flash	32 KB dos quais 0,5 KB usados pelo bootloader	256 KB dos quais 8 KB usados pelo bootloader	512 KB total
SRAM	2 KB	8 KB	96 KB (64+32)
EEPROM	1 KB	4 KB	-
Velocidade do Oscilador	16 MHz	16 MHz	84 MHz

Um dos problemas das plataformas Arduino está na baixa corrente disponível nos pinos de saída. Com o máximo de 800 mA, a mesma não é suficiente para acionamentos industriais, que comumente requerem 3 A.

Criado a partir da mesma base dos Arduinos, há o Maple fabricado pela empresa LeafLabs. Essa placa foi criada como uma alternativa aos Atmel, possuindo um microprocessador STM32F103RB de 32-bit baseado em sistema RISC. O Maple trabalha com voltagem de 3,3 V e a alimentação pode ser entre 3,0 e 12 V. A placa possui 39 pinos de entrada I/O digitais e 16 de entrada analógicas, A memória é dividida em 128 KB de Flash e 20KB de SRAM, a saída é de 800 mA em 3,3 V com suporte para modo de espera. Uma diferença para com os Arduinos é a possibilidade de programação via JTAG, o qual é a denominação comum para a norma IEEE 1149.1 (*LEAFLABS, 2013*).

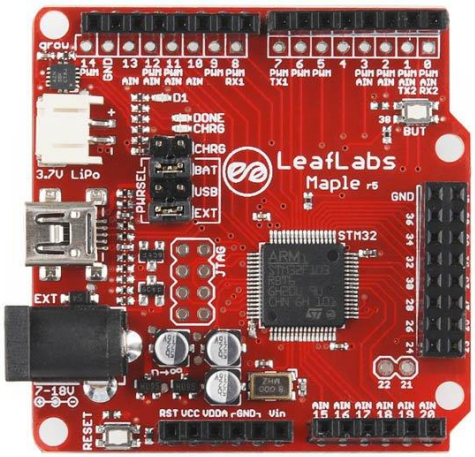

Figura 2.17 – Placa de controle LeafLabs Maple.

Com as melhorias no hardware de celulares, o que os transformam em microcomputadores simples, capazes de realizar tarefas rotineiras, pensou-se na utilização desses sistemas para a criação de microcontroladores. Uma das placas que se utilizam dessa perspectiva é a BeagleBone. Criada em 2011, essa placa se utiliza de um processador Sitara da Texas Instruments, baseado em ARM Cortex-A8 que é similar ao utilizado no celular iPhone 4. Na versão do BeagleBone, este trabalha com uma velocidade do oscilador de 720MHz, além do processador principal ele possui um coprocessador de gráficos 3D PowerVR SGX530, um ARM Cortex-M3 para realizar o gerenciamento de energia e duas CPUs RISC de Programáveis em Tempo Real de 32-bit. A placa possui dois conjuntos de 46 pinos I/O, desses podem ser usados dois I²C, cinco UART, um SPI, I²S, CAN, 66 GPIO de 3,3 V e sete ADC. Ele suporta armazenamento em cartão SD e a programação é feita dentro da própria placa. Como conexões são disponibilizadas uma porta ethernet e

uma porta USB que pode também ser usada como fornecimento de energia para a placa (*BEAGLEBOARD, 2013*).

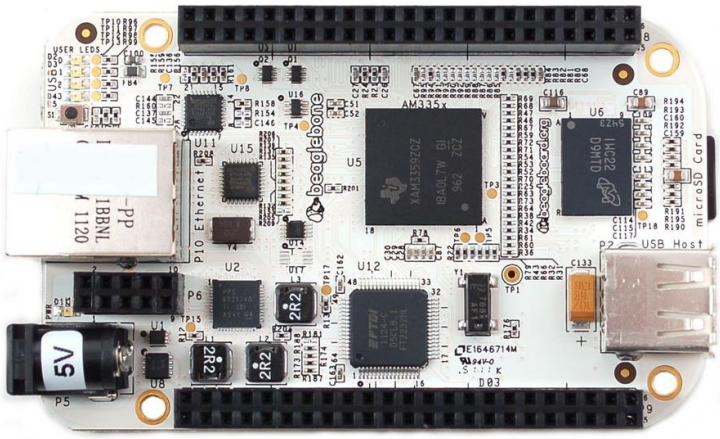

Figura 2.18 – Placa BeagleBone comercializada pela empresa BeagleBoard. (*BEAGLEBOARD, 2013*)

Outra placa computacional disponível é a Raspberry Pi desenvolvida pela Fundação Raspberry que começou a vender o modelo B em Fevereiro de 2012. O dispositivo é baseado no chip Broadcom BCM2835, esse chip integrado conta com uma CPU ARM1176JZF-S de 700MHz e 32-bit, juntamente com uma GPU Videocore 4 para permitir a utilização em displays com resoluções de até 1920x1200. A placa não possui sistema operacional nativo, sendo necessária a instalação de um dos sistemas que estão disponíveis. O armazenamento do sistema e dados é feito através de cartões SD, porém o sistema conta com 512 MB de SRAM. Como a placa suporta um sistema operacional nativo baseado em Linux, não requer nenhum computador ou dispositivo para acessá-lo. Como conexões ele possui uma porta HDMI, Video Composto RCA, duas portas USB, áudio via conector TRS P2 de 3,5mm e Ethernet 10/100. A placa conta com oito GPIO, UART, I^2C, SPI com dois seletores, I^2S, saída 3,3 V, 5 V e terra. Como fonte de energia ele requer suprimento de 5 V que pode ser obtido tanto de uma porta micro-USB quanto através dos pinos de GPIO. Uma limitação do Raspberry Pi é a falta de um chip RTC requerendo que a temporização seja feita na rede, no entanto é possível adicionar um RTC através do uso de I^2C (*RASPBERRY PI, 2013*)

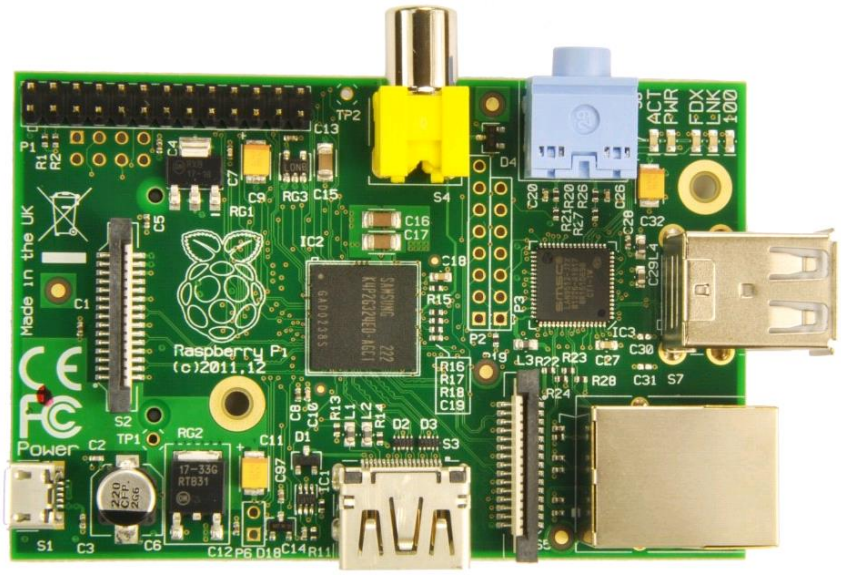

Figura 2.19 – Placa computacional Raspberry Pi.

Capítulo 3 - Metodologia

A fim de desenvolver um sistema de controle autônomo que possa ser usado em placas controladoras de baixo custo, é necessário que o mesmo seja capaz de funcionar com poucos recursos computacionais, visto que em tais sistemas, muitas vezes, não temos o mesmo poder de processamento de um microcomputador.

Apesar dos avanços na comunicação remota de dados no Brasil, ainda está disponível uma banda larga móvel muito precária, diversas vezes não chegando aos 0,4 Mb/s (50 kB/s) sendo que a tecnologia HDSPA (3.5G) permitiria taxas de até 14,4 Mb/s (1800kB/s) (*TAMUSIUNAS, 2013*) Por esse motivo, a compressão de dados se torna um fator muito importante. Contudo, um sistema separado de compressão e auditoria podem consumir recursos computacionais excessivos, inclusive causando o congelamento das funções de controle no caso de sistemas monotarefa. Como a maioria dos sistemas de auditoria para detecção de falhas é baseado em equações, os algoritmos de compactação poderiam ser uma alternativa para a geração dos parâmetros necessários às funções de auditoria, gerando assim um sistema integrado requerendo menos recursos para funcionar. Sendo Assim é necessária uma escolha minuciosa para que o sistema funcione corretamente.

3.1 Escolha dos Métodos e Equipamentos

A avaliação para os métodos e equipamentos necessários deve abrangir três áreas separadas, Compressão, Auditoria e por fim Equipamentos. Como o intuito deste trabalho é o de utilização com sistemas de baixo custo, os principais pontos serão o Requerimento Computacional e Custo dos equipamentos. Dessa forma o peso destes será maior perante os demais itens. O sistema de avaliação será dividido em itens cada qual com notas atribuídas variando de 0 a 5. Nesse, 0 significa que a área em questão não é atendida para o sistema proposto e 5, que a área é plenamente satisfatória. Cada área está detalhada em sua respectiva seção.

3.1.1 Comparativo entre Técnicas de Compressão de Dados

Para que seja possível utilizar as técnicas de compressão para o fornecimento de parâmetros para a auditoria de plantas, precisamos considerar diversos fatores, como:

- Demanda computacional, onde é avaliada a carga computacional necessária para a realização do algoritmo, sendo uma nota baixa correspondente a um algoritmo que consome muito do processamento da unidade.
- Requisitos computacionais, neste são avaliados os requisitos de software, tais como bibliotecas matemáticas e capacidade de implementação das mesmas, sendo uma nota baixa correspondente a um algoritmo necessita várias bibliotecas ou que seja difícil de programa-las.
- Fornecimento de parâmetros, este é a avaliação da facilidade de geração de parâmetros uteis para um sistema de auditoria, sendo uma nota baixa relativa a um algoritmo que não fornece parâmetros, requerendo que estes sejam obtidos de outra forma.
- Erros de reconstrução, neste é avaliado a fidelidade entre a curva real e a curva gerada pelo algoritmo de compressão, uma nota baixa corresponde a uma perda elevada de fidelidade.
- Capacidade de uso *real-time*, onde é avaliada a quantidade de dados necessários para se gerar os pontos da curva comprimida, neste item uma nota baixa significa que são necessários muitos dados para essa geração, o que aumenta significativamente a latência entre um evento e a identificação do mesmo.
- Razão de Compressão, aqui é avaliado o quanto é possível reduzir a quantidade de dados relativos aos originais, neste item uma nota baixa significa que os dados gerados têm o tamanho aproximado do conjunto original.

Como os algoritmos BC e BS foram propostos no mesmo artigo e têm poucas diferenças, podem ser abordados como um único método. Na Tabela 3.1 e Tabela 3.2 serão mostrados os valores atribuídos a cada método:

Tabela 3.1 – Avaliação dos métodos de compressão sem e com perda de dados.

	Sem Perda de Dados			Com Perda de Dados			
	MSE	Lempel-Ziv	BWT	SDT	BC/BS	SLIM	PLOT
Demanda Computacional	3	4	3	4	4	4	4
Requerimentos Computacionais	4	4	3	5	5	4	3
Fornecimento de Parâmetros	2	0	0	4	3	4	3
Erros de Reconstrução	5	5	5	3	2	3	4
Capacidade de uso *Real-Time*	0	1	1	4	5	4	4
Razão de Compressão	1	1	1	3	3	3	4
Média	2,50	2,38	2,00	**3,88**	3,63	3,75	3,63

Tabela 3.2 - Avaliação dos métodos de compressão com base em tendência.

	Com Base em Tendência				
	Quantização	Transformada	Wavelets	MDCT	Spline
Demanda Computacional	1	3	3	3	3
Requerimentos Computacionais	2	3	2	1	3
Fornecimento de Parâmetros	4	4	4	4	4
Erros de Reconstrução	4	4	5	4	4
Capacidade de uso *Real-Time*	3	3	2	3	2
Razão de Compressão	5	4	4	4	4
Média	3,00	3,50	3,38	3,25	3,38

Através dessa avaliação, podemos observar que nos métodos do tipo de compressão sem perda de dados, temos a melhor reconstrução possível, contudo é extremamente complexo para operar em *real-time*, sendo necessário o uso de pequenos blocos de dados para serem comprimidos individualmente. Além disso, a razão de compressão é baixa se comparada com os outros métodos, além de ser difícil de obter parâmetros do processo tais como a tendência, sem que se tenha que descomprimir os dados.

Os algoritmos com perda de dados fornecem parâmetros de forma simples e rápida, por terem cálculos simplificados à demanda computacional é baixa e a maioria dos dispositivos suporta as funções necessárias para implementação dos

mesmos. Em geral o uso dos parâmetros desses algoritmos pode ser direto. Contudo esses algoritmos são os que têm maior erro de desconstrução, porém para processos, onde não se espera grandes variações das variáveis medidas, isso pode não ser muito problemático dado, que os erros de reconstrução diminuem com a estacionariedade do processo.

Para algoritmos com base em tendência, existem parâmetros muito bons para serem utilizados em auditoria de plantas, contudo a demanda computacional pode ser alta, sobretudo se os dispositivos não tiverem implementações próprias para o cálculo dos somatórios comuns a esses métodos. Além disso, o seu uso em *real-time* fica prejudicado pelo fato de requererem blocos para serem calculados. Quando temos dispositivos computacionais com alta capacidade, estes se tornam muito úteis pela capacidade alta de compressão e capacidade de reconstrução sem muitos erros, mesmo quando tomamos curvas de processos com variações elevadas.

Um fator à parte do algoritmo SDT é que o mesmo hoje se encontra em domínio público o que também permite uma redução dos custos da implantação em processos industriais.

Na próxima seção, serão apresentadas alternativas existentes para a realização de autodiagnostico de plantas.

3.1.2 Comparativo entre os Algoritmos de Diagnóstico

A fim de selecionar que técnicas ou conjunto de técnicas para realizar a auditoria de plantas deve-se definir qual algoritmo será a fonte das informações necessárias para os cálculos. Para isso devemos observar os seguintes itens:

- Demanda computacional, onde é avaliada a carga computacional necessária para a realização do algoritmo, sendo uma nota baixa correspondente a um algoritmo que consome muito do processamento da unidade.
- Requisitos computacionais, neste são avaliados os requisitos de software, tais como bibliotecas matemáticas e capacidade de implementação das mesmas, sendo uma nota baixa correspondente a um algoritmo necessita várias bibliotecas ou que seja difícil de programa-las.
- Demanda de dados, este item é a avaliação do comprimento da janela de dados necessária para o algoritmo, notas baixas significam que são necessários muitos dados, o que acarreta em uma latência maior entre os eventos e a identificação dos mesmos.
- Facilidade de detecção de falhas, para essa analise deve ser observado a complexidade da árvore de decisões para ser programada no algoritmo, notas baixas significa a presença de uma árvore complexa, esta pode acarretar que

determinados estados possam estar indefinidos no algoritmo por necessitarem um complexo estudo.

- Facilidade de implementação, neste é observado a complexidade do algoritmo, notas baixas significam que o algoritmo deve sofrer muitas adaptações para poder ser utilizado em uma linguagem de programação, exemplos seriam derivadas parciais, uma vez que a maioria das linguagens não suportam tais cálculos de forma direta.

Na Tabela 3.3 são mostrados os valores atribuídos a cada um dos algoritmos.

Tabela 3.3 – Avaliação dos métodos de diagnostico de falhas.

	Qualitativo		Quantitativo				
	Simples	Híbrido	FDI	PCA	MKPCA	Wavelets	BEMD/GDA /RMV
Demanda Computacional	5	4	3	3	2	2	0
Requerimentos Computacionais	5	4	4	4	2	3	2
Demanda de Dados	2	3	2	3	3	4	4
Facilidade de Detecção de Falhas	1	2	3	3	3	4	5
Facilidade de Implementação	3	4	4	4	3	3	2
Média	3,29	3,43	3,00	3,29	2,57	3,14	2,43

Com essa avaliação, podemos ver que os métodos qualitativos permitem o uso de sistemas simplificados, enquanto métodos muito elaborados necessitam um microcomputador, tanto pela capacidade de processamento elevada que eles exigem, quanto pela disponibilidade de funções mais específicas como integrais e operações com matrizes.

Com relação aos requisitos de dados, métodos complexos podem verificar falhas através de somente os dados do próprio sensor, contudo os métodos qualitativos requerem muitos dados adicionais para poderem fazer o isolamento e definição da falha, por exemplo, em uma tubulação se um sensor registrar um valor nulo seria preciso os dados dos sensores anteriores na linha, para saber se a tubulação está obstruída ou a falha foi do próprio sensor.

Da mesma forma a detecção das falhas se torna mais complexa, pois a tradução de todas as condições depende de um algoritmo com diversos entrelaçamentos em que para uma planta com muitos equipamentos pode ser demasiado complicado e de difícil validação.

Por requererem menos funções que algoritmos elaborados, a implementação em algoritmos qualitativos e quantitativos simples é facilitada tornando-se uma boa alternativa para pequenas plantas descentralizadas. Devido à facilidade de implementação e aos requisitos computacionais reduzidos, mesmo com dificuldade de detecção os algoritmos qualitativos híbridos são a melhor alternativa para observar as falhas em pequenas plantas industriais e consequentemente serão abordados no trabalho.

3.1.3 Comparativo entre Dispositivos de Controle

Nesta seção será feita uma avaliação entre os dispositivos de controle comercialmente disponíveis e apresentados anteriormente. Nesta, os seguintes itens devem ser avaliados:

- Dificuldade de implementação, neste é avaliado a facilidade de um método matemático ser programado no dispositivo onde uma nota baixa significa que a programação dos métodos matemáticos é complexa.
- Capacidade matemática, para este, foi analisado a poder de processamento de operações matemáticas, neste, notas baixas demonstram um dispositivo que não é capaz de realizar muitas operações matemáticas por segundo.
- Conexões externas, a avaliação deste foi feita com relação ao acesso do dispositivo, através de outros dispositivos, notas baixas no mesmo, mostra que a conexão para acesso por outros dispositivos é limitada.
- Conexões GPIO, onde é avaliado o número de conexões para dispositivos e sensores, neste, notas baixas significa que poucos dispositivos podem ser conectados.
- Facilidade de acionamento, neste item é avaliada a comunicação entre o equipamento e os dispositivos, onde notas baixas significam que seria não seria possível comunicar diretamente com os dispositivos conectados.
- Necessidade de adaptação, para este é levado em conta o que é necessário para que os acionamentos sejam possíveis.
- Custo foi avaliado em relação ao preço do equipamento, onde notas baixas significam um custo elevado.

Os resultados podem ser vistos na Tabela 3.4 e Tabela 3.5.

Tabela 3.4 – Avaliação dos dispositivos industriais de controle de plantas.

	PIC	PLC	Datalogger	RTU	Fieldlogger
Dificuldade de Implementação	2	3	3	3	4
Capacidade Matemática	3	3	3	3	4
Conexões Externas	1	2	3	3	3
Conexões GPIO	4	4	3	3	3
Facilidade de Acionamentos	3	4	4	3	3
Necessidade de Adaptação	4	4	3	3	4
Custo	3	2	2	2	1
Média Ponderada	2,17	2,42	2,33	2,17	2,42

Tabela 3.5 – Avaliação dos dispositivos de código aberto disponíveis para controle.

	Arduino	Maple	BeagleBone	Raspberry Pi
Dificuldade de Implementação	3	3	3	4
Capacidade Matemática	3	3	4	5
Conexões Externas	3	3	4	5
Conexões GPIO	3	3	3	2
Facilidade de Acionamentos	2	2	2	2
Necessidade de Adaptação	2	2	2	2
Custo	4	4	3	4
Média Ponderada	2,08	2,08	2,25	**2,58**

As principais vantagens dos dispositivos comerciais são a facilidade de fazer acionamentos de dispositivos e o número de conexões de deste tipo, alguns modelos permitem mais de 100 conexões GPIO, contudo, são em geral dispositivos mais caros. Com os fieldloggers, esse valor pode passar de US$ 1500,00, o que para um grande número de plantas pode ser tornar excessivo. Outro fator é a capacidade matemática. Dispositivos mais simples como os PIC tem uma limitação matemática muito grande visto que normalmente operam basicamente como controladores PID.

As placas de código aberto tem um grande desenvolvimento por permitirem que comunidades de usuários possam contribuir, algumas chegam ter um sistema

operacional de padrões comercial embarcado, os quais podem realizar tarefas matemáticas mais complexas que os dispositivos comerciais. Contudo, por serem projetadas para uso doméstico as suas conexões de GPIO são muito limitadas em alguns casos necessitando de um *driver* externo para realizar o acionamento de motores e válvulas dado que eles não suportam uma corrente muito alta nos pinos. A principal vantagem destas placas está no baixo custo, que muitas vezes é inferior à US$ 50,00. No caso dos Raspberry Pi, suas conexões RCA e de áudio permitem que seja feita inclusive um sistema HMI de baixo custo, tornando a operação de uma planta mais simples.

Pela facilidade de programação e disponibilidade do processamento matemático, juntamente com a variedade de conexões disponíveis juntamente com o baixo custo, para este trabalho optou-se pela placa de código aberto Raspberry Pi, contudo deve ser criado um driver para o acionamento industrial e um conversor ADC para utilização na planta.

3.2 Modificações do Algoritmo SDT

Para poder utilizar o algoritmo SDT no fornecimento de dados para o sistema de auditoria, o mesmo deve ser adaptado, de modo a fornecer os dados de tendência das variáveis de processo, independente do estado de preenchimento do buffer do módulo de compressão. Com essa finalidade, o algoritmo base do SDT precisa ser estruturado em blocos.

O primeiro destes módulos faz o cálculo da tendência da variável tomando por base os dados anteriores. Este cálculo é demonstrado na Equação 3.1, na qual, $Tendencia$ é uma matriz com S linhas e Pos colunas, sendo S o número de sensores, e Pos o número de colunas, assim S_a será o índice do ponto atual e Pos_a o índice da coluna atual no buffer; P_a corresponde ao ponto atual; P_i é o ponto inicial e P_j número de pontos intermediários entre P_i e P_a.

$$Tendencia_{(S_a,Pos_a)} = \frac{P_a - P_i}{\Delta t_{P_a - P_i}} + \sum_{n=1}^{P_j} \left(\frac{P_n - P_i}{\Delta t_{P_n - P_i}} - \frac{P_a - P_i}{\Delta t_{P_a - P_i}} \right) \qquad (3.1)$$

Com isso, ele acessa e armazena em uma variável do tipo matriz as tendências das medições de todos os sensores, permitindo que o módulo de auditoria possa ter acesso aos dados para avaliação. No outro bloco, o SDT acessa esses dados para determinar a situação do ponto para a compressão e reinício do buffer de dados, tendo como finalidade permitir um melhor gerenciamento da memória, como demonstrado na Figura 3.1, diminuindo possíveis problemas relativos ao *garbage collection* (*CHENG,2001*).

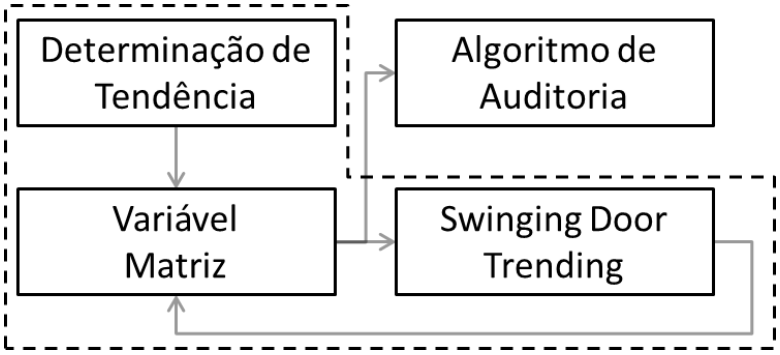

Figura 3.1 – Estrutura do sistema proposto..

Contudo, essa modificação requer que o algoritmo de auditoria conte com um buffer interno para armazenar os dados históricos recentes. Uma alternativa simples é o deslocamento dos dados em uma matriz circular de dimensão fixa. Essa forma armazenamento de dados permite que a memória se mantenha praticamente constante, apesar de requerer um pouco mais de processamento. O sistema de gerenciamento dessa variável no algoritmo de auditoria pode ser visto na Figura 3.2:

Pos. / Sen.	1	2	3	...	n_{-2}	n_{-1}	n
1	V(1,1)	V(2,1)	V(3,1)	...	$V(n_{-2},1)$	$V(n_{-1},1)$	V(n,1)
2	V(1,2)	V(2,2)	V(3,2)	...	$V(n_{-2},2)$	$V(n_{-1},2)$	V(n,2)
...	...	...	...	...	...	...	...
m-1	$V(1,m_{-1})$	$V(2,m_{-1})$	$V(3,m_{-1})$	...	$V(n_{-2},m_{-1})$	$V(n_{-1},m_{-1})$	$V(n,m_{-1})$
m	V(1,m)	V(2,m)	V(3,m)	...	$V(n_{-2},m)$	$V(n_{-1},m)$	V(n,m)

Leitura Sensor 2 - Valor

Pos. / Sen.	1	2	3	...	n_{-2}	n_{-1}	n
1	V(1,1)	V(2,1)	V(3,1)	...	$V(n_{-2},1)$	$V(n_{-1},1)$	V(n,1)
2	V(**2**,2)	V(**3**,2)	V(**4**,2)	...	$V(\mathbf{n}_{-1},2)$	V(**n**,2)	$V(\mathbf{n}_{+1},2)$
...	...	...	...	...	...	...	...
m-1	$V(1,m_{-1})$	$V(2,m_{-1})$	$V(3,m_{-1})$	...	$V(n_{-2},m_{-1})$	$V(n_{-1},m_{-1})$	$V(n,m_{-1})$
m	V(1,m)	V(2,m)	V(3,m)	...	$V(n_{-2},m)$	$V(n_{-1},m)$	V(n,m)

Figura 3.2 – Funcionamento da variável cíclica da auditoria sendo ‘n’ o enésimo valor e ‘m’ o m-ésimo sensor.

Neste caso, sempre que há um acréscimo de dados na variável, todos os dados anteriores são deslocados para a esquerda na tabela, onde o primeiro valor será descartado e o último será o valor atual armazenado.

Outra modificação necessária ao algoritmo SDT está no modo como os ângulos são avaliados. Como um parâmetro adicional para a auditoria, torna-se útil manter todos os cálculos dos ângulos intermediários, de forma que também possa ser avaliado se realmente existe uma mudança de tendência, assim como a rapidez

com que ela ocorre. Outro possível uso está na detecção de *outliers*. Assim o SDT assume a forma demonstrada na Figura 3.3:

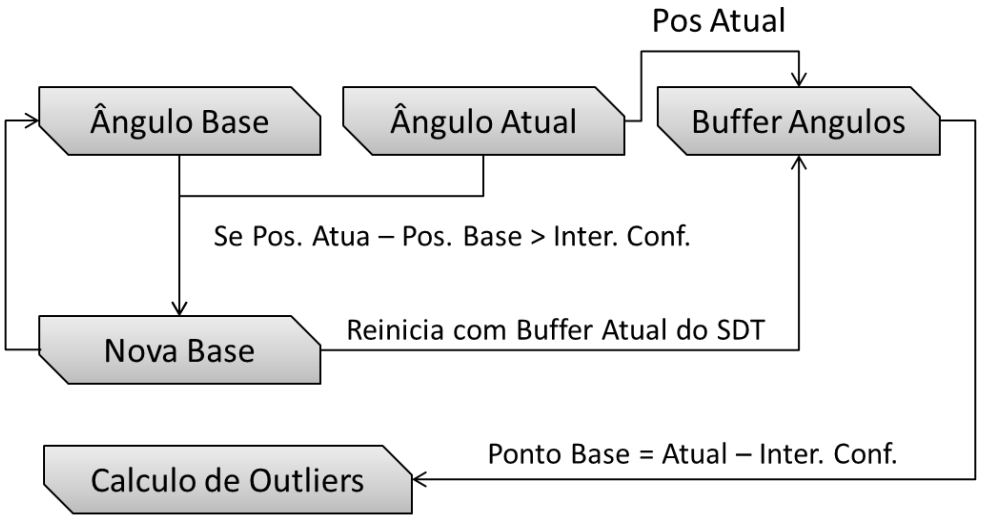

Figura 3.3 – Estrutura do SDT Modificado

3.3 Sistema de Auditoria

Uma das vantagens dessa modificação no algoritmo do *Swinging Door Trending* está na possibilidade de detecções rápidas dos *outliers*, através do uso do histórico de cada bloco. Se for considerado que um processo esteja operando em estado estacionário a função obtida do SDT pode ser reduzida a uma constante uma vez que quando operamos em estado estacionário o ângulo entre um ponto e outro tende a zero:

$$\lim_{\alpha \to 0}(\alpha t + V_p) = V_p \qquad (3.2)$$

Onde α é o ângulo do SDT, t o tempo e V_p a variável de processo.

Contudo, o mesmo não é verdade quando se observa processos com mudanças no valor de variáveis ou em estados de *start-up* que não são rápidos. Nestes casos, tem que observar se a inclinação da rampa é constante; assim, para considerar que a taxa de variação seja constante quando α for constante, que pode ser constatado por:

$$\alpha_x - \alpha_{x-1} = 0 \qquad (3.3)$$

Ao se analisar a igualdade, se um processo tiver uma variação constante então se pode definir que o ângulo entre quaisquer dois pontos da reta tem taxa constante, dessa forma:

$$\alpha_{x1} - \alpha_{x2} = \alpha_{x3} - \alpha_{x4} \qquad (3.4)$$

Como um *outlier* se apresenta como um ponto fora desse padrão, para detectá-lo, pode-se usar os somatórios dos ângulos. Caso os mesmos sejam constantes para

as vizinhanças, ou seja, os valores para os pontos da região anterior sejam iguais ou muito próximos dos pontos da região posterior, o resultado seria expresso por:

$$\sum_{i=m}^{n} \alpha_{x+i} \approx \sum_{i=m}^{n} \alpha_{x-i} \quad (3.5)$$

Onde α é o ângulo da função SDT, x responde pela posição do ponto atual, m o intervalo de tolerância, que para um ponto isolado será 1, e n o intervalo de análise.

Assim, para detectar um ponto *outlier*, o ângulo deste deve ser maior que a média de suas vizinhanças, conforme a seguinte equação:

$$\frac{\sum_{i=1}^{n} \alpha_{x+i} + \sum_{i=m}^{n} \alpha_{x-i}}{2n} < \frac{\sum_{i=0}^{m-1} \alpha_{x-i}}{i} \quad (3.6)$$

Com isso podemos detectar os *outliers* e ao alterar o intervalo de detecção podemos inclusive isolar casos onde mais de um ponto é afetado, alterando-se o intervalo de m.

O cálculo do SDT fornece dois parâmetros para a análise de dados. Pelo ângulo pode-se determinar a trajetória atual do processo e o tipo da exceção que permite avaliar o possível estado futuro da tendência. Ao se tomar o caso de um processo em que a variável avaliada começa a ter um uma mudança elevada no valor de α_x, pode-se avaliar um segundo parâmetro, que é dado pelo sinal positivo do parâmetro de exceção para um máximo, ou negativo para o caso de mínimo, assim pode-se verificar a ocorrência de estabilização. Seja uma variável de processo em estado estacionário que sofra um distúrbio, ocasionando elevação do seu valor. No estágio inicial, o algoritmo gerará uma exceção negativa, contudo, na estabilização a exceção será positiva, como demonstrado na Figura 3.4.

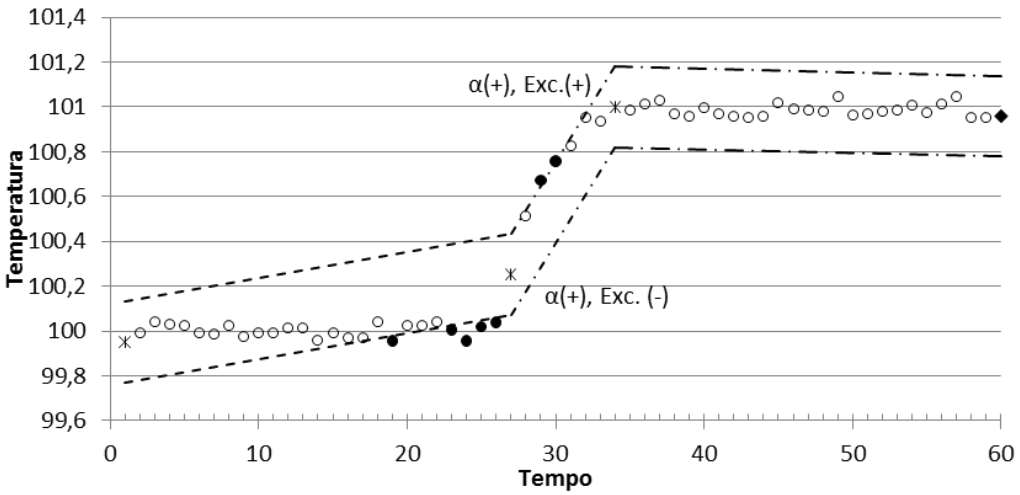

Figura 3.4 – Exemplo de uma mudança positiva na variável avaliada pelo algoritmo SDT, onde se tem o ponto atual (◆), pontos amostrados não gravados (○), pontos gravados (✶) e pontos de exceção (●)

Portanto se multiplicarmos o ângulo pelo o sinal da exceção, podemos definir a possível estabilização sem requerer que sejam necessários pontos posteriores, através da seguinte forma:

$$\alpha_x \times e_c > 0 \quad (3.7)$$

Onde α_x é o ângulo no ponto de exceção e e_c responde pelo valor unitário da exceção.

A vantagem de uma análise nesse formato está no fato de que a mesma será válida também para casos de diminuição no valor da variável, neste caso mesmo com a variação negativa dos dados o ângulo será menor quando o processo tende ao estado estacionário.

Ao se fazer a análise através da Equação 3.7 para o caso da Figura 3.4, no início da mudança obterá em ambas um resultado negativo, porém na estabilização, ambas assumirão valor positivo. Um problema dessa forma de análise está no fato de que o algoritmo SDT deve ser bem ajustado para que detecte somente as mudanças, caso a tolerância seja muito baixa durante a trajetória até a estabilização, podem, haver pontos de exceção indesejados reduzindo a confiabilidade dessa análise.

Para a detecção de possíveis falhas de sensor, um método possível está no monitoramento de mudanças bruscas dos mesmos. Através da tolerância utilizada para o algoritmo SDT, pode-se interpretar que um sensor possa estar em estado de falha quando o valor atual do ângulo for muito maior que o ângulo anterior. Assim generalizando para os casos de aumento e diminuição da variável, pode-se empregar a seguinte inequação para se determinar os eventos de falha:

$$|\alpha_x - \alpha_{x-1}| > T_{SDT} \times M_{LIM} \quad (3.8)$$

Onde T_{SDT} responde pela tolerância do SDT e M_{LIM} o multiplicador para assunção de falha.

Com isso pode-se determinar um novo parâmetro de ajuste que pode ser estabelecido com uma simples alteração de set-point ou na partida do processo, como demonstrado na Figura 3.5. Nesta são mostrados os pontos do SDT e o ângulo quando temos um processo que onde possivelmente há sinais de instabilidade. Esse processo com instabilidade pode então ser avaliado diretamente pelo módulo do ângulo provido pelo SDT.

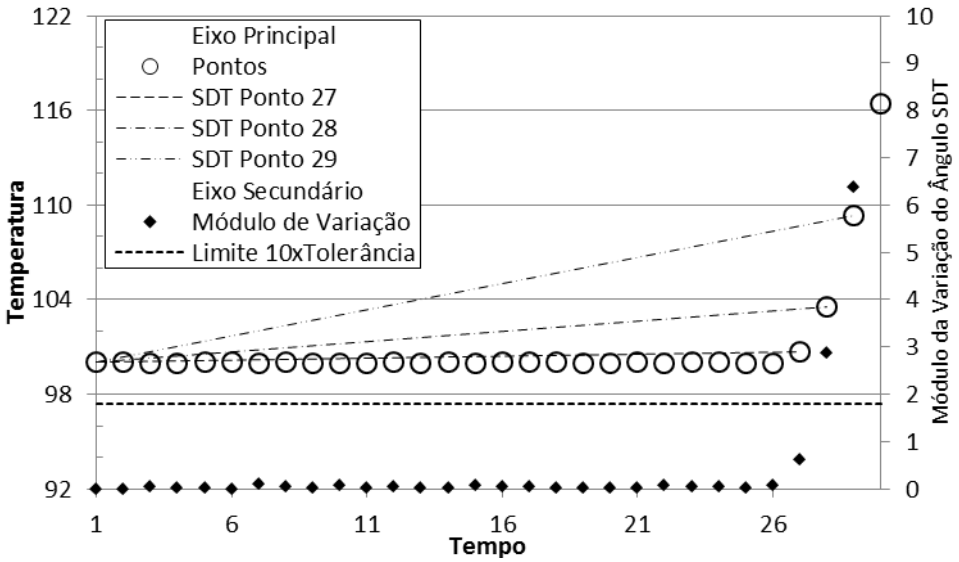

Figura 3.5 - Exemplo de análise de módulo de variação para o ângulo do algoritmo SDT.

Contudo, o método acima pode não ser eficaz quando possuímos processos auto-catalíticos como, por exemplo, na geração do policarbonato. Assumindo que existam sensores correlacionados como, por exemplo, um sensor de temperatura em um reator e um de vazão na entrada do mesmo, assim dispondo dos valores de ambos pode-se usar a mesma analogia da Equação 3.6, logo a variação dos sensores posteriores e anteriores, para o caso de uma falha se sensor, será mínima. Com isso pode-se definir a seguinte equação para o caso de falha.

$$\Delta\alpha_S \gg \frac{\sum_{n=x}^{\Delta Pt} \Delta\alpha_{S_{n\pm\Delta Pt}}}{\Delta Pt} \qquad (3.9)$$

Onde α_S corresponde ao ângulo de um sensor em uma linha, $\Delta\alpha_{S_{n\pm\Delta Pt}}$ corresponde a sensores anteriores ou posteriores.

Assim, quando a equação acima é verdadeira, pode-se determinar que o problema trata-se uma falha de sensor e não a uma causa intrínseca do processo.

Finalmente, pode-se fazer a detecção direta de falha. Grande parte dos sensores comercialmente disponíveis tem interfaces que já possuem mecanismos de detecção básica, como sensores com saída no padrão 4-20 mA. Assim, em operação normal o sensor produzirá saídas nessa faixa, e qualquer valor fora da mesma caracterizaria uma falha do mesmo. Dessa forma, pode-se simplificar o processo de análise para tais casos.

3.4 Algoritmo Integrado

A placa Raspberry Pi possui nativamente um interpretador Python, o qual foi empregado para rodar o algoritmo.

Uma das vantagens das linguagens interpretadas, como Java, Python e Lua reside na facilidade de implementação, pois a existência de comandos simplificados torna acessível à customização dos algoritmos. Por outro lado, uma dificuldade dessas linguagens está no desempenho, já que, ao contrário de linguagens compiladas, elas não são executadas diretamente, e necessitam, portanto, ser enviadas ao interpretador, o qual traduzirá os comandos em linguagem de máquina, para só então ser executado.

O primeiro fator a ser determinado na criação do algoritmo é a respeito do armazenamento das variáveis. No Python, existem três tipos básicos: *Variable*, que corresponde a um valor único, *Array*, que guarda uma lista de variáveis e por fim *List*, que, diferentemente de matrizes, podem apresentar formas complexas (matrizes n-dimensionais).

Desse modo, na declaração de uma variável deve-se definir o formato da mesma. No algoritmo descrito neste trabalho, as variáveis são matrizes bidimensionais e, para melhorar o desempenho, não foi utilizada nenhuma *List* com mais de três dimensões.

Outro fator que requer atenção está nos subtipos das variáveis. Ao contrário das linguagens compiladas, o Python define os subtipos, como *Interger*, *Float* e *String*, durante a execução. Isso pode gerar problemas matemáticos, uma vez que um cálculo pode assumir diferentes resultados de acordo com os subtipos de variável, como demonstrado na Tabela 3.6.

Tabela 3.6 – Precedência de subtipos de variável em linguagem Python

`Var1=10, Var2=20`			`Var1=10.0, Var2=20`			`Var1=10, Var2=20`		
`Divisao=Var1/Var2`			`Divisao=Var1/Var2`			`Divisao=Var1/Var2*1.0`		
`print Divisao`			`print Divisao`			`print Divisao`		
Resulta:			Resulta:			Resulta:		
`Divisao=0`			`Divisao=0.5`			`Divisao=0.5`		
Divisao (Int)	Var1 (Int)	Var2 (Int)	Divisao (Float)	Var1 (Float)	Var2 (Int)	Divisao (Float)	Var1 (Int)	Var2 (Int)

Com isso, pode-se forçar o aplicativo a gerar uma variável *Float* ao multiplicarmos uma equação por $1{,}0$, sobrepondo o fato de que as variáveis envolvidas sejam todas de tipo *Integer*. Deste modo, assegura-se que os cálculos correspondam ao esperado, mesmo quando as variáveis de entrada sejam todas representadas por números inteiros.

Como o objetivo do trabalho é a criação de um sistema de controle e auditoria de plantas para placas de baixo custo, deve-se garantir que a estabilidade do algoritmo não seja afetada durante a execução. Deste modo, a separação dos módulos se torna imprescindível. Uma das formas de se fazer isso é através de *threads*, o que possibilita manter os blocos funcionando autonomamente mesmo que outro esteja em falha. Deste modo, um módulo simples de PID não seria afetado, por exemplo, por erros de execução na interface de modificação de parâmetros.

Para que um determinado *Thread* possa se comunicar com outro, as variáveis têm de ser declaradas como globais, após o mesmo ser iniciado. Dessa forma, para cada função que deve ser separada do algoritmo global, é necessário também o uso de nomes únicos de variável para não haver conflito de valores. Os threads também funcionam de forma assíncrona; assim, para haver um controle temporal devemos usar a biblioteca "*time*", de forma que os cálculos sejam feitos a partir do relógio do dispositivo.

3.4.1 Algoritmo de Controle PID

O módulo do PID é composto de duas partes, uma responsável pelos cálculos do controlador em si e outra responsável por atualizar os comandos externos, sejam do módulo de auditoria ou da interface de usuário.

A parte do cálculo em si baseia-se em um algoritmo simples através das seguintes equações

$$PID_x = \mathrm{K} \times \left(E_{P_x} + \frac{1}{\mathrm{T}_i} \times E_{I_x} + \mathrm{T}_d \times E_{D_x} \right) \times Dir \quad (3.10)$$

$$E_{P_x} = s_p - V_x \quad (3.11)$$

$$E_{I_x} = E_{P_x} + E_{P_{x-1}} \quad (3.12)$$

$$E_{D_x} = (E_{P_x} - E_{P_{x-1}}) \div (x - x_{-1}) \quad (3.13)$$

Onde E_{P_x} responde pelo erro entre o ponto e o *Set-Point* (s_p) no instante x, E_{I_x} é a integral do erro e E_{D_x} a derivada do erro, Dir é a direção de ação (inversa ou direta) e V_x o valor do ponto.

Os parâmetros iniciais do PID são acessados a partir de um arquivo externo e as atualizações externas não modificam esses parâmetros, permitindo que estes possam ser reiniciados. No mesmo conjunto de parâmetros são colocadas variáveis destinadas a armazenar valores anteriores para o cálculo das partes integral e derivativa.

Outro fator importante, é que ao carregar parâmetros de um arquivo, podemos facilmente criar conjuntos de variáveis para diversos estados de operação, tais como, partida de planta, operação normal, parada de planta e parâmetros de

segurança emergenciais, que podem ser carregados mais rápido que um operador poderia fazer.

No caso de uma parada emergencial da planta, foi definido que o programa irá carregar um conjunto de parâmetros específicos de estabilização como, por exemplo, a abertura total da válvula de escape de uma caldeira, a fim de aumentar a segurança de operação.

Outra possibilidade é a gravação do histórico de informações sobre o PID, tais como as alterações feitas pelos usuários. Dessa forma, eventuais alterações errôneas em parâmetros podem ser detectadas em análises posteriores e consequentemente revertidas.

3.4.2 Interface de Usuário

Para a interface de usuário foi utilizado um sistema de *Prompt de Comando* de modo a facilitar o uso do sistema. Primeiramente, o usuário seleciona os modos *Consulta* ou *Atualização*; logo após a seleção, deve-se selecionar o parâmetro a ser verificado, o sensor e por último, caso se esteja fazendo uma atualização, é requisitado o novo valor.

Para evitar erros e acidentes, qualquer valor inválido reinicia o processo sem que haja atualizações. Para os sensores, são considerados os parâmetros de PID do arquivo de configuração do algoritmo integrado, fazendo com que não seja possível selecionarem-se sensores inválidos. Além disso, é possível reiniciar-se os dados relativos aos parâmetros do PID e também selecionar se haverá gravação de um Log com o respectivo *timestamp*, de modo a haver também a data da verificação.

3.5 Funcionamento do Algoritmo

O funcionamento geral do algoritmo segue as seguintes rotinas.

O algoritmo final encontra-se separado em três diferentes threads. A finalidade desse modo é o de aumentar a confiabilidade do sistema, pois, mesmo que a auditoria ou a interface de usuário pare, o PID pode seguir atuando sem ser afetado. O mesmo foi feito para a auditoria para que enquanto o usuário esteja usando a interface, a auditoria continue funcionando. Conforme pode ser visto na Figura 3.6.

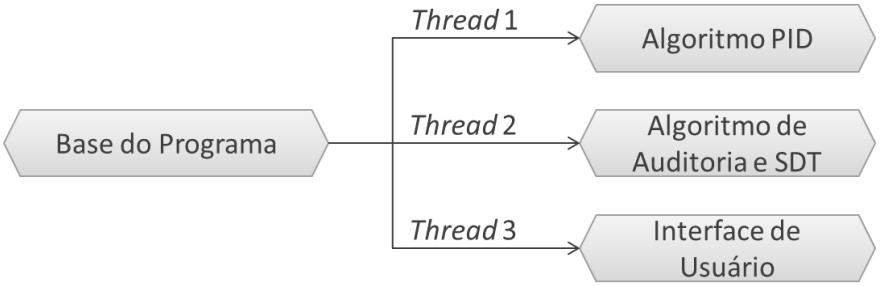

Figura 3.6 - Separação de módulos do algoritmo.

No inicio de operação os parâmetros são carregados para o PID, estes podem ser alterados pela interface de usuário. Durante a operação o sistema de auditoria, caso determinado pode carregar um conjunto de parâmetros para segurança da planta ou então atualizar os parâmetros caso haja instrução conforme visto na Figura 3.7.

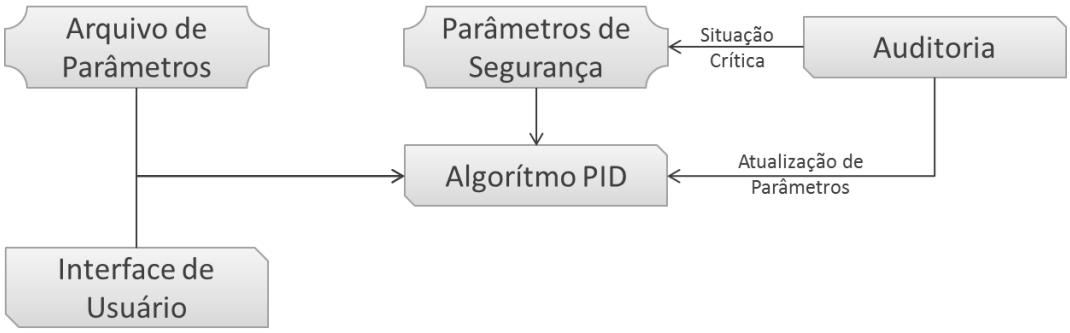

Figura 3.7 - Base do algoritmo PID.

A interface de usuário inicialmente apresenta um conjunto de opções contendo:

- Término do programa, quando se deseja desligar os equipamentos;
- Reinicio dos parâmetros para os valores iniciais;
- Verificação de parâmetros para cada um dos sensores;
- Alteração de parâmetros para Kp, Ki, Kd Set-Point.

Para a verificação é possível gerar um arquivo com o histórico dos parâmetros para poder ser avaliado.

Para cada um das atualizações ou verificações é selecionado somente um sensor, de modo que não seja possível alterar erroneamente a planta inteira.

A rotina é demonstrada na Figura 3.8.

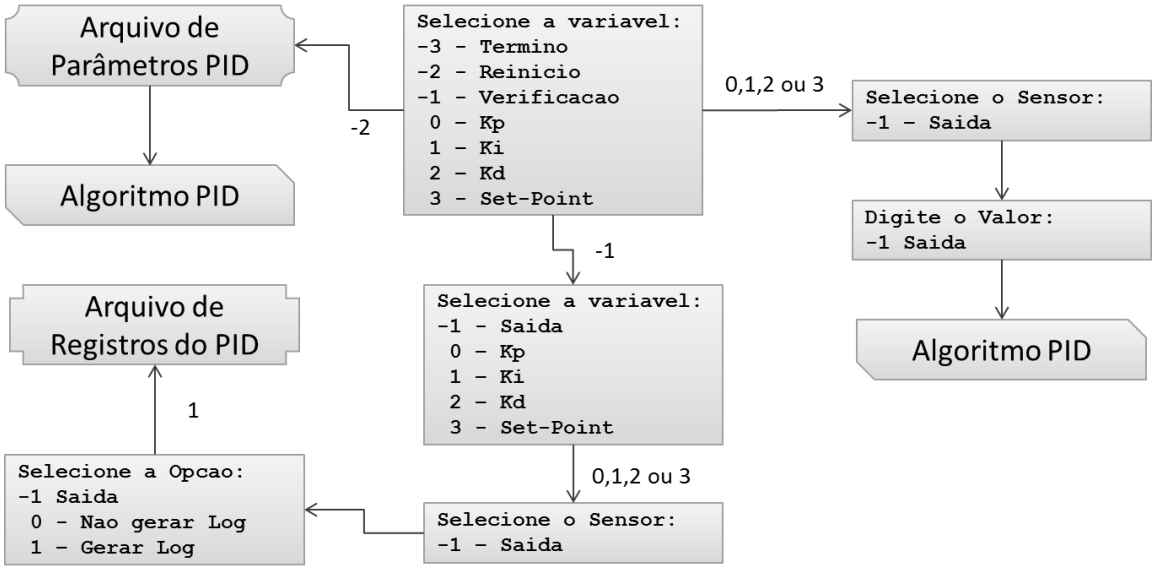

Figura 3.8 - Funcionamento da Interface de usuário.

Para a auditoria, primeiramente são carregados os parâmetros de diagnóstico do algoritmo. Logo após, as variáveis e logs são criadas, dando início à rotina de aquisição dos dados. Após isso é iniciado a Auditoria, para que as tomadas de dados sejam sincronizadas, existe um temporizador que coordena a tomada do estado do sensor. A tomada de dados armazena o dado no buffer do SDT e no buffer da auditoria, cuja função é a de prevenir que aquisições do SDT criem regiões de vazio de pontos. No caso de uma exceção os pontos passam para a auditoria. Para a sincronia dos buffers, ao haver uma gravação do SDT, continuam-se a adicionar os pontos até que o limite mínimo de pontos para um possível novo ponto do SDT seja atingido, cujo mínimo seriam três pontos. Na sincronia, o ângulo base do buffer do algoritmo é trocado pelo atualmente encontrado no SDT, juntamente com os pontos. Quando existe uma exceção que supera os parâmetros, o algoritmo passa para uma análise critica, no qual possa ser definido se a planta deve ter os parâmetros de segurança carregados. Paralelamente são analisados os outliers. Terminado esse processo o algoritmo passa para uma nova aquisição de ponto. Conforme demonstrado na Figura 3.9.

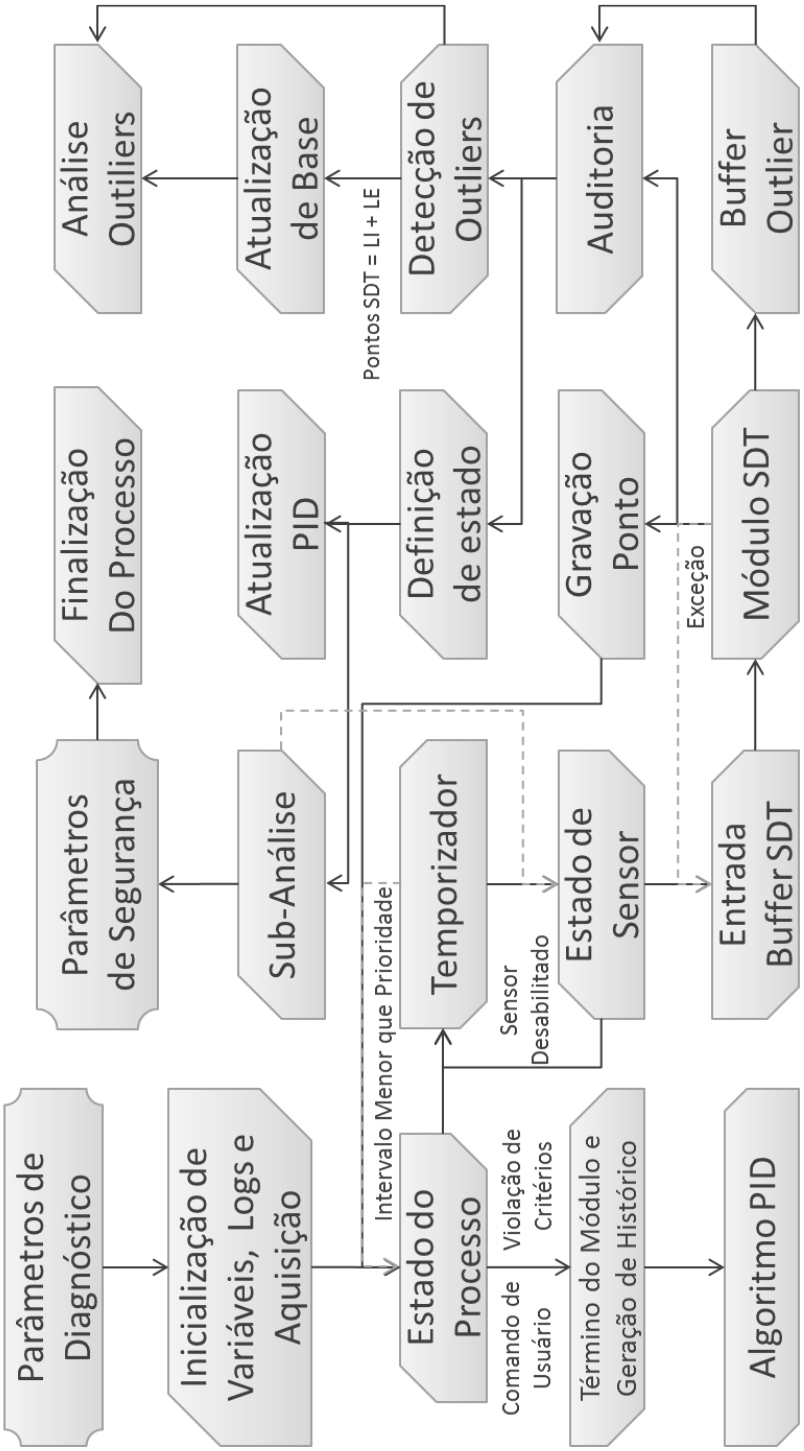

Figura 3.9 - Funcionamento do algoritmo integrado.

O temporizador obedece a uma lógica de prioridade para definir quando é necessária a aquisição de um novo ponto. Se definido que a tomada será feita a cada 5s imediatamente após esse período será feita a aquisição. O funcionamento é demonstrado na Figura 3.10.

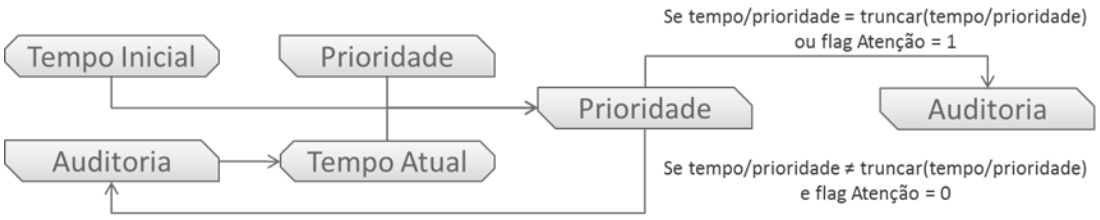

Figura 3.10 - Funcionamento do temporizador.

Para permitir que vários sensores sejam admitidos incorporados, as flags do sistema são feitas com a utilização de bases numéricas. Dessa forma, se houverem dez sensores será utilizada a base 10 para os números, se houverem trinta e cinco sensores serão usados base 35, dessa forma eles podem ser usados matematicamente sem necessidade de alteração manual. Um exemplo para dez sensores pode ser visto na Figura 3. 11.

Diagrama *flat* de auditoria e parada normal, com 10 sensores:

Valor	1	11	111	>8990	9100	9950	9960	10000
Estado	Exceção Positiva	Exceção Negativa	Limite de Buffer	Estado Atenção	Analise Prioritária	Falha de Sensor	Falha de Linha	Sensor Desligado

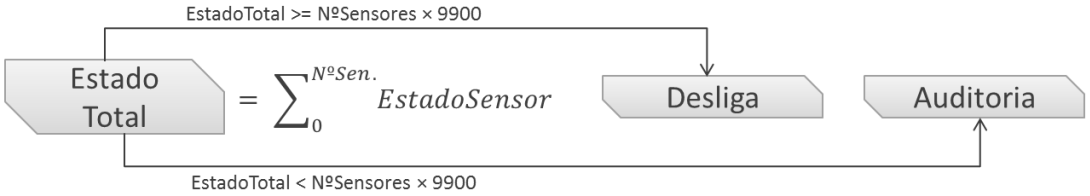

Figura 3. 11 - Sistema de base numérica para flags de sistema.

Para as ações da auditoria, primeiramente é definido qual o tipo de exceção, caso seja normal é feito uma análise superficial dependendo da *flag* do SDT, se for negativa ou positiva. Após isso são feitas as atualizações de histórico e PID caso necessário. Se houver uma *flag* crítica, é feito uma analise estrita onde se necessário o sistema será desligado, se não ele entra em estado de atenção, o qual se mantido carregará os parâmetros de segurança, conforme mostrado na Figura 3.12.

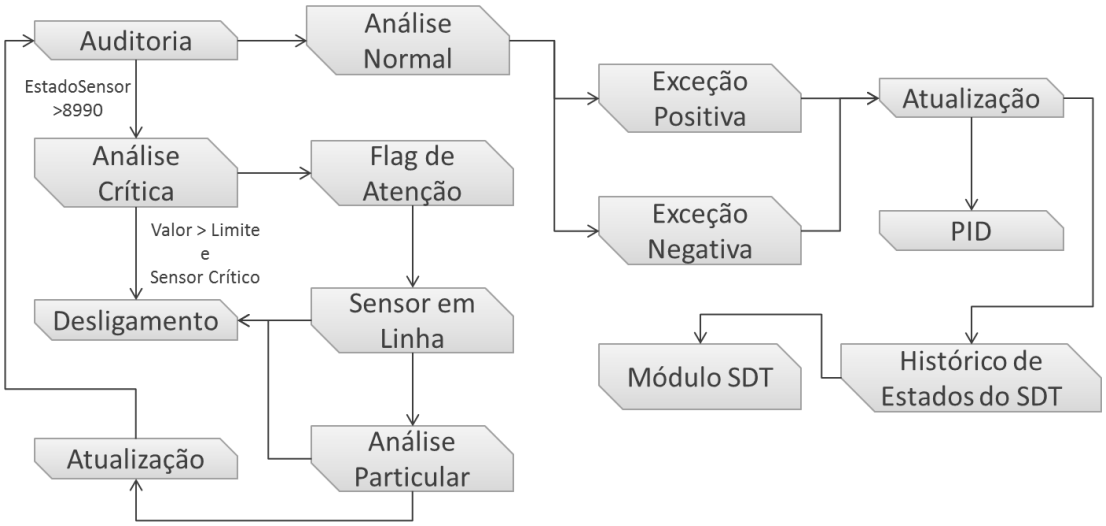

Figura 3.12 - Funcionamento das ações do sistema de auditoria.

Capítulo 4 - Estudos de Caso

Primeiramente, deve-se avaliar se o algoritmo é estável, ou seja, pode operar por longos períodos sem necessidade de intervenção. Essa análise é muito importante, quando se trabalha com plantas descentralizadas, uma vez que se deseja manter o mínimo de funcionários possíveis em cada uma.

4.1 Análise de Desempenho e Estabilidade

Para ser possível analisar esses indicadores, foram tomados dados reais de uma planta que foram processados de forma off-line no algoritmo. Como existe dificuldade em se verificar recursos de sistema na placa Raspberry, foi utilizado um microcomputador para verifica-los, e os dados analisados foram obtidos na própria placa Raspberry Pi, uma vez que ambos possuem a mesma versão do interpretador Python. O computador usado para os testes de estabilidade possui processador Intel® Core™2 Duo E8400 3.00GHz, dois pentes de memória com 1024MB, DDR2 1067MHz, a placa de vídeo é compartilhada On-Board Intel GMA 4500 e um disco rígido SAMSUNG HD161GJ (160 GB, 7200 RPM, SATA-II). O sistema operacional é Windows 7 Professional Service Pack 1 versão 6.1.7601.18205, o Python utilizado encontra-se atualmente na versão 2.7.3, além disso, é rodado também um programa de antivírus F-Secure versão 9.32.

O algoritmo não foi preparado para uso em processadores de múltiplos *cores* por isso o uso se limitava a um único núcleo, o qual, porém foi não exclusivo para o algoritmo, assim tarefas de sistema podiam também usar o mesmo núcleo.

Para o monitoramento dos recursos, foi utilizado o Monitor de Recursos da própria plataforma Windows, na qual é possível verificar diversos itens de desempenho de processos em particular. Para a avaliação do comportamento do algoritmo, como SDT e Auditoria, a própria placa Raspberry Pi foi usada.

4.1.1 Utilização de CPU

A Figura 4.1 demonstra a utilização da CPU pelo processo Python.

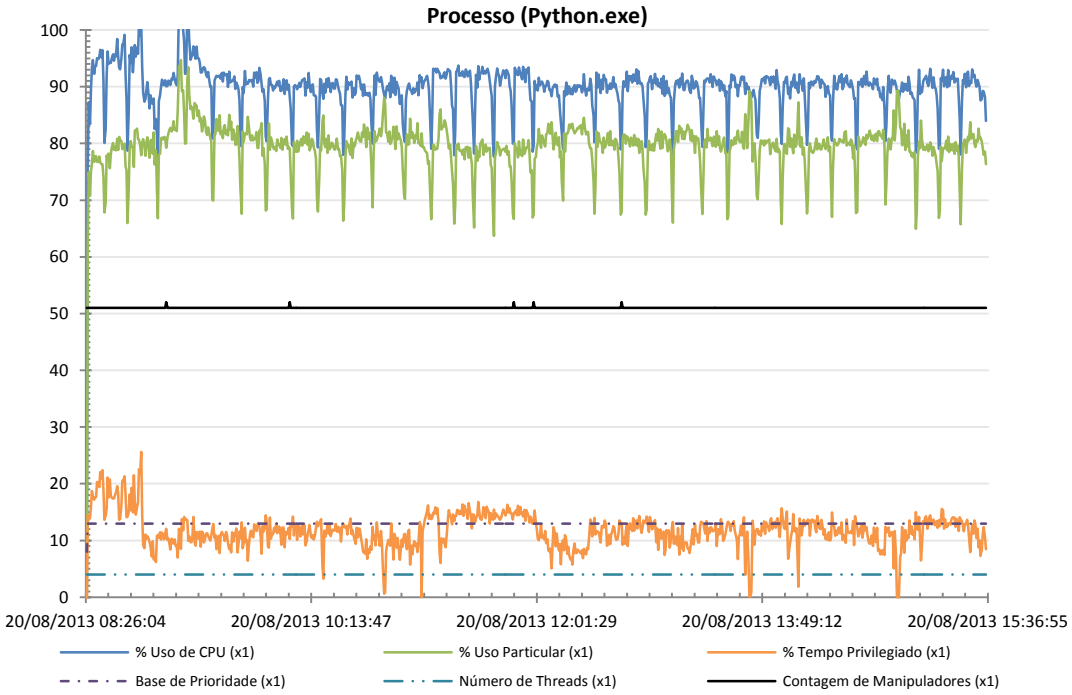

Figura 4.1 – Gráfico da utilização de CPU pelo processo do Python

Neste gráfico, pode ser visto que a utilização da CPU se manteve relativamente constante com picos negativos correspondendo à escrita, dos dados em disco pelo *thread* da auditoria, sendo que o histórico equivalente à operação de quatro dias era gravado a cada 20min aproximadamente. Isso ocorre principalmente por causa da latência de escrita em disco, contudo essa queda não afeta o desempenho dos *threads* responsável pelo PID e o da Interface de usuário. O uso de unidades de armazenamento mais rápidas como cartões SD com classe 6 ou superior, podem reduzir a latência de disco aumentando a desempenho geral do algoritmo.

A base de prioridade demonstra a prioridade perante outros aplicativos em execução os valores possíveis variam de 4 para aplicações suspensas e 24 para uso exclusivo, contudo este valor pode instabilizar o sistema, para segurança do sistema valores de cerca de 13 são considerados de Alta Prioridade e terão preferência perante a maior parte dos aplicativos normais.

Na porcentagem de Tempo Privilegiado é mostrado o quanto do processamento é usado exclusivamente por tarefas do sistema operacional com isso podemos verificar o tempo particular que é exclusivo do algoritmo. Apesar de o programa ter sido feito baseando-se em três *threads*, o Python cria um a mais devido à janela de comando que é feita separadamente. O conjunto de manipuladores, que são objetos criados para monitoramento do aplicativo pelo sistema operacional,

permanece praticamente constante. A base de prioridade foi determinada como Alta (13) e assim foi mantida durante todo o teste.

4.1.2 Desempenho de Memória

A Figura 4.2 mostra como o algoritmo se comporta em termos de uso de memória.

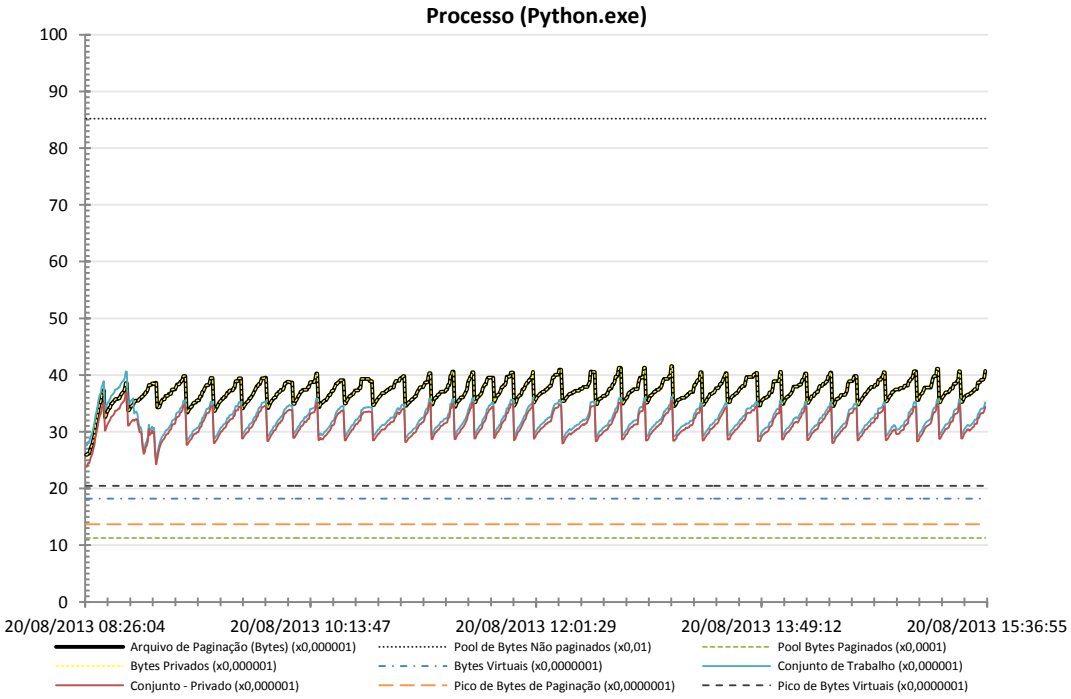

Figura 4.2 – Gráfico do consumo de memória na análise de desempenho.

No início da análise, a memória sofre grandes alterações devido à alocação do próprio Python na memória do computador. Contudo, durante o teste a média de consumo total de memória se manteve entre 30 e 40 MB, grande parte desse volume foi devido ao conjunto de dados utilizados, que consistiam em 240.724 pontos relativos a quatro dias de operação da planta, até que o log de dados seja fechado e gravado em disco. Além disso, foram gerados também logs, armazenados de forma cíclica com os tipos primários das exceções, como exemplo uma exceção de Máximo do SDT, assim como as exceções secundárias geradas pela auditoria em si, onde podem ser demarcadas as possíveis falhas de sensores ou variações abruptas. As variáveis reiniciadas são do tipo Listas complexa para economizar memória uma vez que, mesmo se um sistema de 20 sensores, se um único sensor tiver 50.000 dados enquanto os outros tiverem 1.000 dados ela terá 69.000 valores, em uma lista normal o mesmo sistema alocaria 1.000.000 de valores mesmo que nulos. Assim que o log do histórico é gravado, essa memória é despejada liberando o espaço, isso ocorre devido, ao método de declaração do Python, que, para não

serem quadradas, essas listas tem que ser alocadas coluna a coluna. A formatação final das listas é demonstrada na Tabela 4.1

Tabela 4.1 – Comprimento das linhas das variáveis List do histórico

Sensor -> Comprimento Particular da Lista (Em valores)				
S01 -> 40425	S02 -> 20121	S03 -> 19447	S04 -> 11078	S05 -> 34394
S06 -> 34558	S07 -> 17949	S08 -> 17280	S09 -> 11078	S10 -> 34394

Essa discrepância entre o volume de dados de cada sensor é devido à prioridade de aquisição, como eram relativos a quatro dias de operação, alguns sensores tinham aquisição de 1/20s enquanto outros a aquisição eram de cerca de 1/5s, sendo ajustada para cada um e ajustada para sincronizar o envio dos logs.

4.1.3 Desempenho de Leitura e Escrita em disco

Outro fator necessário para a avaliação do algoritmo é o volume de dados gerados. Para tal fim, o algoritmo foi concebido para escrever em disco, os dados que seriam enviados pela rede, dessa forma, é possível analisarmos o comportamento do mesmo, sem possíveis erros de comunicação, como mostrado na Figura 4.3 e Figura 4.4.

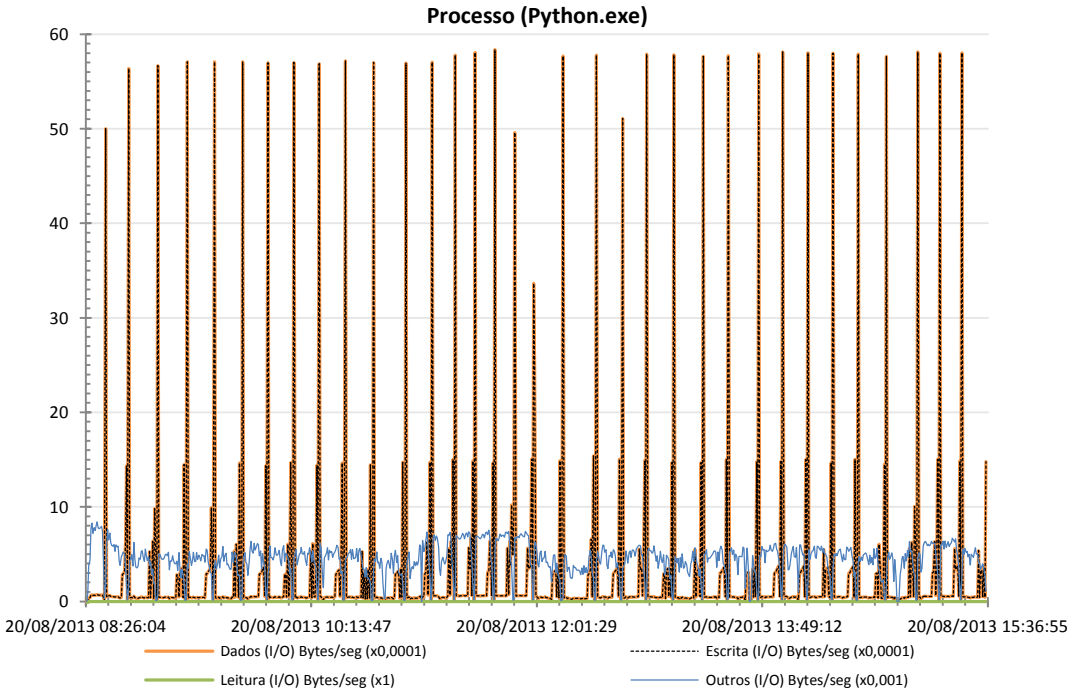

Figura 4.3 – Análise de desempenho de escrita em disco.

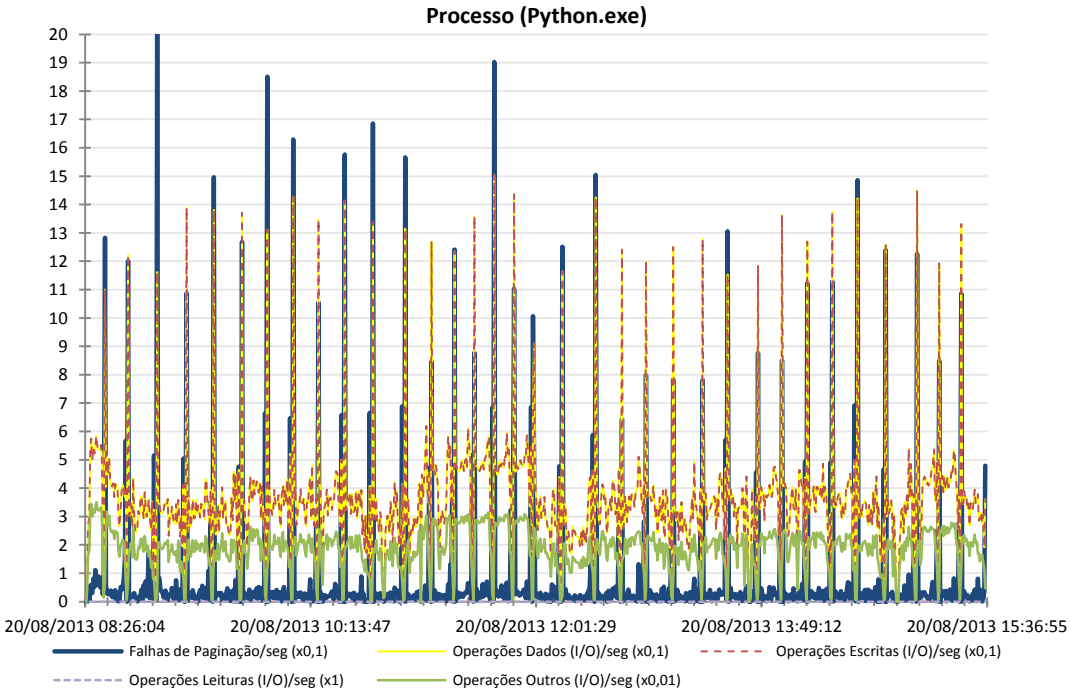

Figura 4.4 – Análise de desempenho de operações I/O.

Nessa análise de escrita em disco, pode ser visto que há uma taxa de escrita constante em disco devido aos blocos de dados comprimidos. Contudo, quando um bloco de dados históricos é gravado, podemos ver picos de volume de dados. Outro fator a ser avaliado está nas falhas de paginação que, embora não danifiquem os dados, geram atrasos na execução do algoritmo. Com o volume grande de dados na escrita do histórico vemos também picos de falhas de paginação, que ocorrem quando um dado da memória virtual não é carregado na memória física do computador. A escrita em execução normal foi em média de 5800 bytes por segundo, enquanto que nos picos da gravação do histórico chegou-se a 47 megabytes. Quanto às falhas de paginação em execução normal, o valor foi de cerca de 3 falhas por segundo com picos de 150 falhas no momento da gravação dos logs históricos, os quais possuem cerca de 12 megabytes cada um. Houve uma discrepância de cerca de três horas e meia de análise, como observado pela redução da escrita e aumento da latência, porém ao compararmos com a Figura 4.1 podemos observar que ocorre no mesmo momento em que há um aumento do uso por parte do sistema operacional, o que pode ser contornado quando o sistema é dedicado ao algoritmo.

4.1.4 Análise de desempenho de processamento

O algoritmo conta com dois parâmetros para sincronia do processador e sensores na auditoria, um é um multiplicador global de velocidade, dependente do processador, isso devido à necessidade de se fazer a aquisição em intervalos bem

regulados, o outro é responsável pelo intervalo de aquisição de cada sensor. Assim quando o multiplicador global é 1 e o de intervalo é 1, obteremos uma amostragem por segundo, ao modificarmos o multiplicador global para 100, mantendo o valor do intervalo, teremos então um total de 100 amostragens por segundo. Para estes testes foi utilizado o conjunto de parâmetros para realização de 400 cálculos de auditoria por segundo divididos 10 sensores, assim poderíamos também avaliar 20 sensores, que estariam com uma taxa de 20 cálculos de auditoria por segundo. Para um teste de rapidez do algoritmo, foi alterado o multiplicador até que o limite de fosse de 630 operações de auditoria por segundo e cerca de 40.000 operações de PID por segundo foi observado que um aumento maior do multiplicador prejudicava o desempenho do algoritmo, o que pode ser devido a dois fatores. O primeiro está na latência de acesso em disco e o segundo na falha de sincronia com o relógio do computador.

Para avaliar as causas reais da perda de desempenho, foi trocado o temporizador por um simples contador não atrelado ao relógio. Nesse teste foi conseguida uma taxa de até 1.000 operações da auditoria por segundo, demonstrando que o desempenho é limitado pela sincronia, e não o disco.

4.2 Análise de Desempenho do SDT Modificado

Para teste, foram usados dados reais de diversos tipos de sensores com diferentes comportamentos, tais como alto ruído e variabilidade. Nas corridas o SDT foi ajustado para ter valor superior a 90% de compressão. Para avaliação do erro foi utilizado o *Two-sample Kolmogorov-Smirnov Test* (*MARSAGLIA, 2003*) entre a curva original e compactada, também para comparação visual o erro entre os dados reais e compactados contra o desvio padrão da corrida foram mostrados. O teste de *Kolmogorov-Smirnov* para duas amostras consiste na seguinte equação:

$$D_{n,n'} = \sup_x |F_{1,n}(x) - F_{2,n'}(x)| \qquad (4.1)$$

Neste $F_{1,n}$ e $F_{2,n'}$ são as distribuições empiricas da amostra principal e comparativa respectivamente, $\sup_x$ corresponde ao *supremum* de x, com isso então a hipotese nula pode ser definica com:

$$D_{n,n'} > c(\alpha)\sqrt{\frac{n+n'}{nn'}} \qquad (4.2)$$

Os valores de $c(\alpha)$ segue os seguintes níveis:

Tabela 4.2 – Valores de α e cutoff normais para o teste TSKT sendo destacado o utilizado para os testes. (*NIST, 2003*)

α	0.10	**0.05**	0.01
$c(\alpha)$	0.37000	**0.41000**	0.49000

4.2.1 Caso 1 - Comportamento de baixa variabilidade e ruído

Para este teste foram utilizados dados de sensores de temperatura. A taxa de compressão foi de 91,66% na redução de dados. Por ser relativamente estável foram usados os parâmetros de 0,3 para a tolerância do SDT e um limite de buffer de 48 pontos. A prioridade de aquisição foi equivalente a 1 para cada 15s, nas duas corridas a taxa de compressão se manteve constante. Para o teste do TKST foram obtidos H = 0, P = 0.4742 e H = 0, P = 0.5603 respectivamente para a primeira e segunda corrida. Na Figura 4.5 e Figura 4.6, podemos ver os gráficos comparativos das curvas de ambos os conjuntos de dados, na Figura 4.7 são mostradas aproximações para melhor visualização do comportamento.

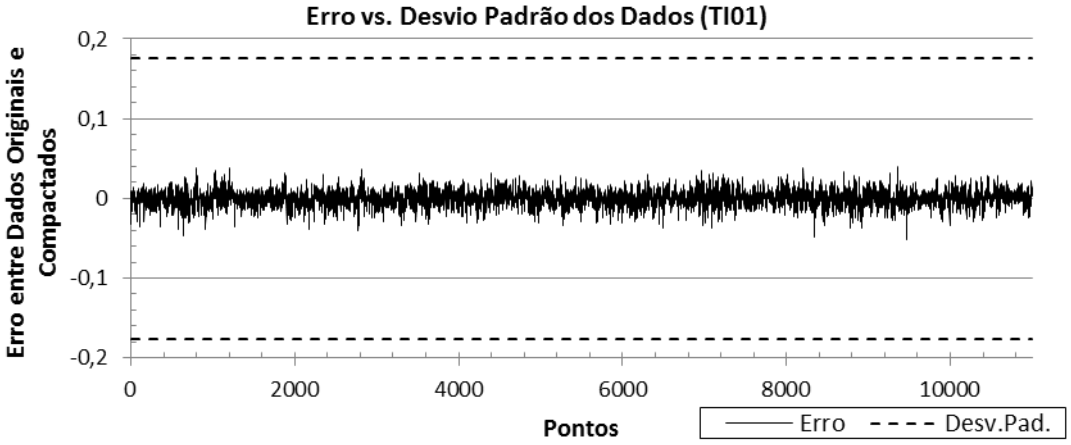

Figura 4.5 – Caso 1 Erro x Desvio Padrão, da primeira corrida.

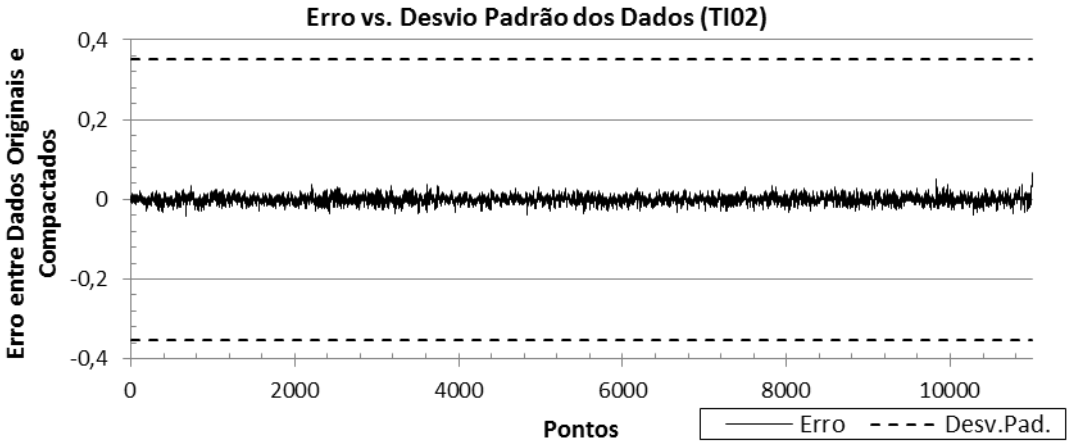

Figura 4.6 – Caso 1 Erro x Desvio Padrão, da segunda corrida.

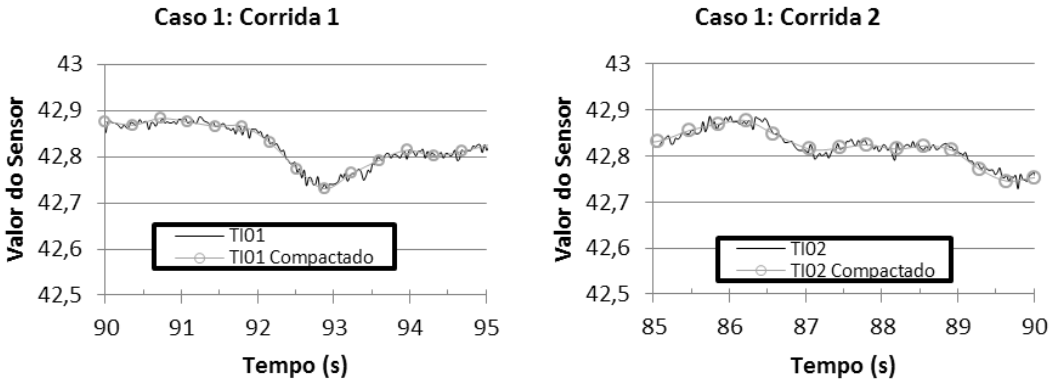

Figura 4.7 –Aproximações das curvas das duas corridas do caso 1.

Nesse caso foi possível captar com acuracidade o comportamento da curva

4.2.2 Caso 2 - Comportamento de baixa variabilidade e médio ruído

Para este teste, foram utilizados dados relativos a um sensor de nível, para o qual a taxa de compressão foi de cerca de 92,5%. Para o teste os parâmetros do SDT foram de 1,0, para a tolerância, buffer máximo de 48 pontos com uma prioridade equivalente a 1 para cada 10s. Foram testados dois conjuntos de dados, para os quais a variação da taxa de compressão foi menor que 1%. Para o teste do TKST foram obtidos H = 0, P = 0.8844 e H = 0, P = 0.9905 respectivamente para a primeira e segunda corrida. O resultado da compressão é mostrado na Figura 4.8 e Figura 4.9 para corrida o primeiro e o segundo conjuntos de dados, respectivamente, na Figura 4.10 são mostradas aproximações para melhor visualização do comportamento.

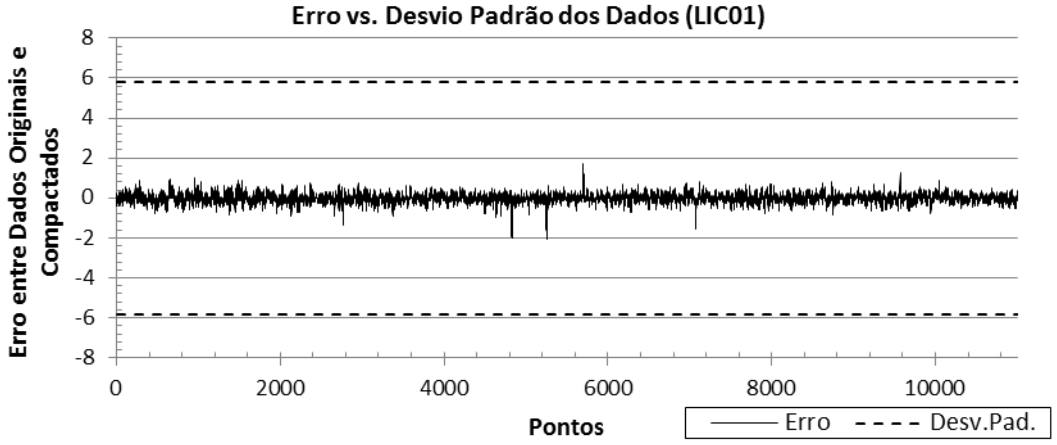

Figura 4.8 – Caso 2 Erro x Desvio Padrão, da primeira corrida.

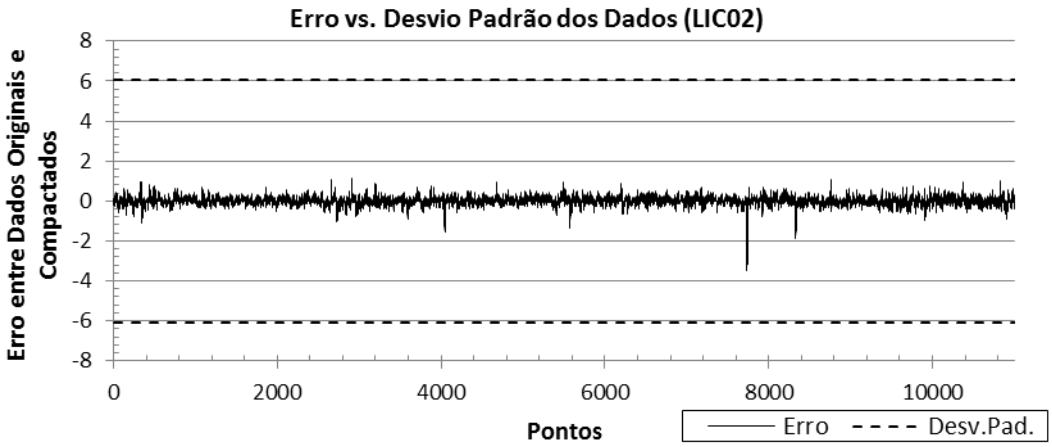

Figura 4.9 – Caso 2 Erro x Desvio Padrão, da segunda corrida.

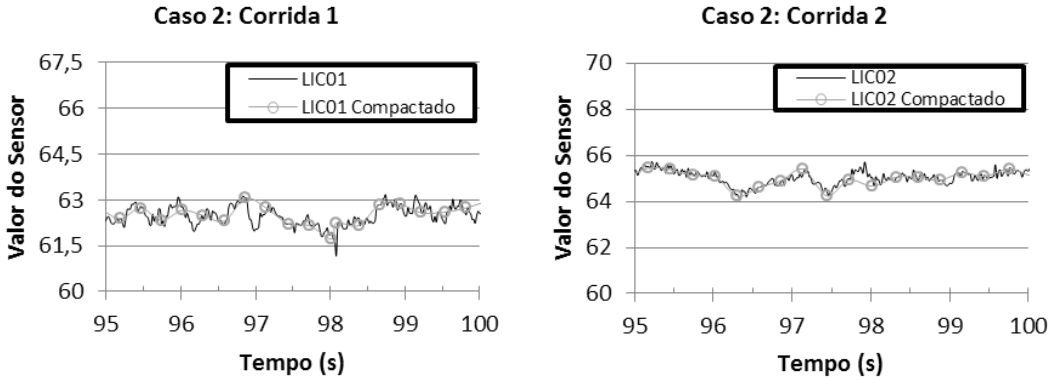

Figura 4.10 –Aproximações das curvas das duas corridas do caso 2.

4.2.3 Caso 3 - Comportamento de baixa variabilidade e alto ruído

Para este teste, foram utilizados dados relativos a um sensor de vazão, no qual a taxa de compressão foi de cerca de 92,6%. Para o teste os parâmetros do SDT foram de 30,0, para a tolerância no ângulo, buffer máximo de 48 pontos com uma prioridade equivalente a 1 para cada 5s. Foram testados dois conjuntos de dados, onde a variação da taxa de compressão foi menor que 1%. Para o teste do TKST foram obtidos H = 1 P = 2.5516e-047, e H = 1, P = 7.7689e-011 respectivamente para a primeira e segunda corrida. O resultado da compressão é mostrado na Figura 4.11 e Figura 4.12 para o primeiro e o segundo conjuntos de dados, respectivamente, na Figura 4.13 são mostradas aproximações para melhor visualização do comportamento.

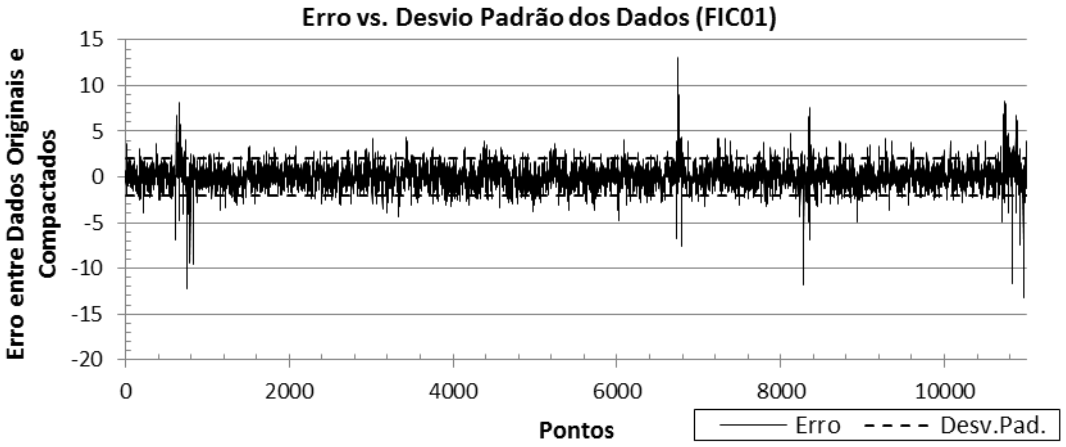

Figura 4.11 – Caso 3 Erro x Desvio Padrão, da primeira corrida.

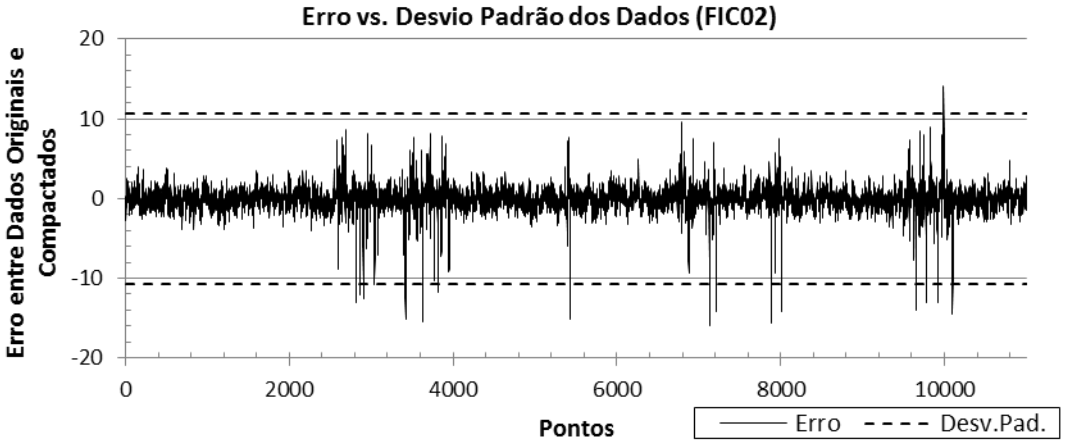

Figura 4.12 – Caso 3 Erro x Desvio Padrão, da segunda corrida.

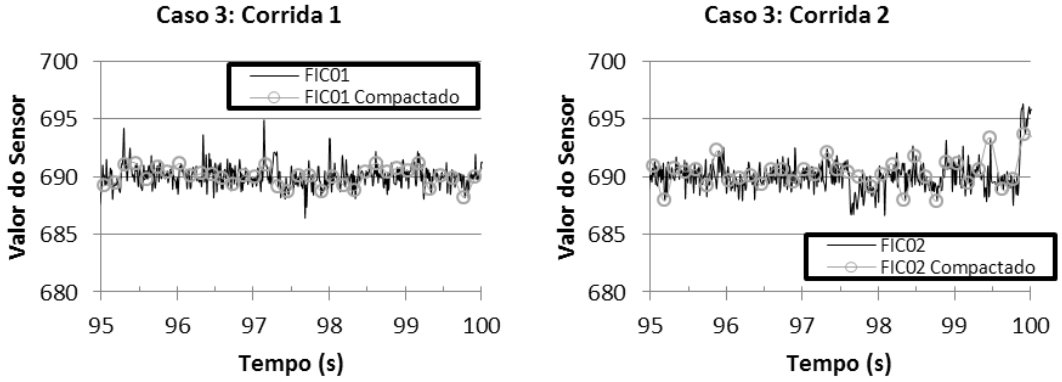

Figura 4.13 –Aproximações das curvas das duas corridas do caso 3.

4.2.4 Caso 4 - Comportamento de alta variabilidade e médio ruído

Para este teste, foram utilizados dados relativos a um sensor vazão, no qual a taxa de compressão foi de cerca de 92,25%. Para o teste, os parâmetros do SDT foram de 50,0, para a tolerância no ângulo, e buffer máximo de 48 pontos com uma

prioridade equivalente a 1 para cada 5s. Foram analisados dois conjuntos de dados, para os quais a variação da taxa de compressão foi menor que 1%. Para o teste do TKST foram obtidos H = 0, P = 0.2700 e H = 0, P = 0.3579 respectivamente para a primeira e segunda corrida. O resultado da compressão é mostrado na Figura 4.14 e Figura 4.15 para o primeiro e o segundo conjuntos de dados, respectivamente, na Figura 4.16 são mostradas aproximações para melhor visualização do comportamento.

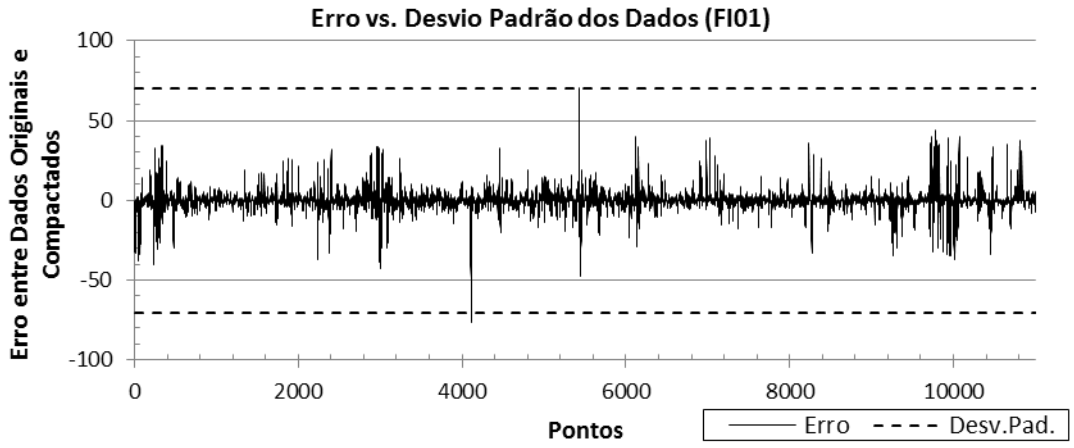

Figura 4.14 – Caso 4 Erro x Desvio Padrão, da primeira corrida.

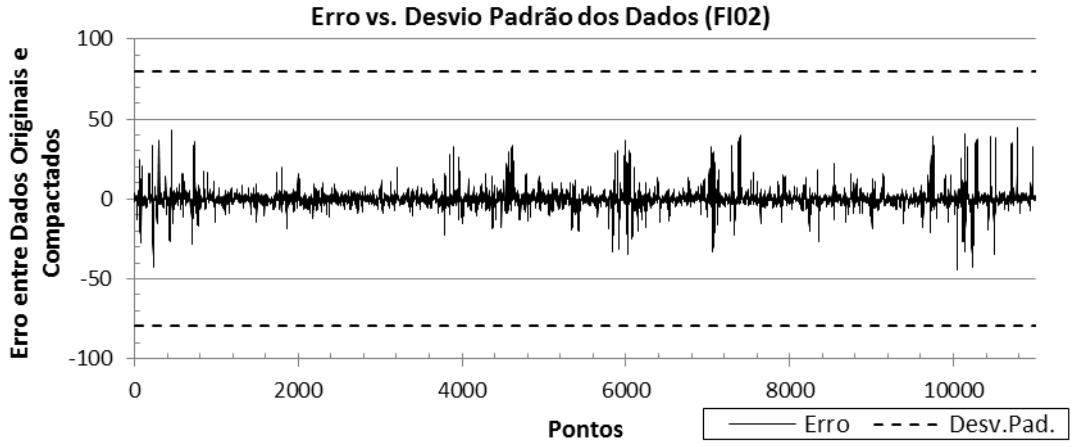

Figura 4.15 – Caso 4 Erro x Desvio Padrão, da segunda corrida.

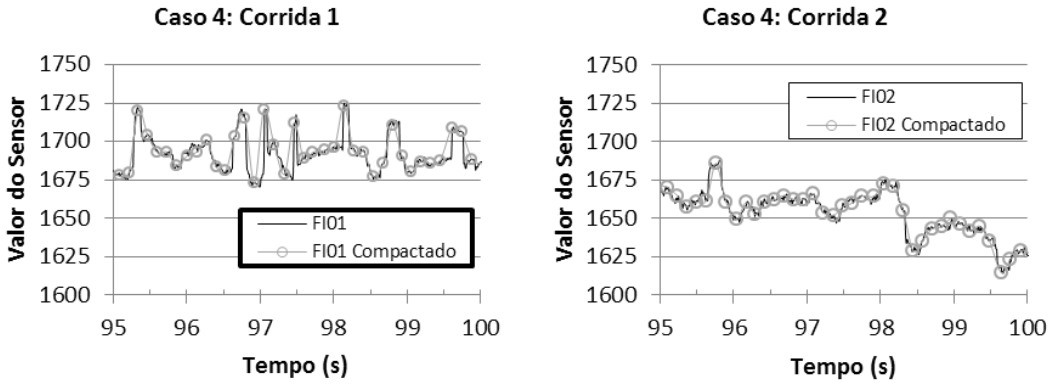

Figura 4.16 –Aproximações das curvas das duas corridas do caso 4.

4.2.5 Caso 5 - Comportamento de alta variabilidade e alto ruído

Para este teste, foram utilizados dados relativos a um sensor de pressão, no qual a taxa de compressão foi de cerca de 91,8%. Para o teste, os parâmetros do SDT foram de 0,3, para a tolerância no ângulo, buffer máximo de 48 pontos com uma prioridade equivalente a 1 para cada 10s. Foram testados dois conjuntos de dados, para os quais a variação da taxa de compressão foi menor que 1%. Para o teste do TKST foram obtidos H = 0, P = 0.2182 e H = 1, P = 0.0144 respectivamente para a primeira e segunda corrida. O resultado da compressão é mostrado na Figura 4.14 e Figura 4.15 para o primeiro e o segundo conjuntos de dados, respectivamente, na Figura 4.19 são mostradas aproximações para melhor visualização do comportamento.

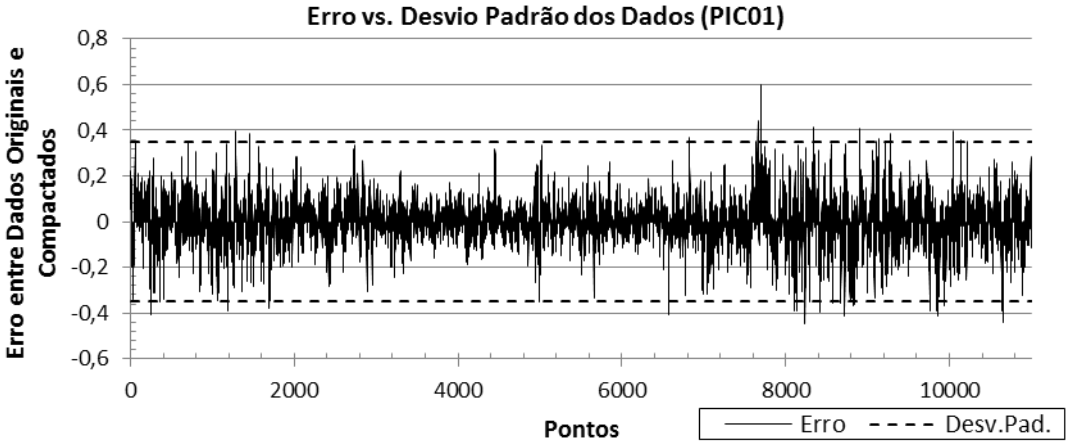

Figura 4.17 – Caso 5 Erro x Desvio Padrão, da primeira corrida.

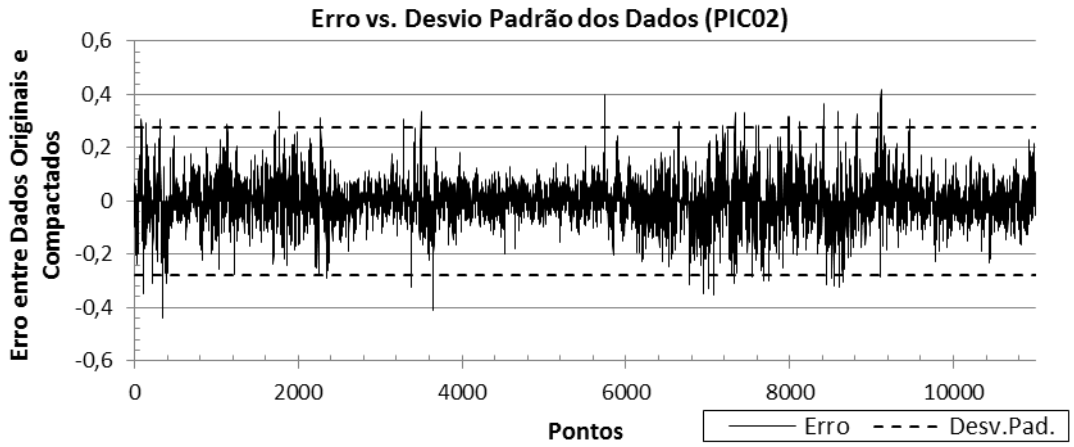

Figura 4.18 – Caso 5 Erro x Desvio Padrão, da segunda corrida.

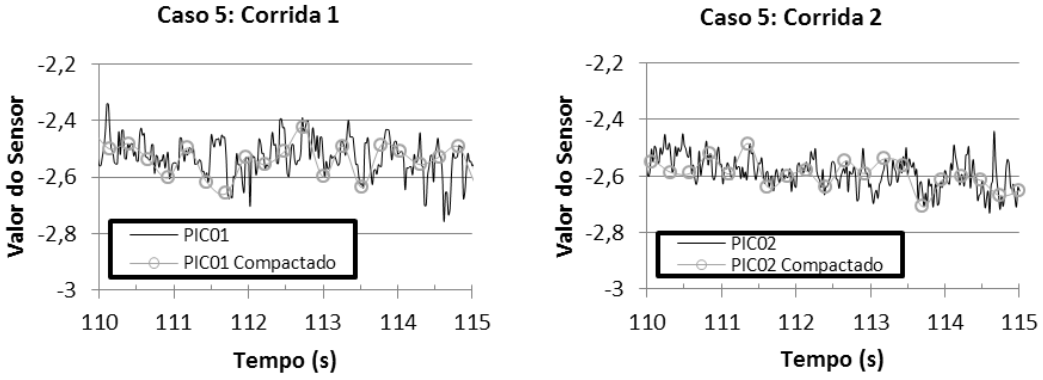

Figura 4.19 –Aproximações das curvas das duas corridas do caso 5.

4.2.6 Desempenho geral do SDT modificado

O algoritmo SDT modificado é capaz de comprimir com uma alta taxa sem uma perda significativa do comportamento da curva, o que foi observado por uma taxa de erro baixa nos conjuntos avaliados. A Tabela 4.3 mostra os coeficientes de determinação para cada uma das corridas.

Tabela 4.3 – Coeficientes de determinação para as duas corridas.

	FI	LIC	PIC	TI	FIC
Corrida 1	0,995047	0,999101	0,965377	0,998296	0,527832
Corrida 2	0,996156	0,999186	0,942231	0,999507	0,979343

Contudo, em processos com altos níveis de ruído e instabilidade, o ajuste se mostra difícil, porém, como a compressão foi superior a 90% em todos os casos, ainda é possível sacrificar a eficiência da compressão para obter uma melhor representação dos dados de planta.

4.3 Desempenho da detecção de Outliers

O teste de detecção foi realizado conjuntamente com os testes de compressão do SDT, portanto os padrões de tolerância do SDT são os mesmos. A seguir serão mostrados gráficos contendo as análises de *outliers*.

4.3.1 Caso 1 - Comportamento de baixa variabilidade e ruído

Para esse teste, foi utilizado um multiplicador com valor de uma vez a tolerância do SDT para o *outlier*, para os limites foi usada um ponto interno e dois externos. Nesse caso, não foi detectado nenhum *outlier* mesmo adotando-se limites baixos, isso se deve ao comportamento estável da curva em que mesmo os distúrbios são de caráter suave.

4.3.2 Caso 2 - Comportamento de alta variabilidade e baixo ruído

Para esse teste, foi utilizado um multiplicador com valor três vezes a tolerância do SDT e limites de um ponto interno e dois externos. Neste caso, foi observado um número considerável de *outliers*, isto ocorre pelo comportamento sem ruído que permite uma fácil detecção, assim mesmo com limites maiores ele consegue distinguir os pontos. Os gráficos das análises podem ser vistos na Figura 4.20 e Figura 4.21. Podemos ver que essa análise de *outlier* sofre um atraso correspondente aos limites impostos. Como o algoritmo utiliza um ponto interno e dois externos, o *outlier* foi marcado dois pontos após local detectado. Além disso, podemos ver na segunda corrida que um ponto não foi tomado, isso se deve ao fato que, mesmo sendo um pico evidente, o intervalo entre um ponto e outro fazia com que a tolerância de três vezes à do SDT não fosse superada assim um novo ajuste dos parâmetros do algoritmo pode fazer com que seja marcado/detectado.

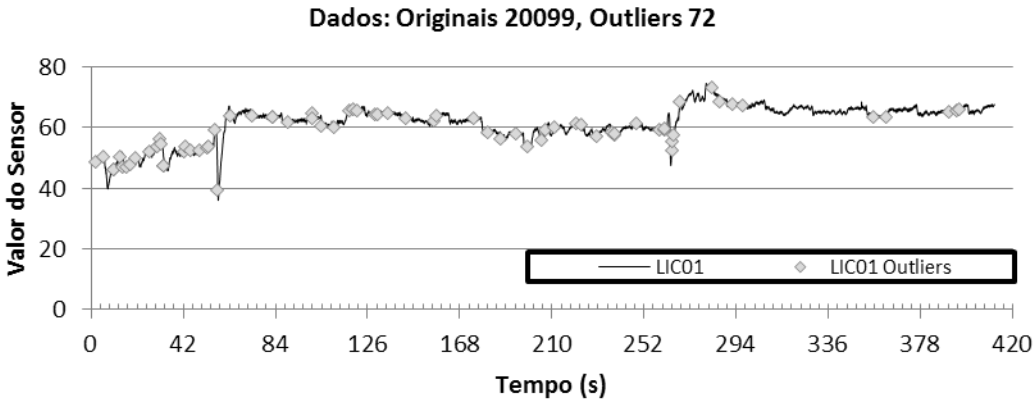

Figura 4.20 – Caso 2 - Gráfico da análise de *outliers* da primeira corrida.

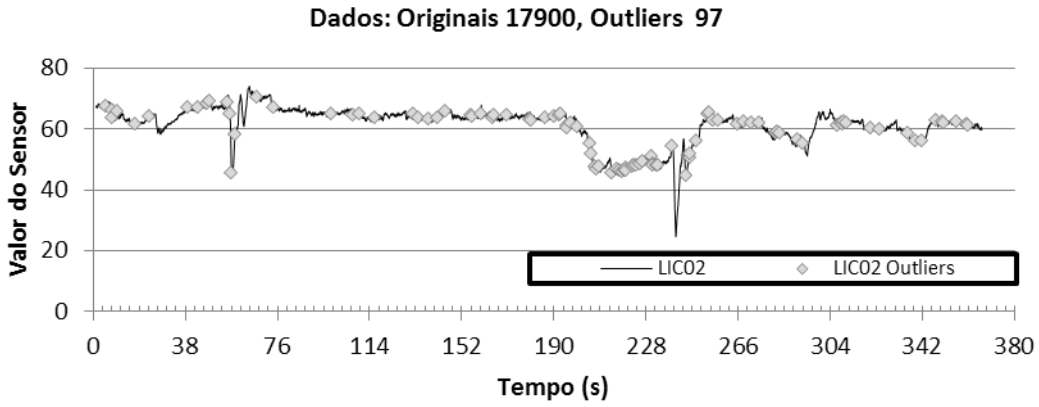

Figura 4.21 – Caso 2 - Gráfico da análise de *outliers* da segunda corrida.

4.3.3 Caso 3 - Comportamento de baixa variabilidade e alto ruído

Para este caso, foi utilizado um multiplicador de duas vezes a tolerância do SDT, e como limites foram utilizados dois pontos internos e quatro externos. Nessa análise não foram encontrados muitos *outliers*, porém alguns foram descartados por causa dos limites tomados internamente e externamente. Apesar disso, a maioria dos *outliers* foi detectada, contudo eles demoraram entre quatro e cinco pontos para serem acionados, devido aos limites internos e externos elevados. As análises podem ser vistas na Figura 4.22 e Figura 4.23.

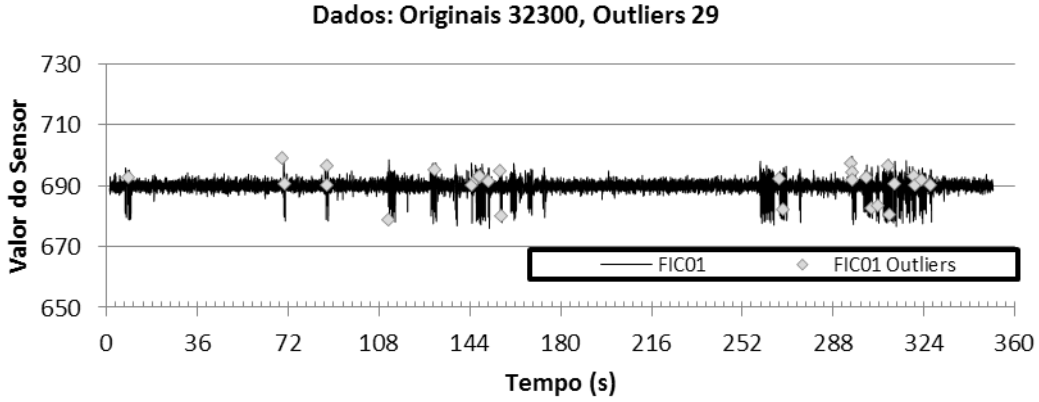

Figura 4.22 – Caso 3 - Gráfico da análise de *outliers* da primeira corrida.

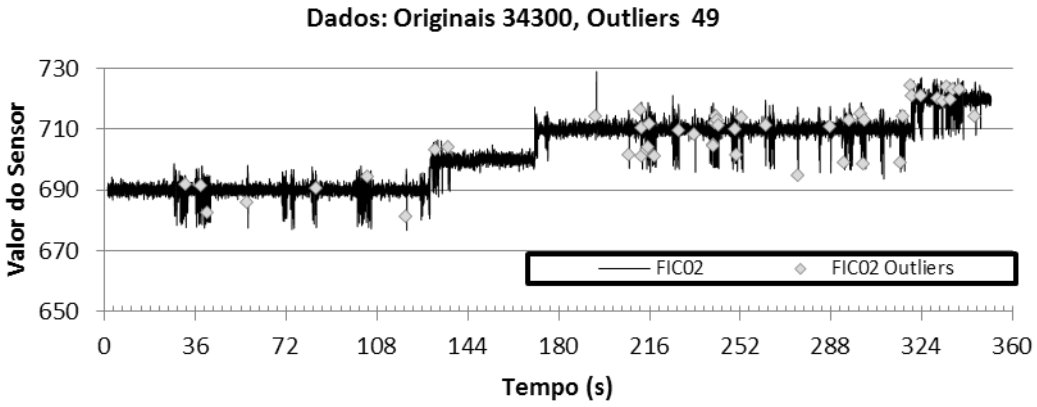

Figura 4.23 – Caso 3 - Gráfico da análise de *outliers* da segunda corrida

4.3.4 Caso 4 - Comportamento de alta variabilidade e médio ruído

Para esta análise, foi utilizado o multiplicador da tolerância como sendo 0,45 vezes a tolerância do SDT. Os limites foram de dois pontos internos e quatro externos. Foram obtidos poucos pontos de *outlier*, contudo a utilização de limites menores como um ponto interno e dois externos faz com que o número de *outliers* seja elevado. Além disso, esse foi o caso com maior dificuldade no ajuste do SDT, como o sistema de detecção é feito a partir da base da compressão, um ajuste pobre da mesma é também espelhado nos *outliers*.

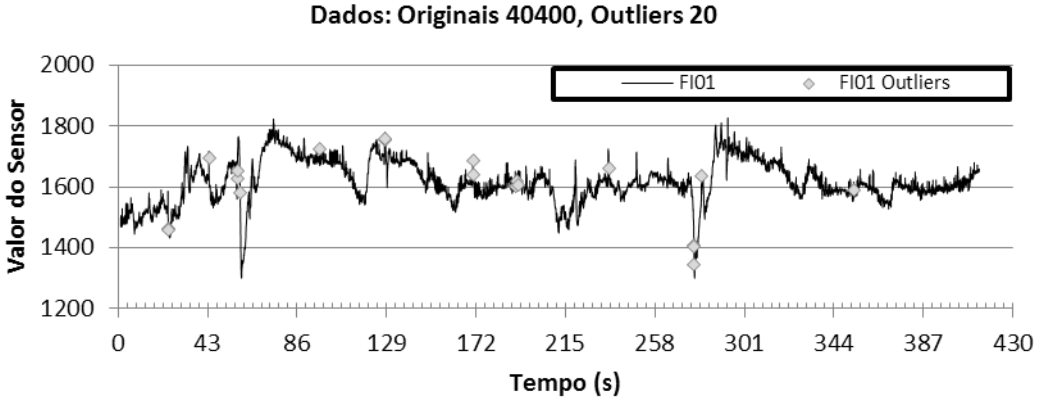

Figura 4.24 – Caso 4 - Gráfico da análise de *outliers* da primeira corrida

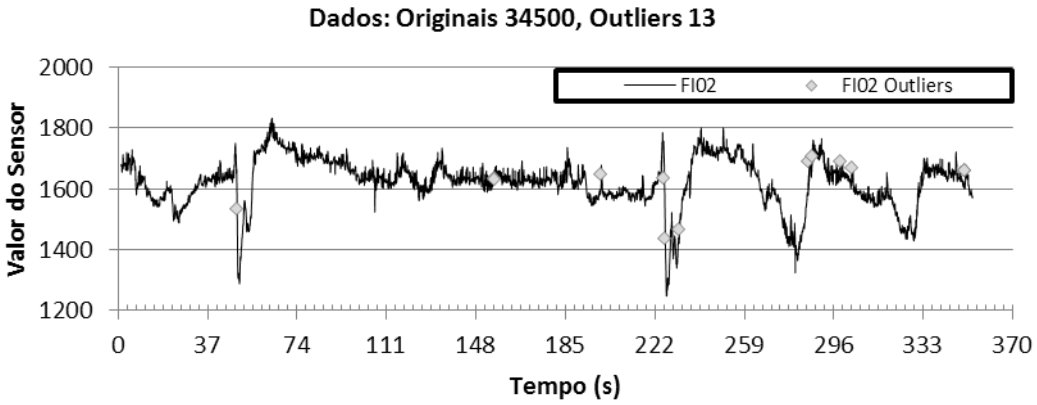

Figura 4.25 – Caso 4 - Gráfico da análise de *outliers* da segunda corrida.

4.3.5 Caso 5 - Comportamento de alta instabilidade e alto ruído

Neste caso, foi utilizado os parâmetros de cinco vezes a tolerância do SDT, sendo que os intervalos internos e externos foram de um e dois pontos respectivamente. Assim como no caso 4, como o processo era altamente instável e com um nível de ruído alto, a detecção dos *outliers* ficou prejudicada. Apesar desses fatores, o algoritmo foi acionado nos pontos de maior diferença e conseguiu descartar os pontos nos quais havia uma mudança do processo, como no caso 3.

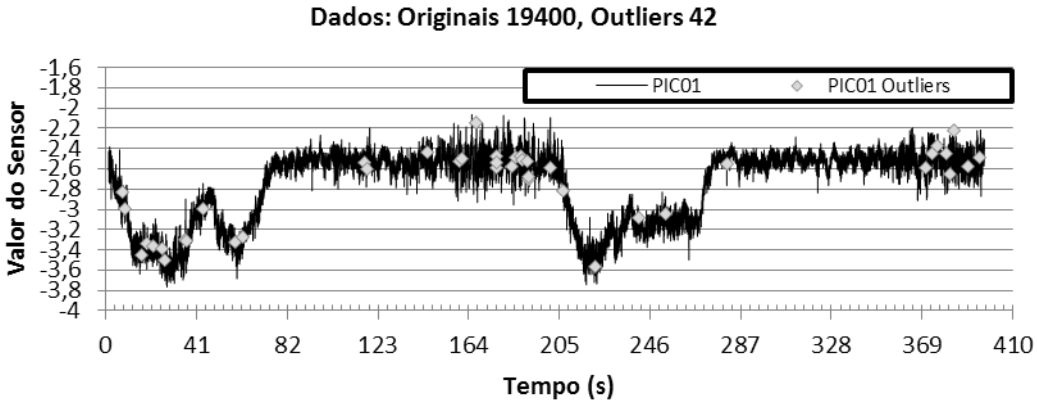

Figura 4.26 – Caso 5 - Gráfico da análise de *outliers* da primeira corrida

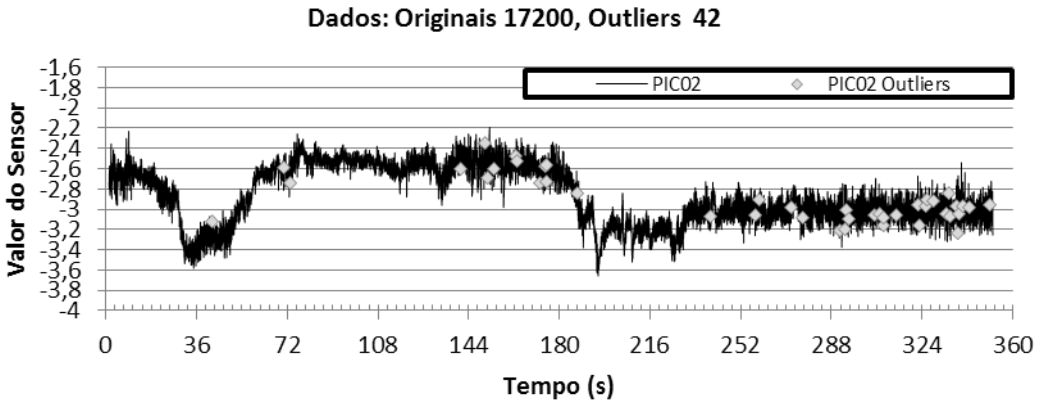

Figura 4.27 – Caso 5 - Gráfico da análise de *outliers* da segunda corrida

4.3.6 Desempenho geral do algoritmo de detecção de outliers

Em sistemas de alta variabilidade, o ajuste para a detecção de *outliers* se mostrou complexa, contudo parte da complexidade se deve ao ajuste do SDT, que em todos os testes foi ajustado para uma compressão superior a 90%. Um ajuste com tolerância menor para o SDT faz com que a detecção dos *outliers* seja simplificada, pois a redução do ruído faz com que os limites internos e externos não sejam suficientes para detecta-lo como tal. Contudo, isso só é possível em sistemas com baixo nível de ruído, ou constante, como o encontrado no caso 2. Uma alternativa seria a de aumentar os limites internos e externos, porém ao fazermos isso estaremos atrasando a detecção. Com limites internos e externos de 10 e 30 pontos respectivamente o atraso da detecção seria entre 30 e 35 pontos, o que pode ser inaceitável na maioria dos casos.

4.4 Desempenho da Auditoria

Para a auditoria foram analisados três áreas distintas, detecção de falhas, ajuste do PID e capacidade de ação em estado crítico.

4.4.1 Detecção de falhas

Na detecção de falhas, o algoritmo detecta uma variação excessiva nos dados. Ao fazer isso, o mesmo, em caráter normal, aciona uma *flag* que coloca em estado de atenção, onde o estado SDT é descartado, fazendo com que as análises sejam feitas a cada aquisição e não somente quando há a gravação de um ponto pelo SDT. Isso possibilitou uma tomada de decisão mais rápida, levando-se em conta os limites equivalentes a cinco pontos que foram utilizados para determinação de um estado de falha. Para processos em que há um ruído muito grande como os dos casos 3 e 5, esse limite foi suficiente para não haver um falso positivo na detecção de falha, devido ao caráter oscilatório da saída. Como demonstrado na Figura 4.28.

Dessa forma, antes de atingir o limite, o algoritmo foi capaz de ignorar os *outliers* e ruídos sem comprometimento. No entanto, para casos como os casos 2 e 4, que possuíam uma grande instabilidade, ela não era rápida o suficiente onde o limite excedia o máximo e o sistema detectava como falha de sensor, como demonstrado na Figura 4.29, a rampa mostrada possui cerca de 60 pontos, dessa forma se faz necessário alterar os limites para quando é possível afirmar a falha, porém isto pode ser feito independentemente para cada um dos sensores. Para o teste do sistema de falha foram gerados também, dados simulados com diversos tipos de situações, nesse ele foi capaz de detectar e agir conforme o que fora programado.

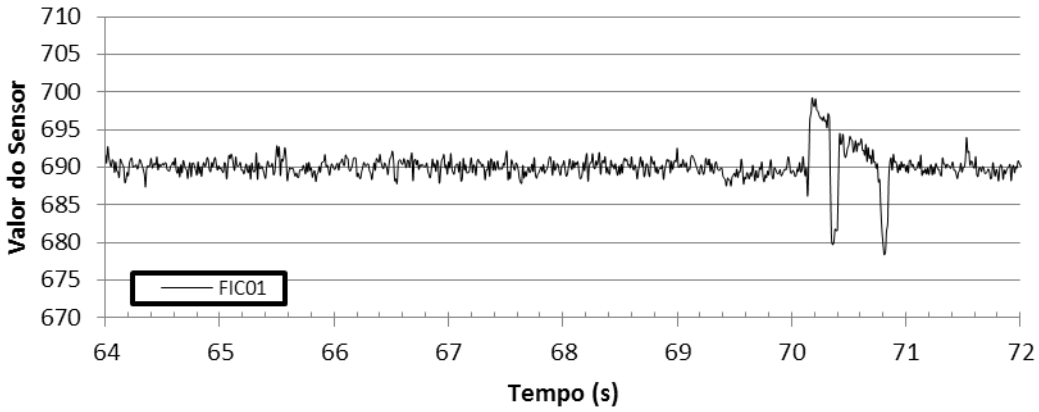

Figura 4.28 – Caráter oscilatório do Caso 3 na primeira corrida.

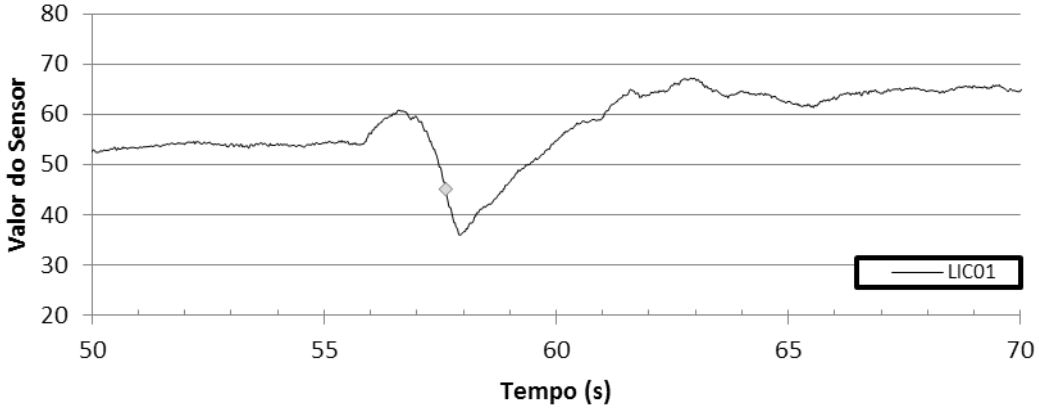

Figura 4.29 – Falha de sensor falso positivo, quando este foi marcado como crítico, detectado pelo algoritmo na primeira corrida do caso 2

4.4.2 Situações críticas

Foram usados primeiramente dados simulados para teste do algoritmo, nestes o comportamento foi o esperado. Um sensor continha o *flat* crítico devido a um evento de alteração muito elevada. Isso levava o algoritmo a carregar os parâmetros de segurança para o PID e após havia o desligamento da planta. No caso

de sensores não críticos, ele avaliava por um intervalo de tempo, então se somente o sensor em questão estava errôneo, o mesmo era marcado como falho. Quando alimentado com medidas de sensores em linha, que fez com que ele tomasse decisões baseadas nos dois sensores, se somente um estivesse com situação de erro o algoritmo continuava a operação. No caso de dois sensores em linha estarem errôneos, ele procedia da mesma forma que um sensor crítico, carregando os parâmetros de segurança e desligando a planta.

Quando testado nos casos descritos, o alto nível de ruído causou instabilidade no algoritmo, fazendo com que sensores marcados como críticos desligassem a planta prematuramente, assim como no caso de haver dois sensores em linha. Para evitar isso, foi necessário atrasar a tomada de decisão, a fim de permitir o isolamento de eventuais problemas causados pelo ruído. Contudo o algoritmo passa para um estado no qual durante esse período a prioridade de aquisição é descartada, assim a taxa de tomada de dados para esses sensores passa a ser maximizada. Essa espera na tomada de decisão pode ser utilizada como um novo parâmetro, uma vez que está atrelada ao ruído do processo.

4.5 Comparativo entre Parâmetros

Para, se obter uma noção de como o algoritmo se comporta conforme os parâmetros selecionados foi selecionado o sensor FIC, o qual foi observado como tendo o pior resultado no teste de TSKST. Os resultados e os parâmetros usados podem ser vistos na Tabela 4.4

Tabela 4.4 – Resultados do sensor FIC, para diversos parâmetros.

Tolerância SDT	Mult. Outlier	Lim. Interno	Lim. Externo	Dados Comp.	Outliers	TSKT (H)	TSKT (P)
25	2	2	4	2460	23	1	4.062 e-45
30	1	2	4	2451	105	1	2.462 e-47
30	2	1	2	2451	41	1	1.757 e-47
30	2	2	4	2451	19	1	3.082 e-47
30	2	3	6	2451	0	1	1.966 e-47
30	3	2	4	2451	4	1	2.462 e-47
35	2	2	4	2430	14	1	1.319 e-46

Capítulo 5 - Conclusões

Com base nos testes realizados, é possível a criação de um sistema integrado de compressão e auditoria que pode operar em sistemas como o do Raspberry Pi, tendo boa estabilidade operacional. Com isso, a sua utilização em plantas que operam de forma descentralizada se torna viável.

Apesar de ser possível esse sistema, para que o mesmo possa operar em tempo real ele necessita de um algoritmo de compressão que também trabalhe em tempo real, como no caso do *Swinging Door Trending*. Contudo, os sistemas podem ter que ser ajustados para que os parâmetros possam ser comunicados entre si e que sejam suficientes. No caso do SDT, foi necessário passar todos os ângulos do intervalo de avaliação. Com essas observações o sistema pode trabalhar por longos períodos de forma segura.

O consumo de recursos de equipamento se mostrou constante, sendo capaz de operar sem falhas por períodos equivalentes a quatro meses de operação, sem a necessidade de intervenção humana. O sistema do SDT modificado foi capaz de conseguir altas taxas de compressão sem a perda significativa do comportamento na maioria dos casos.

A detecção de possíveis falhas está atrelada ao ajuste do SDT, dessa forma, um ajuste falho do SDT, gera uma dificuldade em fazer as análises e operação.

Assim como a auditoria, a detecção de *outliers* depende de um bom ajuste do SDT, porém mesmo em processos que possuem um ruído muito elevado foi possível descartar mudanças no estado de operação da planta. Com isso, o algoritmo se mostra capaz de operar da forma desejada, porém necessita de emprego de técnicas de otimização de parâmetros para um melhor funcionamento.

Em alguns processos, como o do caso 3 em que há alterações elevadas por longos períodos, pode haver falsos positivos para possível falha de sensor. Como

existe um intervalo para que o algoritmo assuma falha, o mesmo pode ser transformado em parâmetro de forma a evitar o problema.

Para futuros trabalhos, o sistema será implementado em uma planta piloto, para a avaliação de problemas em tempo real, assim como ter a possibilidade de avaliar as ações nos equipamentos em si, o detalhamento da planta de produção de etanol é mostrado no Apêndice A.

Outro foco está na criação de um otimizador de parâmetros para evitar os falsos positivos e também manter a maior compactação/fidelidade possível, dos dados obtidos.

Ainda é possível estudar o uso de outras metodologias para realizar a integração, que podem ser combinadas, como o caso das *wavelets*, onde existem trabalhos tanto para compressão quanto para auditoria, que mesmo consumindo muitos recursos poderiam ser utilizadas em grandes unidades.

Por fim, outro foco seria integrar também o algoritmo do PID juntamente com os parâmetros do SDT, alterando a ação derivativa pela tendência do mesmo. Uma proposta de alteração do PID encontra-se no Apêndice B.

Referências

7-Zip, 7-Zip File Achiever, 2012, http://www.7-zip.org/

ALLES, M.; KOGAN, A.; VASARHELYI, M.; *"Black Box Logging and Tertiary Monitoring of Continuous Assurance Systems"*, Information Systems Control Journal, vol.1, 2003, Disponível em: http://www.isaca.org/Journal/Past-Issues/2003/Volume-1/Pages/Black-Box-Logging-and-Tertiary-Monitoring-of-Continuous-Assurance-Systems.aspx

ARDUINO; *"Products Reference"*, Arduino, 2013 Disponível em: http://arduino.cc/en/Main/Products

ASPEN® TECHNOLOGY INC. *"Analysis of Data Storage Technologies for the Management of Real-Time Process Manufacturing Data",* White Paper for InfoPlus® 2.1, Campbell, CA, 2001, Disponível em: http://www.aspentech.com/publication_files/White_Paper_for_IP_21.pdf

BAILEY, S. J.; *"From desktop to plant floor, a CRT is the control operators window on the process"*, Control Engineering, vol.31, ed.6, pag.86-90, 1984

BAKSHI, B. R.; STEPHANOPOULOS G.; *"Compression of chemical process data by functional approximation and feature extraction"*, Aiche Journal, vol.42, ed.2, pag.477-492, 1996, DOI: 10.1002/aic.690420217

BEAGLEBOARD; "BeagleBone Product Details", BeagleBoard, 2013 Disponível em: http://beagleboard.org/bone/

BRISTOL, E. H.; *"Swinging Door Trending: Adaptative Trend Recording?"*, Advances in Instrumentation and Control, Instrument Society of America Paper #90-493, pag.749-754, 1990 Disponível em: http://www.ebristoliclrga.com/PDF/SwDr.pdf

BURROWS, M.; WHEELER, DAVID J.; *"A block-sorting lossless data compression algorithm"*, HP Labs Technical Reports, SRC-RR-124, 1994 Disponível em: http://www.hpl.hp.com/techreports/Compaq-DEC/SRC-RR-124.pdf

CAZAN, R.C.; CHIRILA, L.V.; MARICHESCU, A.; *"Remote terminal unit's Intelligence Evolution"*, IEEE International Conference on Automation, Quality and Testing, Robotics vol.3, ed.1, pag.300-303, 2008, DOI: 10.1109/AQTR.2008.4588932

CHENG, P.; BLELLOCH, G. E.; *"A parallel, real-time garbage collector"*, SIGPLAN 2001 conference on Programming language design and implementation, vol.36, ed.5, pag.125-13, 2001, DOI: 10.1145/378795.378823

CHOW, E. Y.; WILLSKY, A. S.; *"Analytical Redundancy and the Design of Robust Failure Detection Systems"*, IEEE Transactions on Automatic Control, vol.AC-29, ed.7, pag.603-614, 1984, DOI: 10.1109/TAC.1984.1103593

CHOW, T. W S; SHI HAI, *"Induction machine fault diagnostic analysis with wavelet technique"*, IEEE Transactions on Industrial Electronics, vol.51, ed.3, pag.558-565, 2004, DOI: 10.1109/TIE.2004.825325

DAOUD, E.A.; NICOLICI, N.; , "Real-Time Lossless Compression for Silicon Debug", IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol.28, ed.9, pag.1387-1400, 2009, DOI: 10.1109/TCAD.2009.2023198

DE PERSIS, C.; ISIDORI, A.; *"A geometric approach to nonlinear fault detection and isolation"*, IEEE Transactions on Automatic Control, vol.46, ed.6, pag.853-865, 2001, DOI: 10.1109/9.928586

FOWLER, J. E.; YAGEL, R.; *"Lossless compression of volume data*" Proceedings of the 1994 symposium on Volume Visualization, pag.43-50, 1994, DOI: 10.1145/197938.197961

FRANK, P. M.; DING, S. X.; MARCU, T.; *"Model-based fault diagnosis in technical processes"*, Transactions of the Institute of Measurement and Control, vol.22, ed.1, pag.57-101, 2000 DOI: 10.1177/014233120002200104

GOLCHIN, F.; PALIWAL, K.K.; *"Classified adaptive prediction and entropy coding for lossless coding of images"*, Proceedings of International Conference on Image Processing , vol.3, pag.110-113, 1997, DOI: 10.1109/ICIP.1997.632006

HALE, J. C.; SELLARS, H. L.; *"Historical Data Recording for Process Computers"*, Chemical Engineering Progress, vol.77, ed.11, pag.38-43, 1981, OSTI: 5451707

HAMMOURI, H.; KINNAERT, M.; EL YAAGOUBI, E.-H.; *"Observer-based approach to fault detection and isolation for nonlinear systems"*, IEEE Transactions on Automatic Control, vol.44, ed.10, pag.1879-1884, 1999 DOI: 10.1109/9.793728

HENG, G.T.; *"Microcomputer-based remote terminal unit for a SCADA system"*, Microprocessors and Microsystems, vol.20, ed.1, pag.39–45, 1996, DOI: 10.1016/0141-9331(95)01066-1

HIMMELBLAU, D. M.; *"Fault detection and diagnosis in chemical and petrochemical processes"*, Elsevier press, ed.8, Amsterdam, 1979 ISBN-13: 978-0-4444-1747-3

INTERNATIONAL ATOMIC ENERGY AGENCY (IAEA), *"Chernobyl's Legacy: Health, Environmental and Socio-Economic Impacts and Recommendations to the Governments of Belarus, the Russian Federation and Ukraine"*, The Chernobyl Forum: 2003–2005, Vienna, Austria, 2005, Disponível em: http://www.iaea.org/Publications/Booklets/Chernobyl/chernobyl.pdf

JAMES, P. A.; *"Data Compression for Process Historians"* Richmond, CA: Chevron Research and Technology Company, 1995 Disponível em: http://wenku.baidu.com/view/a7415ae881c758f5f61f6793.html

JAYANT, N.; JOHNSTON, J.; SAFRANEK, R., *"Signal compression based on models of human perception"*, Proceedings of the IEEE , vol.81, ed.10, pag.1385-1422, 1993, DOI: 10.1109/5.241504

LAHART, J.; "*Taking an Open-Source Approach to Hardware",* The Wall Street Journal. 2009 Acessado em: 26 de Março de 2013 Disponível em: http://online.wsj.com/article/SB10001424052748703499404574559960271468066.html

LEAFLABS; *"The Maple"*, LeafLabs, 2013 Disponível em: http://leaflabs.com/devices/maple/

LEE, J.-M.; YOO, C.K.; LEE, I.-B.; *"Fault detection of batch processes using multiway kernel principal component analysis"*, Computers & Chemical Engineering, vol.28, ed.9, pag.1837–1847, 2004, DOI: 10.1016/j.compchemeng.2004.02.036

LIN, C. E.; LING, J.-M.; HUANG, C.L.; *"An expert system for transformer fault diagnosis using dissolved gas analysis"*, IEEE Transactions on Power Delivery, vol.8, ed.1, pag.231- 238, 1993 DOI: 10.1109/61.180341

MAGLI, E.; OLMO, G.; QUACCHIO, E.; *"Optimized onboard lossless and near-lossless compression of hyperspectral data using CALIC",* Geoscience and Remote Sensing Letters, vol.1, ed.1, pag. 21- 25, 2004, DOI: 10.1109/LGRS.2003.822312

MAH, R.S.H.; TAMHANE, A.C.; TUNG, S.H.; PATEL, A.N.; *"Process trending with piecewise linear smoothing"*, Computers & Chemical Engineering, vol.19, ed.2, pag.129-137, 1995, DOI: 10.1016/0098-1354(94)E0042-L

MALLAT, S.G., *"A theory for multiresolution signal decomposition: the wavelet representation"*, IEEE Transactions on Pattern Analysis and Machine Intelligence, vol.11, ed.7, pag.674-693, 1989, DOI: 10.1109/34.192463

MANNAN, S.; LEES, F. P.; "Lees' Loss Prevention in the Process Industries: Hazard Identification, Assessment and Control", Elsevier Incorporated, ed.3, 2005, ISBN-13: 978-0-7506-7555-0

MARSAGLIA, G.; TSANG, W.; WANG J.; *"Evaluating Kolmogorov's Distribution"*, Journal of Statistical Software, vol. 8, ed. 18, 2003 Disponivel em: http://www.jstatsoft.org/v08/i18/paper

MAURER, W. D.; *"The effect of the Harvard architecture on the teaching of assembly language"*, Journal of Computing Sciences in Colleges, vol.20, ed.5, pag.79-90, 2005, ISSN:1937-4771, EISSN:1937-4763

MEMON, N.; XUAN KONG; CINKLER, J.; *"Context-based lossless and near-lossless compression of EEG signals"*, IEEE Transactions on Information Technology in Biomedicine, vol. 3, ed. 3, pag. 231-238, 1999 DOI: 10.1109/4233.788586

MEMON, N.D.; SAYOOD, K.; MAGLIVERAS, S.S.; *"Lossless compression of multispectral image data"*, IEEE Transactions on Geoscience and Remote Sensing, vol.32, ed.2, pag.282-289, 1994, DOI: 10.1109/36.295043

MICROCHIP[1]; *"8-bit PIC® Microcontrollers"*, Microchip Technology Incorporated, Chandler, Arizona, USA, Acessado em: 18 de Março de 2013, Disponível em: http://www.microchip.com/pagehandler/en-us/family/8bit/architecture/home.html

MICROCHIP[2]; *"32-bit PIC® Microcontrollers"*, Microchip Technology Incorporated, Chandler, Arizona, USA, Acessado em: 18 de Março de 2013 , Disponível em: http://www.microchip.com/pagehandler/en-us/family/32bit/architecture.html

MINDEK JR., R.B.; *"Development of a programmable logic controller experiential learning platform"*, In Proceedings of the American Society of Engineering Education, Zone I Conference, 2008, Disponível em: https://www.asee.org/documents/zones/zone1/2008/professional/ASEE12008_0055_paper.pdf

NATIONAL SAFETY COUNCIL (NSC); *"Injury facts 2011 Edition"*, Chicago, National Safety Council, 2011 Disponível em: http://www.nsc.org/Documents/Injury_Facts/Injury_Facts_2011_w.pdf

NIST, National Institute of Standards and Technology; "Kolmogorov Smirnov Two Sample", Dataplot, vol.1, 2003 Disponível em: http://www.itl.nist.gov/div898/software/dataplot/refman1/auxillar/ks2samp.htm

NOVUS; *"Controladores Programáveis - Série Dvp-sa / Sx / Sv"*, CLPs e IHMs, Acessado em: 19 de Março de 2013, Disponível em: http://www.novus.com.br/site/default.asp?TroncoID=621808&SecaoID=503607&SubsecaoID=0&Template=../catalogos/layout_produto.asp&ProdutoID=828380

OSISOFT® INC.; "*PI Server Reference Guide*". 2003

RASMUSSEN, J.; *"Information processing and human-machine interaction: an approach to cognitive engineering"*, North-Holland series in system science and engineering, Elsevier Science Limited, vol.12 North Holland, New York, 1986, ISBN-13: 978-0-4440-0987-6

RASPBERRY PI; Consultada em: 27 de Março de 2013 - http://www.raspberrypi.org/

RICHTER, M.; *"Data compression in ALICE by on-line track reconstruction and space point analysis"*, International Conference on Computing in High Energy and Nuclear Physics 2012, vol.396, ed.1, 2012, DOI: 10.1088/1742-6596/396/1/012043

RUSSELL, E. L.; CHIANG, L. H.; BRAATZ, R. D.; *"Fault detection in industrial processes using canonical variate analysis and dynamic principal component analysis"*, vol.51, ed.1, pag.81–93, 2000, DOI: 10.1016/S0169-7439(00)00058-7

SAID, A.; PEARLMAN, W.A.; *"An Image Multiresolution Representation for Lossless and Lossy Compression"*, IEEE Transactions on Image Processing, vol. 5, ed. 9, pag. 1303 – 1310, 1996, DOI: 10.1109/83.535842

SCHNEIDER ELECTRIC, *"Controladores Programáveis para Máquinas Industriais"*, PACs, CLPs e outros Controladores, Acessado em: 19 de Março de 2013, Disponível em: http://download.schneider-electric.com/files?L=en&p=&p_docId=&p_docId=&p_Reference=EIO0000000016&p_EnDocType=User%20guide&p_File_Id=3489350&p_File_Name=EIO0000000016.05.pdf

SCHÖLKOPF, B.; SMOLA, A.; MÜLLER, K.; *"Nonlinear component analysis as a kernel eigenvalue problem"*, Neural Computation, vol.10, ed.5, pag.1299–1399, 1998, DOI: 10.1162/089976698300017467

SEHGAL, V.K.; NITIN; CHAUHAN, D.S.; SHARMA, R.; *"Smart wireless temperature data logger using IEEE 802.15.4/ZigBee protocol"*, TENCON 2008 - 2008 IEEE Region 10 Conference, vol.1, ed.1, pag.1-6, 2008, DOI: 10.1109/TENCON.2008.4766744

SHANNON, C. E.; *"A mathematical theory of communication"*, Bell System Technical Journal, vol.27 pag.379–423,623–656, 1948, DOI Republicação: 10.1145/584091.584093

SHAO, C.; SHAO, Z.; ZHANG, J.; BAI, F.; *"Fault diagnosis design based on qualitative simulation"*, Third International Conference on Intelligent Control and Information Processing (ICICIP), pag.157-161, 2012 DOI: 10.1109/ICICIP.2012.6391507

SHEIKHOLESLAMI, G.; CHATTERJEE, S; ZHANG, A.; *"WaveCluster: A Multi-Resolution Clustering Approach for Very Large Spatial Databases",* Proceedings of the 24th VLDB Conference, 1998 ISBN:1-55860-566-5

SILVEIRA, R. P.; "Novas Metodologias para Compressão de Dados de Processos e para o Ajuste do Sistema PI", Dissertação de Mestrado, Porto Alegre, Universidade Federal do Rio Grande do Sul, 2012

SINGHAL, A.; SEBORG D. E.; *"Effect of data compression on pattern matching in historical data"* Industrial & Engineering Chemistry Research, vol.44, ed.9, pag.3203-3212, 2005 DOI: 10.1021/ie049707a

TAMUSIUNAS, F.; *"Campus Party 6 Medindo a Qualidade da Internet no Brasil"*, Núcleo de Informação e Coordenação do Ponto BR, 29 de Janeiro de 2013,Disponível em:

http://www.nic.br/imprensa/releases/2013/Campus_Party_2013_-_Medindo_a_Qualidade_da_Internet_no_Brasil.pdf

THIELE, G.; RENNER, L.; NIEMEIER, R.; *"Programmable Logic Controllers"*, Control Systems, Robotics and Automation Vol. XXI, EOLSS Publishers Corporation Limited, Reino Unido, 2009 ISBN13: 978-1-84826-610-0

TOYOTA MOTOR CORPORATION; *"Worldwide Operations",* acessado em: 07/02/13, disponível em: http://www.toyota-global.com/company/profile/facilities/worldwide_operations.html

TRAN, V. T.; YANG, B.-S.; GU, F.; BALL A.; *"Thermal image enhancement using bi-dimensional empirical mode decomposition in combination with relevance vector machine for rotating machinery fault diagnosis"*, Mechanical Systems and Signal Processing, vol.38, ed.2, pag.601–614, 2013, DOI: 10.1016/j.ymssp.2013.02.001

USI Usinas Sociais Inteligentes; Porto Alegre, Rio Grande do Sul, Brasil, 2013. Disponível em: http://usibiorefinarias.com.br

VEDAM, H.; VENKATASUBRAMANIAN, *"A B-spline based method for data compression, process monitoring and diagnosis"*, European Symposium on Computer Aided Process Engineering, vol.22, ed.1, pag.S827-S830, 1998, DOI: 10.1016/S0098-1354(98)00158-6

VENKATASUBRAMANIAN, V.; RICH, S.H.; *"An object-oriented two-tier architecture for integrating compiled and deep-level knowledge for process diagnosis"*, Computers & Chemical Engineering, vol.12, ed.9–10, pag.903–921, 1988, DOI: 10.1016/0098-1354(88)87017-0

WALLACE, G. K.; *"The JPEG still picture compression standard"*, Communications of the ACM - Special issue on digital multimedia systems, vol.34, ed.4, pag.30-44, 1991, DOI: 10.1145/103085.103089

WATSON, M.J.; LIAKOPOULOS, A.; BRZAKOVIC, D.; GEORGAKIS, C., *"Wavelet techniques in the compression of process data"*, Proceedings of the 1995 American Control Conference, vol.2, pag.1265-1269, 1995 DOI: 10.1109/ACC.1995.520953

WILLIAMS, M. R.; "A history of computing technology", IEEE Computer Society Press, 1997, ISBN-13: 978-0-8186-7739-7

WINZIP; *"WinZip Feature History"*, WinZip Knowledge Database, atualizado: 16/10/2012, acessado em 22 de Fevereiro de 2013, disponível em: http://kb.winzip.com/kb/entry/294/

ZIV, J.; LEMPEL, A.; *"Compression of individual sequences via variable-rate coding"*, IEEE Transactions on Information Theory, vol.24, ed.5, pag.530- 536, 1978, DOI: 10.1109/TIT.1978.1055934

Apêndice A

Para teste do algoritmo em operação real será usada uma micro destilaria de etanol localizada na Estação Estação Experimental da Universidade Federal do Rio Grande do Sul. A unidade foi fabricada pela USI Usinas Sociais Inteligentes. A unidade em questão possui uma capacidade de até 1000L de etanol diários. Ela opera através da fermentação de insumos agrícolas tais como cana-de-açúcar, batata doce e mandioca.

O sistema é dotado de três dornas de fermentação; uma caldeira; três trocadores de calor em série, operando com água em temperatura ambiente; três bombas, sendo uma delas de maior capacidade para poder alimentar o sistema com o insumo fermentado; existe um destilador de duas etapas. Que conta com uma coluna de destilação aliada a uma coluna de retificação; por último existe uma torre de resfriamento (*USI, 2013*). Como pode ser visto na Figura B.5.1.

Figura B.5.1 – Usinas piloto fabricadas pela USI.(*USI, 2013*)

O sistema original não conta com um sistema de controle, possuindo apenas sensores de temperatura para o destilador. Os acionamentos são feitos de forma manual. Como pode ser observado na Figura B.5.2.

Figura B.5.2 – Sistema de monitoramento e acionamentos da usina piloto. (*USI, 2013*)

Para o teste serão trocadas as válvulas manuais, por válvulas elétricas, serão adicionados sensores de nível, temperatura e concentração em cada uma das três dornas, será colocado um sensor de pressão na caldeira, assim como nas duas partes do destilador. Serão colocados sensores de temperatura nos três trocadores de calor. A bomba para admissão do fermentado será automatizada, porém as bombas de admissão de água para a caldeira e a bomba para retirar o álcool serão mantidas com acionamento manual. Juntamente a isso será colocado um sistema de aquecimento em cada uma das três dornas para que o processo de fermentação possa ser controlado. Um fluxograma simplificado pode ser observado na Figura B.5.3.

Os equipamentos contidos na planta permitem uma avaliação de múltiplas dinâmicas, como a das dornas que tem um caráter lento bem como o destilador que possui dinâmica rápida. Outro fator que esse sistema permite avaliar é o do ruído do processo, que será alto na pressão da caldeira, porém baixo na concentração das dornas.

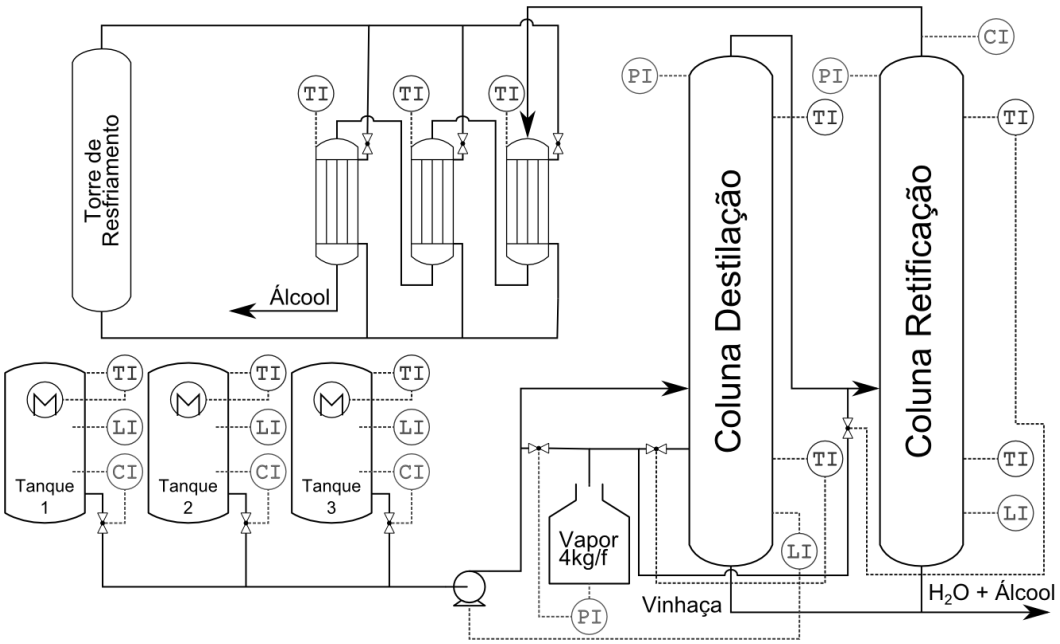

Figura B.5.3 – Sistema de controle simplificado que será instalado na unidade de produção de etanol.

O algoritmo será ajustado para o controle dos 22 sensores e o PID para controlar as 9 válvulas juntamente com a bomba. Além de gerar o histórico dos dados para posterior comparação.

Apêndice B

O algoritmo de SDT fornece parâmetros de tendência nos quais ele se baseia para determinar em um ponto futuro se o mesmo está discrepante dos anteriores. Esse tipo de cálculo é análogo ao parâmetro derivativo do sistema PID.

Se tomarmos o algoritmo tradicional do PID:

$$PID_x = \mathrm{K} \times \left(E_{P_x} + \frac{1}{\mathrm{T}_i} \times E_{I_x} + \mathrm{T}_d \times E_{D_x} \right) \quad (\text{C.1})$$

Temos então que a ação derivativa se apresenta na forma:

$$E_{D_x} = (E_{P_x} - E_{P_{x-1}}) \div (x - x_{-1}) \quad (\text{C.2})$$

Poderíamos então substituir pelo ângulo proveniente do SDT:

$$\alpha_x = (V_{xf} - V_{xi}) \div \Delta x \quad (\text{C.3})$$

Sendo assim o algoritmo dependente do SDT seria um Piα:

$$PI\alpha_x = \mathrm{K} \times \left(E_{P_x} + \frac{1}{\mathrm{T}_i} \times E_{I_x} + \mathrm{T}_\alpha \times E_{\alpha_x} \right) \quad (\text{C.4})$$

Por ser relativo à estabilidade do processo esse algoritmo poderia então, apresentar uma resposta mais adequada ao comportamento. Outro fator para estudo seria a resposta, devido ao intervalo do ângulo do SDT ser variável.

Com isto teríamos um sistema configurável através dos parâmetros do SDT, quando bem ajustados poderiam levar a uma resposta derivativa mais adequada ao processo aumentando o nível de estabilidade da planta.