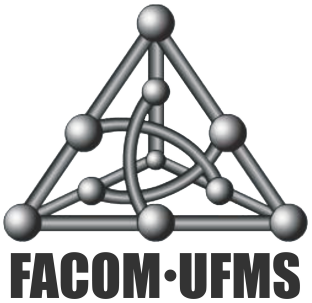
Dissertação de Mestrado

Programação de Microcontroladores Utilizando Técnicas de Tolerância a Falhas

Kleber Kruger

Orientação: Prof. Dr. Fábio Iaione

Área de Concentração: Sistemas de Computação



Faculdade de Computação
Universidade Federal de Mato Grosso do Sul
30 de Julho de 2014

Programação de Microcontroladores Utilizando Técnicas de Tolerância a Falhas

Campo Grande, 30 de Julho de 2014.

Banca Examinadora:

- Prof. Dr. Fábio Iaione (FACOM/UFMS) - Orientador
- Prof. Dr. Irineu Sotoma (FACOM/UFMS)
- Prof. Dr. Gedson Faria (CPCX/UFMS)

Resumo

Os sistemas embarcados abrangem uma grande quantidade de sistemas computacionais e suas aplicações estão cada vez mais presentes no cotidiano das pessoas, principalmente com a expansão da computação ubíqua. Por isso, a ocorrência de falhas nesses sistemas tendem a trazer cada vez mais transtornos e prejuízos financeiros. As falhas podem ocorrer devido aos *bugs* de software, ao envelhecimento dos componentes de hardware, às interferências eletromagnéticas, e por conta de outros fenômenos no meio ambiente que afetam os dispositivos semicondutores. O uso de técnicas de tolerância a falhas aumentam a segurança, pois permite que sistemas continuem funcionando adequadamente após a ocorrência de falhas. Seu princípio básico consiste na redundância, que pode fornecida por meio de hardware, software, dados e processamento. No entanto, a utilização das técnicas de tolerância a falhas é frequente apenas em sistemas de alto risco e de grande porte.

O objetivo deste trabalho foi implementar as técnicas de tolerância a falhas mais viáveis na programação de uma plataforma de prototipagem rápida com microcontroladores. Para avaliar o desempenho das técnicas foi desenvolvido um injetor de falhas por software, e utilizou-se um sistema de estação meteorológica como estudo de caso. Os testes simularam falhas nas leituras dos sensores e falhas nas regiões de memória (por meio da alteração dos dados de alguns endereços de memória) da estação meteorológica.

Ao final, são descritos os resultados mostrando o desempenho do sistema tolerante a falhas em comparação com o sistema não tolerante. Embora o primeiro tenha aumentado levemente o consumo de energia, o tamanho do programa, o uso de memória e o tempo de processamento, o desempenho deste se mostrou eficiente, dado que a quantidade de defeitos diminuiu, principalmente nos testes que injetaram falhas nas leituras dos sensores. Em um teste que injetou 16 falhas na região de memória de dados do microcontrolador a cada ciclo de leitura da estação meteorológica, e 25% de falhas nas leituras dos sensores, o *firmware* sem tolerância a falhas apresentou uma taxa de ocorrência de defeitos de 98,61%, enquanto o *firmware* com tolerância a falhas apresentou 9,21%. Sem a injeção de falhas nas leituras dos sensores e com essa mesma quantidade de falhas injetadas na região de memória de dados, o primeiro apresentou 19,6% enquanto o segundo 4,45%.

Como resultado deste trabalho, uma biblioteca para tolerância e recuperação de falhas, chamada *FaultRecovery* foi criada, a fim de facilitar e auxiliar por meio de um conjunto de classes e macros, a escrita de códigos com implementação de técnicas de tolerância a falhas. Além disso, ela disponibiliza uma estrutura pronta para a recuperação de falhas, baseada em uma máquina de estados.

Palavras-chave: 1) tolerância a falhas; 2) injeção de falhas; 3) sistemas embarcados; 4) programação de microcontroladores.

Abstract

Embedded computers cover a large amount of computational systems and its applications are increasingly present in daily life, particularly with the expansion of ubiquitous computing. Therefore, the occurrence of faults in these systems tends to bring increasingly disorders and financial losses. Faults can occur due to software bugs, aging hardware, electromagnetic interference and others environment phenomena that affect semiconductor devices. The use of fault tolerance techniques increases dependability, because it allows systems to continue functioning properly after the occurrence of faults. Its basic principle consists of redundancy, which may be of hardware, of software, data and processing. However, the use of fault tolerance techniques is frequent only in high risk and large systems.

The aim of this study was to implement the fault tolerance techniques more viable in the programming of a rapid development platform with microcontrollers. To evaluate the performance of the techniques, it was developed a software fault injector, and it was used a weather station system as a study case. The tests simulated faults in the sensor readings and faults in memory regions (through changes of data at some addresses) of the weather station.

Finally, the results showing the performance of the fault tolerant system in comparison with the non-fault tolerant system are described. Although the first one slightly increased the power consumption, the size of the program memory and the processing time, its performance showed to be efficient, since the number of failures decreased, especially in tests that injected faults in sensor readings. In a test that injected 16 faults in the microcontroller data memory region in each read cycle of the weather station, and 25% of faults in sensor readings, the firmware without fault tolerance showed a failure rate of 98,61%, while the firmware with fault tolerance presented 9,21%. Without the injection of faults in sensor readings and with the same number of faults injected into the data memory region, the first one showed 19,6% while the second one 4,45%.

As a result of this work, a library for fault tolerance and recovery, named FaultRecovery was created, in order to facilitate and assist, through a set of classes and macros, the writing of code with implementations of fault tolerance techniques. Furthermore, it offers a structure, ready to the fault recovery, based on a state machine.

Keywords: 1) fault tolerance; 2) fault injection; 3) embedded systems; 4) microcontrollers programming.

Conteúdo

Lista de Figuras **8**

Lista de Tabelas **10**

Lista de Quadros **11**

1 Introdução **12**

1.1 Justificativa 13

1.2 Objetivos 15

1.2.1 Objetivo Geral 15

1.2.2 Objetivos Específicos 15

1.3 Organização da Proposta 16

2 Revisão da Literatura **17**

2.1 Falhas, Erros e Defeitos 17

2.2 Classificação das Falhas 19

2.3 Tolerância a Falhas 20

2.4 Técnicas de Tolerância a Falhas 21

2.4.1 Técnicas Utilizando Redundância de Hardware 21

2.4.2 Técnicas Utilizando Redundância de Software 24

2.4.3 Técnicas Utilizando Redundância de Dados 26

2.4.4 Técnicas Utilizando Redundância de Processamento 29

2.5 Injeção de Falhas 29

2.5.1 Injeção de Falhas por Hardware 30

2.5.2 Injeção de Falhas por Software 31

2.6 Trabalhos Relacionados 32

3 Metodologia **34**

3.1 Sistema de Injeção de Falhas 35

3.1.1 Biblioteca de Injeção de Falhas *FaultInjection* 37

3.1.2 *Firmware* Monitor 41

3.1.3 Programa Monitor de Testes 42

3.2 Técnicas de Tolerância a Falhas Utilizadas 44

3.2.1 Biblioteca de Recuperação de Falhas *FaultRecovery* 45

3.2.2 Alterações no *Firmware* para o Emprego de Tolerância a Falhas . . . 52

3.3 Testes Realizados 57

4 Resultados **60**

4.1 Testes com Injeção de Falhas nas Leituras dos Sensores 60

4.2 Testes com Injeção de Falhas nas Regiões de Memória 63

4.2.1 Injeção de Falhas na Região de Memória de Dados 64

4.2.2 Injeção de Falhas na Região de Memória de Dados e Memória Interna 67

4.3 Testes com Injeção de Falhas nas Leituras dos Sensores e nas Regiões de Memória 70

4.3.1 Injeção de Falhas na Região de Memória de Dados 70

4.3.2 Injeção de Falhas na Região de Memória de Dados e Memória Interna 73

4.4 Consumo de Energia 75

4.5 Comparação do Consumo de Energia, Memória Flash, Memória RAM e Tempo de Processamento 77

5 Conclusão **78**

5.1 Contribuições deste Trabalho 80

5.2 Dificuldades Encontradas 80

5.3 Limitações do Trabalho 81

5.4 Trabalhos Futuros 82

5.4.1 Implementações e Atividades Futuras 82

5.4.2 Redundância de Dados Automática 82

Referências Bibliográficas **87**

Apêndices **90**

A Resultados dos Testes - Eficiência do Método AVG **91**

B Resultados dos Testes - Injeção de Falhas **104**

Lista de Figuras

2.1 Universo dos erros 19
2.2 Redundância modular tripla 22
2.3 Duplicação com comparação 23
2.4 Redundância dinâmica ou ativa 23
2.5 Estados de um sistema com redundância dinâmica 24
2.6 Programação n versões 25
2.7 Blocos de recuperação 25
2.8 Verificação usando um único bit de paridade 26
2.9 Verificação usando paridade bidimensional 27
2.10 Formato dos dados usando código CRC 28
2.11 Exemplo de cálculo CRC 29

3.1 Fluxograma da estação meteorológica original 35
3.2 Sistema de teste 36
3.3 Mapa de memória 38
3.4 Processo de leitura dos sensores na estação meteorológica sem tolerância a falhas 41
3.5 Fluxograma do programa monitor de testes 43
3.6 Recuperação de uma falha 48
3.7 Fluxograma do método *avg*. 51
3.8 Exemplo do método *avg*. 52
3.9 Estação meteorológica com técnicas de tolerância a falhas implementadas . . 53
3.10 Processo de leitura dos sensores 56
3.11 Fluxograma da injeção de falhas no *firmware* sem tolerância a falhas 58
3.12 Fluxograma da injeção de falhas no *firmware* com tolerância a falhas 59

4.1 Registros do método *avg*. 61

4.2 Efetividade do método *avg*. 63

4.3 Resultados dos testes com o *firmware* sem técnicas de tolerância a falhas, com a injeção de falhas na região de memória de dados. 64

4.4 Resultados dos testes com o *firmware* com técnicas de tolerância a falhas, com a injeção de falhas na região de memória de dados. 64

4.5 Resultados dos testes com o *firmware* sem técnicas de tolerância a falhas, com a injeção de falhas na região de memória de dados e memória interna. 67

4.6 Resultados dos testes com o *firmware* com técnicas de tolerância a falhas, com a injeção de falhas na região de memória de dados e memória interna. 68

4.7 Resultados dos testes com o *firmware* sem técnicas de tolerância a falhas, com a injeção de falhas nas leituras dos sensores e na região de memória de dados. 70

4.8 Resultados dos testes com o *firmware* com técnicas de tolerância a falhas, com a injeção de falhas nas leituras dos sensores e na região de memória de dados. 71

4.9 Resultados dos testes com o *firmware* sem técnicas de tolerância a falhas, com a injeção de falhas nas leituras dos sensores e nas regiões de memória de dados e memória interna. 73

4.10 Resultados dos testes com o *firmware* com técnicas de tolerância a falhas, com a injeção de falhas nas leituras dos sensores e nas regiões de memória de dados e memória interna. 73

4.11 Consumo de energia da estação meteorológica. 76

5.1 Saída do exemplo do código *TripledData*. 86

Lista de Tabelas

2.1 Tipos de falhas mais comuns. 32

3.1 Valores das variações de cada parâmetro climático. 44

4.1 Resultados dos testes com o *firmware* sem técnicas de tolerância a falhas, com a injeção de falhas na região de memória de dados. 66

4.2 Resultados dos testes com o *firmware* com técnicas de tolerância a falhas, com a injeção de falhas na região de memória de dados. 66

4.3 Resultados dos testes com o *firmware* sem técnicas de tolerância a falhas, com a injeção de falhas na região de memória de dados e memória interna. 69

4.4 Resultados dos testes com o *firmware* com técnicas de tolerância a falhas, com a injeção de falhas na região de memória de dados e memória interna. 69

4.5 Resultados dos testes com o *firmware* sem técnicas de tolerância a falhas, com a injeção de falhas nas leituras dos sensores e na região de memória de dados. 72

4.6 Resultados dos testes com o *firmware* com técnicas de tolerância a falhas, com a injeção de falhas nas leituras dos sensores e na região de memória de dados. 72

4.7 Resultados dos testes com o *firmware* sem técnicas de tolerância a falhas, com a injeção de falhas nas leituras dos sensores e nas regiões de memória de dados e memória interna. 74

4.8 Resultados dos testes com o *firmware* com técnicas de tolerância a falhas, com a injeção de falhas nas leituras dos sensores e nas regiões de memória de dados e memória interna. 74

4.9 Tempo médio de processamento. 77

4.10 Comparação do consumo de energia, memória *flash*, memória RAM e tempo de processamento. 77

Lista de Quadros

1 Exemplo de código com falha. 18

2 Método injetor de falhas 39

3 Código que configura a máquina de estados 47

4 Método que calcula a média das leituras dos sensores desprezando as leituras fora de uma variação definida 50

5 Método sem tolerância a falhas responsável por ler todos os sensores da estação meteorológica 54

6 Método com tolerância a falhas responsável por ler todos os sensores da estação meteorológica 55

7 Classe que automatiza a redundância de dados das variáveis de tipos primitivos. 84

Capítulo 1

Introdução

O ser humano, ao longo dos tempos, criou diversos mecanismos para facilitar e melhorar as tarefas essenciais do dia a dia e o uso da tecnologia está presente nesse avanço. Seja em tarefas simples ou mais elaboradas, o homem moderno está acostumado ao uso de diversos equipamentos tecnológicos, tais como computadores, celulares, *tablets* e outros dispositivos eletrônicos. O uso de sistemas computacionais pode estar presente, por exemplo, em um escritório, mediante equipamentos de informática e impressoras; em uma fábrica, por meio de máquinas e robôs; e até mesmo em uma rede de processadores que controlam um automóvel, avião ou navio. Quando o equipamento é baseado em um sistema microprocessado e executado em um dispositivo de propósito específico, no qual o computador é totalmente dedicado ao dispositivo, dá-se o nome de sistema embarcado [1].

Segundo Patterson e Hennessy [2], os sistemas embarcados (*embedded computers* ou *embedded systems*) correspondem a maior classe de computadores e abrangem uma grande faixa de aplicações e desempenhos. Estudos indicam que normalmente existem em uma residência, um ou dois computadores *desktop* e uma a duas dezenas de sistemas embarcados (computadores embutidos). Essa diferença tem se tornado ainda maior, pois o número de sistemas embarcados tem crescido nos últimos anos [3], principalmente com a expansão da computação ubíqua, uma vez que alguns equipamentos antes construídos com pouco ou nenhum recurso computacional tornaram-se mais sofisticados, incorporando algum tipo de sistema embarcado.

Diferentemente das outras classes de computadores, em que os sistemas computacionais são preparados para executarem aplicações de propósito geral, nos sistemas embarcados o computador é totalmente dedicado à uma aplicação de propósito específico. Essa aplicação executa um conjunto de tarefas predefinidas e possui requisitos exclusivos que geralmente combinam bom desempenho com rigorosas limitações de custo e consumo de energia. Por esse motivo, dispositivos que executam aplicações embarcadas costumam ser equipados com microcontroladores ao invés de microprocessadores de uso geral. Segundo Ball [4], microcontroladores são componentes eletrônicos que integram, em uma única pastilha, um processador (geralmente de baixa frequência de *clock*) e vários outros dispositivos como, conversores analógico-digital, memórias, temporizadores, interface de comunicação serial, e outros recursos.

Os fabricantes, ao lançarem um produto, se preocupam em definir um *hardware* que

provê eficientemente suas funcionalidades, mas em contrapartida o custo e o consumo de energia geralmente são medidas importantes na definição do projeto. Um telefone celular, por exemplo, até pouco tempo atrás, precisava apenas de um processador rápido o suficiente para atender as funcionalidades de comunicação do aparelho. Com o avanço tecnológico dos aparelhos celulares, esse requisito foi rediscutido, e *smartphones* modernos desempenham funções próximas de um computador *desktop*. Entretanto, outros dispositivos, como tocadores de MP3 e rádios digitais de telecomunicação, continuam com essa abordagem, portanto, para estes equipamentos, minimizar os custos e o consumo de energia são os objetivos mais importantes.

Infelizmente, danos graves causados por falhas em equipamentos já ocorreram na história da humanidade [5] e esses fatos destacam a importância da tolerância a falhas em sistemas computacionais. A tolerância a falhas é a propriedade que permite a um sistema continuar funcionando adequadamente, mesmo que em um nível reduzido, após a manifestação de falhas em alguns de seus componentes [6]. O conceito foi apresentado originalmente por Avizienis em 1967 [7]. Todavia, estratégias para concepção de sistemas mais confiáveis já eram usadas desde a construção dos primeiros computadores [8]. Porém, mesmo sendo estratégias antigas, as técnicas de tolerância a falhas nem sempre são empregadas. Em áreas como telefonia, aviação e finanças é uma prática comum, entretanto no desenvolvimento de sistemas embarcados mais simples isso normalmente não ocorre. Cabe observar que os sistemas embarcados podem ser encontrados também em aplicações críticas, tais como as aplicações militares e aeroespaciais [9].

1.1 Justificativa

Os sistemas embarcados, apesar do baixo custo, precisam tolerar falhas, pois erros em aplicações podem causar danos e prejuízos. Resultados de falhas podem variar de perturbadoras (quando causam transtorno ao usuário) a devastadoras (quando causam prejuízo físico ou financeiro) [2]. Na agricultura, um exemplo de utilização de sistema embarcado são as estações meteorológicas presentes em sistemas de previsão de doenças na lavoura, tais como a ferrugem da soja e outras. Esses sistemas medem os parâmetros climáticos e transmitem a um computador central, que processa os dados e disponibiliza os índices de infecção das doenças para o agricultor em um site ou via telefone. Esses índices ajudam os agricultores na definição do momento adequado à aplicação dos defensivos [10, 11]. Falhas nesses equipamentos podem causar diversos danos e prejuízos. A interrupção na coleta de dados climáticos pode fazer com que o sistema de previsão não detecte um foco de doença antecipadamente e a doença se espalhe rapidamente, destruindo a plantação.

Na indústria de *software* como um todo, existem em média de cinco a vinte falhas para cada mil linhas de código e já existem no mercado sistemas embarcados com *softwares* contendo mais de um milhão de linhas de código. Mesmo com um padrão CMM (*Capability Maturity Model*) nível três (padrão intermediário, em que os processos são definidos e gerenciados) [12] pode-se esperar milhares de problemas e *recalls*, que em termos financeiros resultam em centenas de milhões de dólares de prejuízo [13].

As falhas também podem ser causadas pelo desgaste dos componentes de *hardware* [14] (envelhecimento), por interferências eletromagnéticas [15] e por outros fenômenos no meio

ambiente que afetam os dispositivos semicondutores. A proximidade de circuitos eletrônicos com equipamentos emissores de ruído eletromagnético pode afetar o funcionamento correto do dispositivo, principalmente em equipamentos com pouca ou nenhuma blindagem [16]. A blindagem eletromagnética consiste em colocar um material (normalmente metal) no caminho entre um circuito gerador de ruído e outro que sofra interferência. O quanto a blindagem reduz a intensidade das interferências eletromagnéticas depende das propriedades eletromagnéticas do material da blindagem, da espessura, da distância entre a fonte de interferência e a blindagem, da potência do ruído e de detalhes construtivos [17]. Entretanto, cabe observar que a blindagem não impede totalmente as interferências eletromagnéticas e seu uso tem um custo financeiro e de espaço e peso ao equipamento. Por essas razões, a compatibilidade eletromagnética tem sido bastante estudada e normatizada nas últimas décadas, pois estabelecer a compatibilidade eletromagnética não é uma tarefa simples e precisa acompanhar todas as etapas do projeto, desde sua concepção. A compatibilidade eletromagnética corresponde à capacidade de um equipamento eletrônico funcionar sem ser afetado por ruído eletromagnético, e sem gerar ruído que venha a afetar outros equipamentos [16].

Além disso, no meio ambiente existe uma grande quantidade de partículas atômicas e subatômicas em movimento. Conforme Karnik *et al.* [18], aproximadamente 92% dessas partículas são nêutrons, ∼4% são píons, ∼2% são prótons e ∼2% são múons (ao nível do mar), gerados principalmente pela interação dos raios cósmicos com a atmosfera terrestre. Essas partículas têm a capacidade de gerar a ionização direta de dispositivos semicondutores. Nas memórias e dispositivos semicondutores digitais, essa interação física provoca um fenômeno denominado *bit flip*. Esse fenômeno faz com que uma partícula que atinge sobre um elemento semicondutor, tenha a tendência de mudar o estado, de "0" para "1" ou de "1" para "0", indistintamente [17], e isso, consequentemente, pode causar falhas em um dispositivo de *hardware*.

As falhas naturais provenientes desse fenômeno têm sido descritas teoricamente desde 1962 [19] e falhas reais atribuídas a raios cósmicos têm sido descritas desde 1975 [20]. Karnik *et al.* [18] afirma que a incidência destas falhas na natureza é esporádica e que os dispositivos expostos ao espaço livre ficam mais vulneráveis a este tipo de falhas, já que a atmosfera da Terra atenua grande parte das radiações cósmicas presentes no espaço interplanetário. Cabe observar que o estresse por temperatura, tensão elétrica ou fluxo de corrente, além da exposição do equipamento a lugares úmidos ou com radiação, podem contribuir para o aumento da manifestação das falhas nos dispositivos semicondutores [14].

Há também os problemas de falhas intrínsecas na área da física de semicondutores. As falhas intrínsecas são falhas internas do dispositivo semicondutor e têm sido um assunto constantemente estudado em pesquisas. Um exemplo são as falhas provocadas pelas limitações das propriedades físicas do material semicondutor [14]. Incluso nesses problemas estão: a eletromigração (*Electromigration*), e os fenômenos HDC (*Hot Carrier Degradation*) e TDDB (*Time Dependent Dielectric Breakdown*) [21].

Dado o exposto, viu-se que os sistemas embarcados são a maior classe de computadores e com a expansão da computação ubíqua seu uso é cada vez mais frequente no cotidiano das pessoas. Portanto, com a utilização massiva, as falhas tendem a causar transtornos cada vez maiores, uma vez que os sistemas embarcados estão sujeitos a diversos fatores causadores de falhas, seja em razão de *bugs* de software, fenômenos físicos do meio ambiente que afetam o hardware, interação dos dispositivos em locais onde há grande interferência

eletromagnética, ou o desgaste de seus componentes. Cabe observar que o desenvolvimento de sistemas de *software* e *hardware* com tolerância a falhas e disponibilidade permanente é visto pela Sociedade Brasileira de Computação como um dos grandes desafios da computação no Brasil para o período 2006 a 2016 [22].

1.2 Objetivos

1.2.1 Objetivo Geral

O objetivo deste trabalho foi estudar as técnicas de tolerância a falhas e aplicar as mais viáveis na programação de uma estação meteorológica, usada como um estudo de caso para avaliar o desempenho das técnicas em sistemas embarcados de baixo custo. A fim de testar as técnicas utilizadas, outro objetivo foi criar um sistema de injeção de falhas que fornecesse a localização das mesmas, e um esquema automatizado de testes para monitorar os resultados em tempo de execução.

1.2.2 Objetivos Específicos

- Pesquisar as técnicas de tolerância a falhas presentes na literatura, identificar os pontos mais vulneráveis do *firmware* da estação meteorológica original (não tolerante a falhas), e analisar quais técnicas de tolerância a falhas seriam mais viáveis em um sistema embarcado de baixo custo;
- Alterar o *firmware* original (sem tolerância a falhas) da estação meteorológica, empregando as técnicas de tolerância a falhas definidas (baseadas na redundância de dados e de processamento);
- Criar um sistema eficiente de leituras do sensor que desprezasse valores de leituras acima de uma variação pré-definida;
- Criar um sistema injetor de falhas que fornecesse a localidade das falhas injetadas;
- Criar um sistema para monitorar os resultados dos testes em tempo de execução;
- Dado que algumas falhas travam o equipamento, principalmente quando exposto a uma quantidade excessiva de falhas, foi necessário criar um mecanismo de recuperação que permitisse ao sistema voltar à ação em que estava (mascaramento da falha) antes de ficar travado devido à falha;
- Criar uma biblioteca que facilitasse a implementação de códigos tolerantes e falhas; e
- Identificar quais técnicas mostraram melhor desempenho nos testes.

1.3 Organização da Proposta

Os conceitos e técnicas de tolerância e injeção de falhas são discutidos no Capítulo 2, dividido em seis seções. Na Seção 2.1 conceitua-se os termos comuns ao estudo de falhas, e na Seção 2.2 são definidas as classificações das mesmas. Na Seção 2.3 e na Seção 2.4 são exibidas as técnicas de tolerância a falhas mais conhecidas na literatura. Na Seção 2.5 são explicadas as técnicas de injeção de falhas, necessárias para testes em aplicações. Por último, na Seção 2.6 são descritos alguns trabalhos relacionados.

As implementações deste trabalho são exibidas no Capítulo 3, dividido em três seções. Na Seção 3.1 explica-se o sistema injetor de falhas desenvolvido e utilizado nos testes. Na Seção 3.2 descreve-se as técnicas que foram implementadas no *firmware* da estação meteorológica, e na Seção 3.3 relata-se os testes que foram realizados.

No Capítulo 4 são mostrados os resultados dos testes realizados e uma discussão sobre o desempenho, em termos de segurança, do *firmware* com implementação de técnicas de tolerância a falhas em comparação com o *firmware* sem tolerância a falhas. No Capítulo 5 são expostas as considerações finais deste trabalho. Essa análise final é baseada na comparação entre a quantidade de defeitos ocorridos e no uso de recursos, como memória, consumo de energia e tempo de processamento em ambos os *firmwares*.

Capítulo 2

Revisão da Literatura

Neste capítulo são apresentados os conceitos e técnicas de tolerância a falhas presentes na literatura, e as técnicas de injeção de falhas mais utilizadas. Na Seção 2.1 descreve-se os termos comuns ao estudo de falhas em sistemas computacionais, explicando a diferença entre os termos: falhas, erros e defeitos. Na Seção 2.2 explica-se as classificações das falhas de acordo com sua duração, extensão e natureza. Na Seção 2.3 e na Seção 2.4 são explicados os conceitos básicos da tolerância a falhas e apresentadas as técnicas mais conhecidas na literatura, sejam elas baseadas na redundância de hardware, software, dados e processamento. Na Seção 2.5 são descritas as técnicas de injeção de falhas frequentemente utilizadas para testar as propriedades de segurança de um sistema tolerante a falhas, entre elas, as técnicas de injeção de falhas por hardware e por software. Ao final, na Seção 2.6 são descritos alguns trabalhos relacionados.

2.1 Falhas, Erros e Defeitos

Quando aplicados em sistemas computacionais, os termos falhas, erros e defeitos têm diferentes significados [9]. A seguir aparecem as definições de Laprie [23, 24] e Anderson [25] sobre esses conceitos.

Uma falha (*fault*) ocorre no nível mais baixo do *hardware*, é um defeito físico, uma imperfeição, ou uma vulnerabilidade que ocorre em algum componente de *hardware* (por exemplo, uma flutuação na fonte de alimentação ou a interferência eletromagnética sobre um circuito eletrônico), ou no *software* (a troca de um "$\geq$" por um "$>$" em determinado trecho de código do programa). Desse modo, as falhas estão associadas ao universo físico ou ao projeto do sistema.

A ocorrência de uma falha pode gerar um erro. O erro (*error*) é a representação da falha no universo da informação. Tem-se um erro quando, por consequência de uma falha, a informação é alterada. Portanto, um erro é a manifestação da falha no sistema. Quando o sistema encontra-se nesse estado, a informação está incorreta.

O defeito (*failure*) é o não cumprimento de alguma ação conforme o esperado e ocorre quando a informação incorreta é processada sem o erro ser detectado e tratado por algum sistema de correção de erros. O defeito é perceptível ao usuário, por esse motivo os defeitos

estão associados ao universo do usuário.

Uma falha não necessariamente leva a um estado de erro, pois um trecho de código mal testado com falha pode, por exemplo, pertencer ao bloco básico interno de uma estrutura condicional e nunca ser executado. Essa situação é exemplificada no código do Quadro 1. Considerando-se que o bloco básico interno da estrutura condicional apresenta falhas, este código só será executado se a comparação for verdadeira, ou seja, se a função *fopen* retornar o valor NULL. Um erro também não ocasiona obrigatoriamente um defeito, pois uma informação incorreta pode não ser utilizada [26]. Logicamente, a primeira impressão é que não há sentido produzir informações que não sejam usadas no decorrer do programa, tanto é que os compiladores notificam situações em que uma variável armazena um dado que não é utilizado. No entanto, nem toda informação produzida é utilizada. A função *printf* do exemplo abaixo retorna o número de *bytes* impressos no terminal, mas essa informação não é utilizada.

```
1  int vetor[20];
2
3  FILE *fp = fopen("arquivo.txt", "w");
4
5  if (fp == NULL)
6  {
7      /*
8       * Exemplo de um trecho de código com falha.
9       * O código abaixo acessa uma posição não existente do vetor,
10      * podendo alterar uma região indevida de memória e até mesmo
11      * o valor de outra variável.
12      */
13
14     for (int i = 0; i <=20; i++)
15     {
16         vetor[i] = rand() % 100;
17         printf("%d\n", vetor[i]);
18     }
19 }
```

Quadro 1: Exemplo de código com falha.

Na Figura 2.1 mostra-se uma representação, adotada nesse texto, para os conceitos de falha, erro e defeito. Como mencionado acima, falhas estão associadas ao universo físico, erros ao universo da informação e defeitos ao universo do usuário. Um outro exemplo, dado por Nelson [9] e Weber [27] é um *chip* de memória que apresenta uma falha do tipo "travado-em-zero" (*stuck-at-zero*) em um de seus bits (falha no universo físico), isso pode provocar uma interpretação errada da informação (erro no universo da informação) e como resultado o sistema pode, por exemplo, negar autorização de embarque para todos os passageiros de um voo (defeito no universo do usuário).

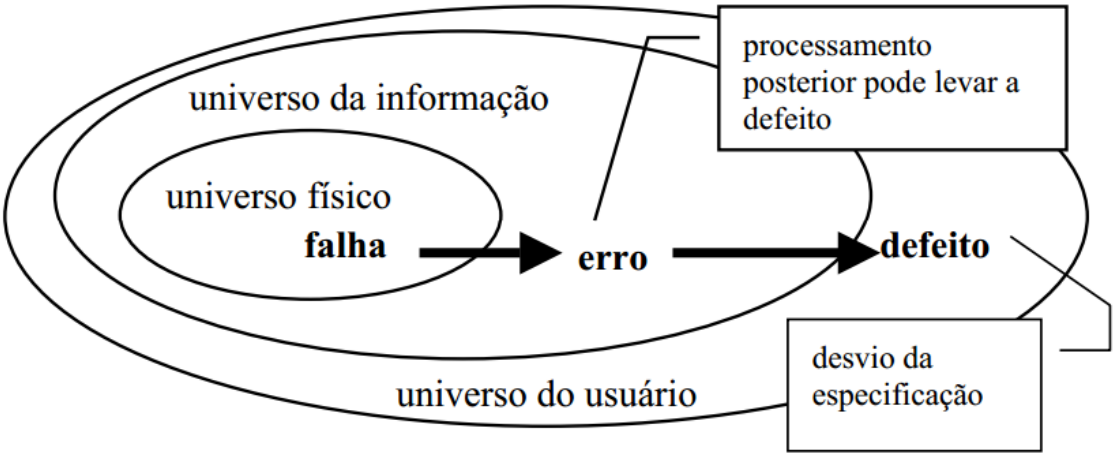


Figura 2.1: Universo dos erros. Retirado de [28].

2.2 Classificação das Falhas

Um sistema computacional pode apresentar falhas tanto no *hardware*, quanto no *software*. As falhas podem surgir durante todas as etapas de concepção: especificação, projeto, desenvolvimento, fabricação, montagem e instalação. A maioria ocorre antes da implantação completa do sistema, sendo frequentemente descobertas e eliminadas ao longo dos testes. As falhas que não são removidas reduzem a segurança do sistema [14].

As falhas apresentam comportamentos distintos, sendo elas classificadas de acordo com sua duração, extensão e natureza. Dentre vários artigos, o que melhor define as propriedades de falhas é o trabalho de Nelson [9]. Nelson explica que a duração de uma falha pode ser transiente, intermitente ou permanente. Uma falha transiente é geralmente provocada por perturbações externas, existe por um período finito de tempo e não é recorrente. Um sistema com uma falha intermitente alterna entre operações com falhas e sem falhas. Falhas permanentes são as que não desaparecem com o tempo, sendo elas resultado de falhas de componentes, danos físicos, ou erros de projeto.

A extensão de uma falha é determinada pela área afetada no nível de abstração considerado: falhas locais afetam os componentes individuais e falhas globais afetam vários componentes [9]. Weber [27] explica que a natureza das falhas é determinada pela natureza de onde a falha foi gerada, podendo ser resultado de falhas em *hardware*, *software*, projeto, entre outros.

Conforme Ziade *et al.* [29], as falhas presentes no *software* são sempre consequência de um erro no projeto, seja na especificação ou na codificação. Ele ainda afirma que muitas delas são latentes e se manifestam apenas durante a operação, especialmente sob cargas de trabalho grandes ou incomuns.

Uma vez que as falhas de *software* são resultados de erros no projeto, pode-se supor que todas as falhas de *software* são permanentes, pois a execução do trecho de código com falhas tende a causar sempre o mesmo defeito. Curiosamente, a prática mostra que apesar da natureza permanente, o seu comportamento é transiente, ou seja, quando um erro ocorre, ele pode não ser observado novamente, mesmo que seja tomado grande cuidado para repetir

a situação em que ocorreu, pois um *bug* pode ser provocado em razão de um conjunto de fatores, envolvendo questões temporais e vários componentes de um sistema, ou por alguma situação rara e irreproduzível [30]. Em outras palavras, algumas falhas podem ser difíceis de serem reproduzidas, pois sua manifestação depende de um conjunto de ações simultâneas.

Velazco *et al* [30] afirma que a maioria das falhas encontradas são permanentes. No entanto, muitos estudos mostram que as falhas transientes e intermitentes são até mais frequentes do que as falhas permanentes. O problema é que elas são muito mais difíceis de serem observadas.

Durante o processo de desenvolvimento de *software*, as falhas podem ser criadas em todas as etapas: definição de requisitos, especificações de requisitos, projeto, implementação, teste e implantação. Essas falhas podem ser classificadas como:

- Falhas de funcionalidade: implementação incorreta ou ausente que requer uma alteração no projeto para ser corrigida.
- Falhas de algoritmo: implementação incorreta ou ausente que pode ser corrigida sem a necessidade de alteração do projeto.
- Falhas de tempo/serialização: serialização ausente ou incorreta de recursos compartilhados.
- Falhas de verificação: validação de dados ausente ou incorreta (incluem, por exemplo, condições incorretas na verificação de entrada).
- Falhas de atribuição: os valores atribuídos são incorretos ou não foram atribuídos. Este erro é bastante comum. Ao declarar uma variável ou um vetor e não inicializá-lo, as regiões de memória podem conter valores inválidos e isso pode passar despercebido na implementação do programa.

2.3 Tolerância a Falhas

Segundo Nelson [9] e Somani *et al.* [31], o objetivo de um projeto tolerante a falhas é melhorar a segurança (do inglês, *dependability*). Alguns autores traduzem este termo para "dependabilidade", porém, por ser uma palavra inexistente na língua portuguesa, será adotada a palavra "segurança" nesse trabalho. A segurança permite a um sistema realizar sua função mesmo na presença de um determinado número de falhas.

Sistemas baseados em tolerância a falhas, de um modo geral, precisam garantir, entre outros requisitos, confiabilidade e disponibilidade. Essas duas características são frequentemente usadas como medidas para avaliar a qualidade de um sistema tolerante a falhas. A confiabilidade, $R(t)$, é a probabilidade condicional de que um sistema pode realizar a sua função no tempo t, uma vez que estava operacional no momento t_0. Assim, $R(t)$ é uma função das falhas afetando o sistema, e de quaisquer mecanismos que toleram falhas quando um imprevisto ocorre. Muitos sistemas de tempo real, tais como os utilizados em uma aeronave ou controle de uma usina nuclear, requerem um alto $R(t)$, pois uma única falha pode ser fatal. A disponibilidade corresponde à probabilidade (representada por um

valor em porcentagem) de um sistema estar operacional em um momento qualquer, definido aleatoriamente ou de acordo a alguma especificação do usuário [9].

O princípio básico da construção de sistemas tolerantes a falhas é prover recursos extras (redundância) para superar os efeitos de um mau funcionamento e aumentar a segurança [9, 27, 32, 31, 33]. A redundância pode assumir a forma de *hardware* adicional, que ignora um sinal errado ou substitui um subsistema com falhas, ou por *software* adicional, que pode permitir a reexecução de um programa após a detecção de uma falha. Neste caso, precisa-se também de alguma redundância de tempo, uma folga incorporada ao projeto, de modo que o sistema tenha tempo para executar novamente o programa sem deixar as especificações de desempenho exigidas [9]. É importante ressaltar que redundâncias introduzem tempo e custos adicionais ao projeto [31], no entanto, aumentam a segurança.

2.4 Técnicas de Tolerância a Falhas

Todas as técnicas de tolerância a falhas envolvem alguma forma de redundância, motivo este que leva os sistemas tolerantes a falhas serem chamados também de "sistemas redundantes" [27]. Segundo Dubrova [34], Nelson [9] e Somani *et al.* [31], a redundância pode ser obtida por meio da:

- Redundância de *hardware*;
- Redundância de *software*;
- Redundância de informação;
- Redundância de processamento.

Weber [27] ainda afirma que todas as formas de redundância têm algum impacto no sistema, seja no custo, no desempenho, na área (tamanho, peso), ou na energia consumida. Assim, apesar de ser uma solução para tolerância a falhas, o uso de redundância em qualquer projeto deve ser bem ponderado. A seguir são explicados os tipos de redundância aqui mencionados.

2.4.1 Técnicas Utilizando Redundância de Hardware

A redundância de *hardware* está baseada na replicação de componentes físicos. Entre as técnicas de redundância de *hardware*, pode-se citar a redundância de *hardware* passiva e a redundância de *hardware* ativa ou dinâmica. Na redundância de *hardware* passiva todos os elementos redundantes executam a mesma tarefa e o resultado é determinado por votação. Exemplos são as técnicas NMR, TMR e DWC.

- Redundância de n módulos (NMR - *N-Modular Redundancy*). Essa técnica consiste em replicar o *hardware* responsável pelo processamento da informação em n módulos. Os módulos executam simultaneamente e o resultado do processamento é determinado

por um sistema de votação que define o resultado da saída mediante a maioria ou do valor médio calculado. Essa forma de construção evita que falhas em apenas um dos componentes gere erros no sistema, uma vez que a falha é encoberta pelo restante do *hardware* redundante no circuito de votação.

- Redundância Modular Tripla - (TMR - *Triple Modular Redundancy*). Versão do NMR com 3 módulos de *hardware* redundantes. Nessa versão o *hardware* responsável pelo processamento é triplicado e o sistema de votação é responsável por definir a saída do processamento [34, 35]. Nesse tipo de implementação o sistema de votação costuma ser o ponto crítico no projeto, pois falhas no votador podem afetar a confiabilidade do sistema. Devido a essa fragilidade, implementações NMR sem sistema de votação têm sido alternativas eficientes, como mostrado em [36]. Na Figura 2.2 ilustra-se um exemplo de TMR.

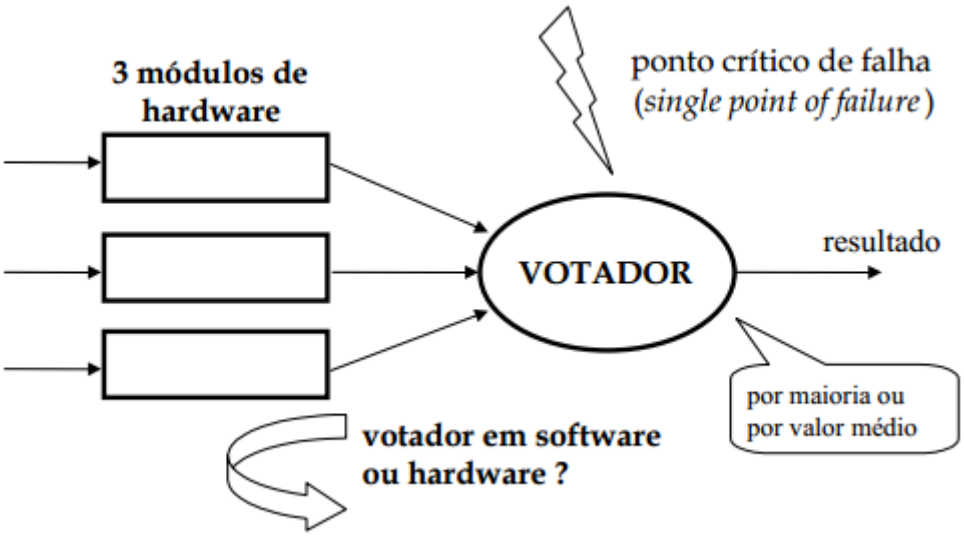


Figura 2.2: Redundância modular tripla. Retirado de [27].

- Duplicação com comparação (DWC - *Duplication With Comparation*). Conforme exibido na Figura 2.3, nessa implementação o *hardware* é apenas duplicado. A comparação das saídas evita que os resultados com erro sejam propagados e gerem defeitos no sistema. Este tipo de implementação apenas evita que o erro seja propagado, pois em caso de falha, o sistema aborta a operação ou toma outra medida de modo a anular a propagação do erro [28, 35].

Na redundância de *hardware* ativa ou dinâmica, a tolerância a falhas é provida por técnicas de detecção, localização e recuperação. Essa técnica é usada em aplicações que suportam permanecer em um estado de erro durante um curto período de tempo. Esse tempo é necessário para a detecção do erro e recuperação para um estado livre de falhas. Um exemplo dessa técnica é a utilização de módulos "estepes", que podem substituir o módulo principal quando o mesmo começa a apresentar defeitos [27]. Este procedimento é chamado de *hot standby*, quando o estepe é conectado eletricamente ao sistema, minimizando a descontinuidade do processamento durante o chaveamento entre os módulos, ou de *cold*

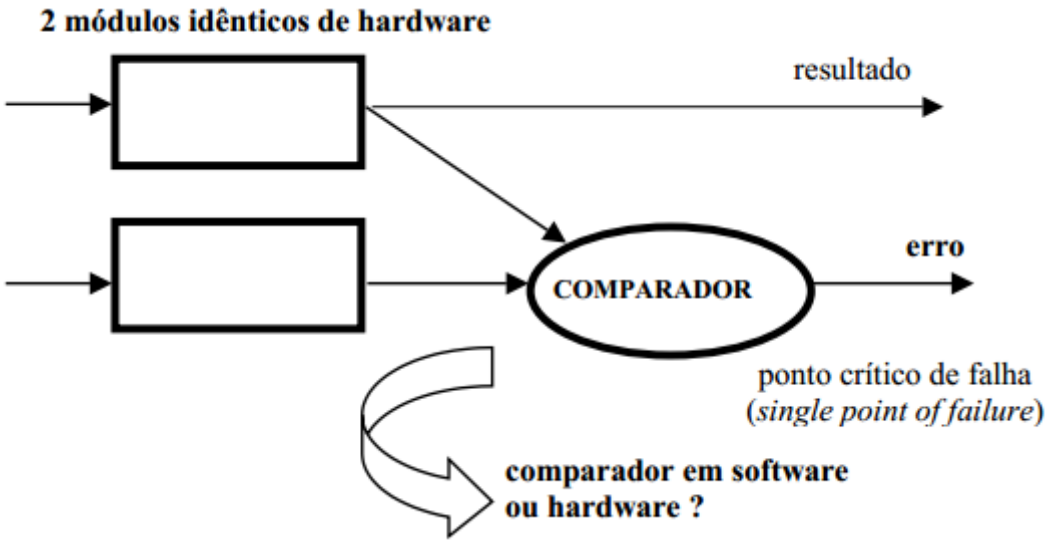


Figura 2.3: Duplicação com comparação (DWC). Retirado de [28].

standby, quando o módulo estepe começa a operar somente depois de conectado após uma falha. Essa estratégia possui um atraso maior, mas aumenta a vida útil dos módulos estepe [28]. Na Figura 2.4 e na Figura 2.5 são mostrados exemplos de utilização dessa técnica.

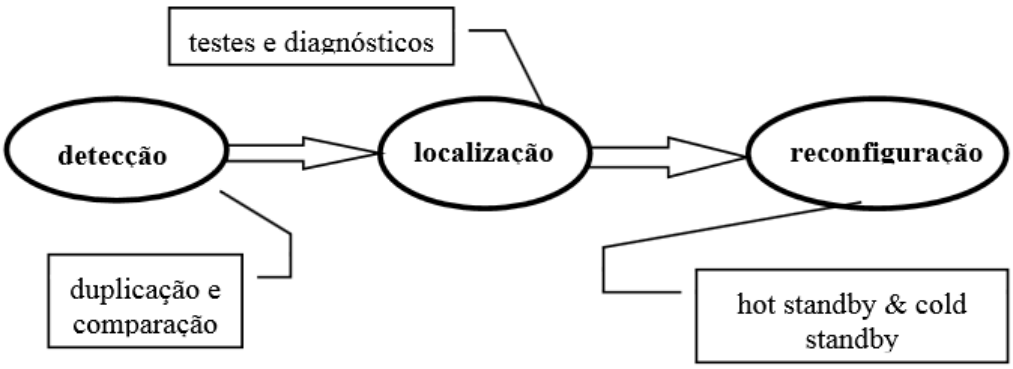


Figura 2.4: Redundância dinâmica ou ativa. Retirado de [28].

Técnicas Utilizando o Temporizador *Watchdog*

Outra técnica baseada em hardware constantemente utilizada na detecção e recuperação de erros é o temporizador *watchdog*. O *watchdog* é um dispositivo eletrônico temporizador usado para detectar erros e reiniciar o equipamento quando o sistema trava. Conceitualmente, a interface de hardware do *watchdog* consiste de um único registrador contendo um valor de limiar. O *watchdog* reinicia o equipamento se o temporizador atinge o valor do limiar. Logo, a menos que o limiar seja aumentado periodicamente, o computador irá reiniciar [37]. No entanto, o mais comum atualmente são *watchdogs* com dois registradores, um guarda o tempo cronometrado pelo temporizador e o outro, o valor de limiar (normalmente um tempo fixo ou ajustado conforme cada operação do programa). O registrador do temporizador deve ser

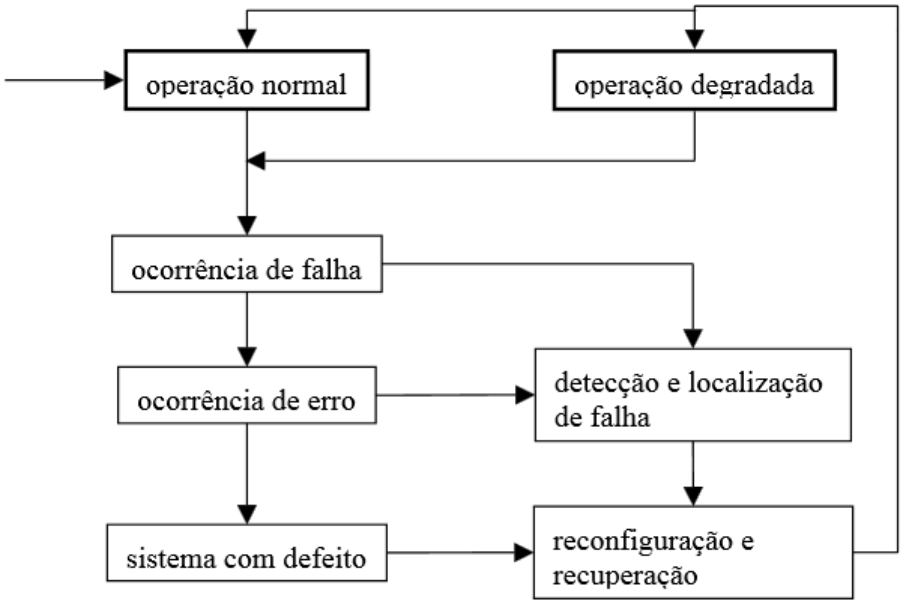


Figura 2.5: Estados de um sistema com redundância dinâmica. Retirado de [28].

reiniciado periodicamente pelo programa antes que atinja o valor do limiar. Nos sistemas embarcados, esta técnica não exige a replicação de hardware, uma vez que a maioria dos microcontroladores já possuem este dispositivo incorporado.

2.4.2 Técnicas Utilizando Redundância de Software

Segundo Brilliant *et al.* [38], a redundância em *software* apresenta características distintas à redundância em *hardware*, pois a replicação de *software* idênticos vão apresentar erros idênticos. Algumas técnicas de redundância em *software* são:

- Diversidade ou programação n-versões. Consiste em usar redundância mediante a implementação de diversas versões do *software*, desenvolvido por diferentes equipes de programadores. O princípio básico dessa técnica é que implementações distintas dificilmente apresentarão as mesmas falhas. Conforme ilustrado na Figura 2.6, as informações são processadas sincronizadamente em cada versão do *software* e os resultados são escolhidos por votação. Experimentos comprovam que o número de falhas idênticas (que não seriam detectadas por essa técnica) é significativamente menor que o número total de falhas. No entanto, existem algumas desvantagens, como o aumento dos custos de produção e manutenção do *software*, e a complexidade de sincronização das versões. Há também variações do uso dessa técnica, podendo a mesma ser utilizada somente na fase de testes, sendo a versão com maior exatidão escolhida para integrar o sistema [39, 38, 31, 40].

- Blocos de recuperação. Essa técnica utiliza n módulos: um módulo principal e outros $n-1$ módulos secundários. Os blocos de recuperação são semelhantes à programação n-versões, porém, apenas o programa primário é executado, sendo necessário o uso dos programas secundários somente quando ele apresentar erro. Na técnica dos blocos de

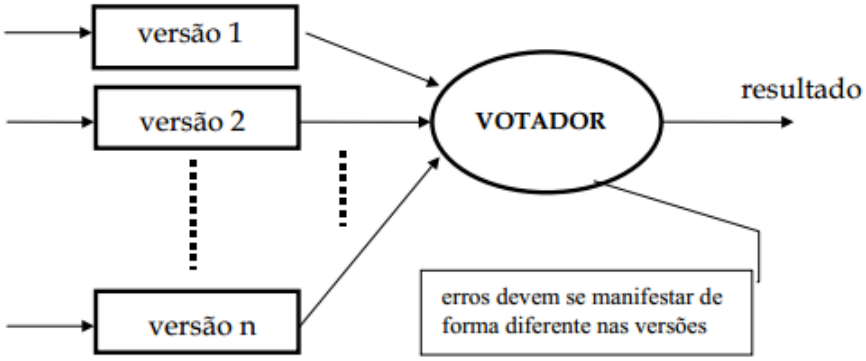


Figura 2.6: Programação n-versões. Retirado de [27]

recuperação, mostrada na Figura 2.7, quando o módulo primário conclui a execução, o seu resultado é verificado por um teste de aceitação. Se a saída não é aceitável, os módulos secundários executam até que a saída de um módulo seja aceita ou até os módulos secundários esgotarem. Essa técnica tolera $n - 1$ falhas, em caso de falhas independentes nas n versões [9, 27, 31].

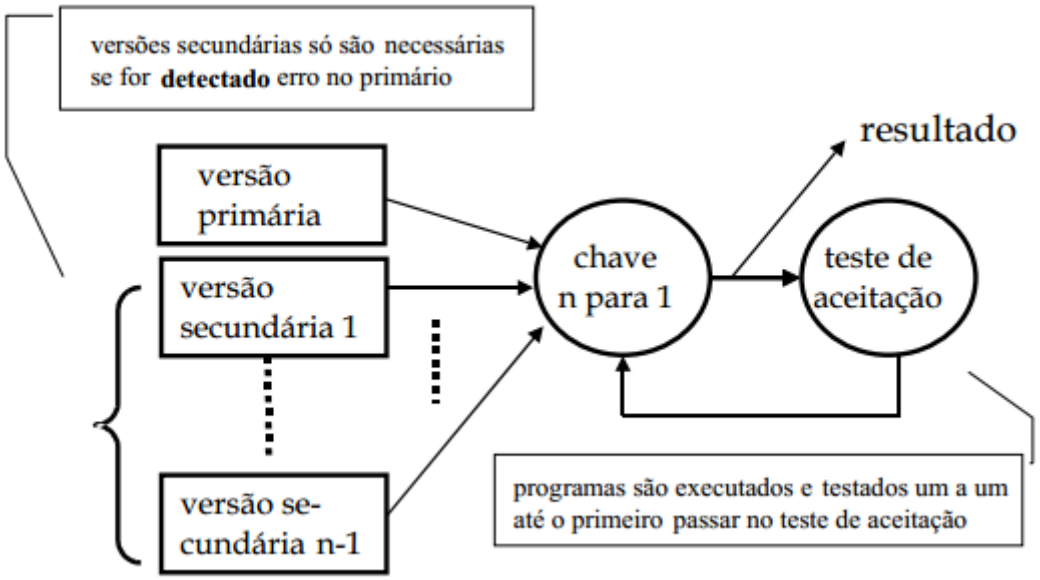


Figura 2.7: Blocos de recuperação. Retirado de [28]

- Verificação de consistência. É uma extensão da programação n-versões. Nessa técnica n equipes implementam separadamente o *software* a partir de uma única especificação, porém, as equipes também desenvolvem um módulo para verificar a saída do seu próprio sistema em tempo de execução. Os *softwares* desenvolvidos pelas equipes são executados paralelamente e as saídas dos módulos são submetidas à verificação. A saída selecionada precisa passar pelo seu próprio teste de verificação e o resultado é definido pelo primeiro módulo que passar ou por um sistema de votação. Com a primeira abordagem, ganha-se em desempenho (rapidez), com a segunda, o benefício é a segurança.

2.4.3 Técnicas Utilizando Redundância de Dados

A redundância de dados consiste em ter dados redundantes na execução do programa. Pode ser provida por meio de códigos de correção de erros e tem sido utilizada exaustivamente em memórias e processadores [27]. Nesse tipo de redundância, bits ou sinais extras podem ser armazenados ou transmitidos junto ao dado para permitir detecção ou correção de erros, e o mascaramento de falhas [31]. Um exemplo clássico são os códigos de paridade, em que para cada n bits são armazenados $n + 1$ bits. O bit extra indica se o número de bits com valor 1 é par (ou ímpar, dependendo do tipo de paridade usada) nos n bits restantes [41].

Em sistemas embarcados, essa técnica é usada com frequência por meio da replicação das informações. Por exemplo, dados de um programa podem ser salvos em diferentes locais, seja memória, arquivos em disco ou quaisquer outros dispositivos de armazenamento. Desse modo, falhas que afetam o valor da informação não necessariamente causam erros no sistema, uma vez que a informação correta ainda pode ser obtida nas outras cópias [42].

Verificação de Paridade

A verificação de paridade é um método amplamente usado na detecção de erros em sistemas que se comunicam por troca de mensagens. Em esquemas simplificados, um único bit de paridade é adicionado a um conjunto de d bits de dados, conforme mostra a Figura 2.8. Kurose [41] explica que em sistemas de paridade par, simplesmente se inclui um bit adicional e escolhe-se o valor desse bit de modo que o número total de "1s" nos $d+1$ bits (a informação original mais um bit de paridade) seja par. Em esquemas de paridade ímpar, o valor do bit de paridade é escolhido de modo que haja um número ímpar de "1s".

Figura 2.8: Verificação usando um único bit de paridade. Retirado de [41].

Para verificar a exatidão dos dados em esquemas de um único bit de paridade, é necessário apenas contar quantos "1s" existem nos $d+1$ bits recebidos. Se ao utilizar-se um esquema de paridade par, for encontrado um número ímpar de bits de valor 1, se saberá que ocorreu um número ímpar de erros de bit. Entretanto, cabe observar que este esquema é ineficaz para um número par de erros de bit, pois em casos como este, os erros não serão detectados. Por esse motivo, essa técnica costuma ser implementada em um esquema bidimensional de bits de paridade. A Figura 2.9 mostra uma generalização bidimensional do esquema de paridade de bit único, no qual o bit com valor 1 na posição (2, 2) está corrompido e mudou para o valor 0.

Em esquemas de paridade bidimensional, os d bits de dados são divididos em i linhas e j colunas, e o valor de paridade é calculado para cada linha e para cada coluna. Os $i+j+1$ bits de paridade resultantes compreendem aos bits de detecção de erros. Kurose [41] explica que com a paridade bidimensional, tanto a paridade da linha quanto a da coluna que contém o

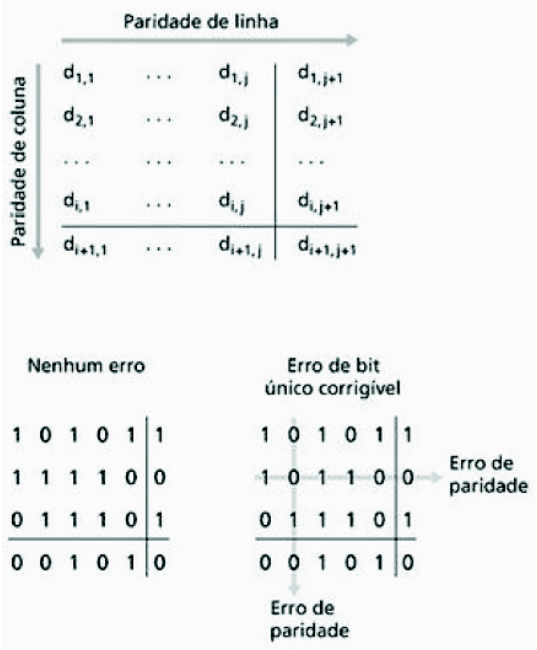


Figura 2.9: Verificação usando paridade bidimensional. Retirado de [41].

bit modificado estarão com erro. Pode-se então não somente detectar que um erro de um bit único ocorreu, mas também usar os índices da linha e da coluna com erros de paridade para identificar o bit que foi corrompido. Vale ressaltar que os erros podem alterar os próprios bits de paridade. Se um erro único nos bits de paridade ocorrer, é possível detectá-lo e corrigi-lo. Todavia, quaisquer combinação de dois erros podem, por este esquema, somente ser detectados.

Soma de Verificação

Em técnicas de soma de verificação, os d bits de dados na Figura 2.8 são tratados como uma sequência de números inteiros de k bits. Um método simples de soma de verificação é somar esses inteiros de k bits e usar o total resultante como bits de detecção de erros. Exemplo de adoção dessa técnica é soma de verificação dos pacotes da Internet, em que *bytes* de dados são tratados como inteiros de 16 bits e somados. O complemento dessa soma forma a soma de verificação do pacote, que é colocado no cabeçalho do segmento. Ao ser recebido, o receptor verifica a soma de verificação calculando os complementos de 1 da soma dos dados recebidos (inclusive a soma de verificação) e verifica se o resultado contém somente bits de valor 1. Se qualquer um dos bits for 0, isso indicará um erro.

Segundo Kurose [41], os métodos de soma de verificação exigem pouca sobrecarga na informação. Entretanto, oferecem proteção relativamente baixa contra erros em comparação com os esquemas de verificação de redundância cíclica.

Verificação de Redundância Cíclica (CRC)

Uma das técnicas amplamente usadas para detecção de erros são os códigos de verificação de redundância cíclica (CRC - *Cyclic Redundancy Check*). Essa técnica é um tipo de função *hash*, que gera um valor expresso em poucos bits em função de um bloco maior de dados. Kurose [41] explica que antes de enviar o dado D de d bits, um código de CRC R com r bits deve ser gerado e anexado ao dado, conforme a Figura 2.10.

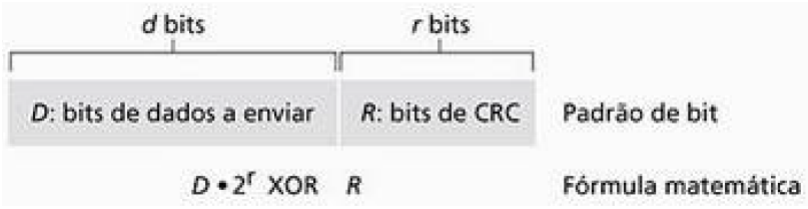


Figura 2.10: Formato dos dados usando código CRC. Retirado de [41].

O receptor e o emissor devem conhecer um padrão de bits G, denominado gerador, com $r + 1$ bits de comprimento, sendo o bit mais significativo igual a 1. Todos os cálculos de CRC são feitos por aritmética de módulo 2 sem "vai 1" nas adições e sem "empresta 1" nas subtrações. Isso significa que a adição e a subtração são idênticas e ambas são equivalentes à operação "ou exclusivo" (XOR) nos bits dos operandos [41]. Por exemplo:

$$1011 \text{ XOR } 0101 = 1110$$

$$1001 \text{ XOR } 1101 = 0100$$

O cálculo do CRC é feito por meio da fórmula abaixo:

$$R = resto\frac{D \cdot 2^r}{G} \tag{2.1}$$

Para se verificar erros com CRC, o receptor divide os $d + r$ bits recebidos por G. Se o resto é diferente de zero, o receptor interpreta os dados como incorretos; caso contrário, os dados são aceitos como corretos [41]. A Figura 2.11 ilustra esses cálculos para o caso de D = 101110, $d = 6$ e $G = 1001$, $r = 3$. Neste exemplo, os bits transmitidos são 101110011.

Padrões internacionais foram definidos para geradores G de 8, 12, 16 e 32 bits. Os mais comuns são:

- CRC 32 bits: 10000010011000001000111011011011 1
- CRC 16 bits: 11000000000000101
- CRC 12 bits: 1000000001011
- CRC 8 bits: 100000111
- CRC 1 bit: 1

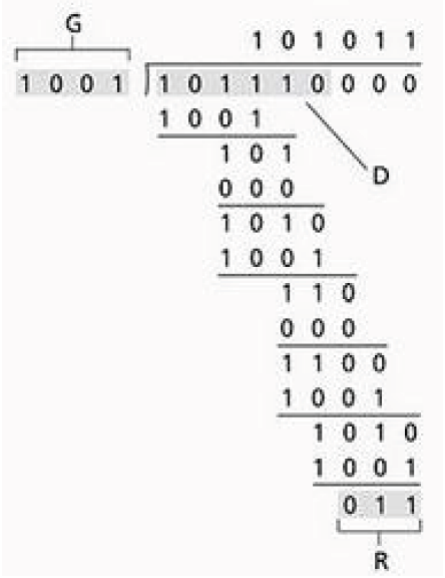


Figura 2.11: Exemplo de cálculo CRC. Retirado de [41].

2.4.4 Técnicas Utilizando Redundância de Processamento

A redundância de processamento (ou redundância temporal) repete a execução do código no tempo. Essa técnica evita custos adicionais relativos ao *hardware*, mas aumenta o tempo necessário para realizar um processamento. É usada em sistemas no qual o tempo não é crítico ou no qual o processador trabalha com ociosidade [27]. Essa técnica é geralmente aplicada em:

- Detecção de falhas transientes: repetindo-se a execução do código, resultados diferentes são uma indicação de uma falha transitória. Essa estratégia não é adequada para falhas permanentes, pois nestas as reexecuções do código apresentarão resultados semelhantes.
- Detecção de falhas permanentes: repete-se o processamento com os dados codificados e decodifica-se o resultado antes da comparação com o resultado anterior. Considerando, por exemplo, um barramento com uma linha travada em zero, deve-se realizar duas transferências: a primeira com o dado original e a segunda com o dado invertido (complemento). Essa estratégia possibilita a detecção de falhas permanentes.

2.5 Injeção de Falhas

Uma etapa importante no desenvolvimento de sistemas tolerantes a falhas é a validação das propriedades de segurança. Entre as possíveis abordagens utilizadas nesse processo, tais como a comprovação ou a modelagem analítica, cuja aplicabilidade e precisão são significativamente limitadas nos sistemas tolerantes a falhas complexos [29], a injeção de falhas é considerada uma abordagem poderosa, pois permite avaliar o comportamento dos sistemas na presença de falhas. Além disso, fornece parâmetros úteis para a avaliação das técnicas de tolerância a falhas (como cobertura e latência) [43] e é recomendada pela maioria dos

padrões de segurança, tais como o padrão ISO 26262 para segurança automotiva [44] e o padrão NASA 8719.13B [45].

Segundo Arlat *et al.* [46], a injeção de falhas é uma técnica de validação da segurança dos sistemas tolerantes a falhas que consiste na realização de experimentos controlados, nos quais observa-se o comportamento dos sistemas na presença de falhas geradas propositalmente (injetadas). As injeções de falhas podem ser feitas via *hardware*, ou via *software*. Entretanto, existem atualmente outros tipos de injeção de falhas, como por exemplo, as injeções de falhas baseadas em simulação. Arlat explica que esse tipo de injeção de falhas se restringe aos modelos de alto nível, na maioria das vezes, modelos VHDL (*VHSIC Hardware Description Language*) - Linguagem de descrição de *hardware* de circuitos de alta velocidade. Esta técnica permite avaliar a segurança quando apenas um projeto do hardware do sistema está disponível.

Embora as técnicas de injeção de falhas sejam reconhecidas há muito tempo como necessárias para validar a segurança de um sistema, é importante observar o conjunto das falhas injetadas para garantir a representatividade das falhas, uma vez que o objetivo é reproduzir as falhas que realmente existem na prática para obter uma avaliação realista. Se as falhas inseridas pelo injetor não ocorrem na prática, ou seja, não são representativas, então é arriscado avaliar a eficiência do sistema usando esse injetor de falhas. Um meio de se garantir representatividade nas falhas é selecionando o tipo de falha a ser injetada ("o que injetar?") e a localização ("onde injetar?") [47]. No entanto, na maioria das vezes, a localização da falha, o tipo de falha e o tempo no qual ela é injetada são selecionados aleatoriamente sem qualquer estudo prévio. Segundo Arlat *et al.* [46] e Kanawati *et al.* [43], cerca de 10% a 60% das falhas injetadas não produzem erros, ou pior, as falhas injetadas não reproduzem situações reais de falha no sistema. Logo, os testes de injeção de falhas nesses casos não apresentam resultados confiáveis.

2.5.1 Injeção de Falhas por Hardware

Devido à necessidade de se testar os equipamentos eletrônicos e os sistemas computacionais em situações de falhas, como as causadas pelo fenômeno *bit flip*, surgiu a necessidade de desenvolvimento de estratégias para ensaios acelerados. Como forma de testar os dispositivos semicondutores, equipamentos geradores de cargas muito mais frequentes do que as naturais foram desenvolvidos. Secall [17] explica que as técnicas de injeção de falhas por *hardware* pressupõem a interação física entre o mecanismo injetor de falhas e o sistema computacional sob teste.

Hsueh *et al.* [15] classifica a injeção de falhas por hardware em duas categorias: com contato e sem contato. Na categoria sem contato estão a interferência eletromagnética e o bombardeamento com radiação ionizante, em que dispositivos semicondutores são bombardeados com partículas atômicas, provocando o *bit flip* (perturbação do *hardware* com parâmetros do ambiente). Outra possibilidade é a injeção de quedas de tensão na fonte de energia do *hardware*, gerando distúrbios de alimentação. Nas técnicas com contato, a mais utilizada é a técnica chamada de *pin-level*, que atua abrindo, curto-circuitando ou injetando sinais interferentes diretamente (por contato) em cada pino do dispositivo sob teste, por meio de pinças de injeção ou posicionamento do circuito sobre soquetes conectados a um injetor de falhas [29, 47].

As técnicas de injeção de falhas com contato e sem contato têm a vantagem de injetar falhas reais no *hardware*, mas podem danificar o componente sob teste. Além disso, requerem o uso de *hardware* especial e as ferramentas de injeção de falhas são dedicadas a determinado sistema [48]. Por esses motivos, as técnicas de injeção de falhas por *software* têm sido mais utilizadas [43].

2.5.2 Injeção de Falhas por Software

As técnicas que emulam falhas por *software* podem ser classificadas conforme o tipo de falha emulada (falha de *hardware* ou falha de *software*). A maioria emula falhas de hardware, modificando os estados do sistema sob teste, fazendo-o agir como se falhas de *hardware* estivessem presentes. Mais recentemente, surgiram também as técnicas de injeção para emular falhas de *software*. Nestas, o programa injetor de falhas introduz pequenas alterações no código original do programa, inserindo *bugs* no sistema.

Técnicas que Emulam Falhas de *Hardware*

Esse método de injeção de falhas, denominado SWIFI (*Software Implemented Fault Injection*) é bastante versátil e tem a capacidade de alterar o valor de registradores e endereços de memória, sendo menos dispendioso em termos de tempo e complexidade em relação as técnicas de *hardware*. Além disso, o sistema em estudo não corre o risco de ser danificado durante a injeção [43].

As técnicas de injeção de falhas por *software*, na maioria das vezes, utilizam a mutação ou corrupção de memória. Para simular essas falhas por intermédio de *software*, pode-se utilizar mecanismos de injeção de falhas na própria aplicação sob testes, mas o mais comum é o uso de *softwares* externos [43]. Alguns exemplos são: NFTAPE [49], Xception [50], GOOFI [51] e Ferrari [43].

Técnicas que Emulam Falhas de *Software*

Nesse tipo de injeção de falhas, denominado SFI (*Software Fault Injection*), um programa injetor emula as falhas no *software* (*bugs*) mediante a introdução de pequenas alterações no código original do programa, criando diferentes versões (cada versão com uma falha de *software* injetada). A maioria dessas técnicas simulam falhas indiretamente, uma vez que apenas emulam os possíveis efeitos das falhas de *software* ao invés de injetar falhas reais. As falhas injetadas podem ser classificadas de acordo com os erros que serão gerados, isso é, erros de dados ou erros de interface. Erros de dados são dados incorretos injetados nos programas em execução, causando desvios do estado correto do sistema. Essa é uma forma de inserir falhas indiretamente, pois o que está sendo inserido não é uma falha em si, mas um possível efeito da falha. A injeção de erros de interface é outra forma de injeção. Nesse caso, o erro é especificamente injetado na interface entre os módulos, geralmente nos parâmetros de funções e APIs (*Application Programming Interface*). Os erros injetados podem ter muitas formas, desde simples alterações de dados até informações válidas, mas com semântica incorreta [47].

Uma técnica SFI amplamente utilizada é a G-SWFIT (*Generic Software Fault Injection*

Technique), que utiliza um conjunto de falhas derivadas dos tipos mais comuns de falhas em *software*. Essa técnica injeta modificações no código, baseando-se em estatísticas de campo sobre a frequência dos tipos de falhas. Os estudos apontaram que a maioria das falhas de *software* achadas em campo pertence a um pequeno conjunto de falhas, descritos na Tabela 2.1, e que outros tipos são raramente encontrados. Essa técnica foca somente nos tipos das falhas e despreza as localizações, inserindo falhas em todos os módulos e em todas as rotinas, sem considerar suas complexidades [47]. Essa é uma forte limitação, pois injetar falhas em todos os locais de código pode se tornar impraticável em um *software* muito grande e complexo.

Tipo	**Descrição**
MFC	Falta da chamada de uma função
MVIV	Falta de inicialização de uma variável usando um valor
MVAV	Falta de atribuição de uma variável usando um valor
MVAE	Falta de atribuição de uma variável usando uma expressão
MIA	Falta da condição "*if*" em volta das instruções
MIFS	Falta da condição "*if*" além das instruções
MIEB	Falta da condição "*if*" e "*else*" antes das instruções
MLC	Falta de "and" ou "or" na cláusula da condição de desvio
MLPA	Falta de pequena e localizada parte de um algoritmo
WVAV	Valor incorreto atribuído a uma variável
WPFV	Valor incorreto de um parâmetro na chamada de função
WAEP	Expressão aritmética incorreta em um parâmetro na chamada de função

Tabela 2.1: Tipos de falhas mais comuns. Retirada de [47].

2.6 Trabalhos Relacionados

Kumar *et al.* [52] introduziram um modelo de *software* baseado em simulação para análise de segurança de um sistema. O modelo representou um programa de aplicação, decomposto em um grafo, constituído por um conjunto de nós, um conjunto de vértices e um mapeamento dos nós na memória. O modelo simulou a execução do programa durante as injeções de falhas nas regiões de memória. Os autores afirmaram que todas as falhas relacionadas ao *hardware* podem ser mapeadas como falhas de memória, mas não provaram isso formalmente. Além disso, os autores não explicam no trabalho quais foram os tipos de falhas injetadas e como a injeção foi realizada.

Delong [53] conduziu um estudo similar para estimar os parâmetros de segurança de um sistema computacional. O sistema de *hardware* foi modelado usando VHDL e as falhas foram injetadas em todos os registradores do computador, simulando os efeitos de falhas permanentes do tipo *stuck-at-0* e *stuck-at-1*. Esse trabalho não cobriu o fenômeno *bit-flip* ou fenômenos de falhas de *hardware* transientes, e não descreveu as localizações e as quantidades

de falhas injetadas nos testes.

Amendola *et al.* [54] conduziram um estudo para investigar o comportamento da falha de um microprocessador. Eles estudaram as falhas localizadas em memórias, em barramentos e em registradores internos do processador. Para isso, falhas do tipo *bit-flip* foram introduzidas no modelo VHDL dos barramentos, memórias e CPU. Esse trabalho demonstrou se o sistema poderia tolerar falhas, porém, mais uma vez, devido a falta de localização das falhas injetadas, não forneceu informações das técnicas que mais contribuíram para a tolerância a falhas. Além disso, outra limitação foi que o trabalho não explicou como as falhas foram distribuídas pelo sistema, ou seja, em quais regiões dos componentes e quais dados necessariamente foram alterados. Correlato ao trabalho de Amendola, D. Gil. *et al.* [55] realizou um experimento de injeção de falhas para analisar o que classificou como "síndrome de erros" de um microcomputador. O modelo também era escrito em VHDL, e o seu experimento injetou falhas de diferentes tipos em diferentes locais da memória do computador.

Secall [17] fez uma avaliação comparativa do impacto do emprego das técnicas de programação defensiva na segurança de sistemas críticos. Para simular falhas semelhantes ao ambiente real, Secall desenvolveu técnicas de injeção de falhas baseadas na injeção de falhas por hardware, injetando falhas por interferências eletromagnéticas, utilizando-se radiofrequência irradiada. Ao final, Secall fez uma avaliação quantitativa da eficácia de algumas técnicas de programação defensiva na capacidade de tolerância a falhas em aplicações críticas. O estudo de caso utilizado no trabalho foi um ambiente metroferroviário.

Poucos são os trabalhos na literatura que abordam a injeção de falhas em sistemas embarcados de pequeno e médio porte. Nenhum dos trabalhos vistos aqui destinaram-se a este nicho, e na literatura, a grande maioria são correspondentes a outras plataformas. Além disso, os trabalhos analisados usaram injetores de falhas que simularam os fenômenos *bit-flip* ou as falhas permanentes, como as de *stuck-at-0* e *stuck-at-1* ("travado em 0 e travado em 1"), mas nenhum forneceu as localizações das falhas injetadas, de forma a trazer uma análise detalhada das regiões atingidas, tais como os impactos disso.

Capítulo 3

Metodologia

Neste trabalho utilizou-se como ponto de partida o sistema embarcado (*hardware* e *firmware*) de uma estação meteorológica usada para coleta de parâmetros climáticos em lavouras. O *firmware* funciona da seguinte maneira: em intervalos de tempo predefinidos, o programa lê todos os sensores da estação meteorológica e escreve o conjunto de leituras, mais um código CRC, em um arquivo binário na memória *flash*, como mostra o fluxograma da Figura 3.1. Além dos códigos CRC, que permitem a detecção de erros no arquivo gravado, o microcontrolador possui um temporizador *watchdog* que reinicia o dispositivo quando o mesmo fica travado deixando de recarregar o temporizador. Nenhuma outra técnica, além dos códigos CRC e do temporizador *watchdog*, foi utilizada para aumentar a confiabilidade do sistema. O *hardware* da estação meteorológica foi baseado na plataforma de prototipagem rápida para microcontroladores mbed [56], modelo NXP LPC1768. O módulo NXP LPC1768 é composto por um núcleo ARM Cortex-M3 de 32 bits e 96 MHz de *clock*, memória *flash* com capacidade de 512 kB, memória SRAM de 64 kB (32 kB SRAM acessível pela CPU e pela controladora DMA - *Direct Memory Access* - destinados ao programa de usuário, e dois blocos adicionais SRAM de 16 kB para uso dos periféricos internos do microcontrolador), além de um conjunto de interfaces, incluindo *built-in Ethernet*, dispositivo e *host* USB (*Universal Serial Bus*), CAN (*Controller Area Network*), SPI (*Serial Peripheral Interface*), I2C (*Inter-Integrated Circuit*), ADC (*Analog-to-Digital Converter*), DAC (*Digital-to-Analog Converter*), PWM (*Pulse-Width Modulation*) e outras interfaces de entrada e saída.

Com base no *firmware* original da estação meteorológica, um outro foi desenvolvido utilizando tolerância a falhas por meio de redundância de dados e redundância de processamento, as duas técnicas mais viáveis em sistemas embarcados de baixo custo. Uma vez que este *firmware* foi desenvolvido, foi preciso definir a ferramenta de injeção de falhas e testes para avaliar o desempenho das técnicas implementadas. Inicialmente, estudou-se a possibilidade do uso de ferramentas de injeções de falhas já existentes, no entanto, a maioria dos injetores de falhas são específicos para outras plataformas (*desktop* e servidores) e possui algumas limitações, como por exemplo, a falta de informação sobre a localidade das falhas injetadas. Devido a esses motivos, verificou-se a necessidade de criar um esquema específico de testes e um injetor de falhas para a plataforma mbed. Para isso, três programas precisaram ser desenvolvidos: uma biblioteca de injeção de falhas, um *firmware* monitor e um programa monitor de testes, explicados detalhadamente na Seção 3.1. Técnicas de injeção de falhas por *hardware* não foram utilizadas, pois além de exigirem *hardware* adicional (alguns não

viáveis a este trabalho, pela não disponibilidade e custo elevado) oferecem risco de danificar o equipamento sob teste. Além disso, os trabalhos analisados incentivam o uso de técnicas de injeção de falhas por *software*, já que as falhas de *hardware* podem ser facilmente emuladas. Mais ainda, as ferramentas de injeção de falhas por *software* permitem injeção de falhas e relatórios mais precisos, uma vez que pode-se escolher o que e onde inserir, isto é, os tipos e as localizações das falhas.

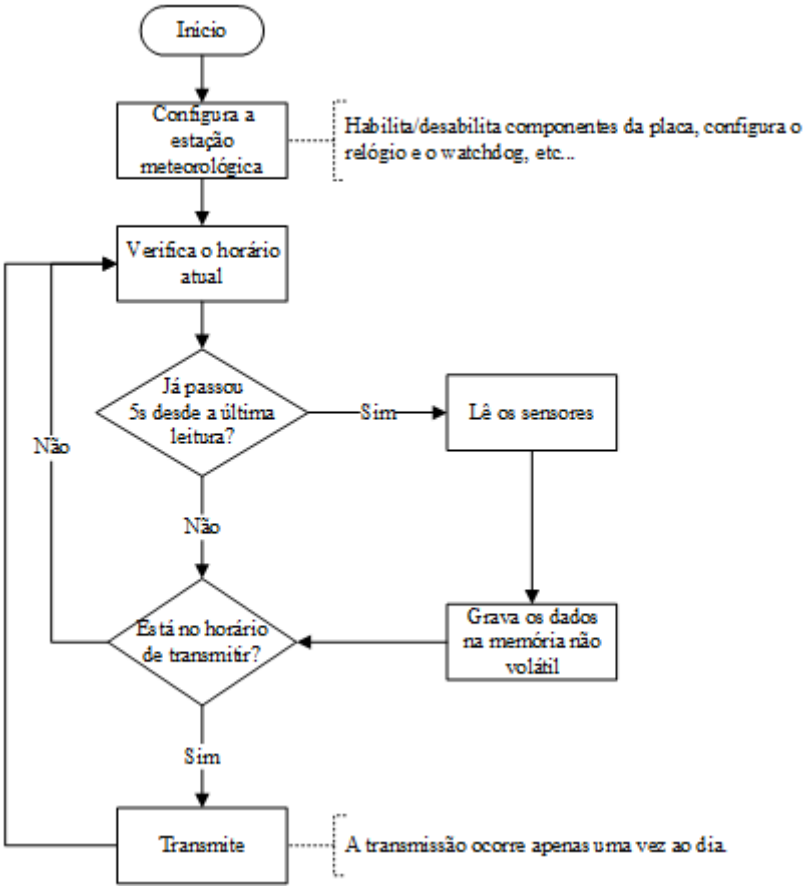


Figura 3.1: Fluxograma da estação meteorológica original. A cada 5 segundos, o programa lê todos os sensores da estação meteorológica e grava os valores das leituras em um arquivo binário na memória *flash*. Uma vez ao dia, o arquivo com os dados das leituras é enviado ao servidor por meio de comunicação GPRS ou satélite. Cabe observar que o intervalo entre leituras normalmente é de alguns minutos e foi reduzido para 5 segundos para acelerar o teste.

3.1 Sistema de Injeção de Falhas

O sistema de injeção de falhas e testes desenvolvido neste trabalho é composto por três partes: a biblioteca de injeção de falhas, o *firmware* monitor e o programa monitor de testes. A biblioteca de injeção de falhas, denominada *FaultInjector*, atua inserindo falhas em regiões de memória do microcontrolador para alterar os dados e causar erros no programa sob teste, simulando os efeitos do fenômeno *bit-flip*. O *firmware* monitor, executado em outro módulo mbed (mbed monitor), serve para emular os sinais (pulsos) de pluviometria e anemometria,

e para reiniciar o *firmware* da estação meteorológica quando necessário, isto é, quando falhas são detectadas pelo programa monitor. O programa monitor, executado em um computador pessoal (IBM-PC), monitora os resultados dos testes em tempo de execução, analisando o arquivo de dados gravado pela estação. Quando uma falha é encontrada, este programa emite um sinal para o *firmware* monitor por intermédio da porta serial (virtual), que reinicia a estação meteorológica, conforme mostra a Figura 3.2. Ao final da execução de um número predefinido de testes, um relatório detalhado é gravado no computador mostrando a quantidade de falhas ocorridas juntamente com os tipos das falhas. Os módulos mbed foram conectados ao computador via USB e a comunicação de entrada e saída com o mbed foi estabelecida pela aplicação de terminal TeraTerm, versão 4.8.3. O TeraTerm [57] é uma aplicação de terminal e módulo de SSH (*Secure Shell*), executada em um PC hospedeiro. Esse tipo de aplicação fornece um meio para o microcontrolador estabelecer comunicação de entrada e saída com o computador. Cada programa registra seus eventos relevantes em um arquivo de texto, chamado de *log*, salvo na sistema local de arquivos do dispositivo. Todos os programas foram implementados em linguagem C/C++.

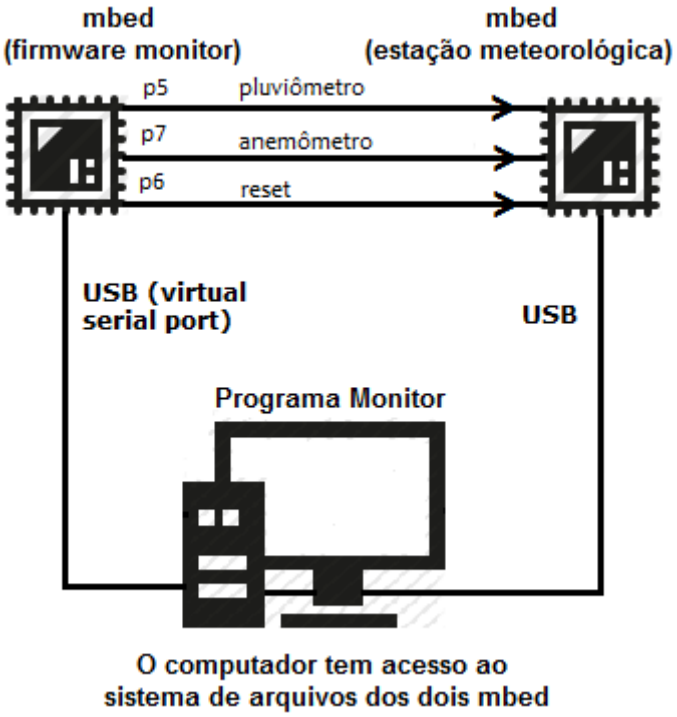


Figura 3.2: Sistema de teste. Os módulos mbed (*firmware* monitor e estação meteorológica) são conectados ao computador via USB. O computador executa o programa de teste que analisa os arquivos de dados gravados pela estação meteorológica em seu sistema de arquivos. Enquanto isso, o *firmware* monitor (à esquerda) simula os pulsos do pluviômetro e anemômetro por meio dos pinos p5 e p7, respectivamente. O pino p6 é conectado à estação meteorológica, a fim de reiniciar o *firmware* da mesma quando uma falha é encontrada.

3.1.1 Biblioteca de Injeção de Falhas *FaultInjection*

A biblioteca de injeção de falhas *FaultInjection* [42] tem como objetivo inserir, em tempo de execução, um número predefinido de falhas em quaisquer regiões de memória do módulo mbed. As injeções de falhas são realizadas durante a execução do programa alvo, em momentos específicos ou em intervalos de tempos aleatórios, por meio de interrupções. Nos testes de injeção de falhas efetuados, as injeções de falhas ocorreram em intervalos de tempo aleatórios, para simular uma situação mais real da ocorrência de falhas no sistema.

No *log* do sistema injetor de falhas, é possível visualizar todos os endereços que foram atingidos pela injeção de falhas, e também os dados originais e modificados. Isso permite verificar a localização e mediante essa informação pode-se analisar quais regiões de memória são mais vulneráveis quando seus dados são modificados pelo injetor.

Além de falhas por alterações dos dados, o sistema permite a injeção de mais dois tipos de falhas: a que trava o microcontrolador (por meio de uma instrução que faz acesso a um endereço de memória inválido) e a que prende a execução do programa em um laço infinito. A primeira é útil para testar a capacidade de restauração do programa em situações de "travamento" total. A segunda permite testar as técnicas de recuperação por *watchdog*. Neste caso, é recomendável que diversas instruções de recarga do *watchdog* sejam dispostas em diferentes trechos do código, ao invés de ser uma única instrução colocada em uma função executada periodicamente por interrupção. Essa estratégia garante que o programa reinicie se o mesmo ficar preso em um laço infinito, devido a alguma falha ou *bug* do software.

A Figura 3.3 mostra o mapa de memória da família mbed LPC17xx. O modelo LPC1768, usado neste trabalho, possui 3 regiões de memória, sendo 32 kB reservados para o programa de usuário e duas regiões de 16 kB destinadas ao uso interno da plataforma de prototipagem rápida mbed. Os dados na memória *flash* também são endereçáveis. Cabe observar que por não haver memória virtual, os acessos aos endereços de memória dentro do programa correspondem aos endereços físicos. A injeção de falhas é realizada pelo método *injectFaults*, conforme mostra o código do Quadro 2.

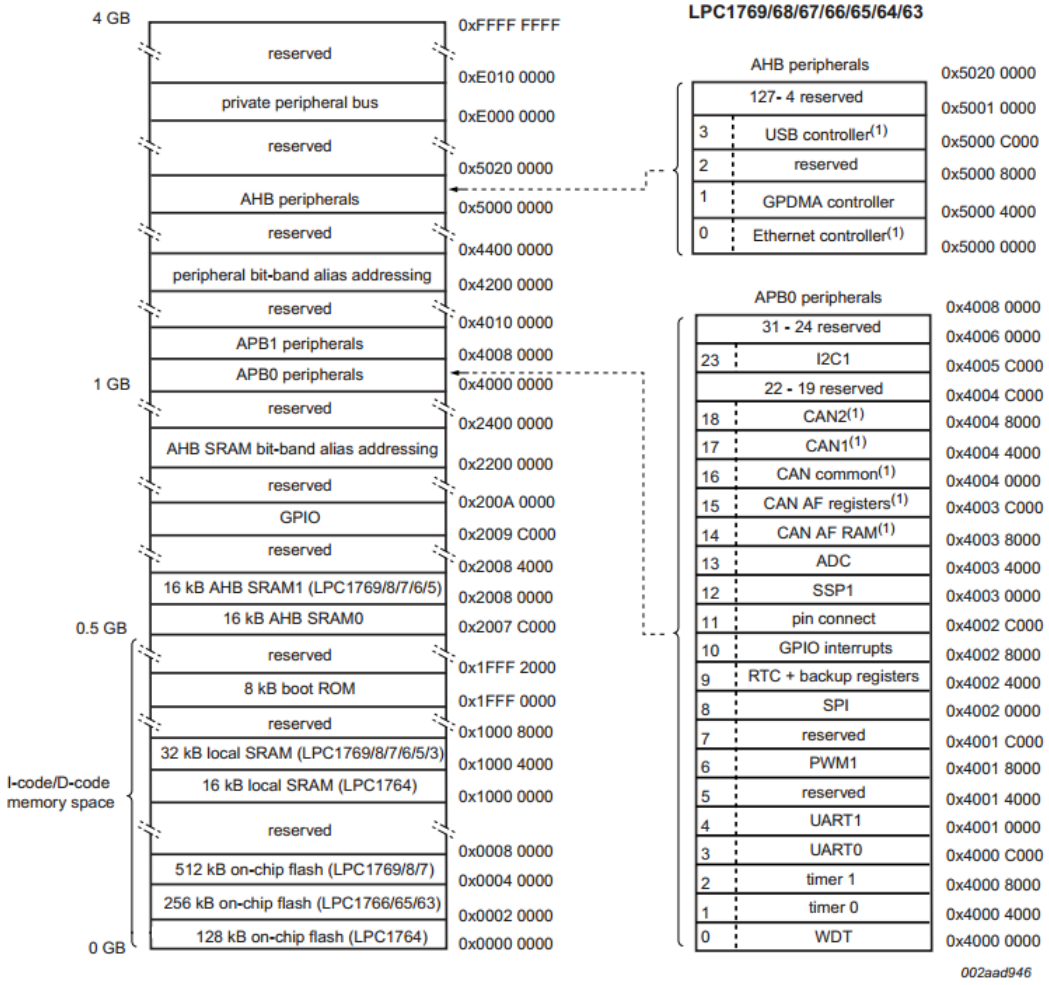


Figura 3.3: Mapa de memória da família mbed LPC17xx. O mbed NXP LPC1768 usado neste trabalho possui 3 regiões de memória. Os 32 kB da região local SRAM ($0x10000000$ até $0x10008000$) são reservados para o *firmware* do usuário. As regiões SRAM0 ($0x2007C000$ até $0x2008000$) e SRAM1 ($0x20080000$ até $0x20084000$) contém 16 kB cada, e são destinadas ao uso interno da plataforma de prototipagem rápida mbed.

```
1  /*************************************************************
2  Arquivo: FaultInjector.h
3  *************************************************************/
4  static const int DEFAULT_CHANGED_BYTES = 1;
5  static const int DEFAULT_CHANGED_BITS = 0; // (0..8) 0 = Aleatório.
6
7  /*************************************************************
8  Arquivo: FaultInjector.cpp
9  *************************************************************/
10 void FaultsInjector::injectFaults(unsigned int startAddr, unsigned int
       endAddr, unsigned int changedBytes, unsigned int changedBits) {
11
12     int count, temp, position;
13     unsigned char *addr;
14     char buffer[32];
15     bool drawn[8];
16
17     if (changedBytes < 1 || changedBytes > memorySize)
18         changedBytes = DEFAULT_CHANGED_BYTES;
19
20     if (changedBits > 8)
21         changedBits = DEFAULT_CHANGED_BITS;
22
23     temp = changedBits;
24
25     FILE *fp = fopen("/local/faults.log", "a");
26
27     if (fp != NULL) {
28         time_t seconds = time(NULL);
29         strftime(buffer, 32, "%d/%m/%Y %H:%M:%S", localtime(&seconds));
30
31         fprintf(fp, "%s\n", buffer);
32         fprintf(fp, "Start address: %d\n", startAddr);
33         fprintf(fp, "End address:   %d\n", endAddr);
34         fprintf(fp, "Bytes changed:  %d\n", changedBytes);
35         fprintf(fp, "Bits changed:   %d\n\n", changedBits);
36
37         for (int i = 0; i < changedBytes; i++)
38         {
39             char value[9];
40             addr = (unsigned char *) getByteMemory(startAddr, endAddr);
41             itoa(*addr, value, 2);
42
43             fprintf(fp, "%d\n", i + 1);
44             fprintf(fp, "Address (byte): %p\n", addr);
45             fprintf(fp, "Correct value:  %s\n", value);
```

```
46              if (changedBits == 0)
47                  temp = getRandomIntPositive(1, 8);
48
49              for (count = 0; count < 8; count++)
50                  drawn[count] = false;
51
52              count = 0;
53
54              while (count < temp) {
55                  position = getRandomIntPositive(0, 7);
56                  if (drawn[position] != true)
57                  {
58                      drawn[position] = true;
59                      count++;
60                      (*addr) ^= (1 << position);
61                  }
62              }
63              itoa(*addr, value, 2);
64              fprintf(fp, "Changed value: %s\n\n", value);
65          }
66          fclose(fp);
67      }
68  }
```

Quadro 2: Método injetor de falhas. Os parâmetros *startAddr* e *endAddr* determinam os limites da região de memória onde as falhas serão injetadas; *changedBytes* corresponde ao número de endereços que serão alterados na região. Para cada endereço, *changedBits* serão alterados. Caso o valor de algum parâmetro seja inválido, o valor é substituído pelo padrão.

O método *injectFaults* altera o conteúdo de alguns locais da memória e registra as alterações em um arquivo de *log*. Os parâmetros desse método são explicados a seguir:

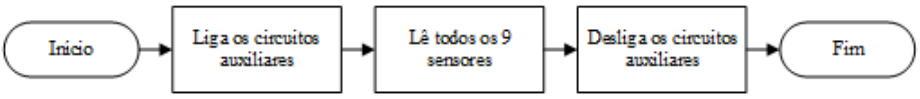
- *startAddr*. Endereço inicial da memória de dados. Este valor deve ser definido por um número inteiro entre 0 (que indica o endereço inicial da primeira região de memória) e o tamanho total da memória física (64 kB) - 1.
- *endAddr*. Endereço final da memória de dados. Este valor deve ser definido por um número inteiro entre *startAddr* (endereço inicial) e o tamanho total da memória (64 kB - 1).

Os dois parâmetros acima são utilizados para definir o início e o fim da região de memória onde as falhas são inseridas. Por padrão, é considerado o espaço total da memória (64 kB) do módulo mbed ($startAddr = 0$, 64 kB - 1).

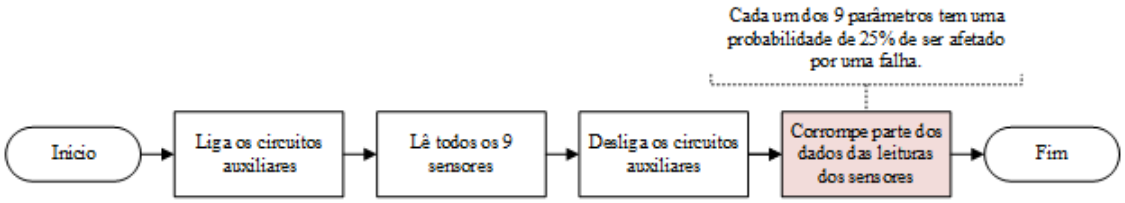
- *changedBytes*. Número inteiro que define o número de endereços (posições de memória) alterados em cada injeção de falhas. Este valor deve ser entre 1 (valor padrão) e $endAddr - startAddr$ (todos os *bytes* da região são alterados). Se um número inválido é inserido, o programa assume o valor padrão ($changedBytes = 1$).

- *changedBits*. Número de bits que serão alterados em cada *byte*. Este valor deve ser um número inteiro entre 0 (padrão) e 8. O valor 1 altera um único bit aleatório e o valor 8 inverte todos os bits do *byte* de dados (por exemplo: os bits 01001101 serão alterados para 10110010). O valor 0 altera n bits, no qual n é um número aleatório entre 1 e 8.

Além das injeções de falhas em regiões de memória do microcontrolador, alguns dos testes realizados neste trabalho emularam a presença de falhas em leituras dos sensores. Essas falhas em um ambiente real ocorrem principalmente devido aos ruídos causados por interferências externas do ambiente no momento da leitura de um sensor, gerando valores incorretos. Logo, para simular esses efeitos, o método *readSensors*, responsável por ler os 9 sensores da estação meteorológica, foi levemente alterado nos *firmwares* com e sem tolerância a falhas, de forma que, após a leitura de cada sensor, o método *injectFaultAtFloat* da biblioteca *FaultInjector* seja executado. Este método recebe dois parâmetros, o primeiro corresponde ao endereço da variável do tipo *float* a sofrer alteração, isto é, a receber uma falha, o segundo, a probabilidade de isso acontecer, sendo esta probabilidade um número flutuante na escala de 0,0 a 1,0, no qual o valor 0 não afeta o endereço especificado e 1 certamente o altera. A alteração do dado consiste em substituir o valor original por qualquer valor possível de uma variável *float*. Na Figura 3.4a é mostrado resumidamente o processo de leitura dos sensores na estação meteorológica sem tolerância a falhas e na Figura 3.4b é exibido um fluxograma do mesmo *firmware* recebendo a injeção de falhas nas leituras dos sensores.



(a) Processo de leitura dos sensores na estação meteorológica sem tolerância a falhas.



(b) Processo de leitura dos sensores na estação meteorológica com a injeção de falhas. Após a leitura dos sensores, os dados sofrem alterações de acordo com uma porcentagem definida de injeção de falhas.

Figura 3.4: Processo de leitura dos sensores na estação meteorológica sem tolerância a falhas

3.1.2 *Firmware* Monitor

O *firmware* monitor, executado em outro mbed NXP LPC1768 (mbed monitor), é responsável por emular os sinais do pluviômetro e do anemômetro durante os testes da estação meteorológica e reiniciá-la quando necessário. Ele é conectado ao mbed da estação meteorológica através dos pinos p5 (pluviômetro), p6 (*reset*) e p7 (anemômetro). Para emular o sinal do pluviômetro, o programa escreve alternadamente os valores 0 e 1 no pino p5 a

cada 150 milissegundos. O mesmo acontece no pino p7 que emula o anemômetro, porém, os intervalos que mantém o nível lógico do pino em 1 são de 450 milissegundos enquanto o nível lógico 0 é mantido durante os mesmo 150 milissegundos. Além disso, o *firmware* monitor realiza uma escuta na porta serial do dispositivo, que ao receber um sinal do programa monitor, reinicia a estação meteorológica. Para reiniciar a estação meteorológica, escreve-se nível lógico 1 no pino p6 durante 500 milissegundos e retorna-se para nível lógico 0 em seguida.

3.1.3 Programa Monitor de Testes

O programa monitor, executado em um computador pessoal (IBM-PC), tem como objetivo automatizar o processo de testes monitorando os resultados em tempo de execução. Os resultados são analisados individualmente a cada ciclo de leitura da estação meteorológica. Cada ciclo de leitura corresponde aos passos: 1) verificar o horário atual, 2) ler todos os 9 sensores da estação meteorológica (temperatura do ar, umidade do ar, pluviometria, anemometria, temperatura do solo, umidade do solo, irradiação solar, molhamento foliar e tensão da bateria) e 3) gravar os dados no arquivo binário de leituras. Antes de iniciar os testes, o programa lê um um arquivo binário, contendo as medidas corretas (valores de referência). Essas informações servem, posteriormente, como base para a análise dos dados.

Ao iniciar o programa monitor de testes, o mesmo lê um arquivo de configuração contendo informações necessárias, tais como o nome da porta de comunicação do mbed monitor, o caminho do arquivo gerado pela estação meteorológica, a quantidade de ciclos de leituras a serem analisadas, o tempo total de cada ciclo de leitura, e outras. A cada ciclo de leitura, o programa monitor procura o arquivo binário contendo os dados das leituras. Este arquivo é escrito periodicamente na memória *flash* pela estação meteorológica. Se em um tempo estipulado (tempo total de um ciclo de leitura somado a um valor de tolerância de 1 segundo) o arquivo não for encontrado, o programa monitor identifica que houve uma falha durante a execução do teste, provavelmente resultado de uma falha que causou um defeito no sistema. Caso contrário, as informações do arquivo binário escrito pela estação meteorológica são lidas e verificadas pelo programa monitor, que em seguida, apaga o arquivo. Ao verificar o arquivo, confere-se primeiramente sua integridade, calculando novamente o código CRC e comparando-o ao código CRC encontrado no arquivo. Após isso, o valor de cada parâmetro climático é comparado com os dados do arquivo de referência, lido na inicialização do programa. Esse processo de verificação aceita pequenas diferenças entre os valores de referência e os valores armazenados pela estação. Isso é necessário devido às variações normais das leituras de conversores analógicos-digitais. Quando uma falha é encontrada, este programa emite um sinal, escrevendo o caractere 'R' na porta serial (virtual) em que está conectado o *firmware* (mbed) monitor, que reinicia a estação meteorológica. O programa termina quando atinge a quantidade de ciclos de leituras a serem analisadas. A Figura 3.5 mostra um fluxograma do programa monitor de testes.

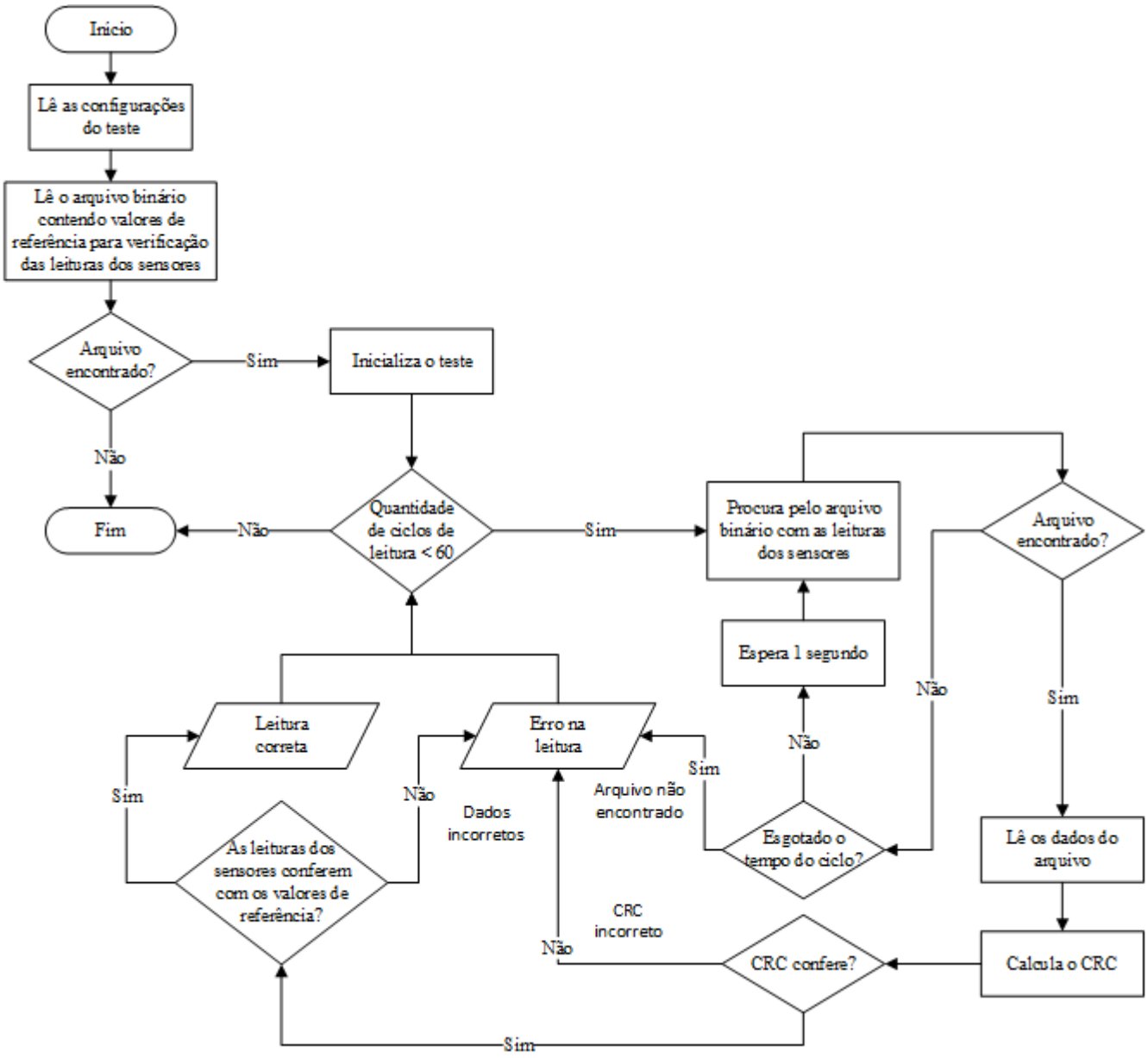


Figura 3.5: Fluxograma do programa monitor de testes.

O resultado de cada ciclo de leitura é classificado em:

- TEST_UNPERFORMED. Teste não realizado. Este valor é retornado quando o arquivo binário que contém os valores de referência das leituras não é encontrado. Se isso ocorrer, nenhum teste é executado.
- TEST_SUCCESSFULLY. Teste realizado com sucesso. Neste caso, o ciclo de leitura não apresentou nenhum erro.
- ERROR_DATA_NOT_FOUND. Arquivo de dados não encontrado. Este erro ocorre quando uma falha trava o sistema e o arquivo não é gerado pela estação meteorológica.

- ERROR_INCORRECT_CRC. CRC inválido. Este erro ocorre quando o CRC de um ciclo de leitura é inválido.
- ERROR_INCORRECT_DATA. Dados incorretos. Neste caso, os dados foram gerados, mas não correspondem aos valores esperados (referência).

Nos testes, os sensores foram substituídos por sinais que emulam valores fixos dos parâmetros climáticos para facilitar a verificação dos dados. A Tabela 3.1 mostra os sinais utilizados.

Parâmetro	Variação	Emulação
Temperatura do ar	5%	Não emulado (SPI)
Umidade do ar	5%	Não emulado (SPI)
Pluviometria	30%	Sinal de frequência (3,33 Hz)
Anemometria	5%	Sinal de frequência (1,66 Hz)
Temperatura do solo	5%	Tensão fixa
Umidade do solo	20%	Tensão fixa
Irradiação solar	5%	Tensão fixa
Molhamento foliar	5%	Resistência fixa
Tensão da bateria	10%	Não emulado (medida real)

Tabela 3.1: Valores das variações de cada parâmetro climático. Alguns parâmetros suportam variações maiores devido ao valor de referência ser um número muito pequeno.

3.2 Técnicas de Tolerância a Falhas Utilizadas

Como já mencionado, o princípio básico de sistemas tolerantes a falhas é prover a redundância de recursos, seja por *hardware*, *software*, dados ou processamento extra. As técnicas podem ser utilizadas em conjunto para aumentar a confiabilidade de um sistema, no entanto, é importante estabelecer uma relação de custo-benefício considerando as vantagens e desvantagens de cada uma. Entre as técnicas de tolerância a falhas vistas na literatura, verificou-se que algumas eram inviáveis a este trabalho. Criar redundância de *hardware* ou *software*, em geral, acrescentaria custos não cabíveis, pois não haveria disponibilidade de recursos, tempo e equipes para replicação de *hardware* ou diversas versões de *software*. Já as técnicas baseadas na redundância de dados e de processamento mostraram-se adequadas, uma vez que os consumos de memória RAM e memória *flash* no programa da estação meteorológica são pequenos, não havendo limitações nesse requisito. Além disso, o tempo de processamento do sistema não é crítico, pois o processador trabalha com ociosidade na maior parte do tempo.

Para adicionar redundância de dados ao programa, todas as variáveis efetivamente usadas foram triplicadas, mantendo as cópias dos dados em diferentes regiões da memória, com isso, a possibilidade das falhas que atingem regiões próximas de memória afetarem as cópias simultaneamente diminuiu consideravelmente. O valor de cada variável efetivamente utilizada foi obtido mediante um sistema de votação. Considerando-se três cópias dos dados, representadas por x_1, x_2 e x_3, esses valores podem sofrer alterações devido a quantidade de falhas inseridas ao longo do teste. No entanto, se uma falha altera o valor de um dado em uma posição x_1 de memória, para essa falha causar um erro não percebido pelo sistema é

preciso que a posição x_2 ou x_3 sejam alteradas exatamente com o mesmo valor de x_1, (ou que x_2 e x_3 apresentem o mesmo valor incorreto) o que dificilmente ocorre. Se apenas uma cópia for corrompida, as outras duas mantém os valores corretos da informação e o erro é evitado. Em situações em que duas das três cópias são alteradas com valores incorretos, se os valores dos dados modificados forem diferentes (situação mais provável) o sistema então assume um estado de erro e toma uma medida de correção, refazendo a operação (redundância de processamento) ou apenas evitando a propagação do erro. A redundância de processamento foi implementada pela reexecução das funções até suas tarefas serem realizadas com sucesso ou o limite de tentativas ser atingido. As técnicas utilizadas para empregar tolerância a falhas no estudo de caso deste trabalho é visto detalhadamente na Subseção 3.2.1 e na Subseção 3.2.2.

Embora as técnicas de tolerância a falhas possam ser facilmente empregadas em um projeto, seu uso requer tempo e esforços adicionais. Reescrever um código não tolerante a falhas aplicando as técnicas de tolerância a falhas pode custar tempo e um esforço considerável. Por esse motivo, viu-se a necessidade de criar uma biblioteca para tolerância e recuperação de falhas, descrita na Subseção 3.2.1.

3.2.1 Biblioteca de Recuperação de Falhas *FaultRecovery*

A biblioteca de recuperação de falhas *FaultRecovery* foi desenvolvida a fim de facilitar a implementação das técnicas de tolerância a falhas em códigos desenvolvidos para a plataforma mbed. Ela é composta por um conjunto de classes e macros que possibilitam a adição de tais técnicas com pouca reescrita de código. Para isso, deve-se importá-la do repositório de códigos mbed, disponível no endereço `http://mbed.org/users/kleberkruger/code/FaultRecovery/`.

Embora a implementação de técnicas de tolerância a falhas melhore a segurança do sistema, esta biblioteca não garante que as falhas não ocorram. Por esse motivo, pensou-se em uma estratégia que permitisse a um sistema se recuperar de falhas que provocam o travamento. Ao travar o dispositivo, o temporizador *watchdog* reinicia o *firmware* e o restaura ao ponto em que estava. Essa técnica permite mascarar falhas, muitas das vezes, de modo imperceptível ao usuário. Considerando-se o programa da estação meteorológica, uma falha durante o envio dos dados ao servidor pode causar um transtorno muito grande. Ao deixar de recarregar o temporizador watchdog, o mesmo irá reiniciar o *firmware* depois de um tempo predefinido. Se o horário de envio estiver dentro desse intervalo em que o *firmware* ficou em estado de erro, isso causará um transtorno muito grande, já que os dados normalmente são configurados para serem enviados apenas uma vez ao dia. Com este sistema de recuperação, ao reiniciar-se de uma falha, o programa carrega o último estado em que estava e volta do ponto em que parou, isto é, de envio dos dados ao servidor. A implementação dessa técnica pode ser feita utilizando a classe *FaultRecovery* da biblioteca. Entretanto, cabe observar que esse procedimento de recuperação deve ser rigorosamente implementado, pois se a falha que travou o dispositivo for uma falha permanente localizada neste ponto do sistema, em razão de um *bug* no software ou problema no hardware de transmissão, ao executar novamente o envio dos dados, a falha novamente ocorrerá. Logo, o ideal é tomar uma estratégia secundária ao refazer a operação ou ignorar este procedimento, caso o mesmo não seja crucial.

A classe *FaultRecovery* permite ao sistema se recuperar ao ponto em que estava quando o mesmo é reiniciado pelo *watchdog* em razão de uma falha. Essa classe deve ser herdada pela

classe principal do programa que ao iniciar, configura uma máquina de estados, adicionando em uma lista todos os estados do programa previamente conhecidos. No estudo de caso deste trabalho, identificou-se que o sistema da estação meteorológica é formado por seis estados, descritos a seguir:

- STATE_NOT_CONFIGURED. Este é o primeiro estado do sistema. Nele, o *firmware* está iniciando e a estação meteorológica ainda não foi configurada.
- STATE_CONFIGURED. Neste segundo estado, a estação meteorológica se encontra configurada, isto é, o método *config*, que liga a energia dos circuitos auxiliares, desabilita o GPS e a *ethernet* para reduzir o consumo de energia e acerta o relógio, foi executado. Neste estado, a estação meteorológica se encontra pronta para a leitura dos sensores e para o envio dos dados ao servidor.
- STATE_READ_SENSORS. O programa entra nesse estado quando está no momento de ler todos os sensores da estação meteorológica.
- STATE_SAVE_DATA. Após ler os sensores, o programa atinge este estado, indicando que os dados devem ser escritos no dispositivo local de armazenamento (memória *flash*).
- STATE_DATA_SAVED. Este estado indica que os dados de leitura foram armazenados corretamente no dispositivo local de armazenamento. Se no momento da leitura dos sensores a estação meteorológica falhar antes de chegar a este estado, o processo de leitura deve ser refeito novamente.
- STATE_SEND_DATA. Este estado indica que a estação meteorológica encontra-se no momento de envio dos dados ao servidor por meio de comunicação GPRS ou satélite.

Os estados previamente conhecidos (definidas geralmente por uma *enum*) são adicionados em uma lista de estados de *FaultRecovery* conforme o método *addState*, exibido no Quadro 3. Este método recebe como parâmetro o número do estado, a função de *callback* (função executada conforme a ocorrência de um evento predefinido), executada quando o sistema é recuperado a partir daquele ponto, e um número *float* opcional. Este número, quando indicado e maior que zero, define o limite de tempo em que o estado pode permanecer, sendo reiniciado pelo *watchdog* quando esse tempo é atingido.

A classe *FaultRecovery* também possui um atributo do tipo inteiro, chamado *state* que serve para guardar a identificação do estado atual. Essa variável é triplicada na memória e o acesso a ela é feito por um sistema de votação. Ao executar procedimentos que alterem o estado atual do programa, realiza-se um procedimento de mudança de estado. Esse procedimento consiste em atualizar as 3 cópias da variável e escrever essa informação em um arquivo no dispositivo local de armazenamento (memória *flash*). Quando uma falha ocorre, o arquivo no dispositivo local de armazenamento é lido e o programa volta ao estado em que estava, executando a função de *callback* configurada no início do programa.

No código do Quadro 3, os estados `STATE_NOT_CONFIGURED`, `STATE_CONFIGURED` e `STATE_DATA_SAVED` adicionaram a mesma função de *callback*, pois quando o sistema é reiniciado durante esses estados, o único procedimento a se tomar é configurar novamente a estação meteorológica, uma vez que o sistema não estava durante um procedimento de leitura ou envio. Os

estados STATE_READ_SENSORS e STATE_SAVE_DATA adicionam o método de *callback* *readSensorsAgain*. No primeiro caso, o sistema estava no momento de leitura dos sensores e no segundo, os dados lidos ainda não haviam sido gravados na memória não volátil, portanto, todo o procedimento de leitura deve ser retomado. No estado STATE_SEND_DATA, a falha ocorreu durante o envio dos dados ao servidor e este processo terá que ser refeito novamente. Cabe observar que ao se recuperar de uma falha, o sistema sempre deverá se reconfigurar novamente invocando o método *config* e que embora o estado de envio tenha sido adicionado na máquina de estados, este método não foi implementado devido a não disponibilidade do módulo GPRS. Em vez disso, apenas uma mensagem é impressa na tela. A Figura 3.9 mostra um fluxograma detalhado do uso dessa biblioteca de recuperação de falhas no *firmware* tolerante a falhas.

```
1  WeatherStationFT::WeatherStationFT() : FaultRecovery() {
2
3      FaultRecovery::addState(STATE_NOT_CONFIGURED, &configAgain, 1.0);
4      FaultRecovery::addState(STATE_CONFIGURED, &configAgain, 10.0);
5      FaultRecovery::addState(STATE_READ_SENSORS, &readSensorsAgain, 10.0);
6      FaultRecovery::addState(STATE_SAVE_DATA, &readSensorsAgain, 3.0);
7      FaultRecovery::addState(STATE_DATA_SAVED, &configAgain, 1.0);
8      FaultRecovery::addState(STATE_SEND_DATA, &sendAgain, 5.0);
9
10     ...
11 }
```

Quadro 3: Código que configura a máquina de estados. O primeiro parâmetro corresponde ao número do estado. O segundo, a função de recuperação que será executada se o firmware travar durante esse estado. O terceiro parâmetro determina o tempo limite em que o estado pode permanecer. Esse valor é usado como limitador do temporizador *watchdog*.

Uma vantagem dessa técnica é que além de geralmente mascarar as falhas do sistema, muitas vezes, de modo imperceptível ao usuário, seu uso fornece registros de *log* que facilitam o diagnóstico de erros do programa. A Figura 3.6 mostra um *log* do sistema recuperando-se de uma falha. Nele, é possível observar que a estação meteorológica, ao ser reiniciada pelo *watchdog*, retomou sua tarefa do ponto em que travou. No entanto, embora dito que esta técnica geralmente permite o mascaramento das falhas, em algumas situações isso pode não ser verdade, principalmente em situações em que o atraso para a execução de uma tarefa ocasione em erros. Um exemplo são os testes com o programa monitor. Como mencionado na Subseção 3.1.3, o programa monitor de testes espera pelo arquivo binário a cada 11 segundos. Se o arquivo não é gerado nesse tempo, um defeito é diagnosticado. Portanto, essa estratégia não impediu a identificação de defeitos pelo programa monitor, mas em uma situação real, em que um pequeno atraso é tolerável, como a do exemplo de envio dos dados ao servidor, essa estratégia evitaria prejuízos.

Na Figura 3.6, o estado ("state") número 2 indica que o programa, ao ser reiniciado pelo temporizador *watchdog* devido a uma falha, estava durante o processo de leituras dos sensores. Ao reiniciar e constatar que o programa foi reiniciado pelo *watchdog* (essa informação é armazenada em um registrador específico), o mesmo executa a função de recuperação programada, que reconfigura a estação e em seguida, executa novamente o método *readSensors*,

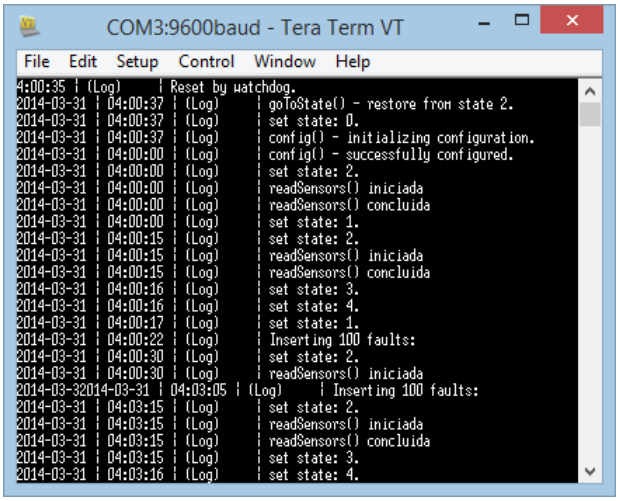


Figura 3.6: Arquivo de *log* que mostra a recuperação de uma falha. Ao ser reiniciado pelo *watchdog* devido a uma falha, o programa lê o arquivo de configuração e volta ao estado em que estava (leitura dos sensores) antes da falha travar o dispositivo.

responsável por ler todos os sensores da estação meteorológica.

Além das classes mencionadas, esta biblioteca é composta por um conjunto de macros que possibilita a adição de técnicas de tolerância a falhas no código. O termo macro é descrito para descrever uma abstração que define como um padrão de entrada deve ser substituído por um padrão de saída. As macros existentes nesta biblioteca são:

- EXECUTE: esta macro deve ser adicionada antes da chamada de uma função (ou método) que define o ponto de partida para um novo estado do programa. Ao compilar o código, os trechos onde a macro foi escrita serão substituídos por um procedimento que primeiro altera o estado do programa para o estado informado, e em seguida, executa a função. Considerando-se que a estação meteorológica altera do estado STATE_CONFIGURED para o estado STATE_READ_SENSORS ao chamar o método *readSensors*, deve-se substituir o trecho da chamada de *readSensors* por:

```
EXECUTE(readSensors(), STATE_READ_SENSORS);
```

- EXECUTE_AND_NEXTSTATE: similar a anterior, no entanto, além de alterar o programa para um estado x_1, a macro também o define para um novo estado x_2 caso uma condição seja satisfeita pelo retorno da função. Um exemplo de utilização é: após ler os sensores da estação meteorológica, a função *saveData* é executada para gravar os dados das leituras no dispositivo local de armazenamento. A estação meteorológica só deve ir para o próximo estado "STATE_DATA_SAVED" se a função conseguir gravar corretamente os dados na memória *flash*, caso contrário, o estado atual deve permanecer o mesmo ("STATE_SAVE_DATA"). A chamada para a função *saveData* deve ser realizada da seguinte maneira:

```
EXECUTE_AND_NEXTSTATE(saveData(), STATE_SAVE_DATA,
                                  STATE_DATA_SAVED, true);
```

- TRY_EXECUTING_N_TIMES: executa uma função (ou método) sob a redundância de processamento. A função é executada até um número informado de vezes ou até que sua resposta seja igual a uma condição estabelecida. Esta macro tem como parâmetros: a função a ser executada, o novo estado que ela define, uma variável recebendo o retorno da função, uma condição de parada e um número limite de tentativas;

- TRY_EXECUTING_N_TIMES_AND_NEXTSTATE: similar a anterior, porém esta ainda altera o programa para um novo se o retorno da função satisfazer uma condição informada;

- READ_SENSOR: lê um sensor utilizando a redundância de processamento para obter uma média precisa, desprezando leituras com valores acima de uma variação informada. Com a utilização dessa macro, o valor da leitura de um sensor, em vez de ser obtido por uma única leitura sujeita a erros, é calculado pela média de um conjunto de leituras, efetuadas em intervalos de i milissegundos e submetidas ao método *avg*. O método *avg* calcula a média dos valores de uma amostra A de tamanho n, desprezando os valores que estejam fora de uma variação v. Este método retorna a média calculada ou o valor NaN, quando os dados analisados não permitem um resultado confiável. Um resultado é classificado como confiável quando, dos n elementos na amostra A, s estão dentro da variação v. O valor da variação v, o valor de s, o tempo do intervalo i entre as leituras e o tamanho n das amostras são configuráveis, tendo como restrições $v \geq 0$ e $n/2 < s \leq n$.

O código do Quadro 4 mostra a implementação em linguagem C++ do método *avg*, responsável por calcular a média de um conjunto de leituras (amostras) de um sensor, desprezando as leituras fora de uma variação estipulada.

Os parâmetros do método *avg* são:

- A: vetor com os valores das leituras.

- n: tamanho do vetor.

- s: quantidade mínima de elementos que deverão pertencer a média.

- v: determina a variação máxima entre quaisquer elementos que pertencem a média.

```
1  float FaultRecovery::avg(float A[], int n, int s, float v) {
2
3      int i, j, left, right, cr;
4      float result = 0;
5
6      qsort(A, n, sizeof(int), compare);
7
8      left = right = n / 2;
9
10     for (cr = 1, i = left; i >= 0; i--) {
11         for (j = i + 1; j < n && A[j] <= A[i] + v; j++);
12
13         if (j - i > cr) {
14             cr = j - i;
15             left = i;
16             right = j - 1;
17         }
18     }
19
20     if (cr < s) return NAN;
21
22     for (i = left; i <= right; i++)
23         result += A[i];
24
25     if (cr < n) logger.log("Recovery:AVG");
26
27     result = result / cr;
28     return result;
29 }
```

Quadro 4: Método que calcula a média das leituras dos sensores desprezando as leituras fora de uma variação definida.

O método *avg* primeiramente ordena em ordem crescente os elementos do vetor de tamanho n, e em seguida, escolhe o elemento central como referência. Quando n é um valor par, escolhe-se o elemento central que está na posição $n/2$. Como toda amostra com análise satisfatória deve possuir a maioria dos elementos corretos (pelo menos metade mais um dos elementos apresentam valores dentro da variação v), pode-se deduzir que o elemento central, necessariamente pertence à solução. Após isso, a partir do elemento central, procura-se o conjunto que abrigue a maior quantidade de elementos dentro da variação v. Caso o tamanho do maior conjunto seja inferior a s, o método retorna o valor NaN, significando que não pôde calcular uma média precisa, pois a quantidade de elementos dentro da variação é menor que o mínimo definido. Na Figura 3.7 define-se um fluxograma do método *avg* e na Figura 3.8 mostra-se um exemplo, no qual $n = 7$, $s = 6$ e $v = 2$.

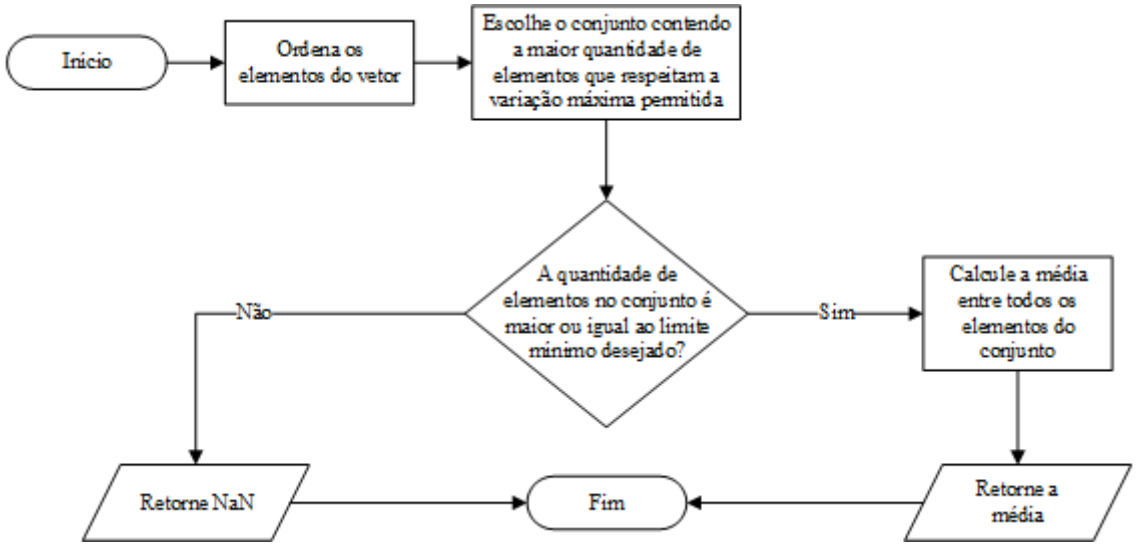


Figura 3.7: Fluxograma do método *avg*. O método ordena os elementos do vetor de tamanho n e escolhe o elemento central como referência. A partir dele, procura o arranjo que abriga a maior quantidade de elementos dentro da variação máxima v. Caso o tamanho do maior conjunto encontrado seja maior ou igual a s, o método retorna a média de todos os elementos contidos no conjunto, caso contrário, retorna o valor NaN, significando que o método não pôde calcular uma média confiável.

A leitura de um sensor ainda pode utilizar mais uma redundância de processamento: ao preencher o vetor com as amostras e enviá-las ao método *avg*, os dados podem não ser satisfatórios e o método retornará o valor NaN. Nesse caso, o processo pode ser repetido até um número máximo de tentativas.

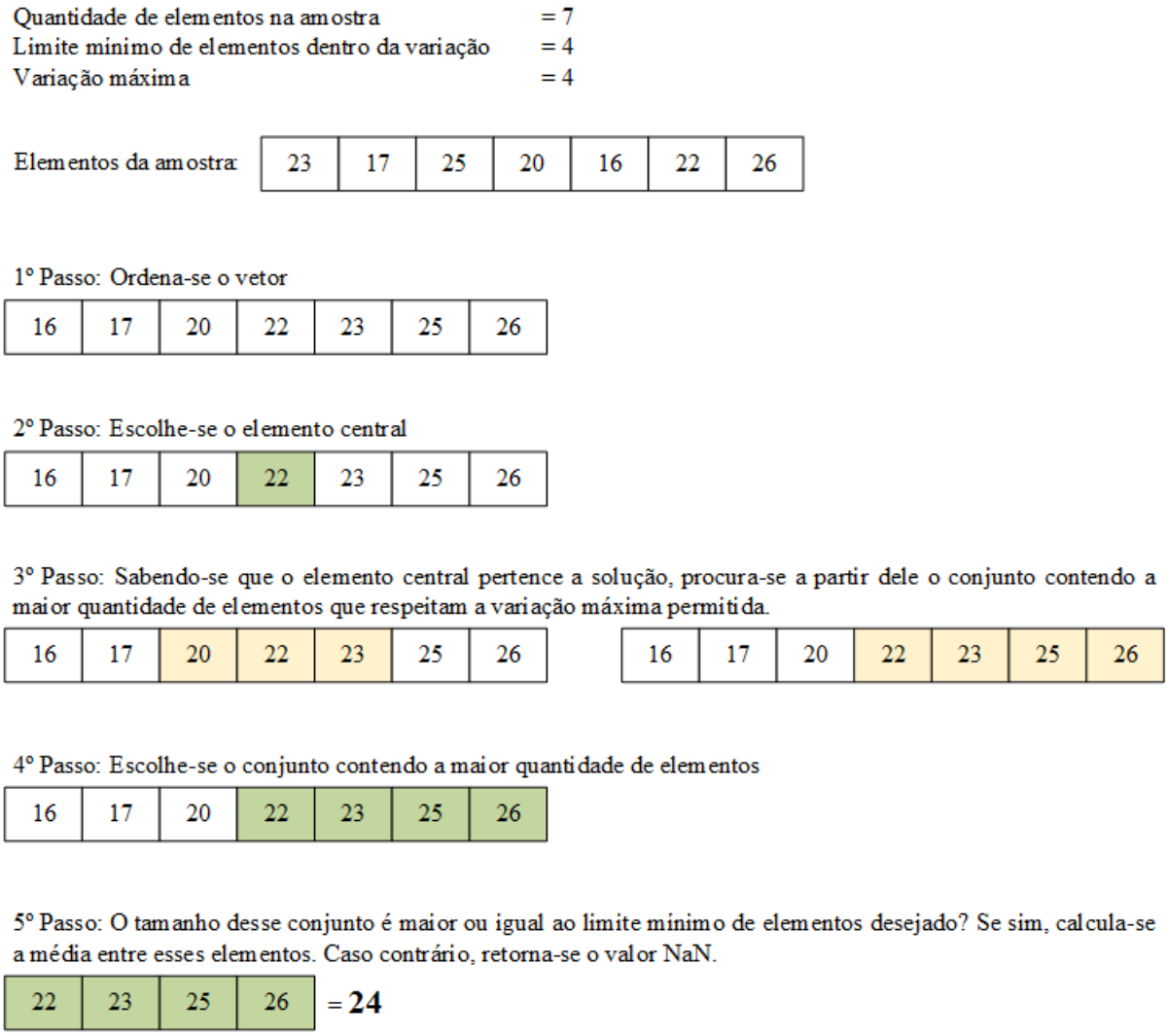


Figura 3.8: Exemplo do método *avg*. O método escolhe sempre o melhor conjunto de elementos, isto é, o conjunto com a maior quantidade de elementos dentro da variação máxima.

3.2.2 Alterações no *Firmware* para o Emprego de Tolerância a Falhas

O primeiro passo para a reescrita do código do *firmware* inicial que não empregou nenhuma técnica de tolerância a falhas (exceto o temporizador *watchdog*) foi identificar as áreas do código com maior vulnerabilidade. Para isso, no início do programa foram colocadas instruções que imprimiram os endereços de memória de cada variável efetivamente usada no programa e injetou-se falhas pelo sistema injetor de falhas *FaultInjector*. Analisando o *log* do sistema injetor de falhas, foi possível comparar os endereços e verificar quais variáveis foram mais sujeitas a falhas.

Após os dados mais vulneráveis serem identificados, o segundo passo consistiu em empregar as técnicas baseadas na redundância de dados e de processamento, uma vez que as técnicas baseadas na redundância de hardware e software não eram viáveis a este trabalho.

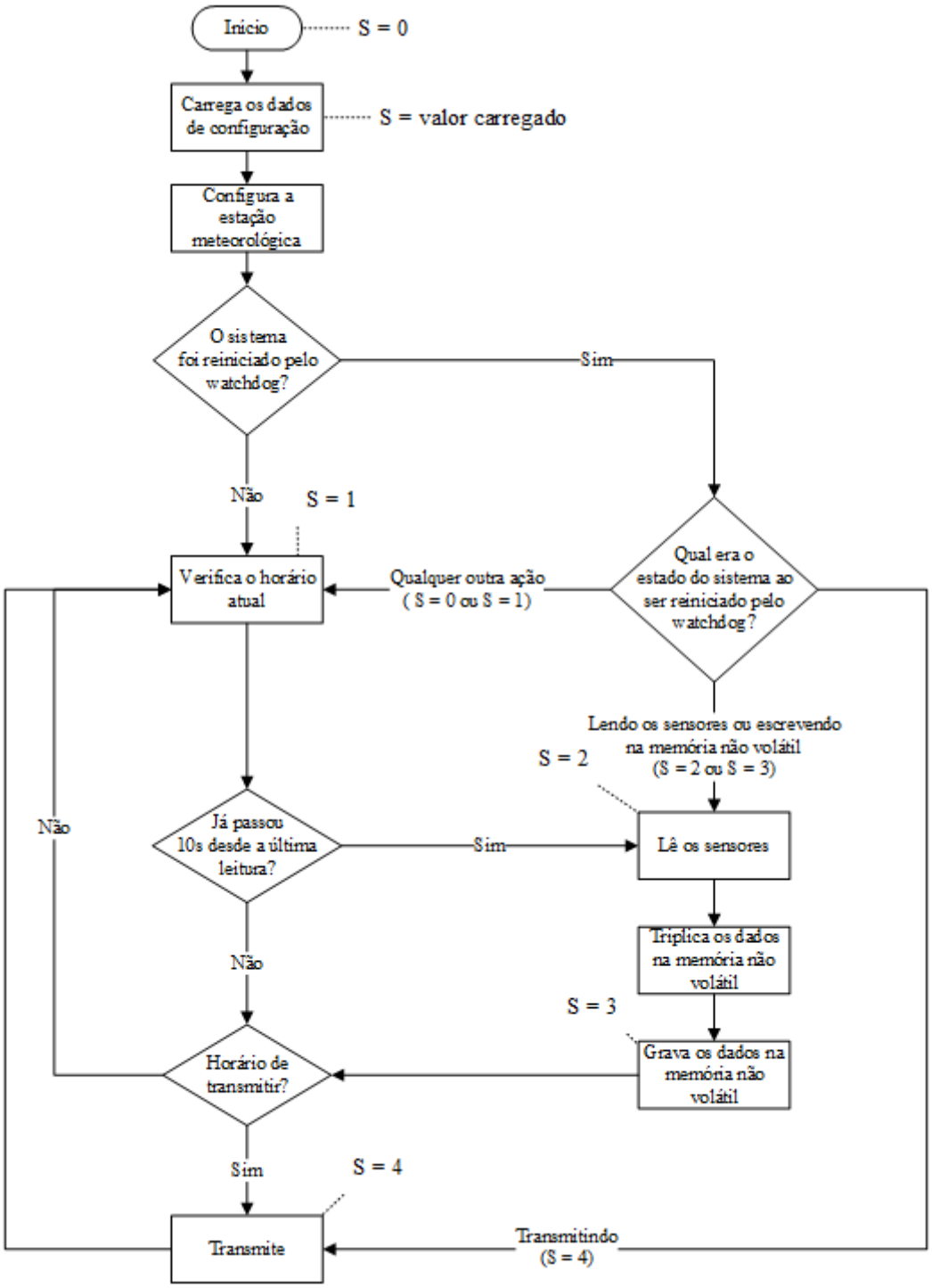


Figura 3.9: Estação meteorológica com técnicas de tolerância a falhas implementadas. Uma máquina de estado permite que o sistema volte ao estado em que estava ao ser reiniciado pelo *watchdog*. *S* corresponde ao estado atual do sistema. A cada alteração de estado, todas as cópias da variável são atualizadas e escritas em um arquivo de configuração. A redundância de dados é aplicada pela triplicação dos dados na memória *flash*.

O programa da estação meteorológica é preso a uma estrutura de repetição que em seu bloco básico chama constantemente um método para verificar o horário atual e conferir se

está no momento de ler os sensores da estação ou enviar os dados das leituras para o servidor. Este método contém apenas variáveis locais, usadas por um período muito curto de tempo, e seus valores são escritos e lidos de forma consecutiva. Logo, como a informação é adquirida e consumida muito rapidamente, a probabilidade de falhas atingir essas variáveis no momento em que o dado está sendo usado é pequena, por isso esse trecho do código foi inalterado.

Verificou-se que o trecho de maior atividade no programa é executado durante a leitura dos sensores. Essa atividade é feita pelo método *readSensosrs*. O Quadro 5 mostra o método *readSensors* no *firmware* sem tolerância a falhas. Cabe observar que qualquer falha durante a leitura de um sensor resulta no armazenamento incorreto da informação. Outro erro ocorre quando uma falha corrompe a área de memória da variável "data". Após uma falha atingir a região de memória da variável, o programa obtém um valor incorreto e o erro é propagado.

```
1  void WeatherStation::readSensors() {
2      logger.log("readSensors() - initializing readings.");
3      powerBattery(POWER_ON);
4
5      SHTx::SHT15 sensorTE_UR(p29, p30);
6      sensorTE_UR.setOTPReload(false);
7      sensorTE_UR.setResolution(true);
8
9      Anemometer anem(p21);
10     Wetting wet(p19, p26, p25);
11     wet.config(100, 1000, 1, 3000);
12
13     data.setTime(time(NULL));
14     sensorTE_UR.update();
15     sensorTE_UR.setScale(false);
16
17     data.setAnemometer(anem.read());
18     data.setPluviometer(pluv.read());
19     data.setWetting(wet.read() / 1000);
20     data.setTemperature(sensorTE_UR.getTemperature());
21     data.setHumidity(sensorTE_UR.getHumidity());
22     data.setSoilTemperaure(readSensor(17, 5, 0.320512821, 50, 3.205128205));
23     data.setSoilHumidity(readSensor(16, 1.1, 0, 5.54, 1.0));
24     data.setSolarRadiation(readSensor(18, 0, 0, 1500, 1.5));
25     data.setBatteryVoltage(readSensor(15, 0, 0, 15.085714286, 3.3));
26
27     data.setCRC(data.calculateCRC());
28
29     powerBattery(POWER_OFF);
30     logger.log("readSensors() - finished.");
31 }
```

Quadro 5: Método sem tolerância a falhas responsável por ler todos os sensores da estação meteorológica. Qualquer falha durante a leitura de um dos sensores pode gerar um erro no arquivo binário de leituras.

A solução encontrada para resolver esse problema foi adotar redundância replicando-se os dados e o processamento. O código alterado com a implementação dessas técnicas é exibido na Quadro 6. Dentre as modificações, a macro READ_SENSOR da biblioteca *FaultRecovery* foi usada na leitura de cada sensor. O intervalo entre as leituras foi definido em 5 milissegundos. Cada amostra foi composta por 4 leituras e a quantidade de elementos dentro da variação máxima foi definido em 75%, isto é, para um conjunto de 4 leituras ser classificado como confiável, 3 precisaram estar dentro da variação máxima. Caso contrário, uma nova amostra, formada por outras quatro leituras serão submetidas novamente ao método *avg*, até que se encontre um resultado confiável ou o número de tentativas, definido em 3, seja atingido. Se isso acontecer, o sensor recebe o valor NaN, indicando que todas as tentativas de leitura daquele sensor falhou. Ao final da leitura dos sensores, os dados foram triplicados na memória RAM (volátil). A Figura 3.10 mostra um fluxograma desse processo do *firmware* da estação meteorológica com tolerância a falhas.

```
1  void WeatherStationFT::readSensors() {
2
3      int i, att, rdIntv = 5;
4      float result;
5      float samples[getNumberOfReadings()];
6
7      logger.log("readSensors() - initializing readings.");
8      powerBattery(POWER_ON);
9
10     SHTx::SHT15 sensorTE_UR(p29, p30);
11     sensorTE_UR.setOTPReload(false);
12     sensorTE_UR.setResolution(true);
13
14     Anemometer anem(p21);
15     Wetting wet(p19, p26, p25);
16     wet.config(100, 1000, 1, 3000);
17
18     data.setTime(time(NULL));
19     sensorTE_UR.update();
20     sensorTE_UR.setScale(false);
21
22     READ_SENSOR(result, rdIntv, 3, anem.read(), getNumberOfReadings(),
23                 getMinCorrectReadings(), 5);
24     data.setAnemometer(result);
25     if (!isnan(result) && att > 1)
26         logger.log("Recovery: AVG Redundancy. [Anemometer]");
27
28     READ_SENSOR(result, rdIntv, 3, pluv.read(), getNumberOfReadings(),
29                 getMinCorrectReadings(), 30);
30     data.setPluviometer(result);
31     if (!isnan(result) && att > 1)
32         logger.log("Recovery: AVG Redundancy. [Pluviometer]");
33
34     // Lê o restante dos sensores usando a macro READ_SENSOR...
```

```
35      data.setCRC(data.calculateCRC());
36
37      powerBattery(POWER_OFF);
38
39      memcpy(&data_copy_1, &data, sizeof(ReadingData));
40      memcpy(&data_copy_2, &data, sizeof(ReadingData));
41
42      logger.log("readSensors() finished.");
43  }
```

Quadro 6: Método com tolerância a falhas responsável por ler todos os sensores da estação meteorológica. Cada sensor é lido com o uso da macro READ_SENSOR e ao final, os dados são triplicados na memória volátil. A variação máxima de cada parâmetro climático foi definido de acordo com a Tabela 3.1.

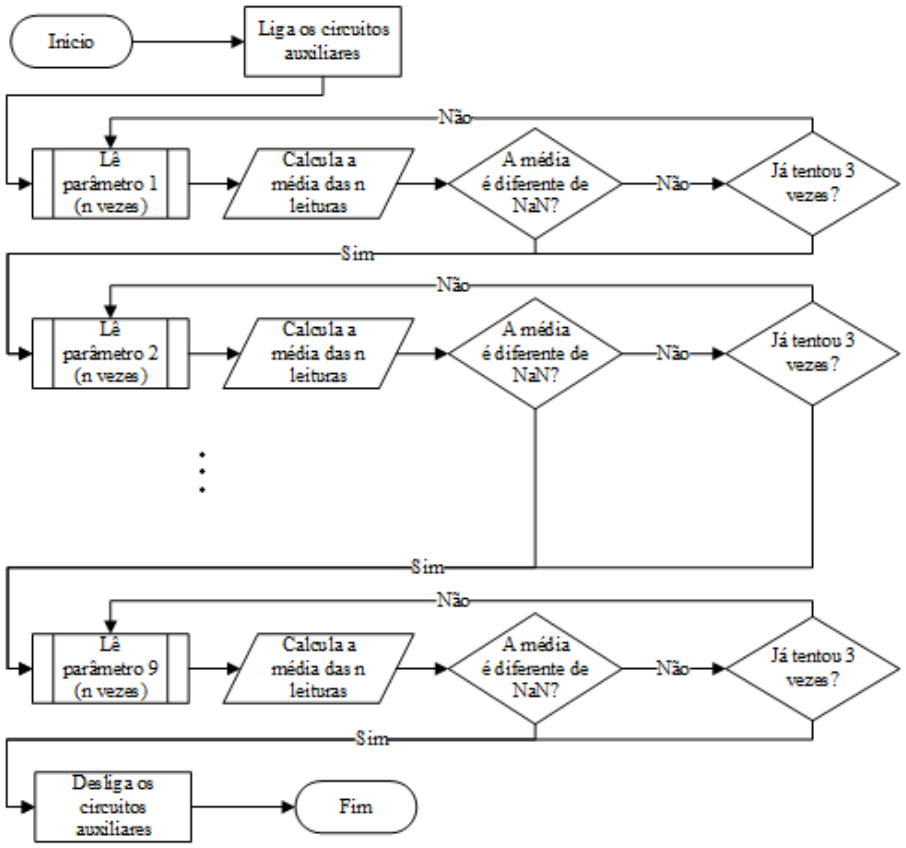


Figura 3.10: Processo de leituras dos sensores. O valor de cada um dos parâmetros climáticos é definido conforme a média de uma série de leituras. As amostras são obtidas por uma sequência de leituras feitas em intervalos de 5 milissegundos, e submetidas ao método *avg*. O método *avg* calcula o valor médio desconsiderando os valores que estão fora de uma variação. Este processo pode ser repetido até que o método retorne um resultado satisfatório ou o número máximo de tentativas seja atingido. Os sensores correspondem à temperatura do ar, umidade, precipitação, velocidade do vento, umidade do solo, temperatura do solo, radiação solar, molhamento foliar, a voltagem da bateria.

Cabe observar que em alguns trechos a redundância de hardware é essencial, como por

exemplo, o método *send*, responsável por enviar os dados ao servidor. A falha no módulo GPRS impossibilita o envio dos dados. Nesse caso, é fundamental a presença de hardware adicional. Este é apenas um exemplo, pois como já mencionado, esta funcionalidade não foi implementada.

3.3 Testes Realizados

Os testes elaborados neste trabalho realizaram, cada um, 60 ciclos de leituras. O primeiro conjunto de testes injetou via software no *firmware* tolerante a falhas 10%, 20%, 25%, 30%, 40% e 50% de falhas nas leituras dos sensores. O objetivo foi analisar a capacidade de tolerância a falhas do método *avg*, juntamente com a redundância de processamento sobre ela, tendo como entrada uma quantidade (em porcentagem) definida de falhas nas leituras dos sensores. Esse conjunto de testes não utilizou a biblioteca *FaultInjector*, pois as falhas provocadas pela biblioteca danificam várias partes do sistema, podendo causar até mesmo o travamento do equipamento.

O segundo conjunto de testes utilizou a biblioteca *FaultInjection* para injetar falhas em regiões de memória do microcontrolador com o objetivo de simular os efeitos do fenômeno *bit-flip* e avaliar o desempenho dos *firmwares* sem tolerância a falhas e com tolerância a falhas. A quantidade de falhas inseridas em cada ciclo foi de 0, 1, 2, 4, 8, 16, 32, 64, 128 e 256. Os testes foram realizados em duas configurações de injeção de falhas: apenas memória de dados; e memória de dados e memória interna.

O terceiro conjunto de testes, similar ao anterior, diferiu-se pela injeção adicional de 25% de falhas nas leituras dos sensores. Utilizou-se esse valor porque o tempo de execução de cada conjunto de testes é elevado, conforme exibido no Apêndice B, e por não haver tempo hábil para o aumento da quantidade de testes, escolheu-se como medida o valor médio, uma vez que o primeiro conjunto de testes trabalhou com quantidades variadas de falhas nas leituras dos sensores.

Para permitir uma grande quantidade de testes em um curto período de tempo, a duração de cada ciclo, que corresponde ao intervalo entre as leituras da estação meteorológica, foi definida em 5 segundos no *firmware* sem tolerância a falhas. Dado que as técnicas de redundância a falhas acrescentam tempo de processamento, o tempo do ciclo no *firmware* com tolerância a falhas precisou ser redefinido para 10 segundos. Este tempo corresponde a duração máxima (com a adição de uma pequena folga) de um ciclo completo de leitura que o sistema consumiu em situações com extremas quantidades de falhas injetadas.

A Figura 3.11 e a Figura 3.12 apresentam, respectivamente, um fluxograma do *firmware* sem tolerância a falhas e com tolerância a falhas, executando com injeções de falhas pela biblioteca *FaultInjection*.

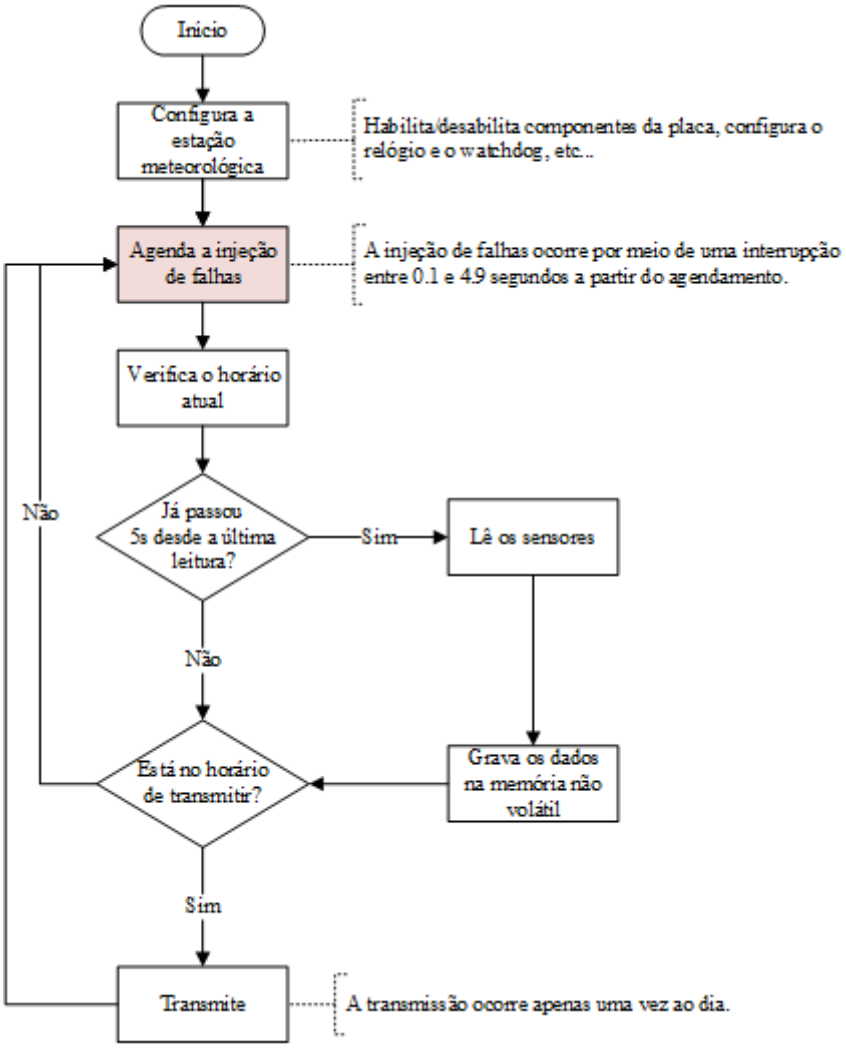


Figura 3.11: Fluxograma do *firmware* não tolerante a falhas com a injeção de falhas nas regiões de memória do microcontrolador. As falhas são injetadas por uma interrupção em intervalo de tempo aleatório entre 0,1 a 4,9 segundos.

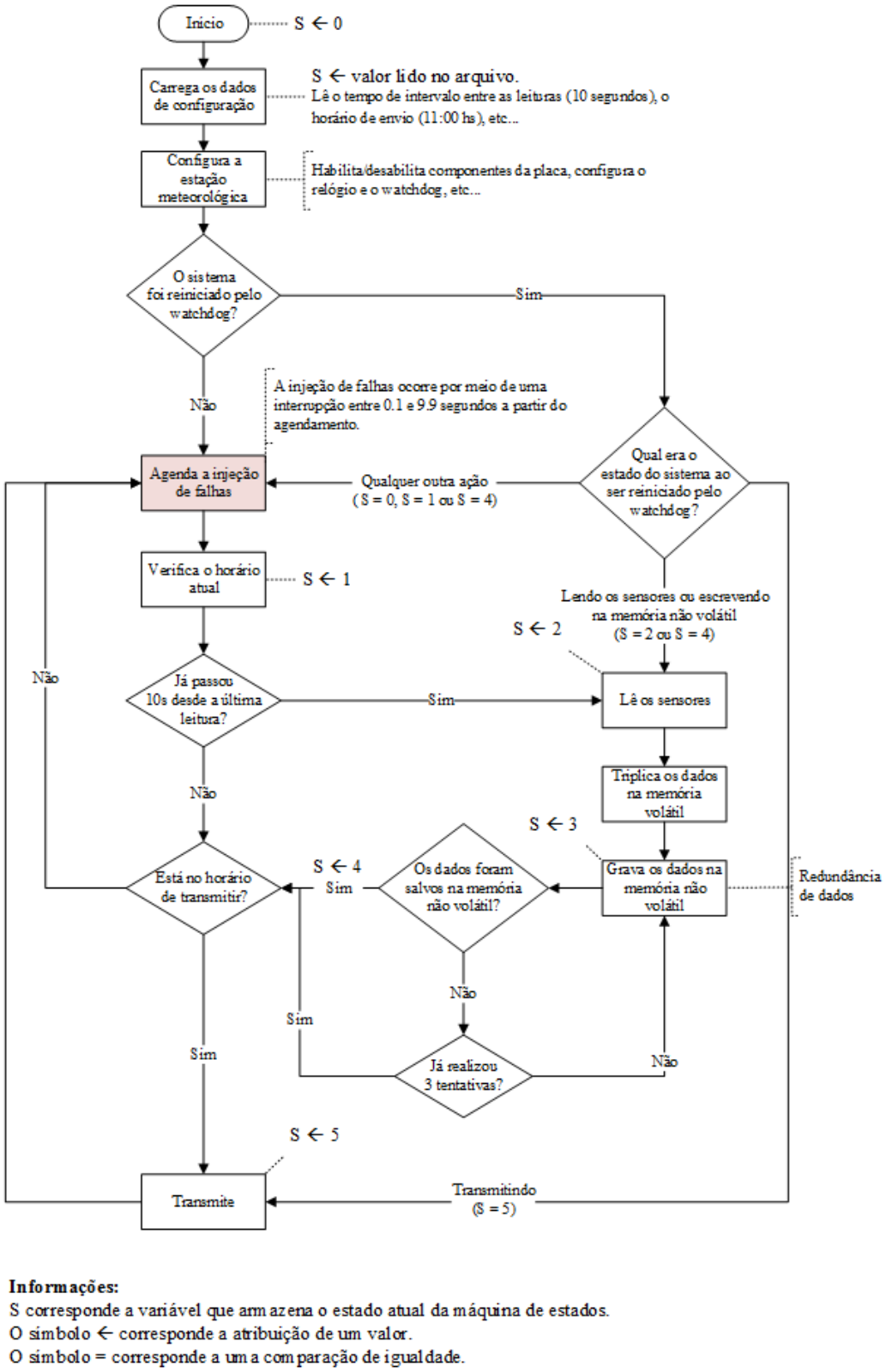


Figura 3.12: Fluxograma do *firmware* tolerante a falhas com a injeção de falhas nas regiões de memória do microcontrolador. A máquina de estados faz parte do sistema de recuperação de falhas da biblioteca *FaulRecovery*. As falhas são injetadas por uma interrupção em intervalo de tempo aleatório entre 0,1 a 9,9 segundos.

Capítulo 4

Resultados

Neste capítulo são apresentados os resultados dos testes realizados. Na Seção 4.1 mostra-se os resultados dos testes com injeção de falhas nas leituras dos sensores. Na Seção 4.2, os resultados dos testes que utilizaram a biblioteca *FaultInjector* para injetar falhas nas regiões de memória do microcontrolador, e na Seção 4.3, os resultados dos testes com injeção de falhas nas leituras dos sensores e nas regiões de memória. Na Seção 4.4 são exibidas as medições do consumo de energia, e na Seção 4.5 são mostradas as vantagens e desvantagens do uso das técnicas de tolerância a falhas, considerando o aumento dos recursos, como memória, tempo de processamento, consumo de energia, e a diminuição dos defeitos.

4.1 Testes com Injeção de Falhas nas Leituras dos Sensores

Os testes explicados nesta seção injetaram falhas nas leituras dos sensores, afetando 10%, 20%, 25%, 30%, 40% e 50% das leituras. Como já mencionado, esse conjunto de teste foi realizado sob duas configurações: na primeira, cujos resultados aparecem na Figura 4.2a, o método *avg* foi configurado com uma taxa de exatidão (percentual de leituras dentro da faixa aceitável) de 66,6%, e na segunda, a taxa de exatidão foi definida em 75% (Figura 4.2b). A escolha dessas configurações baseou-se principalmente no tempo adicional gerado pela redundância de processamento. Os valores definidos permitem amostras com dados suficientes para expressar um resultado confiável e em contrapartida, não adicionam tempo demasiado.

Para se calcular a eficiência do método *avg*, um detalhado esquema de *log* foi implementado. Este *log* é registrado a cada 60 ciclos de leitura pela estação meteorológica. Na Figura 4.1 mostra-se o resultado de um teste que injetou 10% de falhas nas leituras dos sensores. Neste, o método *avg* foi configurado com uma taxa de exatidão de 66,6%. Os dados mostram que ao longo dos 60 ciclos de leituras, foram injetadas 346 falhas, e o método *avg* foi invocado 553 vezes (540 devido às leituras dos 9 sensores a cada ciclo de leitura, e mais 13 em razão da redundância de processamento, efetuada 11 vezes durante o teste). Das 11 vezes que ocorreu redundância de processamento sobre o método *avg*, em nove a média confiável foi obtida na segunda tentativa e em outras duas, apenas na terceira. Devido a eficiência dessa técnica, nenhum erro foi diagnosticado.

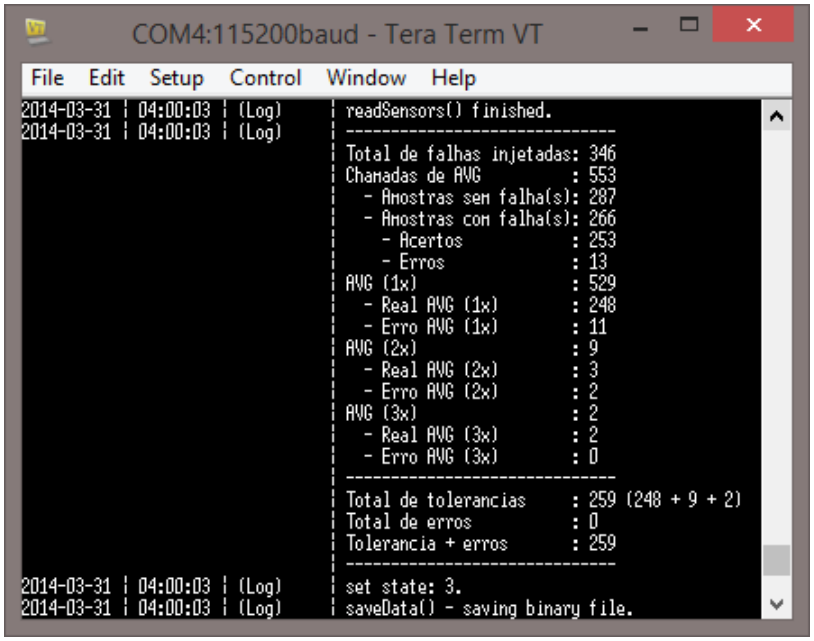


Figura 4.1: Registros do método *avg*. O *log* mostra uma análise detalhada dos eventos que ocorreram durante o processo de leituras dos sensores.

As informações descritas no *log* correspondem a:

- Total de falhas injetadas: quantidade total de falhas injetadas durante a execução do programa;
- Chamadas de AVG: quantidade de vezes que o método *avg* foi invocado;
  - Amostras sem falha(s): número de vezes em que todos os elementos da amostra estavam dentro da faixa aceitável (todos os dados corretos);
  - Amostras com falha(s): quantidade de vezes em que no mínimo um elemento na amostra estava acima da variação máxima, ou seja, fora da faixa aceitável. A soma desse parâmetro com o anterior equivale à quantidade de vezes que o método foi invocado;
    - Acertos: número total de vezes que o método *avg* tolerou falhas, seja na primeira, segunda ou terceira tentativa. Os casos em que todos os elementos da amostra estão dentro da faixa aceitável não entram nessa contagem, pois nessas situações, o método *avg* não teve interferência no resultado;
    - Erros: quantidade total de vezes que o método *avg* não encontrou uma média confiável.
- AVG (1x): quantidade de vezes que o método *avg* precisou ser chamado apenas uma vez, seja porque o mesmo tolerou falhas ou porque todos os elementos da amostra estavam dentro da faixa aceitável.
  - Real AVG (1x): número de vezes que o método *avg* realmente tolerou falhas. Os casos em que todos os elementos da amostra estão dentro da variação máxima não

entram nessa contagem, pois nessas situações, o método *avg* não teve interferência no resultado;

- Erro AVG (1x): número de vezes que o método *avg* não calculou uma média confiável na primeira tentativa;

- AVG (2x): número de vezes que a execução do método *avg* pela segunda vez (redundância de processamento) trouxe uma média confiável;

  - Real AVG (2x): número de vezes em que o método *avg* tolerou falhas em sua segunda tentativa. No exemplo da Figura 4.1, das 9 vezes em que a segunda tentativa do método *avg* forneceu uma média confiável, apenas em 3 a amostra continha elemento fora da faixa. No restante, todos os elementos estavam dentro da variação máxima definida;
  - Erro AVG (2x): número de vezes que o método *avg* não conseguiu calcular uma média confiável na segunda tentativa;

- AVG (3x): número de vezes que a execução do método *avg* pela terceira vez (redundância de processamento) trouxe uma média confiável;

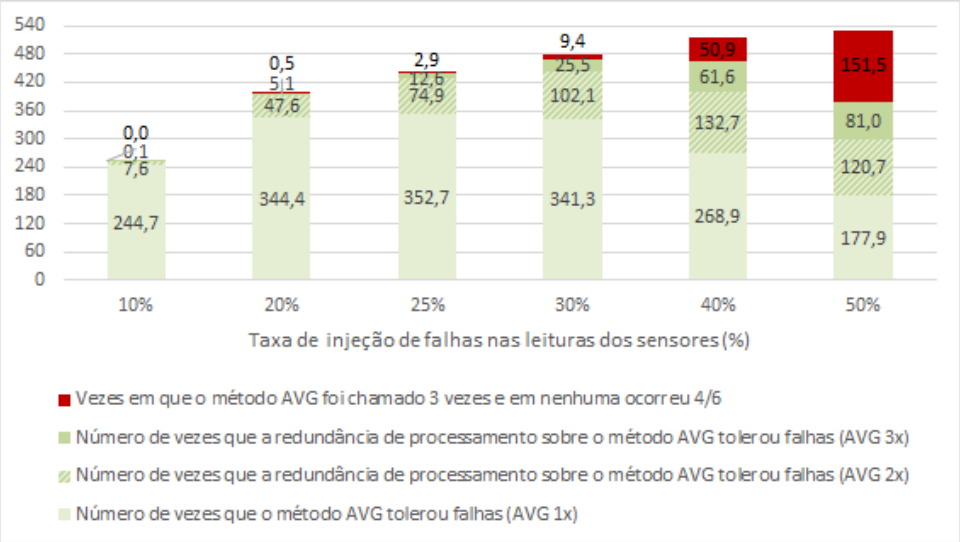
  - Real AVG (3x): número de vezes em que o método *avg* realmente tolerou falhas em sua terceira tentativa;
  - Erro AVG (3x): número de vezes que o método *avg* não calculou uma média confiável na terceira tentativa, gerando um erro no programa;

- Total de tolerâncias: número total de falhas toleradas pelo método *avg* juntamente com a redundância de processamento sobre ele.

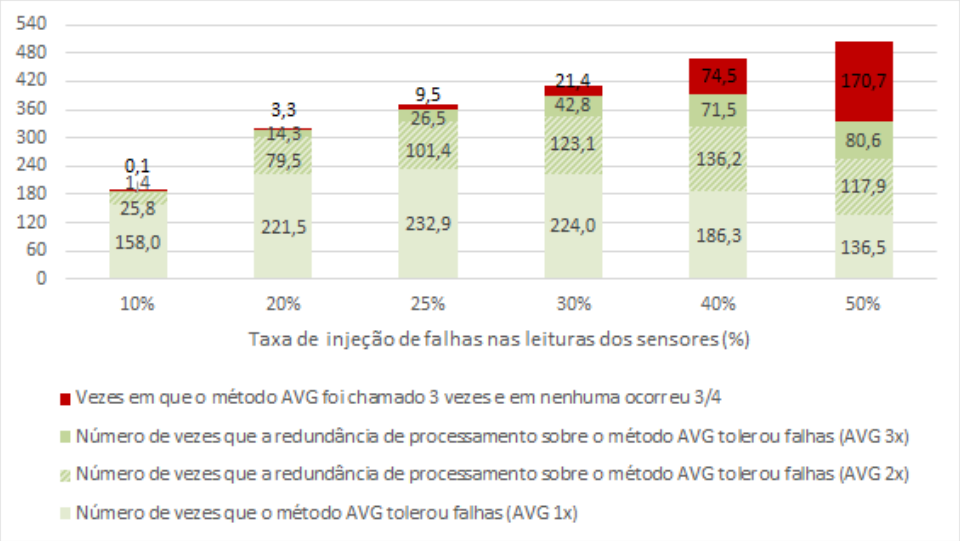
- Total de erros: número de vezes que o método *avg*, mesmo com as 3 tentativas, não conseguiu calcular uma média confiável;

- Tolerância + erros: soma do número de vezes que o método *avg* com a redundância de processamento tolerou falhas, mais o número de erros ocorridos;

As colunas do gráfico da Figura 4.2a e da Figura 4.2b contém os valores médios obtidos de 30 testes individuais, cada um durou 60 ciclos de leitura. Os valores exibidos nas colunas empilhadas correspondem ao número de vezes que o método *avg* tolerou falhas, ao número de vezes que a redundância de processamento tolerou falhas, e a quantidade total de erros ocorridos durante a execução do programa.



(a) Percentual de leituras dentro da faixa aceitável: 66,6% (4/6).



(b) Percentual de leituras dentro da faixa aceitável: 75% (3/4)

Figura 4.2: Efetividade do método *avg* às leituras com falhas. Os dados são referentes aos valores médios de 30 testes, cada realizou 540 leituras nos sensores. O número de falhas aumentou conforme a porcentagem de falhas injetadas nas leituras dos sensores. Os resultados individuais dos 30 testes estão no Apêndice A.

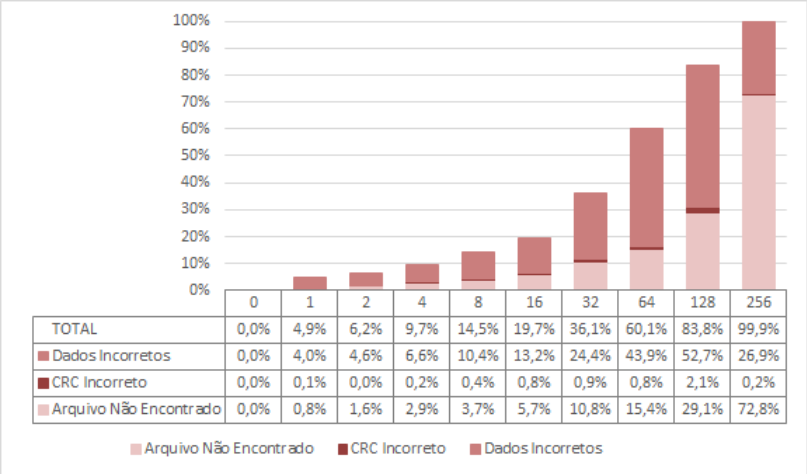
4.2 Testes com Injeção de Falhas nas Regiões de Memória

Esta seção mostra os resultados dos testes que utilizaram a biblioteca *FaultInjection* para alterar os dados nas regiões de memória e simular os efeitos do fenômeno *bit-flip* sobre o equipamento. Na Subseção 4.2.1 mostra-se os resultados dos testes com a injeção de falhas na região de memória de dados, e na Subseção 4.2.2, os resultados com a injeção de falhas nas regiões de memória de dados e memória interna. Como mencionado na Subseção 3.1.3, uma vez que o teste é executado, cada ciclo de leitura pode resultar em 3 tipos de defeitos: "dados não encontrados", "CRC incorreto" e "dados incorretos". Os resultados são representados

por gráficos de colunas empilhadas, mostrados da Figura 4.3 à Figura 4.10 e seus dados correspondem aos valores médios de um conjunto de 30 testes. Os resultados individuais de cada teste estão no Apêndice B deste trabalho. Nas Tabelas 4.1 à 4.8 mostram-se as médias de cada defeito, mais um intervalo de confiança, calculado para um nível de confiança de 99%.

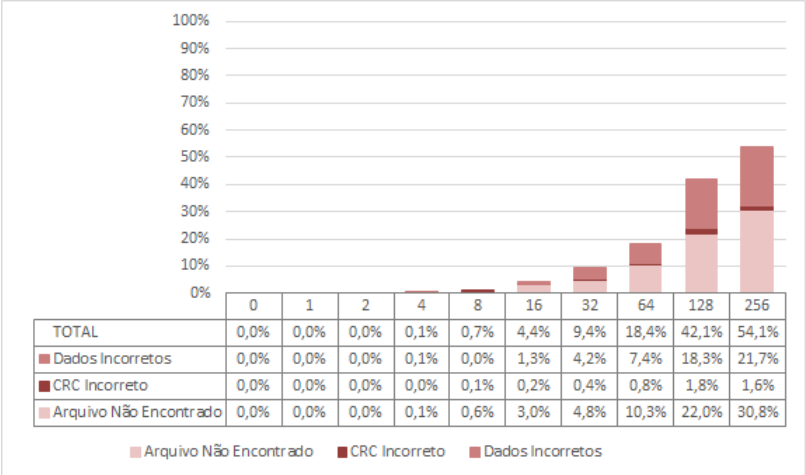
4.2.1 Injeção de Falhas na Região de Memória de Dados

Os resultados dos testes com injeção de falhas na região de memória de dados do microcontrolador são mostrados na Figura 4.3 e na Figura 4.4.



	0	1	2	4	8	16	32	64	128	256
TOTAL	0,0%	4,9%	6,2%	9,7%	14,5%	19,7%	36,1%	60,1%	83,8%	99,9%
Dados Incorretos	0,0%	4,0%	4,6%	6,6%	10,4%	13,2%	24,4%	43,9%	52,7%	26,9%
CRC Incorreto	0,0%	0,1%	0,0%	0,2%	0,4%	0,8%	0,9%	0,8%	2,1%	0,2%
Arquivo Não Encontrado	0,0%	0,8%	1,6%	2,9%	3,7%	5,7%	10,8%	15,4%	29,1%	72,8%

Figura 4.3: Resultados dos testes com o *firmware* sem técnicas de tolerância a falhas, com a injeção de falhas na região de memória de dados. Os dados deste gráfico correspondem aos valores médios de um conjunto de 30 testes. Os resultados individuais estão no Apêndice B.



	0	1	2	4	8	16	32	64	128	256
TOTAL	0,0%	0,0%	0,0%	0,1%	0,7%	4,4%	9,4%	18,4%	42,1%	54,1%
Dados Incorretos	0,0%	0,0%	0,0%	0,1%	0,0%	1,3%	4,2%	7,4%	18,3%	21,7%
CRC Incorreto	0,0%	0,0%	0,0%	0,0%	0,1%	0,2%	0,4%	0,8%	1,8%	1,6%
Arquivo Não Encontrado	0,0%	0,0%	0,0%	0,1%	0,6%	3,0%	4,8%	10,3%	22,0%	30,8%

Figura 4.4: Resultados dos testes com o *firmware* com técnicas de tolerância a falhas, com a injeção de falhas na região de memória de dados. Os dados deste gráfico correspondem aos valores médios de um conjunto de 30 testes. Os resultados individuais estão no Apêndice B.

Os resultados apontam que o *firmware* com tolerância a falhas apresentou, nos testes com até 8 falhas injetadas por ciclo de leitura, menos de um por cento de defeitos, sendo que nos testes com até duas falhas injetadas, os defeitos não ocorreram. O primeiro defeito só ocorreu a partir do teste com 4 falhas injetadas, e mesmo assim, esse número foi quase nulo (menos de 0,1%), conforme mostra-se na Figura 4.4. Os defeitos somente apresentaram valores significativos a partir da injeção de 16 (4,4%) e 32 falhas (9,6%). Em contrapartida, o *firmware* não tolerante a falhas não escapou de defeitos em nenhum dos testes (Figura 4.3). Com a injeção de duas falhas por ciclo de leitura (que não gerou nenhum defeito no primeiro), o número de defeitos foi em média, 6,2%. Vale lembrar que as falhas são injetadas a cada ciclo de leitura da estação meteorológica.

Nos testes acima de 16 falhas, o número de defeitos do *firmware* tolerante a falhas também se manteve menor em relação ao outro, mas não na mesma proporção dos testes anteriores com menores quantidades de falhas. A partir de 32 e 64 falhas injetadas, o número de defeitos no primeiro se manteve entre três a quatro vezes menor, e nos casos mais extremos (128 e 256 falhas), o número de defeitos manteve-se aproximadamente pela metade. O número de injeção de falhas por ciclo de leitura limitou-se em 256, pois a partir desse número o *firmware* sem tolerância travou completamente em todos os testes, não respondendo sequer ao comando de *reset* do *firmware* monitor, não havendo, portanto, mais possibilidade de comparação.

Com o *firmware* sem tolerância a falhas, o tipo de defeito mais frequente foi o de dados incorretos, com exceção aos testes com 128 e 256 falhas injetadas, isso porque os testes com essas quantidades excessivas de falhas provocaram constantemente o travamento do dispositivo, não gerando o arquivo binário com os dados das leituras. Os defeitos de CRC incorreto foram encontrados poucas vezes, possivelmente porque os dados usados para se gerar o código CRC são mantidos por pouco tempo na memória.

Nas Tabelas 4.1 e 4.2 pode-se observar a média de defeitos com os intervalo de confiança e o tempo total dos testes. O tempo dos testes com o *firmware* sem tolerância a falhas variou de quase duas horas e meia a aproximadamente seis horas e quinze minutos, e com o *firmware* tolerante a falhas, o tempo variou de quase cinco horas a nove horas e 24 minutos. Os tempos dos testes com o segundo *firmware* foram maiores, porque o intervalo entre as leituras foi ajustado de 5 para 10 segundos, em razão do tempo adicional de processamento. Testes com maior número de defeitos consomem mais tempo, pois quando um defeito é encontrado, o *firmware monitor*, executado em outro módulo mbed, reinicia a estação meteorológica, e o programa monitor de testes, executado no computador, fica parado esperando o tempo necessário para a estação meteorológica reiniciar e estabilizar todos os sensores.

As primeiras versões do *firmware* tolerante a falhas não apresentaram o mesmo desempenho mostrado aqui, pois os testes com uma ou duas falhas injetadas exclusivamente na região de memória de dados provocavam alguns defeitos de "arquivo não encontrado". Por esse motivo, foi necessário identificar todas as vulnerabilidades do código, buscando trechos em que uma única falha poderia destruir um dado não redundante. A redundância de dados foi então implementada não somente nas regiões que armazenavam os dados das leituras, mas também nos endereços de memória que armazenavam variáveis de configuração do programa, como por exemplo, o tempo de intervalo entre as leituras, a quantidade de sensores, o tempo do *watchdog*, e outras. A alteração do dado que guarda o tempo de intervalo entre as leituras pode fazer com que a estação meteorológica não faça as leituras no tempo correto.

Quantidade de falhas	**Arquivo Não Encontrado**	**CRC Incorreto**	**Dados Incorretos**	**Total de Defeitos**	**Tempo Total**
0	0	0	0	0	02:28:56
1	$0,83 \pm 0,57$	$0,06 \pm 0,14$	$4,00 \pm 1,12$	$4,89 \pm 1,50$	02:38:39
2	$1,61 \pm 0,78$	0	$4,56 \pm 1,22$	$6,17 \pm 1,78$	02:41:32
4	$2,94 \pm 0,84$	$0,17 \pm 0,24$	$6,61 \pm 1,36$	$9,72 \pm 1,85$	02:50:27
8	$3,72 \pm 1,02$	$0,39 \pm 0,40$	$10,39 \pm 1,30$	$14,50 \pm 2,15$	03:01:48
16	$5,67 \pm 1,26$	$0,83 \pm 0,61$	$13,17 \pm 1,73$	$19,67 \pm 2,85$	03:11:06
32	$10,78 \pm 2,66$	$0,89 \pm 0,57$	$24,39 \pm 3,33$	$36,06 \pm 4,20$	03:49:18
64	$15,44 \pm 3,13$	$0,78 \pm 0,57$	$43,89 \pm 5,98$	$60,11 \pm 7,53$	04:48:50
128	$29,06 \pm 9,14$	$2,06 \pm 1,38$	$52,67 \pm 7,45$	$83,78 \pm 5,81$	05:37:05
256	$72,78 \pm 5,40$	$0,22 \pm 0,34$	$26,89 \pm 5,27$	$99,89 \pm 0,20$	06:14:05

Tabela 4.1: Resultados dos testes com o *firmware* sem técnicas de tolerância a falhas, com a injeção de falhas na região de memória de dados. Os dados desta tabela correspondem aos valores médios de um conjunto de 30 testes. Os resultados individuais estão no Apêndice B.

Quantidade de falhas	**Arquivo Não Encontrado**	**CRC Incorreto**	**Dados Incorretos**	**Total de Defeitos**	**Tempo Total**
0	0	0	0	0	04:57:28
1	0	0	0	0	04:57:54
2	0	0	0	0	04:57:49
4	$0,06 \pm 0,14$	0	$0,06 \pm 0,14$	$0,11 \pm 0,20$	04:57:26
8	$0,61 \pm 0,48$	$0,06 \pm 0,14$	0	$0,67 \pm 0,49$	05:00:10
16	$3 \pm 1,22$	$0,17 \pm 0,24$	$1,28 \pm 0,87$	$4,44 \pm 1,27$	05:22:24
32	$4,78 \pm 1,45$	$0,44 \pm 0,35$	$4,17 \pm 1,25$	$9,39 \pm 2,47$	05:29:23
64	$10,28 \pm 2,14$	$0,78 \pm 0,53$	$7,39 \pm 1,99$	$18,44 \pm 2,56$	06:19:00
128	$22,00 \pm 4,34$	$1,78 \pm 0,94$	$18,28 \pm 3,05$	$42,06 \pm 4,44$	07:44:02
256	$30,83 \pm 5,39$	$1,56 \pm 0,71$	$21,72 \pm 3,38$	$54,11 \pm 5,92$	09:21:24

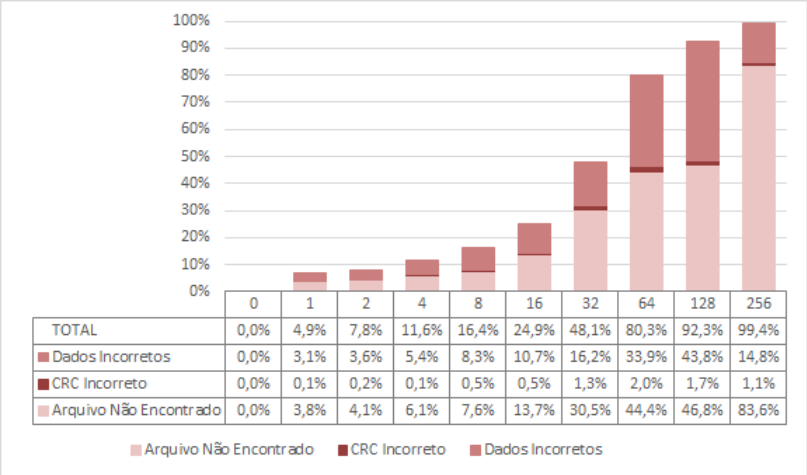
Tabela 4.2: Resultados dos testes com o *firmware* com técnicas de tolerância a falhas, com a injeção de falhas na região de memória de dados. Os dados deste gráfico correspondem aos valores médios de um conjunto de 30 testes. Os resultados individuais estão no Apêndice B.

Quanto mais se aprimorou isso, mais satisfatório se tornaram os resultados. O sistema de votação também precisou ser melhorado. Nos primeiros testes, o sistema de votação apenas obtinha o resultado da saída conforme a maioria. Porém, com a contínua injeção de falhas, todas as cópias eram afetadas, tornando-o ineficiente, gerando muitos erros. A modificação

realizada consistiu em corrigir a cópia com falha, caso possível, no momento da votação, isto é, se duas das três cópias mostrarem consenso e a terceira um valor divergente, a terceira é atualizada com o valor das outras duas, deixando todas as cópias com o mesmo valor ao final da votação. Com essa alteração, os resultados mostraram um desempenho superior, uma vez que os erros são rapidamente corrigidos quando a variável é constantemente acessada.

4.2.2 Injeção de Falhas na Região de Memória de Dados e Memória Interna

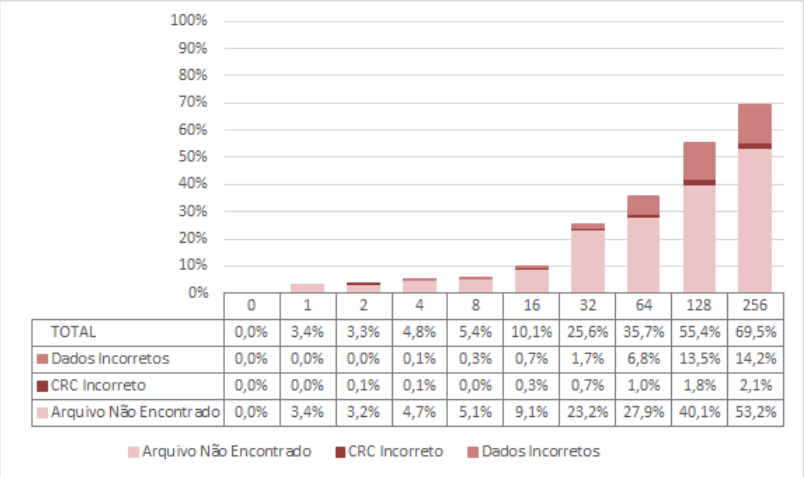
Os resultados dos testes com a injeção de falhas nas regiões de memória de dados e memória interna são exibidos na Figura 4.5 e na Figura 4.6.



	0	1	2	4	8	16	32	64	128	256
TOTAL	0,0%	4,9%	7,8%	11,6%	16,4%	24,9%	48,1%	80,3%	92,3%	99,4%
Dados Incorretos	0,0%	3,1%	3,6%	5,4%	8,3%	10,7%	16,2%	33,9%	43,8%	14,8%
CRC Incorreto	0,0%	0,1%	0,2%	0,1%	0,5%	0,5%	1,3%	2,0%	1,7%	1,1%
Arquivo Não Encontrado	0,0%	3,8%	4,1%	6,1%	7,6%	13,7%	30,5%	44,4%	46,8%	83,6%

Figura 4.5: Resultados dos testes com o *firmware* sem técnicas de tolerância a falhas, com a injeção de falhas na região de memória de dados e memória interna. Os dados deste gráfico correspondem aos valores médios de um conjunto de 30 testes. Os resultados individuais estão no Apêndice B.

Nos testes com injeção de falhas nas duas regiões de memória, cujos resultados são mostrados na Figura 4.5 e na Figura 4.6, os defeitos mais frequentes foram os do tipo "arquivo não encontrado", pois as falhas injetadas na região de memória interna do microcontrolador, geralmente provocaram o travamento do dispositivo. Nenhuma estratégia de redundância foi utilizada para tolerar falhas na região de memória interna do microcontrolador, pois, por ser uma plataforma de prototipagem rápida proprietária, o fabricante disponibiliza poucas informações de como essa região de memória é usada pelo microcontrolador. Entretanto, os testes com o *firmware* tolerante a falhas, em média, apresentaram menos erros em todas as categorias. No teste com 32 falhas injetadas, por exemplo, o número de defeitos do tipo "arquivos não encontrados" foi 30,5% no *firmware* sem tolerância e 23,2% no com tolerância. O motivo disso é que falhas injetadas na região de memória de dados também geram defeitos de arquivo não encontrado. Portanto, o emprego de redundâncias na região de memória de dados contribuiu para essa redução. Os erros de "dados incorretos" também foram menores no *firmware* com tolerância a falhas. Nesse mesmo teste com a injeção de 32 falhas por ciclo de leitura, o número de erros do sistema não tolerante foi de 16,2%. Em comparação ao teste com o mesmo *firmware* que injetou falhas apenas na região de memória de dados, esse número é menor (24,4% contra 16,2%) pois a probabilidade das falhas atingirem os dados do



	0	1	2	4	8	16	32	64	128	256
TOTAL	0,0%	3,4%	3,3%	4,8%	5,4%	10,1%	25,6%	35,7%	55,4%	69,5%
Dados Incorretos	0,0%	0,0%	0,0%	0,1%	0,3%	0,7%	1,7%	6,8%	13,5%	14,2%
CRC Incorreto	0,0%	0,0%	0,1%	0,1%	0,0%	0,3%	0,7%	1,0%	1,8%	2,1%
Arquivo Não Encontrado	0,0%	3,4%	3,2%	4,7%	5,1%	9,1%	23,2%	27,9%	40,1%	53,2%

Figura 4.6: Resultados dos testes com o *firmware* com técnicas de tolerância a falhas, com a injeção de falhas na região de memória de dados e memória interna. Os dados deste gráfico correspondem aos valores médios de um conjunto de 30 testes. Os resultados individuais estão no Apêndice B.

programa diminuiu em 50% com o aumento das regiões para a injeção de falhas (de 32 kB foi para 64 kB). Fazendo a comparação da quantidade de defeitos do tipo "dados incorretos" entre os *firmwares* com e sem tolerância a falhas, o desempenho do tolerante reduziu de 16,2% para 1,7%.

Na Tabela 4.3 e na Tabela 4.4 exibe-se, respectivamente, os resultados dos testes com o *firmware* sem tolerância a falhas e com tolerância a falhas, juntamente a um intervalo de confiança e o tempo total dos testes.

Quantidade de falhas	**Arquivo Não Encontrado**	**CRC Incorreto**	**Dados Incorretos**	**Total de Defeitos**	**Tempo Total**
0	0	0	0	0	02:28:56
1	$3,78 \pm 1,43$	$0,11 \pm 0,20$	$3,06 \pm 0,97$	$6,94 \pm 1,50$	02:46:04
2	$4,11 \pm 1,35$	$0,17 \pm 0,24$	$3,56 \pm 1,28$	$7,83 \pm 1,92$	02:46:07
4	$6,06 \pm 1,82$	$0,11 \pm 0,20$	$5,44 \pm 1,72$	$11,61 \pm 2,38$	02:54:58
8	$7,61 \pm 1,62$	$0,50 \pm 0,47$	$8,33 \pm 1,99$	$16,44 \pm 2,26$	03:05:35
16	$13,67 \pm 1,81$	$0,50 \pm 0,47$	$10,72 \pm 2,29$	$24,89 \pm 2,37$	03:24:50
32	$30,50 \pm 4,06$	$1,33 \pm 0,86$	$16,22 \pm 2,92$	$48,06 \pm 5,48$	04:19:14
64	$44,39 \pm 5,29$	$2 \pm 0,75$	$33,89 \pm 5,88$	$80,28 \pm 2,59$	05:27:45
128	$46,83 \pm 6,74$	$1,72 \pm 0,73$	$43,78 \pm 6,65$	$92,33 \pm 2,88$	05:56:14
256	$83,61 \pm 3,10$	$1,06 \pm 0,81$	$14,78 \pm 2,46$	$99,44 \pm 1,43$	06:16:21

Tabela 4.3: Resultados dos testes com o *firmware* sem técnicas de tolerância a falhas, com a injeção de falhas na região de memória de dados e memória interna. Os dados desta tabela correspondem aos valores médios de um conjunto de 30 testes. Os resultados individuais estão no Apêndice B.

Quantidade de falhas	**Arquivo Não Encontrado**	**CRC Incorreto**	**Dados Incorretos**	**Total de Defeitos**	**Tempo Total**
0	0	0	0	0	04:57:28
1	$3,39 \pm 1,31$	0	0	$3,39 \pm 1,31$	05:10:11
2	$3,22 \pm 1,16$	$0,06 \pm 0,14$	0	$3,28 \pm 1,23$	05:10:37
4	$4,67 \pm 1,49$	$0,06 \pm 0,14$	$0,06 \pm 0,14$	$4,78 \pm 1,54$	05:19:48
8	$5,11 \pm 1,94$	0	$0,28 \pm 0,36$	$5,39 \pm 2,02$	05:28:04
16	$9,11 \pm 1,94$	$0,33 \pm 0,38$	$0,67 \pm 0,64$	$10,11 \pm 1,85$	05:49:00
32	$23,22 \pm 3,60$	$0,67 \pm 0,53$	$1,72 \pm 0,95$	$25,61 \pm 3,33$	06:52:54
64	$27,94 \pm 4,34$	$1 \pm 0,53$	$6,78 \pm 2,43$	$35,72 \pm 4,05$	07:38:56
128	$40,11 \pm 6,07$	$1,83 \pm 0,86$	$13,50 \pm 2,79$	$55,44 \pm 5,68$	09:05:44
256	$53,17 \pm 4,71$	$2,11 \pm 1,07$	$14,22 \pm 3,49$	$69,50 \pm 5,26$	10:05:25

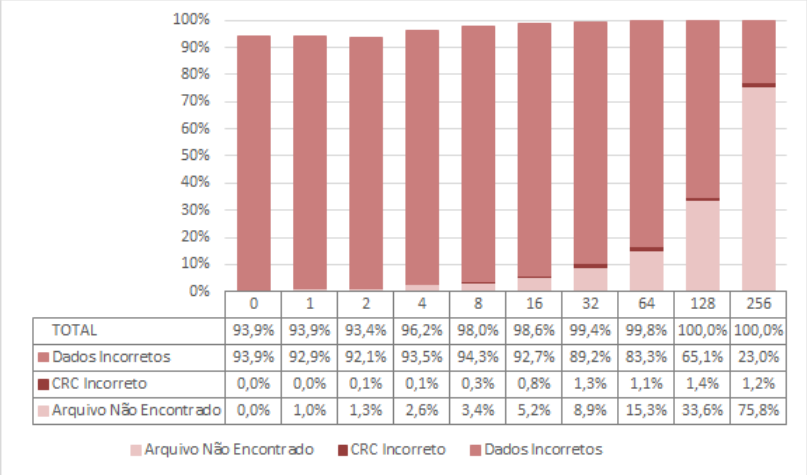
Tabela 4.4: Resultados dos testes com o *firmware* com técnicas de tolerância a falhas, com a injeção de falhas na região de memória de dados e memória interna. Os dados desta tabela correspondem aos valores médios de um conjunto de 30 testes. Os resultados individuais estão no Apêndice B.

4.3 Testes com Injeção de Falhas nas Leituras dos Sensores e nas Regiões de Memória

Os testes que injetaram falhas nas regiões de memória do microcontrolador em conjunto com as injeções de falhas nas leituras dos sensores, em geral, mostraram um aumento excessivo dos defeitos do tipo "dados incorretos" no *firmware* sem tolerância a falhas. A injeção de falhas nas leituras dos sensores atingiram 25% das leituras. Na Subseção 4.3.1 exibe-se os resultados dos testes com injeção de falhas na região de memória de dados do microcontrolador, e na Subseção 4.3.2, os testes com injeção de falhas nas regiões de memória de dados e memória interna.

4.3.1 Injeção de Falhas na Região de Memória de Dados

Na Figura 4.7 mostra-se os resultados dos testes com o *firmware* sem tolerância a falhas. Nestes, injetou-se falhas nas leituras dos sensores e na região de memória de dados do microcontrolador.



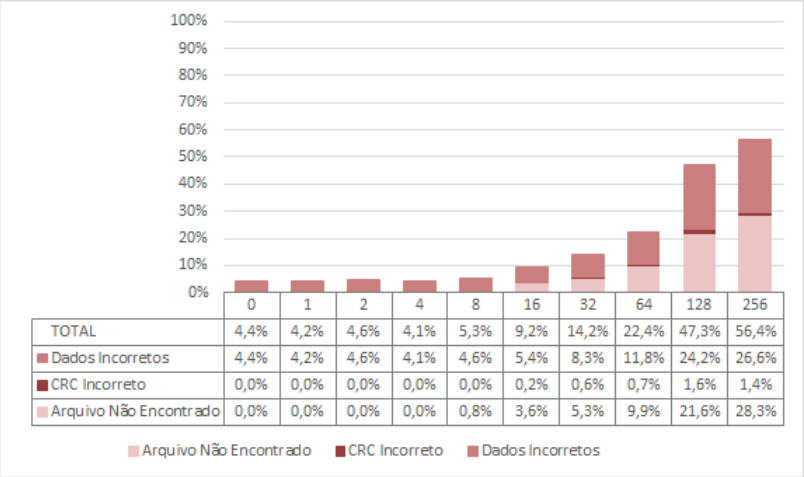
	0	1	2	4	8	16	32	64	128	256
TOTAL	93,9%	93,9%	93,4%	96,2%	98,0%	98,6%	99,4%	99,8%	100,0%	100,0%
Dados Incorretos	93,9%	92,9%	92,1%	93,5%	94,3%	92,7%	89,2%	83,3%	65,1%	23,0%
CRC Incorreto	0,0%	0,0%	0,1%	0,1%	0,3%	0,8%	1,3%	1,1%	1,4%	1,2%
Arquivo Não Encontrado	0,0%	1,0%	1,3%	2,6%	3,4%	5,2%	8,9%	15,3%	33,6%	75,8%

Figura 4.7: Resultados dos testes com o *firmware* sem técnicas de tolerância a falhas, com a injeção de falhas nas leituras dos sensores e na região de memória de dados. Os dados deste gráfico correspondem aos valores médios de um conjunto de 30 testes. Os resultados individuais estão no Apêndice B.

Conforme é mostrado na Figura 4.7, as injeções de falhas nas leituras dos sensores causaram grande quantidade de defeitos no *firmware* sem tolerância a falhas. Comparado ao teste exibido na Figura 4.3, que injetou falhas exclusivamente na região de memória de dados, identificou-se que as injeções de falhas nas leituras dos sensores aumentaram em aproximadamente 90% o número de defeitos. No teste com apenas uma falha injetada na memória e 25% de falhas nas leituras dos sensores, o número de defeitos aumentou de 4,9% para 93,9%. Todos os testes mostrados neste gráfico apresentaram um número de defeitos acima de 93%, e o defeito mais comum foi o de "dados incorretos". Cabe observar, que quanto maior as injeções de falhas nas regiões de memória, menor foi a quantidade de defeitos do tipo "dados incorretos", isto porque as falhas nas regiões de memória começaram a travar ou

causar um funcionamento incorreto do programa, provocando mais defeitos de "arquivo não encontrado". No teste que injetou 256 falhas por ciclo de leitura, por exemplo, o número de defeitos do tipo arquivo "não encontrado" foi de 75% e o de "dados incorretos", 23%.

Na Figura 4.8 mostra-se os resultados dos testes com o *firmware* tolerante a falhas. Nestes, injetou-se falhas nas leituras dos sensores e na região de memória de dados do microcontrolador.



	0	1	2	4	8	16	32	64	128	256
TOTAL	4,4%	4,2%	4,6%	4,1%	5,3%	9,2%	14,2%	22,4%	47,3%	56,4%
Dados Incorretos	4,4%	4,2%	4,6%	4,1%	4,6%	5,4%	8,3%	11,8%	24,2%	26,6%
CRC Incorreto	0,0%	0,0%	0,0%	0,0%	0,0%	0,2%	0,6%	0,7%	1,6%	1,4%
Arquivo Não Encontrado	0,0%	0,0%	0,0%	0,0%	0,8%	3,6%	5,3%	9,9%	21,6%	28,3%

Figura 4.8: Resultados dos testes com o *firmware* com técnicas de tolerância a falhas, com a injeção de falhas nas leituras dos sensores e na região de memória de dados. Os dados deste gráfico correspondem aos valores médios de um conjunto de 30 testes. Os resultados individuais estão no Apêndice B.

Os testes com o *firmware* tolerante a falhas, cujos resultados são mostrados na Figura 4.8, em todos os casos, apresentaram menor quantidade de defeitos comparado ao *firmware* sem tolerância a falhas. Fazendo uma comparação entre eles, no teste com uma única falha injetada a cada ciclo de leitura e 25% de falhas nas leituras dos sensores, este apresentou 4,2% de defeitos, enquanto o *firmware* não tolerante, uma média de 93,9%. O tipo de defeito mais evitado foi o de "dados incorretos", graças a eficiência do método *avg* com a redundância de processamento. Este mesmo teste, se comparado aos da Figura 4.4 (que utilizando o mesmo *firmware*, injetou falhas apenas na região de memória de dados), o número de defeitos foi muito próximo, exceto os de arquivo incorreto, que foram aproximadamente 5% superiores com a injeção de falhas nos sensores.

Na Tabela 4.5 e na Tabela 4.6 são exibidas, respectivamente, as médias de defeitos com os intervalos de confiança obtidos nos testes com o *firmware* não tolerante a falhas e do *firmware* com tolerância a falhas.

Quantidade de falhas	**Arquivo Não Encontrado**	**CRC Incorreto**	**Dados Incorretos**	**Total de Defeitos**	**Tempo Total**
0	0	0	$93,94 \pm 1,98$	$93,94 \pm 1,98$	06:04:52
1	$1 \pm 0,76$	0	$92,94 \pm 2,18$	$93,94 \pm 2,24$	06:02:35
2	$1,28 \pm 0,84$	$0,06 \pm 0,14$	$92,06 \pm 1,67$	$93,39 \pm 1,64$	05:58:32
4	$2,56 \pm 1,04$	$0,11 \pm 0,20$	$93,5 \pm 1,93$	$96,17 \pm 1,85$	06:02:32
8	$3,44 \pm 1,07$	$0,28 \pm 0,36$	$94,28 \pm 1,31$	$98 \pm 0,83$	06:07:11
16	$5,17 \pm 1,6$	$0,78 \pm 0,49$	$92,67 \pm 1,7$	$98,61 \pm 0,99$	06:10:52
32	$8,89 \pm 2,28$	$1,33 \pm 0,56$	$89,17 \pm 2,32$	$99,39 \pm 0,6$	06:16:01
64	$15,33 \pm 3,25$	$1,11 \pm 0,59$	$83,33 \pm 3$	$99,78 \pm 0,34$	06:14:19
128	$33,56 \pm 8,98$	$1,39 \pm 0,8$	$65,06 \pm 8,62$	100	06:13:27
256	$75,78 \pm 4,97$	$1,22 \pm 0,94$	$23 \pm 4,26$	100	06:15:04

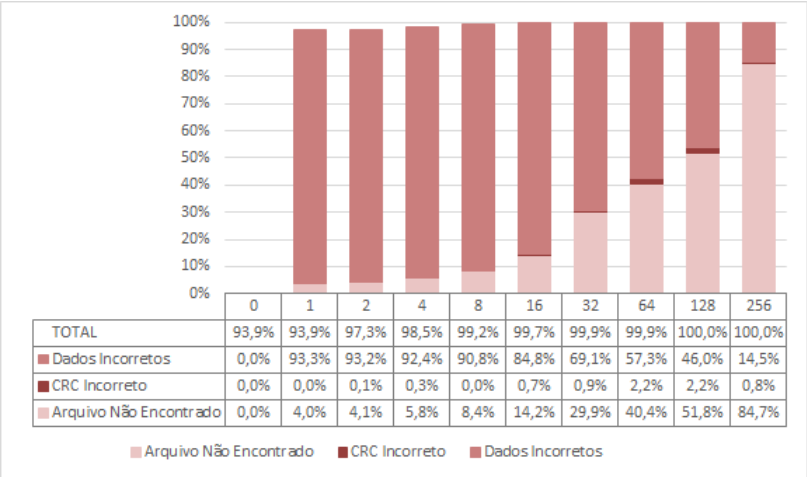
Tabela 4.5: Resultados dos testes com o *firmware* sem técnicas de tolerância a falhas, com a injeção de falhas nas leituras dos sensores e na região de memória de dados. Os dados desta tabela correspondem aos valores médios de um conjunto de 30 testes. Os resultados individuais estão no Apêndice B.

Quantidade de falhas	**Arquivo Não Encontrado**	**CRC Incorreto**	**Dados Incorretos**	**Total de Defeitos**	**Tempo Total**
0	0	0	$4,39 \pm 1,37$	$4,39 \pm 1,37$	05:14:24
1	0	0	$4,17 \pm 1,31$	$4,17 \pm 1,31$	05:11:09
2	0	0	$4,61 \pm 1,16$	$4,61 \pm 1,16$	05:12:30
4	0	0	$4,11 \pm 1,16$	$4,11 \pm 1,16$	05:13:07
8	$0,78 \pm 0,57$	0	$4,56 \pm 1,18$	$5,33 \pm 1,42$	05:23:13
16	$3,61 \pm 1,43$	$0,22 \pm 0,27$	$5,39 \pm 1,69$	$9,22 \pm 2,23$	05:49:21
32	$5,28 \pm 1,20$	$0,61 \pm 0,52$	$8,28 \pm 1,91$	$14,17 \pm 2,34$	06:09:10
64	$9,94 \pm 2,3$	$0,67 \pm 0,53$	$11,83 \pm 2,2$	$22,44 \pm 2,74$	06:40:36
128	$21,61 \pm 4,86$	$1,56 \pm 0,74$	$24,17 \pm 3,32$	$47,33 \pm 4,37$	08:31:24
256	$28,33 \pm 4,38$	$1,44 \pm 0,76$	$26,61 \pm 3,52$	$56,39 \pm 5,02$	09:12:55

Tabela 4.6: Resultados dos testes com o *firmware* com técnicas de tolerância a falhas, com a injeção de falhas nas leituras dos sensores e na região de memória de dados. Os dados desta tabela correspondem aos valores médios de um conjunto de 30 testes. Os resultados individuais estão no Apêndice B.

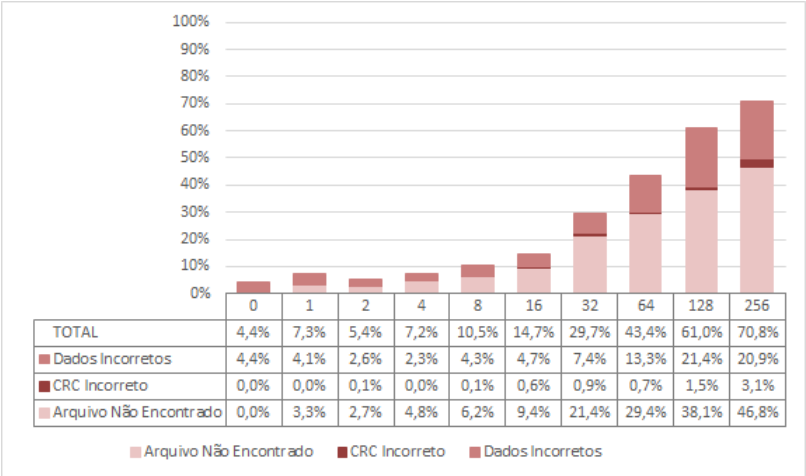
4.3.2 Injeção de Falhas na Região de Memória de Dados e Memória Interna

Na Figura 4.9 e na Figura 4.10 são exibidos os resultados dos testes com a injeção de falhas nas leituras dos sensores, em conjunto com a injeção de falhas na região de memória de dados e memória interna. Na primeira (Figura 4.9), os testes foram executados com o *firmware* sem tolerância a falhas e na segunda, utilizou-se o *firmware* com tolerância. A Tabela 4.7 e a Tabela 4.8 complementam esses resultados exibindo os intervalos de confiança e o tempo total de cada teste.



	0	1	2	4	8	16	32	64	128	256
TOTAL	93,9%	93,9%	97,3%	98,5%	99,2%	99,7%	99,9%	99,9%	100,0%	100,0%
Dados Incorretos	0,0%	93,3%	93,2%	92,4%	90,8%	84,8%	69,1%	57,3%	46,0%	14,5%
CRC Incorreto	0,0%	0,0%	0,1%	0,3%	0,0%	0,7%	0,9%	2,2%	2,2%	0,8%
Arquivo Não Encontrado	0,0%	4,0%	4,1%	5,8%	8,4%	14,2%	29,9%	40,4%	51,8%	84,7%

Figura 4.9: Resultados dos testes com o *firmware* sem técnicas de tolerância a falhas, com a injeção de falhas nas leituras dos sensores e nas regiões de memória de dados e memória interna. Os dados deste gráfico correspondem aos valores médios de um conjunto de 30 testes. Os resultados individuais estão no Apêndice B.



	0	1	2	4	8	16	32	64	128	256
TOTAL	4,4%	7,3%	5,4%	7,2%	10,5%	14,7%	29,7%	43,4%	61,0%	70,8%
Dados Incorretos	4,4%	4,1%	2,6%	2,3%	4,3%	4,7%	7,4%	13,3%	21,4%	20,9%
CRC Incorreto	0,0%	0,0%	0,1%	0,0%	0,1%	0,6%	0,9%	0,7%	1,5%	3,1%
Arquivo Não Encontrado	0,0%	3,3%	2,7%	4,8%	6,2%	9,4%	21,4%	29,4%	38,1%	46,8%

Figura 4.10: Resultados dos testes com o *firmware* com técnicas de tolerância a falhas, com a injeção de falhas nas leituras dos sensores e nas regiões de memória de dados e memória interna. Os dados deste gráfico correspondem aos valores médios de um conjunto de 30 testes. Os resultados individuais estão no Apêndice B.

Quantidade de falhas	**Arquivo Não Encontrado**	**CRC Incorreto**	**Dados Incorretos**	**Total de Defeitos**	**Tempo Total**
0	0	0	$93,94 \pm 1,98$	$93,94 \pm 1,98$	06:04:52
1	$4 \pm 1,46$	0	$93,28 \pm 1,59$	$97,28 \pm 1,48$	06:03:13
2	$4,06 \pm 1,51$	$0,06 \pm 0,14$	$93,22 \pm 1,54$	$97,33 \pm 1,48$	06:05:11
4	$5,78 \pm 1,73$	$0,28 \pm 0,3$	$92,44 \pm 1,41$	$98,5 \pm 0,97$	06:09:45
8	$8,44 \pm 2,01$	0	$90,78 \pm 1,79$	$99,22 \pm 0,57$	06:08:54
16	$14,17 \pm 2,38$	$0,67 \pm 0,57$	$84,83 \pm 2,24$	$99,67 \pm 0,38$	06:12:16
32	$29,89 \pm 4,09$	$0,94 \pm 0,7$	$69,06 \pm 3,75$	$99,89 \pm 0,2$	06:16:16
64	$40,39 \pm 5,99$	$2,22 \pm 1,08$	$57,33 \pm 6,17$	$99,94 \pm 0,14$	06:14:14
128	$51,78 \pm 5,46$	$2,22 \pm 1,26$	$46 \pm 5,53$	100	06:12:59
256	$84,67 \pm 3,16$	$0,83 \pm 0,73$	$14,5 \pm 2,76$	100	06:16:47

Tabela 4.7: Resultados dos testes com o *firmware* sem técnicas de tolerância a falhas, com a injeção de falhas nas leituras dos sensores e nas regiões de memória de dados e memória interna. Os dados desta tabela correspondem aos valores médios de um conjunto de 30 testes. Os resultados individuais estão no Apêndice B.

Quantidade de falhas	**Arquivo Não Encontrado**	**CRC Incorreto**	**Dados Incorretos**	**Total de Defeitos**	**Tempo Total**
0	0	0	$4,39 \pm 1,37$	$4,39 \pm 1,37$	05:14:24
1	$3,28 \pm 1,12$	0	$4,06 \pm 1,28$	$7,33 \pm 1,8$	05:24:05
2	$2,67 \pm 1,06$	$0,11 \pm 0,2$	$2,61 \pm 0,94$	$5,39 \pm 1,39$	05:20:25
4	$4,83 \pm 1,15$	0	$2,33 \pm 1,14$	$7,17 \pm 1,53$	05:30:33
8	$6,17 \pm 1,54$	$0,06 \pm 0,14$	$4,28 \pm 1,34$	$10,5 \pm 2,01$	06:05:49
16	$9,39 \pm 2,22$	$0,56 \pm 0,56$	$4,72 \pm 1,32$	$14,67 \pm 2,53$	06:43:14
32	$21,39 \pm 3,03$	$0,94 \pm 0,73$	$7,39 \pm 1,77$	$29,72 \pm 3,46$	07:12:50
64	$29,39 \pm 3,14$	$0,67 \pm 0,49$	$13,33 \pm 2,25$	$43,39 \pm 3,5$	08:11:51
128	$38,11 \pm 4,55$	$1,5 \pm 0,93$	$21,39 \pm 4,29$	$61 \pm 6,65$	09:31:16
256	$46,78 \pm 4,91$	$3,06 \pm 1,15$	$20,94 \pm 3,36$	$70,78 \pm 4,94$	10:15:10

Tabela 4.8: Resultados dos testes com o *firmware* com técnicas de tolerância a falhas, com a injeção de falhas nas leituras dos sensores e nas regiões de memória de dados e memória interna. Os dados desta tabela correspondem aos valores médios de um conjunto de 30 testes. Os resultados individuais estão no Apêndice B.

Os resultados exibidos na Figura 4.9 e na Figura 4.10 tiveram um comportamento análogo aos testes realizados na Subseção 4.2.2. Mais uma vez, nos testes com o *firmware* sem tolerância a falhas, os defeitos do tipo "dados incorretos" mantiveram-se acima dos 90% nos testes com até 8 falhas injetadas nas regiões de memória, sendo sucessivamente reduzidos conforme o aumento das falhas injetadas na memória. A razão do número de defeitos de "arquivo não encontrado" ter ultrapassado os de "dados incorretos" com a injeção de 128 falhas é que, como já mencionado, este trabalho não aplicou estratégias de redundância de dados na região de memória interna do microcontrolador, e como este conjunto de testes injetou falhas nesta região, o microcontrolador travou mais rapidamente.

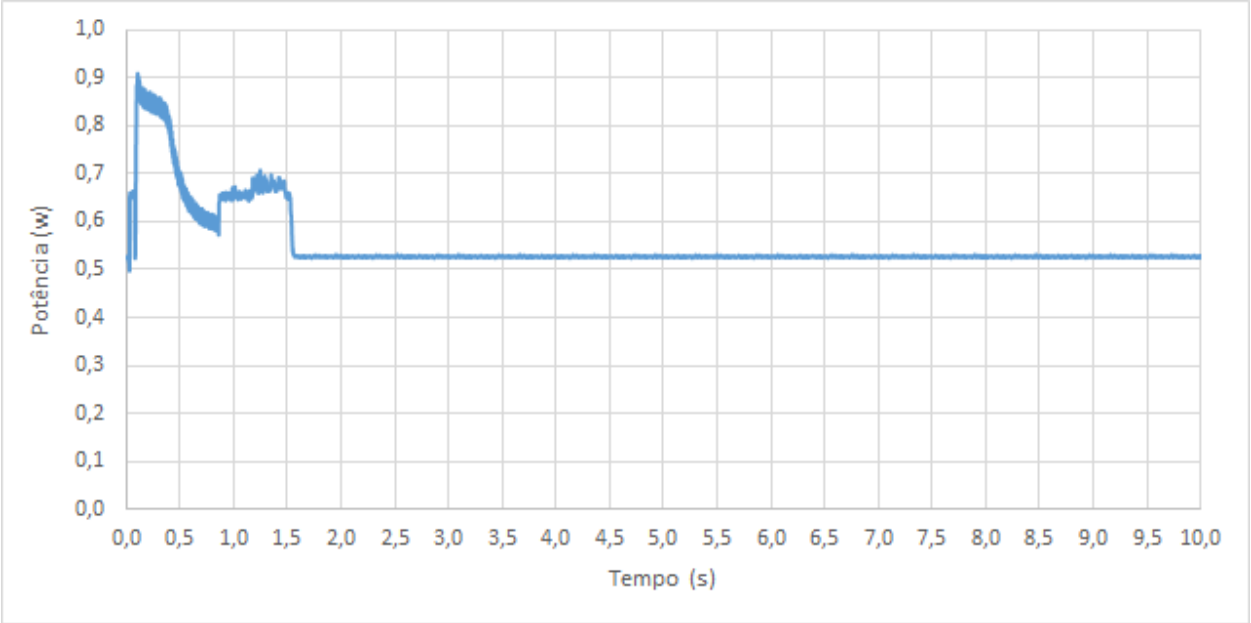
No *firmware* com tolerância e as mesmas 8 falhas injetadas nas regiões de memória, o total de defeitos manteve-se em menos de 11%. Cabe destacar que enquanto o *firmware* não tolerante apresentou mais de 97% de defeitos a partir do teste com 2 injeções de falhas, o *firmware* com tolerância a falhas, com 256 falhas injetadas, teve em média 70% de defeitos. Novamente, cabe observar que estes números mencionados de injeção de falhas na memória equivalem a quantidade de falhas injetadas a cada ciclo de leitura.

4.4 Consumo de Energia

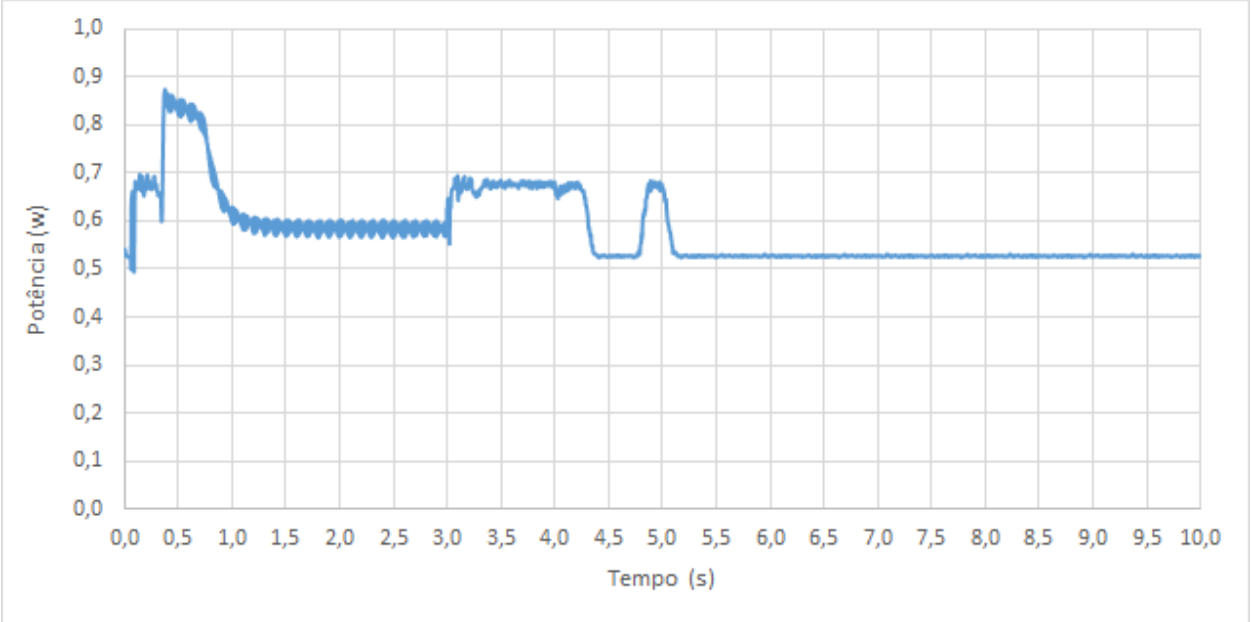
As medições do consumo de energia mostraram que a estação meteorológica, executando o *firmware* com e sem tolerância a falhas, manteve um consumo fixo em cerca de 0,54W durante o período inativo. Em razão dos circuitos auxiliares da placa serem ligados ao início (e desligados ao final) do processo de leituras dos sensores, nesta etapa de atividade, a energia consumida subiu de uma média de 1,08J no *firmware* sem tolerância a falhas, para 3,26J no *firmware* com tolerância a falhas. A energia consumida pelo segundo foi superior devido ao maior tempo de processamento durante a leitura dos sensores.

Conforme mostra-se na Figura 4.11, a energia total consumida em cada ciclo de leitura (energia consumida para ler/processar sensores mais energia em modo inativo) com o *firmware* sem tolerância a falhas foi de 5,63J em média, e com o *firmware* tolerante a falhas, o consumo médio apresentado foi de 5,88J. Estes valores foram calculados considerando-se um ciclo de leitura de 10 segundos para ambos os *firmwares*.

Uma estação meteorológica em uma situação real de trabalho normalmente coleta os parâmetros climáticos em intervalos de minutos, e não em segundos como se utilizou nos testes deste trabalho. Logo, o tempo de inatividade nesses casos é maior e, portando, o impacto no consumo de energia é menor.



(a) Consumo de energia do *firmware* sem tolerância a falhas.



(b) Consumo de energia do *firmware* com tolerância a falhas.

Figura 4.11: Consumo de energia da estação meteorológica. Esses dados correspondem a média coerente (ou promediação temporal) do consumo de energia medido para 30 ciclos de leituras dos sensores, utilizando-se um osciloscópio e um resistor de 9 ohms em série com a fonte de energia.

4.5 Comparação do Consumo de Energia, Memória Flash, Memória RAM e Tempo de Processamento

Conforme visto da Seção 4.1 à Seção 4.3, as técnicas de tolerância a falhas aumentaram a segurança do sistema, pois o número de defeitos gerados com a presença de falhas diminuiu. Porém, as alterações (principalmente a redundância de processamento) afetaram o tempo total de processamento. Na Tabela 4.9 pode-se conferir o tempo necessário para se ler todos os sensores e gravar os dados das leituras em um arquivo na memória *flash*, para estação meteorológica sem tolerância a falhas; com tolerância, mas sem injeção de falhas; e, com tolerância e injeção de falhas (um caso médio e um extremo).

Tolerância a falhas	**Número de falhas injetadas na memória**	**Porcentagem de falhas injetadas nas leituras dos sensores**	**Tempo médio de processamento (segundos)**
Não	0	0%	1,78
Sim	0	0%	5,03
Sim	16	25%	5,76
Sim	128	50%	8,62

Tabela 4.9: Tempo médio de processamento. Esse período corresponde ao tempo (em segundos) necessário para se ler todos os sensores da estação meteorológica e gravar os dados das leituras em um arquivo binário na memória *flash*.

Conforme mostrado na Tabela 4.10, as técnicas de tolerância a falhas também aumentaram o consumo de memória *flash*, o consumo de memória RAM e o consumo de energia. Contudo, é importante observar que o ciclo de leitura do *firmware* sem tolerância a falhas tem duração de 5 segundos, enquanto no *firmware* com tolerância a falhas, a duração é de 10 segundos. Logo, o consumo a cada um minuto é quase idêntico.

Firmware	**Consumo (energia)**	**Consumo (*flash*)**	**Consumo (RAM)**	**Tempo de processamento**
sem tolerância a falhas	33,78J	39,6 kB	1,7 kB	1,78s
com tolerância a falhas	35,28J	68,4 kB	3,9 kB	5,03s

Tabela 4.10: Comparação do consumo de energia (a cada um minuto), memória flash, memória RAM e tempo de processamento entre os *firmwares* sem tolerância a falhas e com tolerância a falhas. Considerando-se um tempo de ciclo de leitura igual a 10 segundos, com o *firmware* sem tolerância a falhas o consumo apresenta uma média de 5,63J, enquanto com o *firmware* com tolerância a falhas este consumo é em média 5,88J.

Capítulo 5

Conclusão

Este trabalho simulou a ocorrência de falhas em um sistema embarcado pode meio de ferramentas que automatizaram o processo de testes e permitiram avaliar o desempenho de um sistema tolerante a falhas, em comparação a outro sem tolerância. As ferramentas desenvolvidas e usadas neste processo, tais como o sistema injetor de falhas e o programa monitor de testes possibilitaram uma análise detalhada das falhas injetadas, disponibilizando entre outras informações, a localização de falhas e os tipos de defeitos gerados. Com a localização das falhas, foi possível descobrir os endereços de memória que mais tiveram impacto no surgimentos de defeitos e com essa informação foi possível descobrir vulnerabilidades no programa. Embora existam outras ferramentas na literatura para este propósito, a maioria são destinadas a outras plataformas, sendo escassas as ferramentas designadas aos sistemas embarcados, tanto que dentre grande parte das ferramentas de diagnóstico frequentemente usadas no auxílio da programação, poucas são compatíveis com a plataforma mbed.

Os resultados deste trabalho apontaram que os programas sem técnicas de tolerância a falhas podem facilmente travar ou produzir resultados incorretos (defeitos) quando expostos a situações de falhas. A presença de defeitos pode causar transtornos aos usuários, e em alguns casos, danos graves e prejuízos. Entretanto, o uso das técnicas baseadas na redundância de dados e de processamento trouxeram uma melhoria significativa na segurança do sistema usado como estudo de caso neste trabalho. Nos testes realizados, o número de falhas na estação meteorológica diminuiu drasticamente, especialmente nos testes que realizaram a injeção de falhas nas leituras dos sensores. Nesses testes, com uma taxa de 25% de injeção de falhas nas leituras dos sensores, a quantidade de defeitos foi em média de 93,95% no *firmware* não tolerante a falhas, sendo reduzido para apenas 4,38% no *firmware* tolerante a falhas.

Nos outros dois conjuntos de testes que injetaram falhas nas regiões de memória do microcontrolador, o primeiro injetou falhas apenas na região de memória destinada ao programa do usuário, e pôde-se perceber que enquanto o *firmware* sem tolerância a falhas apresentou defeitos desde os primeiros testes, com pequenas quantidades de falhas injetadas, os defeitos no *firmware* com tolerância a falhas tornaram-se significativos somente após a injeção de uma quantidade intermediária de falhas. No teste com injeção de 16 falhas por ciclo de leitura, por exemplo, o número de defeitos encontrados foi cerca de 5 vezes menor (média de 2,67 falhas contra 11,8) em comparação com o *firmware* sem tolerância a falhas. Logicamente, com o aumento das quantidades de falhas injetadas, o número de defeitos tornam-se maiores

em ambos os *firmwares*. Cabe observar que no último teste com o *firmware* sem tolerância a falhas, que injetou 256 falhas por ciclo de leitura, estas travaram constantemente o equipamento, exigindo que ele fosse reiniciado manualmente. Em uma situação real, isto causaria um problema grave ao usuário, uma vez que o equipamento do estudo de caso, muitas vezes, está localizado em regiões muito distantes. O mesmo teste com o *firmware* tolerante a falhas apresentou cerca de 55% de defeitos.

Nos testes em que as falhas foram injetadas em todas as regiões de memória, os resultados foram menos satisfatórios, o que demonstra ainda mais a importância das técnicas de tolerância a falhas, uma vez que nas duas regiões de 16 kB, destinadas a memória interna do microcontrolador, não foi utilizado nenhum tipo de redundância, sendo esta, a área de maior vulnerabilidade no sistema. No entanto, a estratégia da máquina de estados foi adotada como uma forma de mascarar as falhas que afetaram o dispositivo.

Os testes mostraram que a técnica com maior eficiência foi o método *avg*, juntamente com a redundância de processamento sobre ele. Embora isso tenha aumentado a segurança, esta e as outras alteração afetaram o tempo de processamento (tempo total de processamento para ler todos os sensores e gravar os dados das leituras em um arquivo na memória *flash*). Nos testes sem injeção de falhas, o tempo de processamento aumentou de 1,57s para 5,14s no *firmware* com tolerância a falhas. Além disso, o firmware com tolerância a falhas aumentou o consumo de memória *flash* (tamanho do programa) de 39,6 kB para 86,4 kB, o consumo de memória (de 1,7 kB para 3,9 kB) e o consumo de energia (de 5,63J para 5,88J). Entretanto, mesmo com maior consumo desses recursos o ganho ainda é superior, considerando o desempenho dessas técnicas nos testes com injeção de falhas. É importante observar que como o *firmware* tolerante a falhas ocupa mais espaço memória RAM, ele foi provavelmente mais atingido por falhas nos testes com injeção de falhas na memória.

Embora algumas técnicas de tolerância a falhas requeiram baixa complexidade de implementação e pouco ou nenhum custo adicional ao projeto, elas geralmente não são empregadas em sistemas embarcados de pequeno ou médio porte, seja pela falta de tempo ou desconhecimento das equipes de programação. A biblioteca de tolerância e recuperação de falhas desenvolvida, chamada de *FaultRecovery*, auxilia por meio de um conjunto de classes e macros a escrita de códigos com implementação de técnicas de tolerância a falhas. Além disso, ela disponibiliza uma estrutura pronta para a recuperação de falhas.

Tanto a biblioteca de injeção de falhas *FaultInjector* quanto a biblioteca de recuperação de falhas *FaultRecovery* desenvolvidas neste trabalho, podem ser adicionadas facilmente a um projeto mbed por meio do repositório oficial da comunidade, disponível, respectivamente, nos endereços `https://mbed.org/users/kleberkruger/code/FaultInjector/` e `https://mbed.org/users/kleberkruger/code/FaultRecovery/`. Os *firmwares* com e sem tolerância a falhas também estão disponíveis.

O desempenho das técnicas de tolerância a falhas podem trazer resultados ainda mais significativos com a implementação de outras técnicas presentes na literatura, como por exemplo, as técnicas baseadas na redundância de hardware ou de software. Para a continuação deste trabalho deve-se estudar alternativas que melhorem ainda mais a segurança do sistema, principalmente sobre as falhas inseridas no espaço reservado a pilha e aos recursos internos do microcontrolador. Uma classe chamada *TripledData* encontra-se em fase experimental, mas pode futuramente trazer mais facilidade ao uso da redundância de dados em

um programa.

Nota-se que, com a expansão da computação ubíqua, as falhas em sistemas embarcados provocarão cada vez mais transtornos na vida cotidiana das pessoas, pois os sistemas embarcados estarão sendo usados o tempo todo. Além disso, as plataformas de prototipagem rápida com microcontroladores de diferentes fabricantes contém bibliotecas com funções e classes prontas. Devido à rapidez com que estas plataformas são colocadas no mercado, a probabilidade de erros de programação é maior e a utilização de técnicas de tolerância a falhas torna-se fundamental.

5.1 Contribuições deste Trabalho

- Uma biblioteca de injeção de falhas, chamada *FaultInjector* que permite, em tempo de execução, injetar um número predefinido de falhas em quaisquer regiões de memória do módulo mbed, além de outros dois tipos de falhas (a que trava o microcontrolador e falha de *loop* infinito). Essa ferramenta disponibiliza uma informação detalhada das falhas injetadas e pode ser facilmente importada a um projeto mbed.

- Uma biblioteca de recuperação e tolerância a falhas, composta por um conjunto de classes e macros que possibilitam empregar as técnicas de tolerância a falhas em um sistema não tolerante com pouca reescrita de código. Esta biblioteca pode ser importada a um projeto por meio do repositório de códigos oficial da comunidade mbed. Suas contribuições mais significativas são:

  - O esquema de recuperação de falhas, baseado em uma máquina de estados.
  - O algoritmo do método *avg*, já utilizado e referenciado em outros trabalhos, que permite um resultado confiável da leitura de um sensor, desprezando leituras com valores acima de uma variação máxima definida.
  - A classe *TripledData* pode ser usada na automatização da redundância de dados em trabalhos futuros. No entanto, embora suas técnicas (redundância de dados e acesso aos dados por sistema de votação) tenham sido constantemente utilizadas neste trabalho, sua implementação foi realizada somente após os testes realizados. A ideia da classe *TripledData* é facilitar e automatizar a redundância de dados (aplicadas neste trabalho) em trabalhos posteriores. Um exemplo da ideia de implementação inicial dessa classe pode ser visto na Subseção 5.4.2.

- Resultados dos testes. Através dos resultados desses testes é possível analisar o desempenho das técnicas de tolerância a falhas baseadas na redundância de dados e de processamento quando empregadas a um sistema embarcado.

5.2 Dificuldades Encontradas

Os primeiros testes com o *firmware* tolerante a falhas apresentaram resultados poucos satisfatórios, pois as falhas injetadas não geravam qualquer informação de localização. Foi

preciso alterar a biblioteca de injeção de falhas e analisar detalhadamente os *logs* para descobrir novos pontos de vulnerabilidade no programa. Descobriu-se que as áreas das regiões de memória que guardavam as configurações do programa, como tempo do intervalo entre as leituras, quantidade de sensores lidos, valor limite do temporizador *watchdog*, e outras, eram constantemente afetadas por falhas, e por esse motivo, o programa apresentava muitos erros.

Outro problema constantemente encontrado foi no uso das bibliotecas importadas por outros desenvolvedores, no repositório de códigos da plataforma mbed. Não existe nenhum teste de aceitação para submeter uma biblioteca ao repositório de códigos. A biblioteca que controla o módulo gps, por exemplo, apresentava uma falha de ficar travado em um laço infinito, tentando conectar ao módulo gps, mesmo ele não estando presente. A biblioteca de manipulação do arquivo de configuração continha falhas de gerência de memória. Essas falhas são comuns em códigos de programação, e ocorrem quando, por exemplo, um programa lê uma posição a mais de um vetor alocado ("*invalid read*"), escreve em uma região de memória pertencente a outra variável ("*invalid write*)", ou tenta desalocar regiões que já foram desalocadas ("*invalid free*").

5.3 Limitações do Trabalho

Uma limitação deste trabalho foi a não realização de testes em um ambiente real. Esse procedimento, embora custoso em termos de tempo, proporcionaria uma avaliação precisa dos efeitos de falhas em cenário real. No entanto, colocar o equipamento exposto em um ambiente aberto esperando o surgimento de falhas levaria um tempo considerável, e cada teste levaria meses, dado que as falhas, embora aconteçam, sua frequência em ambientes reais é menor, conforme cita experimentos práticos na literatura. Isso, aliás, foi um dos principais motivos que incentivaram o desenvolvimento de injetores de falhas. Com estes, é possível bombardear o dispositivo com geradores de falhas muito mais frequentes do que as naturais para acelerar o processo de testes. Além disso, as falhas naturais não geram qualquer informação de tipo e localização. O que dificulta o diagnóstico das vulnerabilidades de um sistema.

Outra limitação foi uso de nenhuma estratégia de redundância nas regiões de memória interna do microcontrolador. Esta limitação deveu-se por conta da falta de informações de como essa região é utilizada pelo microcontrolador, pois, em razão da placa de prototipagem rápida usada neste trabalho pertencer a um projeto proprietário, os fabricantes pouco disponibilizam informações de seu funcionamento interno. Adotando-se estratégias de redundância nos dados de todas as regiões de memória, provavelmente o ganho de desempenho em relação a tolerância a falhas seria ainda maior, dado que os testes que injetaram falhas apenas na região de memória com redundância controlada (memória de dados) teve o número de defeitos bastante reduzido.

5.4 Trabalhos Futuros

5.4.1 Implementações e Atividades Futuras

Como trabalhos futuros, pode-se:

- Ampliar o uso das técnicas de tolerância a falhas na região de memória interna do microcontrolador;
- Estender o conjunto de testes em um ambiente real;
- Ampliar a automatização do emprego das técnicas de tolerância a falhas com o uso da biblioteca *FaultRecovery* para que não seja preciso o uso de macros pelo programador; O ideal seria a própria biblioteca efetuar a alteração dos estados;
- Criar um "identificador" de falhas que analise os traços de execução do programa, observando os valores do registrador PC (*Program Counter*); Com isso será possível identificar que o programa ficou preso em um laço infinito ou que, devido a uma falha, tomou um caminho incorreto de execução;
- Implementar estratégias na classe *TripledData* para aplicar a redundância de dados em variáveis homogêneas e objetos;

5.4.2 Redundância de Dados Automática

A classe *TripledData* foi criada com o objetivo de automatizar a redundância de dados das variáveis de tipos primitivos (*int*, *float*, *long*, *double*, *char*, *bool*). Ao criar um objeto *TripledData*, três cópias do dado primitivo são armazenadas em diferentes regiões da memória e seus dados são obtidos mediante um sistema de votação, realizado implicitamente a cada acesso à variável. Um exemplo simplificado da implementação dessa classe é mostrado no Quadro 7.

Ao instanciar um objeto do tipo *TripledData*, deve-se especificar o tipo do dado redundante: TripledData<tipo_da_variável> var. O comando __attribute__((section("X"))), na linha 24, 25 e 26 é responsável por armazenar cada cópia da variável em diferentes regiões da memória, de forma a evitar que falhas em localizações próximas afetem todas as cópias. Uma limitação dessa classe é que apenas os tipos de variáveis primitivas (mais especificamente, variáveis homogêneas) são suportadas como parâmetro do *template*.

Como o dado é replicado em 3 diferentes regiões de memória, é preciso manter todas as cópias atualizadas a cada alteração do valor da variável. Da mesma forma, o valor da mesma deve ser obtido mediante um sistema de votação, já que as falhas podem atingir suas regiões de memória. Para implementar esse mecanismo, utilizou-se a sobrescrita de operadores, disponível na linguagem C++. Ao alterar o valor da variável, o método *setData* é invocado para alterar simultaneamente o valor de todas as cópias, e ao acessá-la, o método *getData* obtém o valor por um sistema de votação, que atualiza a cópia com erro caso necessário. Cabe observar que nos primeiros testes, o sistema de votação apenas obtinha o resultado da saída conforme a maioria. No entanto, com a contínua injeção de falhas,

todas as cópias eram afetadas e isso trouxe pouca eficiência. Com a correção do dado errado a cada acesso, os resultados mostraram um bom desempenho, uma vez que os erros são rapidamente corrigidos quando a variável é constantemente acessada. Caso mais de uma cópia seja atingida por falha, o sistema de votação identifica o erro e gera uma exceção. Outro ponto a se observar, é que a reescrita do código se torna muito trivial: o usuário apenas precisa substituir as declarações das variáveis primitivas pelo objeto *TripledData*, passando o tipo da variável primitiva ao *template* (entre os símbolos de <>).

```
1  template <typename T>
2  class TripledData {
3  public:
4
5      TripledData() {}
6
7      T getData();
8      void setData(T data);
9      void setData(T *data);
10     void injectFault();
11     void print();
12
13     TripledData& operator= (T data) {
14         setData(data);
15         return *this;
16     }
17     TripledData& operator= (TripledData& obj) {
18         setData(obj.getData());
19         return *this;
20     }
21     operator T() { return getData(); }
22
23 private:
24     T d1 __attribute__((section("SRAM")));
25     T d2 __attribute__((section("AHBSRAM0")));
26     T d3 __attribute__((section("AHBSRAM1")));
27
28     T getByVotting();
29 };
30
31 template <typename T>
32 T TripledData<T>::getByVotting() {
33     if (memcmp(&d1, &d2, sizeof (T)) == 0) {
34         if (memcmp(&d1, &d3, sizeof (T)) != 0) {
35             memcpy(&d3, &d1, sizeof (T));
36         }
37     } else if (memcmp(&d1, &d3, sizeof (T)) == 0)
38         memcpy(&d2, &d1, sizeof (T));
39     else if (memcmp(&d2, &d3, sizeof (T)) == 0)
40         memcpy(&d1, &d2, sizeof (T));
41     else {
42         throw new data_exception();
43         return 0;
44     }
45     return d1;
46 }
```

```
47 template <typename T>
48 T TripledData<T>::getData() {
49     return getByVotting();
50 }
51
52 template <typename T>
53 void TripledData<T>::setData(T data) {
54     memcpy(&d1, &data, sizeof (T));
55     memcpy(&d2, &data, sizeof (T));
56     memcpy(&d3, &data, sizeof (T));
57 }
58 template <typename T>
59 void TripledData<T>::setData(T *data) {
60     memcpy(&d1, data, sizeof (T));
61     memcpy(&d2, data, sizeof (T));
62     memcpy(&d3, data, sizeof (T));
63 }
64
65 template <typename T>
66 void TripledData<T>::injectFault() {
67     srand(time(NULL));
68     d1 = rand() % 65535;
69 }
70 template <typename T>
71 void TData<T>::print() {
72     printf("d1 = %d", d1);
73     printf("d2 = %d", d2);
74     printf("d3 = %d", d3);
75 }
76
77 int main(int argc, char** argv) {
78
79     TripledData<int> var;
80     TripledData<int> *var_ptr;
81     var_ptr = &var;
82     var.setData(50);
83     var = 90 + 10;
84     var.injectFault();
85     printf("Valor de var = %d", var);
86     printf("Valor de var = %d", *var_ptr);
87     var.injectFault();
88
89     return 0;
90 }
```

Quadro 7: Classe que automatiza a redundância de dados das variáveis de tipos primitivos.

A Figura 5.1 mostra a saída da execução do código do Quadro 7.

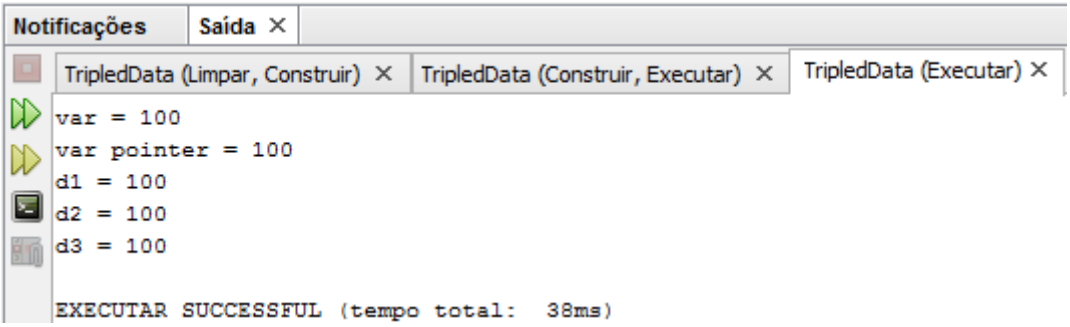


Figura 5.1: Saída do exemplo do código *TripledData*. Mesmo com a injeção de falha na cópia $d1$, o valor da variável se encontra correto no final do programa, pois o acesso a ela implicitamente a corrige pelo sistema de votação.

Bibliografia

[1] MARWEDEL, P. *Embedded system design*. Secaucus, NJ, USA: Springer-Verlag New York, Inc., 2006.

[2] PATTERSON, D. A.; HENNESSY, J. L. *Computer organization and design: The hardware/software interface*. 3rd. ed. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2007.

[3] CHETAN, S.; RANGANATHAN, A.; CAMPBELL, R. Towards fault tolerance pervasive computing. *Technology and Society Magazine, IEEE*, v. 24, n. 1, p. 38–44, Spring 2005.

[4] BALL, S. *Embedded microprocessor systems: Real world design*. 3rd. ed. Newton, MA, USA: Butterworth-Heinemann, 2002.

[5] SINGH, A. D.; MURUGESAN, S. Fault-tolerant systems. *Computer*, Los Alamitos, CA, USA, v. 23, p. 15–17, jul 1990.

[6] JOHNSON, B. Fault-tolerant microprocessor-based systems. *Micro, IEEE*, v. 4, n. 6, p. 6–21, dec 1984.

[7] AVIZIENIS, A. Design of fault-tolerant computers. *Fall Joint Computer Conference*, v. 31, p. 733–743, 1967.

[8] VON NEUMANN, J. Probabilistic logics and the synthesis of reliable organisms from unreliable components. *Automata Studies*, p. 43–98, 1956.

[9] NELSON, V. P. Fault-tolerant computing: Fundamental concepts. *Computer*, Los Alamitos, CA, USA, v. 23, p. 19–25, jul 1990.

[10] IAIONE, F.; LIMA, D. G. D.; GASSEN, F. R. Equipamento para coleta de dados e previsão de doenças na lavoura, 1999.

[11] REIS, E. *Previsão de doenças de plantas*. UPF EDITORA, 2004.

[12] PRESSMAN, R. *Software engineering: A practitioner's approach*. 6. ed. New York, NY, USA: McGraw-Hill, Inc., 2005.

[13] TAURION, C. *Software embarcado: oportunidades e potencial de mercado*. Rio de Janeiro: Brasport, 2005.

[14] HUANG, B.; LI, X.; LI, M.; BERNSTEIN, J.; SMIDTS, C. Study of the impact of hardware fault on software reliability. *2013 IEEE 24th International Symposium on (ISSRE)*, Los Alamitos, CA, USA, v. 0, p. 63–72, 2005.

[15] HSUEH, M.-C.; TSAI, T. K.; IYER, R. K. Fault injection techniques and tools. *Computer*, Los Alamitos, CA, USA, v. 30, n. 4, p. 75–82, apr 1997.

[16] LOPES, D. C. *Estimação da robustez de sistemas eletrônicos via injeção de interferência eletromagnética*. 2005. Dissertação de Mestrado - PUCRGS, Porto Alegre, 2005.

[17] SECALL, J. M. *Avaliação comparativa do impacto do emprego de programação defensiva na segurança de sistemas críticos*. 2007. Tese de Doutorado - Escola Politécnica da Universidade de São Paulo, São Paulo, 2007.

[18] KARNIK, T.; HAZUCHA, P.; PATEL, J. Characterization of soft errors caused by single event upsets in cmos processes. *IEEE Trans. Dependable Secur. Comput.*, Los Alamitos, CA, USA, v. 1, n. 2, p. 128–143, apr 2004.

[19] WALLMARK, J.; MARCUS, S. M. Minimum size and maximum packing density of nonredundant semiconductor devices. *Proceedings of the IRE*, v. 50, n. 3, p. 286–298, March 1962.

[20] BINDER, D.; SMITH, E.; HOLMAN, A. B. Satellite anomalies from galactic cosmic rays. *Nuclear Science, IEEE Transactions on*, v. 22, n. 6, p. 2675–2680, Dec 1975.

[21] J. SRINIVASAN, S. V. ADVE, P. B. J. R.; HU, C.-K. Ramp: A model for reliability aware microprocessor design, December 2003.

[22] CARVALHO, A. Grandes Desafios da Pesquisa em Computação no Brasil - 2006 – 2016: Relatório sobre o Seminário realizado em 8 e 9 de maio de 2006. Porto Alegre: SBC, 2006. Disponível em: <http://www.sbc.org.br>, Acesso em: 01/05/2010.

[23] LAPRIE, J. C. Dependable computing and fault-tolerance: concepts and terminology. *Annual International Symposium on Fault Tolerant Computing*, v. 15, p. 2–11, jun 1985.

[24] LAPRIE, J.-C. Dependable computing and fault tolerance: Concepts and terminology. In: . c1995. p. 2–.

[25] ANDERSON, T.; LEE, P. *Fault tolerance, principles and practice*. Prentice/Hall International, 1981.

[26] KOREN, I.; KRISHNA, C. M. *Fault-tolerant systems*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2007.

[27] WEBER, T. S. Tolerância a falhas: conceitos e exemplos. In: *Intech Brasil*. Distrito 4 da ISA (The Instrumentation, System and Automation Society), 2003. v. 52, p. 32–42.

[28] WEBER, T. S. Um roteiro para exploração dos conceitos básicos de tolerância a falhas. Publicação online, 2002. Acesso em: 15/07/2014.

[29] ZIADE, H.; AYOUBI, R. A.; VELAZCO, R. A survey on fault injection techniques. *Int. Arab J. Inf. Technol.*, , n. 2, p. 171–186, 2004.

[30] VELAZCO, R.; REZGUI, S. Transient bitflip injection in microprocessor embedded applications. In: . IOLTW '00. Washington, DC, USA: IEEE Computer Society, c2000. p. 80–.

[31] SOMANI, A. K.; VAIDYA, N. H. Understanding fault tolerance and reliability. *Computer*, Los Alamitos, CA, USA, v. 30, n. 4, p. 45–50, apr 1997.

[32] LAPRIE, J.-C.; BÉOUNES, C.; KANOUN, K. Definition and analysis of hardware- and software-fault-tolerant architectures. *Computer*, Los Alamitos, CA, USA, v. 23, n. 7, p. 39–51, jul 1990.

[33] SALEWSKI, F.; TAYLOR, A. Fault handling in fpgas and microcontrollers in safety-critical embedded applications: A comparative survey. In: . DSD '07. Washington, DC, USA: IEEE Computer Society, c2007. p. 124–131.

[34] DUBROVA, E. Fault tolerant design: An introduction. In: . Department of Microelectronics and Information Technology Royal Institute of Technology Stockholm, Sweden: Stockholm, Sweden: Kluwer Academic Publishers, c2007.

[35] PHAM, H.-M.; PILLEMENT, S.; PIESTRAK, S. Low-overhead fault-tolerance technique for a dynamically reconfigurable softcore processor. *IEEE Trans. Comput.*, Washington, DC, USA, v. 62, n. 6, p. 1179–1192, jun 2013.

[36] DESOUSA, P. T.; MATHUR, F. P. Modular redundancy without voters decreases complexity of restoring organ. In: . AFIPS '77. New York, NY, USA: ACM, c1977. p. 801–806.

[37] FETZER, C. Perfect failure detection in timed asynchronous systems. *IEEE Trans. Comput.*, Washington, DC, USA, v. 52, n. 2, p. 99–112, feb 2003.

[38] BRILLIANT, S. S.; KNIGHT, J. C.; LEVESON, N. G. Analysis of faults in an n-version software experiment. *IEEE Trans. Softw. Eng.*, Piscataway, NJ, USA, v. 16, n. 2, p. 238–247, feb 1990.

[39] AVIZIENIS, A. The n-version approach to fault-tolerant software. *IEEE Trans. Softw. Eng.*, Piscataway, NJ, USA, v. 11, n. 12, p. 1491–1501, dec 1985.

[40] ROMANOVSKY, A. Diversely designed classes for use by multiple tasks. *Ada Lett.*, New York, NY, USA, v. XX, n. 1, p. 25–37, mar 2000.

[41] KUROSE, J. F.; ROSS, K. W. *Computer networking: A top-down approach*. 5th. ed. USA: Addison-Wesley Publishing Company, 2009.

[42] KRUGER, K.; IAIONE, F. Development of a fault injection system to test a weather station based on rapid prototyping platform. In: . High Performance Computing and Communications & Embedded and Ubiquitous Computing, c2013. p. 1652–1657.

[43] KANAWATI, G. A.; KANAWATI, N. A.; ABRAHAM, J. A. Ferrari: A flexible software-based fault and error injection system. *IEEE Trans. Comput.*, Washington, DC, USA, v. 44, n. 2, p. 248–260, feb 1995.

[44] International Organization for Standardization, “Product de-velopment: software level”, ISO/DIS 26262-6, 2009.

[45] NASA Software Safety Guidebook, NASA-GB-8719.13, 2004.

[46] ARLAT, J.; AGUERA, M.; AMAT, L.; CROUZET, Y.; FABRE, J.-C.; LAPRIE, J.-C.; MARTINS, E.; POWELL, D. Fault injection for dependability validation: A methodology and some applications. *IEEE Transactions on Software Engineering*, Los Alamitos, CA, USA, v. 16, n. 2, p. 166–182, 1990.

[47] NATELLA, R.; COTRONEO, D.; DURAES, J. A.; MADEIRA, H. S. On fault representativeness of software fault injection. *IEEE Transactions on Software Engineering*, Los Alamitos, CA, USA, v. 39, n. 1, p. 80–96, 2013.

[48] SOTOMA, I. *Afids - arquitetura para injeção de falhas em sistemas distribuídos*. 1997. Dissertação de Mestrado - UFRGS, Porto Alegre, 1997.

[49] STOTT, D.; FLOERING, B.; BURKE, D.; KALBARCZPK, Z.; IYER, R. Nftape: a framework for assessing dependability in distributed systems with lightweight fault injectors. In: . c2000. p. 91–100.

[50] CARREIRA, J.; MADEIRA, H.; SILVA, J. G. Xception: A technique for the experimental evaluation of dependability in modern computers. *IEEE Trans. Software Eng.*, v. 24, n. 2, p. 125–136, 1998.

[51] AIDEMARK, J.; VINTER, J.; FOLKESSON, P.; KARLSSON, J. Goofi: generic object-oriented fault injection tool. In: . c2001. p. 83–88.

[52] GOSWAMI, K.; IYER, R. Simulation of software behavior under hardware faults. In: . c1993. p. 218–227.

[53] DELONG, T. A.; JOHNSON, B. W.; PROFETA III, J. A. A fault injection technique for vhdl behavioral-level models. *IEEE Des. Test*, Los Alamitos, CA, USA, v. 13, n. 4, p. 24–33, dec 1996.

[54] AMENDOLA, A.; BENSO, A.; CORNO, F.; IMPAGLIAZZO, L.; MARMO, P.; PRINETTO, P.; REBAUDENGO, M.; REORDA, M. Fault behavior observation of a microprocessor system through a vhdl simulation-based fault injection experiment. In: . c1996. p. 536–541.

[55] GIL, D.; BARAZA, J.-C.; BUSQUETS, J. V.; GIL, P. Fault injection into vhdl models: analysis of the error syndrome of a microcomputer system. In: . c1998. v. 1. p. 418–425 vol.1.

[56] MBED. Disponível em: <https://mbed.org/>. Acesso em: 15/07/2014.

[57] TERATERM. Disponível em: <http://en.sourceforge.jp/projects/ttssh2/>. Acesso em: 15/07/2014.

Apêndice A

Resultados dos Testes - Eficiência do Método AVG

As tabelas deste apêndice mostram os resultados individuais dos testes que injetaram 10%, 20%, 25%, 30%, 40% e 50% de falhas nas leituras dos sensores.

As colunas das tabelas correspondem a:

- Teste: número do teste;
- Real AVG (1x): número de vezes que o método *avg* realmente tolerou falhas. Os casos em que todos os elementos da amostra estão dentro da variação máxima não entram nessa contagem, pois nessas situações, o método *avg* não teve interferência no resultado;
- AVG (2x): número de vezes que a execução do método *avg* pela segunda vez (redundância de processamento) trouxe uma média confiável;
- AVG (3x): número de vezes que a execução do método *avg* pela terceira vez (redundância de processamento) trouxe uma média confiável;
- Total de erros: número de vezes que o método *avg*, mesmo com as 3 tentativas, não conseguiu calcular uma média confiável;
- Tolerância + erros: soma do número de vezes que o método *avg* com a redundância de processamento tolerou falhas, mais o número de erros ocorridos.

Percentual de falhas injetadas:					**10%**
Percentual de leituras dentro da faixa aceitável:					**66,6%** (4/6)
Teste	Real AVG (1x)	AVG (2x)	AVG (3x)	Total de erros	Tolerância + erros
1	273	9	0	0	282
2	238	11	0	0	249
3	275	6	0	0	281
4	220	5	0	0	225
5	246	8	0	0	254
6	242	6	0	0	248
7	242	3	0	0	245
8	235	6	1	0	242
9	231	9	0	0	240
10	247	9	0	0	256
11	243	5	0	0	248
12	246	7	0	0	253
13	241	8	0	0	249
14	271	9	0	0	280
15	237	9	1	0	247
16	243	3	0	0	246
17	251	9	1	0	261
18	229	7	0	0	236
19	252	12	0	0	264
20	230	8	0	0	238
21	248	12	0	0	260
22	264	6	0	0	270
23	242	6	1	0	249
24	254	8	0	0	262
25	215	13	0	0	228
26	243	8	0	0	251
27	255	5	0	0	260
28	257	8	0	0	265
29	230	8	0	0	238
30	240	5	0	0	245

Percentual de falhas injetadas:					**20%**
Percentual de leituras dentro da faixa aceitável:					**66,6%** (4/6)
Teste	Real AVG (1x)	AVG (2x)	AVG (3x)	Total de erros	Tolerância + erros
1	361	45	4	0	410
2	334	52	3	0	389
3	351	32	8	1	392
4	337	55	5	0	397
5	356	44	4	0	404
6	336	45	0	1	382
7	326	61	3	2	392
8	332	56	7	2	397
9	319	61	9	0	389
10	316	52	3	1	372
11	345	43	10	0	398
12	344	48	2	1	395
13	354	34	6	1	395
14	323	58	9	0	390
15	346	42	5	1	394
16	346	46	6	0	398
17	382	37	4	0	423
18	347	55	3	2	407
19	352	47	5	0	404
20	376	33	5	0	414
21	337	45	8	1	391
22	362	41	6	0	409
23	328	53	2	0	383
24	340	40	4	0	384
25	356	44	8	0	408
26	350	62	3	0	415
27	344	49	2	1	396
28	342	44	8	0	394
29	354	51	5	0	410
30	336	53	5	0	394

Percentual de falhas injetadas:					**25%**
Percentual de leituras dentro da faixa aceitável:					**66,6%** (4/6)
Teste	Real AVG (1x)	AVG (2x)	AVG (3x)	Total de erros	Tolerância + erros
1	349	78	14	3	444
2	362	79	8	6	455
3	345	59	16	2	422
4	333	76	18	2	429
5	355	69	10	3	437
6	343	87	9	2	441
7	347	79	13	2	441
8	374	68	10	2	454
9	358	66	11	2	437
10	371	62	12	2	447
11	371	76	14	1	462
12	347	64	19	1	431
13	349	83	11	10	453
14	349	87	11	5	452
15	365	71	15	4	455
16	357	83	9	3	452
17	352	70	14	3	439
18	365	66	17	4	452
19	348	79	13	1	441
20	341	91	14	3	449
21	371	70	7	1	449
22	332	74	15	3	424
23	355	66	8	2	431
24	343	72	15	3	433
25	337	87	13	2	439
26	355	87	12	3	457
27	351	82	15	3	451
28	351	76	12	3	442
29	354	70	13	1	438
30	352	69	11	4	436

Percentual de falhas injetadas:					**30%**
Percentual de leituras dentro da faixa aceitável:					**66,6%** (4/6)
Teste	Real AVG (1x)	AVG (2x)	AVG (3x)	Total de erros	Tolerância + erros
1	324	114	28	11	477
2	343	103	24	10	480
3	348	97	23	6	474
4	329	102	30	11	472
5	335	101	27	11	474
6	351	82	24	7	464
7	350	92	28	8	478
8	339	99	22	12	472
9	356	85	23	7	471
10	333	119	25	8	485
11	336	108	28	14	486
12	344	98	21	15	478
13	351	109	23	14	497
14	346	110	24	9	489
15	324	111	31	12	478
16	333	105	32	6	476
17	333	115	24	11	483
18	342	96	28	8	474
19	335	106	28	8	477
20	362	91	17	5	475
21	343	92	39	4	478
22	324	106	20	6	456
23	349	100	23	10	482
24	356	95	24	8	483
25	331	115	23	9	478
26	342	100	29	8	479
27	342	110	16	9	477
28	349	104	28	13	494
29	345	98	27	12	482
30	345	100	25	11	481

Percentual de falhas injetadas:					**40%**
Percentual de leituras dentro da faixa aceitável:					**66,6%** (4/6)
Teste	Real AVG (1x)	AVG (2x)	AVG (3x)	Total de erros	Tolerância + erros
1	298	108	54	52	512
2	251	134	70	57	512
3	277	125	55	56	513
4	257	133	74	45	509
5	258	138	66	50	512
6	278	131	57	47	513
7	278	137	56	50	521
8	264	136	70	48	518
9	267	132	58	53	510
10	280	136	53	45	514
11	251	136	71	55	513
12	263	140	71	44	518
13	243	148	68	50	509
14	266	133	56	53	508
15	252	149	62	54	517
16	258	130	61	61	510
17	269	147	56	44	516
18	264	143	49	54	510
19	267	148	51	52	518
20	272	135	58	44	509
21	268	141	61	54	524
22	279	111	76	52	518
23	273	132	58	51	514
24	283	129	64	42	518
25	275	129	57	48	509
26	276	120	67	50	513
27	281	126	61	50	518
28	263	132	65	58	518
29	276	128	61	44	509
30	279	114	61	63	517

Percentual de falhas injetadas:					**50%**
Percentual de leituras dentro da faixa aceitável:					**66,6%** (4/6)
Teste	Real AVG (1x)	AVG (2x)	AVG (3x)	Total de erros	Tolerância + erros
1	163	116	81	170	530
2	184	130	63	156	533
3	187	117	80	145	529
4	176	127	85	146	534
5	204	97	70	159	530
6	182	137	82	128	529
7	156	113	104	162	535
8	178	124	93	137	532
9	183	121	72	158	534
10	168	111	81	169	529
11	184	113	83	153	533
12	183	118	76	150	527
13	187	117	87	138	529
14	177	112	79	162	530
15	188	121	72	147	528
16	162	131	83	154	530
17	183	109	84	155	531
18	162	110	92	166	530
19	176	131	72	154	533
20	161	123	81	163	528
21	184	122	84	140	530
22	184	108	92	151	535
23	169	134	75	158	536
24	161	141	73	152	527
25	166	136	81	152	535
26	175	112	88	158	533
27	189	123	81	138	531
28	175	137	82	133	527
29	195	110	74	153	532
30	196	119	79	137	531

Percentual de falhas injetadas:					**10%**
Percentual de leituras dentro da faixa aceitável:					**75,0%** (3/4)
Teste	Real AVG (1x)	AVG (2x)	AVG (3x)	Total de erros	Tolerância + erros
1	161	28	1	0	190
2	147	30	0	0	177
3	149	19	0	0	168
4	166	29	1	0	196
5	166	24	0	0	190
6	164	21	3	0	188
7	155	28	1	0	184
8	161	29	0	1	191
9	152	24	2	0	178
10	167	22	1	0	190
11	164	18	0	0	182
12	152	21	2	0	175
13	163	21	1	0	185
14	163	28	2	0	193
15	165	20	1	0	186
16	178	21	0	0	199
17	156	25	0	0	181
18	159	31	1	0	191
19	140	28	2	0	170
20	170	31	1	0	202
21	159	17	2	0	178
22	146	36	4	0	186
23	151	27	1	0	179
24	160	36	0	0	196
25	149	25	3	0	177
26	155	30	3	0	188
27	159	18	3	0	180
28	160	22	5	0	187
29	149	34	1	1	185
30	155	31	1	0	187

Percentual de falhas injetadas:					**20%**
Percentual de leituras dentro da faixa aceitável:					**75,0%** (3/4)
Teste	Real AVG (1x)	AVG (2x)	AVG (3x)	Total de erros	Tolerância + erros
1	219	76	16	6	317
2	205	92	7	2	306
3	221	78	21	3	323
4	233	56	13	2	304
5	224	92	19	2	337
6	213	88	18	5	324
7	229	76	10	7	322
8	228	79	18	3	328
9	207	76	18	3	304
10	212	78	15	5	310
11	193	68	14	3	278
12	245	60	12	4	321
13	218	88	16	3	325
14	212	83	21	3	319
15	214	89	11	2	316
16	213	91	12	2	318
17	220	78	12	6	316
18	238	74	14	1	327
19	227	85	17	8	337
20	243	69	10	3	325
21	213	78	12	3	306
22	218	68	18	1	305
23	223	83	12	2	320
24	236	76	17	2	331
25	225	87	15	1	328
26	213	86	11	4	314
27	220	102	17	3	342
28	237	73	15	4	329
29	214	81	7	1	303
30	232	76	12	5	325

Percentual de falhas injetadas:					**25%**
Percentual de leituras dentro da faixa aceitável:					**75,0%** (3/4)
Teste	Real AVG (1x)	AVG (2x)	AVG (3x)	Total de erros	Tolerância + erros
1	246	112	23	7	388
2	231	90	34	17	372
3	248	91	18	12	369
4	224	107	21	9	361
5	250	100	23	8	381
6	224	104	23	5	356
7	239	104	27	8	378
8	222	98	32	10	362
9	244	99	25	6	374
10	228	107	35	6	376
11	236	82	27	12	357
12	248	105	20	10	383
13	244	102	33	10	389
14	251	101	26	17	395
15	243	89	29	4	365
16	225	109	24	8	366
17	228	93	26	9	356
18	229	104	25	12	370
19	231	103	31	12	377
20	227	109	28	8	372
21	233	98	28	4	363
22	227	117	20	10	374
23	222	94	37	11	364
24	235	103	38	9	385
25	211	107	26	12	356
26	226	101	29	6	362
27	229	102	19	8	358
28	237	97	14	12	360
29	218	110	27	12	367
30	231	105	26	12	374

Percentual de falhas injetadas:					**30%**
Percentual de leituras dentro da faixa aceitável:					**75,0%** (3/4)
Teste	Real AVG (1x)	AVG (2x)	AVG (3x)	Total de erros	Tolerância + erros
1	216	125	45	18	404
2	239	130	45	13	427
3	234	114	47	26	421
4	238	109	35	23	405
5	230	106	42	23	401
6	219	127	42	19	407
7	234	111	38	18	401
8	237	117	48	19	421
9	227	121	32	24	404
10	227	122	53	18	420
11	234	117	38	22	411
12	210	136	46	27	419
13	230	109	48	19	406
14	222	132	45	24	423
15	207	125	39	23	394
16	214	146	45	20	425
17	227	123	53	19	422
18	219	121	35	29	404
19	215	109	39	29	392
20	215	127	41	29	412
21	218	125	51	17	411
22	214	136	35	20	405
23	227	126	37	16	406
24	224	136	44	17	421
25	216	114	40	27	397
26	228	118	44	23	413
27	215	130	44	25	414
28	224	133	43	17	417
29	230	133	43	19	425
30	231	116	46	19	412

Percentual de falhas injetadas:					**40%**
Percentual de leituras dentro da faixa aceitável:					**75,0%** (3/4)
Teste	Real AVG (1x)	AVG (2x)	AVG (3x)	Total de erros	Tolerância + erros
1	213	127	67	72	479
2	192	147	67	79	485
3	184	142	62	74	462
4	182	140	64	76	462
5	192	146	54	76	468
6	186	152	71	70	479
7	175	137	65	76	453
8	177	137	78	72	464
9	176	145	72	78	471
10	193	140	68	85	486
11	190	141	80	62	473
12	181	136	70	70	457
13	195	137	65	75	472
14	171	146	71	78	466
15	188	140	75	60	463
16	173	128	81	76	458
17	207	120	76	69	472
18	174	134	79	92	479
19	178	147	76	74	475
20	184	145	71	69	469
21	183	142	71	62	458
22	187	123	79	86	475
23	177	126	93	77	473
24	190	125	76	82	473
25	189	123	68	80	460
26	186	135	73	73	467
27	163	158	69	68	458
28	203	130	70	56	459
29	200	128	63	83	474
30	200	109	72	85	466

Percentual de falhas injetadas:					**50%**
Percentual de leituras dentro da faixa aceitável:					**75,0%** (3/4)
Teste	Real AVG (1x)	AVG (2x)	AVG (3x)	Total de erros	Tolerância + erros
1	133	132	82	159	506
2	134	128	73	173	508
3	144	124	72	165	505
4	122	116	87	187	512
5	138	118	85	164	505
6	137	115	98	154	504
7	155	105	70	169	499
8	146	115	81	169	511
9	141	103	93	169	506
10	145	137	66	149	497
11	134	120	80	173	507
12	122	127	88	175	512
13	135	114	96	154	499
14	146	117	68	175	506
15	104	125	76	194	499
16	120	115	87	180	502
17	140	111	83	181	515
18	141	116	74	171	502
19	145	111	71	171	498
20	143	102	81	185	511
21	143	112	81	173	509
22	132	101	86	182	501
23	146	133	71	161	511
24	143	105	81	183	512
25	123	123	86	179	511
26	122	136	86	165	509
27	140	124	79	162	505
28	129	106	88	182	505
29	142	130	66	162	500
30	150	115	83	154	502

Apêndice B

Resultados dos Testes - Injeção de Falhas

Os apêndices deste trabalho mostram os resultados individuais dos testes que injetaram falhas nas regiões de memória e nas leituras dos sensores.

As siglas correspondem a:

- ANE: Arquivo Não Encontrado.
- CI: CRC Incorreto.
- DI: Dados incorretos.
- T: Total de defeitos.

O resultado dos 30 testes é representado por:

$$M \pm IC(NS) \tag{B.1}$$

onde:

- M = média
- IC = Intervalo de Confiança
- NS = Nível de confiança

Estação Meteorológica Sem Técnicas de Tolerância a Falhas

Falhas inseridas na região de memória	:	Nenhuma
Quantidade de ciclos em cada teste	:	60
Quantidade de falhas injetadas na memória em cada round	:	**0**

Taxa de falhas nas leituras dos sensores:				**0%**	
Número do Teste	**Tipos de Erros**				**Tempo**
	ANE	**CI**	**DI**	**T**	
1	0	0	0	0	00:04:59
2	0	0	0	0	00:04:57
3	0	0	0	0	00:04:57
4	0	0	0	0	00:04:58
5	0	0	0	0	00:04:59
6	0	0	0	0	00:04:56
7	0	0	0	0	00:04:58
8	0	0	0	0	00:05:00
9	0	0	0	0	00:04:58
10	0	0	0	0	00:04:58
11	0	0	0	0	00:04:57
12	0	0	0	0	00:04:57
13	0	0	0	0	00:04:58
14	0	0	0	0	00:04:56
15	0	0	0	0	00:04:59
16	0	0	0	0	00:04:59
17	0	0	0	0	00:04:58
18	0	0	0	0	00:04:56
19	0	0	0	0	00:05:00
20	0	0	0	0	00:04:59
21	0	0	0	0	00:04:59
22	0	0	0	0	00:04:56
23	0	0	0	0	00:04:56
24	0	0	0	0	00:05:00
25	0	0	0	0	00:04:58
26	0	0	0	0	00:04:59
27	0	0	0	0	00:04:58
28	0	0	0	0	00:04:58
29	0	0	0	0	00:04:56
30	0	0	0	0	00:04:57
	Tempo Total				02:28:56
	M ± IC (NS)				**DP**
ANE	0,00				0,00
CI	0,00				0,00
DI	0,00				0,00
T	0,00				0,00

Taxa de falhas nas leituras dos sensores:				**25%**	
Número do Teste	**Tipos de Erros**				**Tempo**
	ANE	**CI**	**DI**	**T**	
1	0	0	59	59	00:12:38
2	0	0	53	53	00:11:32
3	0	0	52	52	00:11:26
4	0	0	54	54	00:11:59
5	0	0	60	60	00:12:45
6	0	0	55	55	00:11:57
7	0	0	59	59	00:12:47
8	0	0	60	60	00:12:41
9	0	0	58	58	00:12:23
10	0	0	59	59	00:12:35
11	0	0	56	56	00:11:59
12	0	0	55	55	00:11:52
13	0	0	56	56	00:12:01
14	0	0	55	55	00:11:56
15	0	0	58	58	00:12:23
16	0	0	59	59	00:12:25
17	0	0	58	58	00:12:44
18	0	0	55	55	00:11:55
19	0	0	57	57	00:12:19
20	0	0	55	55	00:12:07
21	0	0	53	53	00:11:56
22	0	0	52	52	00:11:33
23	0	0	55	55	00:11:49
24	0	0	57	57	00:12:12
25	0	0	52	52	00:11:41
26	0	0	55	55	00:11:44
27	0	0	57	57	00:12:06
28	0	0	58	58	00:12:18
29	0	0	59	59	00:12:22
30	0	0	60	60	00:12:47
	Tempo Total				06:04:52
	M ± IC (NS)				**DP**
ANE	0,00				0,00
CI	0,00				0,00
DI	56,37 ± 1,19 (0,01)				2,53
T	56,37 ± 1,19 (0,01)				2,53

Estação Meteorológica Com Técnicas de Tolerância a Falhas		
Falhas inseridas na região de memória	:	Nenhuma
Quantidade de ciclos em cada teste	:	60
Quantidade de falhas injetadas na memória em cada round	:	**0**

Taxa de falhas nas leituras dos sensores:				**0%**	
Número do Teste	**Tipos de Erros**				**Tempo**
	ANE	**CI**	**DI**	**T**	
1	0	0	0	0	00:09:54
2	0	0	0	0	00:09:56
3	0	0	0	0	00:09:52
4	0	0	0	0	00:09:57
5	0	0	0	0	00:09:56
6	0	0	0	0	00:09:56
7	0	0	0	0	00:09:57
8	0	0	0	0	00:09:57
9	0	0	0	0	00:09:52
10	0	0	0	0	00:09:55
11	0	0	0	0	00:09:58
12	0	0	0	0	00:09:59
13	0	0	0	0	00:09:59
14	0	0	0	0	00:09:53
15	0	0	0	0	00:09:54
16	0	0	0	0	00:09:51
17	0	0	0	0	00:09:51
18	0	0	0	0	00:09:55
19	0	0	0	0	00:09:52
20	0	0	0	0	00:09:57
21	0	0	0	0	00:09:51
22	0	0	0	0	00:09:58
23	0	0	0	0	00:09:54
24	0	0	0	0	00:09:53
25	0	0	0	0	00:09:54
26	0	0	0	0	00:09:59
27	0	0	0	0	00:09:52
28	0	0	0	0	00:09:51
29	0	0	0	0	00:09:58
30	0	0	0	0	00:09:57
	Tempo Total				04:57:28
	M ± IC (NS)				**DP**
ANE	0,00				0,00
CI	0,00				0,00
DI	0,00				0,00
T	0,00				0,00

Taxa de falhas nas leituras dos sensores:				**25%**	
Número do Teste	**Tipos de Erros**				**Tempo**
	ANE	**CI**	**DI**	**T**	
1	0	0	5	5	00:11:15
2	0	0	3	3	00:10:44
3	0	0	2	2	00:10:29
4	0	0	1	1	00:10:08
5	0	0	4	4	00:10:43
6	0	0	1	1	00:10:14
7	0	0	4	4	00:10:51
8	0	0	2	2	00:10:22
9	0	0	2	2	00:10:26
10	0	0	0	0	00:09:53
11	0	0	3	3	00:10:36
12	0	0	4	4	00:10:39
13	0	0	5	5	00:10:45
14	0	0	0	0	00:09:57
15	0	0	1	1	00:10:11
16	0	0	3	3	00:09:40
17	0	0	5	5	00:10:52
18	0	0	5	5	00:11:03
19	0	0	3	3	00:10:41
20	0	0	3	3	00:10:33
21	0	0	1	1	00:10:06
22	0	0	3	3	00:10:36
23	0	0	4	4	00:10:42
24	0	0	1	1	00:10:15
25	0	0	0	0	00:09:51
26	0	0	1	1	00:10:07
27	0	0	2	2	00:10:19
28	0	0	2	2	00:10:23
29	0	0	7	7	00:11:39
30	0	0	2	2	00:10:24
	Tempo Total				05:14:24
	M ± IC (NS)				**DP**
ANE	0,00				0,00
CI	0,00				0,00
DI	2,63 ± 0,82 (0,01)				1,75
T	2,63 ± 0,82 (0,01)				1,75

Estação Meteorológica Sem Técnicas de Tolerância a Falhas		
Falhas inseridas na região de memória	:	Memória de dados
Quantidade de ciclos em cada teste	:	60
Quantidade de falhas injetadas na memória em cada round	:	**1**

Taxa de falhas nas leituras dos sensores:				**0%**	
Número do Teste	**Tipos de Erros**				**Tempo**
	ANE	**CI**	**DI**	**T**	
1	2	0	4	6	00:05:40
2	0	0	2	2	00:05:10
3	0	0	0	0	00:04:58
4	1	0	5	6	00:05:38
5	0	0	1	1	00:05:05
6	0	0	3	3	00:05:15
7	0	0	2	2	00:05:12
8	1	0	3	4	00:05:29
9	1	0	2	3	00:05:23
10	2	0	3	5	00:05:38
11	0	0	1	1	00:05:06
12	0	0	3	3	00:05:19
13	1	0	4	5	00:05:35
14	0	0	2	2	00:05:16
15	2	0	5	7	00:05:50
16	0	0	1	1	00:05:04
17	0	0	2	2	00:04:07
18	0	0	4	4	00:05:28
19	2	0	5	7	00:05:48
20	0	1	2	3	00:05:19
21	0	0	0	0	00:04:57
22	0	0	1	1	00:05:06
23	1	0	1	2	00:05:14
24	0	0	3	3	00:05:20
25	0	0	2	2	00:05:09
26	0	0	3	3	00:05:21
27	1	0	0	1	00:05:08
28	0	0	2	2	00:05:11
29	1	0	3	4	00:05:29
30	0	0	3	3	00:05:24
	Tempo Total				02:38:39
	M ± IC (NS)				**DP**
ANE	0,50 ± 0,34 (0,01)				0,73
CI	0,03 ± 0,09 (0,01)				0,18
DI	2,40 ± 0,67 (0,01)				1,43
T	2,93 ± 0,90 (0,01)				1,91

Taxa de falhas nas leituras dos sensores:				**25%**	
Número do Teste	**Tipos de Erros**				**Tempo**
	ANE	**CI**	**DI**	**T**	
1	0	0	57	57	00:12:18
2	0	0	52	52	00:11:39
3	0	0	56	56	00:12:01
4	0	0	57	57	00:12:14
5	2	0	54	56	00:12:01
6	1	0	56	57	00:12:09
7	0	0	53	53	00:11:56
8	0	0	57	57	00:12:11
9	0	0	57	57	00:12:05
10	0	0	58	58	00:12:17
11	2	0	57	59	00:12:25
12	0	0	58	58	00:12:25
13	1	0	53	54	00:11:46
14	0	0	55	55	00:11:59
15	0	0	58	58	00:12:16
16	0	0	56	56	00:12:00
17	4	0	56	60	00:12:29
18	0	0	60	60	00:12:23
19	1	0	52	53	00:11:25
20	0	0	51	51	00:11:18
21	1	0	51	52	00:11:15
22	1	0	56	57	00:12:09
23	0	0	58	58	00:12:16
24	2	0	54	56	00:11:50
25	0	0	53	53	00:12:00
26	0	0	60	60	00:12:37
27	1	0	59	60	00:12:43
28	0	0	51	51	00:11:28
29	2	0	58	60	00:12:31
30	0	0	60	60	00:12:29
	Tempo Total				06:02:35
	M ± IC (NS)				**DP**
ANE	0,60 ± 0,46 (0,01)				0,97
CI	0,00				0,00
DI	56,37 ± 1,19 (0,01)				2,78
T	56,37 ± 1,19 (0,01)				2,86

Estação Meteorológica Com Técnicas de Tolerância a Falhas		
Falhas inseridas na região de memória	:	Memória de dados
Quantidade de ciclos em cada teste	:	60
Quantidade de falhas injetadas na memória em cada round	:	**1**

Taxa de falhas nas leituras dos sensores:				**0%**	
Número do Teste	**Tipos de Erros**				**Tempo**
	ANE	**CI**	**DI**	**T**	
1	0	0	0	0	00:09:59
2	0	0	0	0	00:09:53
3	0	0	0	0	00:09:53
4	0	0	0	0	00:09:59
5	0	0	0	0	00:09:56
6	0	0	0	0	00:09:58
7	0	0	0	0	00:09:55
8	0	0	0	0	00:09:57
9	0	0	0	0	00:09:55
10	0	0	0	0	00:09:58
11	0	0	0	0	00:09:59
12	0	0	0	0	00:09:56
13	0	0	0	0	00:09:59
14	0	0	0	0	00:09:53
15	0	0	0	0	00:09:54
16	0	0	0	0	00:09:51
17	0	0	0	0	00:09:51
18	0	0	0	0	00:09:52
19	0	0	0	0	00:09:54
20	0	0	0	0	00:09:58
21	0	0	0	0	00:09:58
22	0	0	0	0	00:09:59
23	0	0	0	0	00:09:58
24	0	0	0	0	00:09:53
25	0	0	0	0	00:09:59
26	0	0	0	0	00:09:55
27	0	0	0	0	00:09:56
28	0	0	0	0	00:09:51
29	0	0	0	0	00:09:57
30	0	0	0	0	00:09:58
	Tempo Total				04:57:54
	M ± IC (NS)				**DP**
ANE	0,00				0,00
CI	0,00				0,00
DI	0,00				0,00
T	0,00				0,00

Taxa de falhas nas leituras dos sensores:				**25%**	
Número do Teste	**Tipos de Erros**				**Tempo**
	ANE	**CI**	**DI**	**T**	
1	0	0	1	1	00:10:04
2	0	0	1	1	00:10:09
3	0	0	0	0	00:09:52
4	0	0	3	3	00:10:21
5	0	0	2	2	00:10:16
6	0	0	3	3	00:10:28
7	0	0	2	2	00:10:19
8	0	0	2	2	00:10:23
9	0	0	1	1	00:10:04
10	0	0	4	4	00:10:38
11	0	0	3	3	00:10:31
12	0	0	1	1	00:10:09
13	0	0	6	6	00:11:01
14	0	0	2	2	00:10:13
15	0	0	3	3	00:10:23
16	0	0	3	3	00:10:25
17	0	0	3	3	00:10:19
18	0	0	2	2	00:10:15
19	0	0	3	3	00:10:32
20	0	0	3	3	00:10:26
21	0	0	8	8	00:11:33
22	0	0	2	2	00:10:14
23	0	0	4	4	00:10:46
24	0	0	0	0	00:09:57
25	0	0	4	4	00:10:38
26	0	0	2	2	00:10:19
27	0	0	3	3	00:10:20
28	0	0	0	0	00:09:55
29	0	0	2	2	00:10:18
30	0	0	2	2	00:10:21
	Tempo Total				05:11:09
	M ± IC (NS)				**DP**
ANE	0,00				0,00
CI	0,00				0,00
DI	2,50 ± 0,79 (0,01)				1,68
T	2,50 ± 0,79 (0,01)				1,68

Estação Meteorológica Sem Técnicas de Tolerância a Falhas		
Falhas inseridas na região de memória	:	Memória de dados e memória interna
Quantidade de ciclos em cada teste	:	60
Quantidade de falhas injetadas na memória em cada round	:	**1**

Taxa de falhas nas leituras dos sensores:				**0%**	
Número do Teste	**Tipos de Erros**				**Tempo**
	ANE	**CI**	**DI**	**T**	
1	2	0	2	4	00:05:34
2	3	0	1	4	00:05:28
3	1	0	2	3	00:05:23
4	2	0	3	5	00:05:37
5	4	0	1	5	00:05:39
6	5	0	0	5	00:05:41
7	0	0	2	2	00:05:12
8	2	0	3	5	00:05:41
9	2	0	0	2	00:05:14
10	3	0	4	7	00:05:56
11	4	1	2	7	00:05:51
12	5	0	1	6	00:05:50
13	1	0	3	4	00:05:25
14	0	0	3	3	00:05:18
15	2	0	0	2	00:05:11
16	3	0	2	5	00:05:43
17	6	0	2	8	00:06:10
18	3	0	2	5	00:05:45
19	0	0	4	4	00:05:30
20	1	1	3	5	00:05:30
21	1	0	0	1	00:05:06
22	2	0	2	4	00:05:31
23	1	0	2	3	00:05:25
24	3	0	1	4	00:05:35
25	0	0	0	0	00:04:59
26	1	0	3	4	00:05:30
27	2	0	0	2	00:05:14
28	0	0	3	3	00:05:23
29	7	0	1	8	00:06:06
30	2	0	3	5	00:05:37
	Tempo Total				02:46:04
	M ± IC (NS)				**DP**
ANE	2,27 ± 0,86 (0,01)				1,82
CI	0,07 ± 0,12 (0,01)				0,25
DI	1,83 ± 0,58 (0,01)				1,23
T	4,17 ± 0,9 (0,01)				1,91

Taxa de falhas nas leituras dos sensores:				**25%**	
Número do Teste	**Tipos de Erros**				**Tempo**
	ANE	**CI**	**DI**	**T**	
1	1	0	57	58	00:11:48
2	0	0	56	56	00:11:32
3	4	0	55	59	00:11:36
4	3	0	57	60	00:11:59
5	3	0	57	60	00:12:45
6	7	0	53	60	00:11:57
7	1	0	58	59	00:12:47
8	2	0	53	55	00:12:32
9	4	0	55	59	00:12:23
10	2	0	58	60	00:12:35
11	2	0	58	60	00:11:59
12	3	0	56	59	00:11:52
13	0	0	56	56	00:12:01
14	1	0	57	58	00:11:56
15	2	0	57	59	00:12:13
16	0	0	57	57	00:12:15
17	1	0	55	56	00:12:44
18	3	0	54	57	00:11:55
19	8	0	52	60	00:12:19
20	2	0	58	60	00:12:07
21	2	0	58	60	00:11:56
22	3	0	56	59	00:11:33
23	1	0	55	56	00:11:49
24	4	0	56	60	00:12:12
25	3	0	52	55	00:11:41
26	4	0	56	60	00:11:44
27	3	0	57	60	00:12:06
28	1	0	59	60	00:12:18
29	0	0	59	59	00:12:22
30	2	0	52	54	00:12:17
	Tempo Total				06:03:13
	M ± IC (NS)				**DP**
ANE	2,4 ± 0,88 (0,01)				1,87
CI	0,00				0,00
DI	55,97 ± 0,95 (0,01)				2,03
T	58,37 ± 0,89 (0,01)				1,88

Estação Meteorológica Com Técnicas de Tolerância a Falhas		
Falhas inseridas na região de memória	:	Memória de dados e memória interna
Quantidade de ciclos em cada teste	:	60
Quantidade de falhas injetadas na memória em cada round	:	**1**

Taxa de falhas nas leituras dos sensores:				**0%**	
Número do Teste	**Tipos de Erros**				**Tempo**
	ANE	**CI**	**DI**	**T**	
1	3	0	0	3	00:10:29
2	1	0	0	1	00:10:08
3	0	0	0	0	00:09:58
4	2	0	0	2	00:10:12
5	2	0	0	2	00:10:18
6	4	0	0	4	00:10:48
7	3	0	0	3	00:10:38
8	2	0	0	2	00:10:22
9	0	0	0	0	00:09:53
10	6	0	0	6	00:11:20
11	1	0	0	1	00:10:09
12	0	0	0	0	00:09:51
13	0	0	0	0	00:09:54
14	3	0	0	3	00:10:44
15	1	0	0	1	00:10:06
16	2	0	0	2	00:10:18
17	3	0	0	3	00:10:26
18	3	0	0	3	00:10:30
19	0	0	0	0	00:09:55
20	5	0	0	5	00:10:58
21	2	0	0	2	00:10:21
22	1	0	0	1	00:10:05
23	2	0	0	2	00:10:12
24	3	0	0	3	00:10:24
25	0	0	0	0	00:09:53
26	4	0	0	4	00:10:55
27	1	0	0	1	00:10:09
28	5	0	0	5	00:11:01
29	0	0	0	0	00:09:54
30	2	0	0	2	00:10:20
	Tempo Total				05:10:11
	M ± IC (NS)				**DP**
ANE	2,03 ± 0,79 (0,01)				1,67
CI	0,00				0,00
DI	0,00				0,00
T	2,03 ± 0,79 (0,01)				1,67

Taxa de falhas nas leituras dos sensores:				**25%**	
Número do Teste	**Tipos de Erros**				**Tempo**
	ANE	**CI**	**DI**	**T**	
1	0	0	2	2	00:10:17
2	3	0	4	7	00:11:26
3	2	0	1	3	00:10:33
4	2	0	2	4	00:10:39
5	0	0	2	2	00:10:15
6	1	0	3	4	00:10:34
7	4	0	3	7	00:11:24
8	1	0	1	2	00:10:08
9	2	0	2	4	00:10:30
10	2	0	1	3	00:10:28
11	0	0	4	4	00:10:36
12	3	0	3	6	00:11:15
13	2	0	0	2	00:10:14
14	0	0	1	1	00:10:10
15	1	0	1	2	00:10:18
16	4	0	3	7	00:11:28
17	0	0	5	5	00:11:06
18	3	0	1	4	00:10:30
19	2	0	3	5	00:10:49
20	0	0	1	1	00:10:06
21	4	0	2	6	00:11:09
22	2	0	3	5	00:10:55
23	1	0	6	7	00:11:22
24	1	0	3	4	00:10:48
25	5	0	4	9	00:11:51
26	3	0	0	3	00:10:27
27	2	0	4	6	00:11:08
28	2	0	0	2	00:10:22
29	3	0	2	5	00:10:55
30	4	0	6	10	00:12:22
	Tempo Total				05:24:05
	M ± IC (NS)				**DP**
ANE	1,97 ± 0,67 (0,01)				1,43
CI	0,00				0,00
DI	2,43 ± 0,77 (0,01)				1,63
T	4,4 ± 1,08 (0,01)				2,30

Estação Meteorológica Sem Técnicas de Tolerância a Falhas		
Falhas inseridas na região de memória	:	Memória de dados
Quantidade de ciclos em cada teste	:	60
Quantidade de falhas injetadas na memória em cada round	:	**2**

Taxa de falhas nas leituras dos sensores:				**0%**	
Número do Teste	**Tipos de Erros**				**Tempo**
	ANE	**CI**	**DI**	**T**	
1	0	0	2	2	00:05:11
2	2	0	7	9	00:05:58
3	1	0	3	4	00:05:29
4	0	0	2	2	00:05:12
5	0	0	1	1	00:05:05
6	1	0	3	4	00:05:29
7	0	0	4	4	00:05:22
8	1	0	2	3	00:05:22
9	3	0	3	6	00:05:23
10	0	0	0	0	00:04:56
11	0	0	2	2	00:05:14
12	0	0	2	2	00:05:09
13	2	0	5	7	00:05:48
14	1	0	4	5	00:05:31
15	1	0	3	4	00:05:22
16	2	0	2	4	00:05:33
17	3	0	5	8	00:05:54
18	0	0	3	3	00:05:19
19	0	0	0	0	00:04:58
20	1	0	3	4	00:05:23
21	2	0	3	5	00:05:39
22	2	0	4	6	00:05:36
23	1	0	4	5	00:05:40
24	1	0	3	4	00:05:28
25	3	0	3	6	00:05:35
26	0	0	2	2	00:05:09
27	0	0	2	2	00:05:12
28	0	0	0	0	00:04:56
29	1	0	4	5	00:05:26
30	1	0	1	2	00:05:13
	Tempo Total				02:41:32
	M ± IC (NS)				**DP**
ANE	0,97 ± 0,47 (0,01)				1,00
CI	0,00				0,00
DI	2,73 ± 0,73 (0,01)				1,55
T	3,7 ± 1,07 (0,01)				2,28

Taxa de falhas nas leituras dos sensores:				**25%**	
Número do Teste	**Tipos de Erros**				**Tempo**
	ANE	**CI**	**DI**	**T**	
1	0	0	56	56	00:11:49
2	1	0	57	58	00:12:10
3	0	0	54	54	00:11:45
4	2	0	52	54	00:11:37
5	0	0	58	58	00:12:05
6	3	0	56	59	00:12:11
7	1	0	54	55	00:11:46
8	0	0	57	57	00:12:09
9	0	0	57	57	00:12:03
10	1	0	52	53	00:11:38
11	2	0	54	56	00:11:53
12	0	0	59	59	00:12:30
13	1	0	55	56	00:11:56
14	0	0	59	59	00:11:46
15	0	0	57	57	00:12:06
16	4	0	54	58	00:12:15
17	1	0	55	56	00:11:58
18	0	0	56	56	00:12:01
19	0	0	58	58	00:12:21
20	0	1	57	58	00:12:19
21	1	0	55	56	00:11:53
22	3	0	56	59	00:12:27
23	1	0	54	55	00:11:40
24	0	0	54	54	00:11:36
25	0	0	55	55	00:11:48
26	1	0	52	53	00:11:25
27	0	0	55	55	00:11:45
28	0	0	51	51	00:11:18
29	0	0	56	56	00:12:24
30	1	0	52	53	00:11:58
	Tempo Total				05:58:32
	M ± IC (NS)				**DP**
ANE	0,77 ± 0,5 (0,01)				1,07
CI	0,03 ± 0,09 (0,01)				0,18
DI	55,23 ± 1 (0,01)				2,13
T	56,03 ± 0,98 (0,01)				2,09

Estação Meteorológica Com Técnicas de Tolerância a Falhas		
Falhas inseridas na região de memória	:	Memória de dados
Quantidade de ciclos em cada teste	:	60
Quantidade de falhas injetadas na memória em cada round	:	**2**

Taxa de falhas nas leituras dos sensores:				**0%**	
Número do Teste	**Tipos de Erros**				**Tempo**
	ANE	**CI**	**DI**	**T**	
1	0	0	0	0	00:09:59
2	0	0	0	0	00:09:58
3	0	0	0	0	00:09:54
4	0	0	0	0	00:09:58
5	0	0	0	0	00:09:53
6	0	0	0	0	00:09:59
7	0	0	0	0	00:09:57
8	0	0	0	0	00:09:58
9	0	0	0	0	00:09:55
10	0	0	0	0	00:09:58
11	0	0	0	0	00:09:52
12	0	0	0	0	00:09:55
13	0	0	0	0	00:09:52
14	0	0	0	0	00:09:57
15	0	0	0	0	00:09:50
16	0	0	0	0	00:09:58
17	0	0	0	0	00:09:58
18	0	0	0	0	00:09:53
19	0	0	0	0	00:09:58
20	0	0	0	0	00:09:55
21	0	0	0	0	00:09:51
22	0	0	0	0	00:09:55
23	0	0	0	0	00:09:59
24	0	0	0	0	00:09:51
25	0	0	0	0	00:09:54
26	0	0	0	0	00:09:53
27	0	0	0	0	00:09:55
28	0	0	0	0	00:09:57
29	0	0	0	0	00:09:58
30	0	0	0	0	00:09:59
	Tempo Total				04:57:49
	M ± IC (NS)				**DP**
ANE	0,00				0,00
CI	0,00				0,00
DI	0,00				0,00
T	0,00				0,00

Taxa de falhas nas leituras dos sensores:				**25%**	
Número do Teste	**Tipos de Erros**				**Tempo**
	ANE	**CI**	**DI**	**T**	
1	0	0	3	3	00:10:28
2	0	0	7	7	00:11:21
3	0	0	3	3	00:10:24
4	0	0	3	3	00:10:30
5	0	0	3	3	00:10:33
6	0	0	4	4	00:10:33
7	0	0	1	1	00:10:06
8	0	0	2	2	00:10:13
9	0	0	5	5	00:10:55
10	0	0	4	4	00:10:41
11	0	0	2	2	00:10:11
12	0	0	0	0	00:09:52
13	0	0	2	2	00:10:16
14	0	0	0	0	00:09:54
15	0	0	2	2	00:10:14
16	0	0	2	2	00:10:15
17	0	0	4	4	00:10:42
18	0	0	4	4	00:10:37
19	0	0	2	2	00:10:13
20	0	0	3	3	00:10:20
21	0	0	2	2	00:10:17
22	0	0	3	3	00:10:28
23	0	0	3	3	00:10:23
24	0	0	2	2	00:10:11
25	0	0	2	2	00:10:11
26	0	0	4	4	00:10:39
27	0	0	3	3	00:10:35
28	0	0	5	5	00:11:01
29	0	0	2	2	00:10:18
30	0	0	1	1	00:10:09
	Tempo Total				05:12:30
	M ± IC (NS)				**DP**
ANE	0,00				0,00
CI	0,00				0,00
DI	2,77 ± 0,7 (0,01)				1,48
T	2,77 ± 0,7 (0,01)				1,48

Estação Meteorológica Sem Técnicas de Tolerância a Falhas		
Falhas inseridas na região de memória	:	Memória de dados e memória interna
Quantidade de ciclos em cada teste	:	60
Quantidade de falhas injetadas na memória em cada round	:	**2**

Taxa de falhas nas leituras dos sensores:				**0%**	
Número do Teste	**Tipos de Erros**				**Tempo**
	ANE	**CI**	**DI**	**T**	
1	5	0	3	8	00:06:02
2	1	0	2	3	00:05:18
3	2	0	1	3	00:05:19
4	2	0	1	3	00:05:14
5	3	0	6	9	00:06:05
6	3	0	0	3	00:05:18
7	5	0	2	7	00:05:45
8	2	0	2	4	00:05:25
9	0	0	1	1	00:05:08
10	2	0	1	3	00:05:16
11	4	0	3	7	00:05:49
12	2	0	1	3	00:05:25
13	3	0	4	7	00:05:49
14	2	0	6	8	00:05:51
15	6	1	1	8	00:05:56
16	5	0	4	9	00:06:10
17	5	0	3	8	00:06:00
18	3	0	1	4	00:05:30
19	1	0	1	2	00:05:14
20	3	0	0	3	00:05:19
21	3	0	3	6	00:05:45
22	0	1	0	1	00:05:05
23	1	0	3	4	00:05:29
24	2	0	1	3	00:05:20
25	0	0	2	2	00:05:07
26	1	1	2	4	00:05:31
27	0	0	3	3	00:05:19
28	5	0	1	6	00:05:45
29	2	0	1	3	00:05:19
30	1	0	5	6	00:05:34
	Tempo Total				02:46:07
	M ± IC (NS)				**DP**
ANE	2,47 ± 0,81 (0,01)				1,72
CI	0,1 ± 0,14 (0,01)				0,31
DI	2,13 ± 0,77 (0,01)				1,63
T	4,7 ± 1,15 (0,01)				2,45

Taxa de falhas nas leituras dos sensores:				**25%**	
Número do Teste	**Tipos de Erros**				**Tempo**
	ANE	**CI**	**DI**	**T**	
1	3	0	56	59	00:11:58
2	2	0	55	57	00:11:32
3	1	0	58	59	00:11:51
4	2	0	54	56	00:11:46
5	1	0	57	58	00:12:45
6	4	0	56	60	00:12:03
7	1	0	57	58	00:12:17
8	8	0	52	60	00:12:32
9	0	0	59	59	00:12:23
10	1	0	59	60	00:12:35
11	3	0	57	60	00:12:30
12	1	0	56	57	00:11:52
13	4	0	56	60	00:12:11
14	1	0	52	53	00:11:56
15	2	0	58	60	00:12:23
16	3	0	57	60	00:12:15
17	6	0	54	60	00:12:44
18	4	0	56	60	00:12:35
19	1	0	57	58	00:12:19
20	0	0	55	55	00:12:07
21	5	0	54	59	00:12:06
22	2	0	58	60	00:12:18
23	2	1	52	55	00:11:49
24	1	0	56	57	00:12:12
25	3	0	55	58	00:11:41
26	0	0	57	57	00:11:44
27	6	0	54	60	00:12:06
28	2	0	58	60	00:12:18
29	2	0	58	60	00:12:22
30	2	0	55	57	00:12:01
	Tempo Total				06:05:11
	M ± IC (NS)				**DP**
ANE	2,43 ± 0,9 (0,01)				1,92
CI	0,03 ± 0,09 (0,01)				0,18
DI	55,93 ± 0,92 (0,01)				1,96
T	58,4 ± 0,89 (0,01)				1,89

Estação Meteorológica Com Técnicas de Tolerância a Falhas		
Falhas inseridas na região de memória	:	Memória de dados e memória interna
Quantidade de ciclos em cada teste	:	60
Quantidade de falhas injetadas na memória em cada round	:	**2**

Taxa de falhas nas leituras dos sensores:				**0%**	
Número do Teste	**Tipos de Erros**				**Tempo**
	ANE	**CI**	**DI**	**T**	
1	2	0	0	2	00:10:25
2	2	0	0	2	00:10:18
3	2	0	0	2	00:10:21
4	1	0	0	1	00:10:06
5	1	0	0	1	00:10:09
6	4	0	0	4	00:10:53
7	2	0	0	2	00:10:20
8	4	0	0	4	00:10:52
9	2	0	0	2	00:10:18
10	0	0	0	0	00:09:52
11	1	0	0	1	00:10:05
12	0	0	0	0	00:09:58
13	1	0	0	1	00:10:06
14	0	0	0	0	00:09:56
15	3	0	0	3	00:10:41
16	3	0	0	3	00:10:38
17	4	0	0	4	00:10:50
18	1	0	0	1	00:10:08
19	0	0	0	0	00:09:52
20	1	0	0	1	00:10:07
21	1	0	0	1	00:10:09
22	0	0	0	0	00:09:55
23	4	0	0	4	00:10:56
24	2	0	0	2	00:10:16
25	4	0	0	4	00:10:46
26	2	0	0	2	00:10:21
27	5	1	0	6	00:11:15
28	1	0	0	1	00:10:05
29	1	0	0	1	00:10:06
30	4	0	0	4	00:10:53
	Tempo Total				05:10:37
	M ± IC (NS)				**DP**
ANE	1,93 ± 0,7 (0,01)				1,48
CI	0,03 ± 0,09 (0,01)				0,18
DI	0,00				0,00
T	1,97 ± 0,74 (0,01)				1,56

Taxa de falhas nas leituras dos sensores:				**25%**	
Número do Teste	**Tipos de Erros**				**Tempo**
	ANE	**CI**	**DI**	**T**	
1	2	0	4	6	00:11:29
2	0	0	2	2	00:10:23
3	3	0	2	5	00:11:11
4	1	0	1	2	00:10:25
5	3	0	2	5	00:11:16
6	1	0	0	1	00:10:02
7	1	0	4	5	00:11:03
8	1	0	2	3	00:10:38
9	1	0	2	3	00:10:37
10	0	1	4	5	00:10:59
11	2	0	0	2	00:10:20
12	3	0	0	3	00:10:39
13	1	0	0	1	00:10:04
14	2	0	2	4	00:10:49
15	4	0	3	7	00:11:38
16	3	0	0	3	00:10:40
17	1	0	1	2	00:10:24
18	1	0	1	2	00:10:28
19	1	0	2	3	00:10:40
20	5	0	1	6	00:11:18
21	2	0	1	3	00:10:38
22	0	0	2	2	00:10:24
23	3	0	2	5	00:11:05
24	3	0	3	6	00:11:20
25	0	0	2	2	00:10:26
26	1	0	1	2	00:10:18
27	0	0	1	1	00:10:01
28	0	1	0	1	00:10:08
29	0	0	1	1	00:10:06
30	3	0	1	4	00:10:56
	Tempo Total				05:20:25
	M ± IC (NS)				**DP**
ANE	1,6 ± 0,64 (0,01)				1,35
CI	0,07 ± 0,12 (0,01)				0,25
DI	1,57 ± 0,56 (0,01)				1,19
T	3,23 ± 0,83 (0,01)				1,77

Estação Meteorológica Sem Técnicas de Tolerância a Falhas		
Falhas inseridas na região de memória	:	Memória de dados
Quantidade de ciclos em cada teste	:	60
Quantidade de falhas injetadas na memória em cada round	:	**4**

Taxa de falhas nas leituras dos sensores:				**0%**	
Número do Teste	**Tipos de Erros**				**Tempo**
	ANE	**CI**	**DI**	**T**	
1	2	0	3	5	00:05:31
2	2	0	4	6	00:05:45
3	1	0	3	4	00:05:27
4	2	1	3	6	00:05:45
5	3	0	4	7	00:05:46
6	1	0	4	5	00:05:33
7	0	0	3	3	00:05:22
8	2	0	9	11	00:06:21
9	3	0	4	7	00:05:45
10	1	0	5	6	00:05:41
11	1	0	3	4	00:05:22
12	2	0	2	4	00:05:29
13	2	0	3	5	00:05:38
14	2	0	5	7	00:05:45
15	3	1	5	9	00:06:08
16	2	0	4	6	00:05:43
17	3	0	1	4	00:05:33
18	4	1	4	9	00:06:10
19	2	0	5	7	00:05:45
20	3	0	8	11	00:06:17
21	1	0	4	5	00:05:35
22	3	0	4	7	00:05:52
23	0	0	3	3	00:05:21
24	1	0	1	2	00:05:11
25	1	0	4	5	00:05:40
26	0	0	3	3	00:05:19
27	3	0	7	10	00:06:11
28	0	0	3	3	00:05:16
29	2	0	3	5	00:05:37
30	1	0	5	6	00:05:39
	Tempo Total				02:50:27
	M ± IC (NS)				**DP**
ANE	1,77 ± 0,5 (0,01)				1,07
CI	0,1 ± 0,14 (0,01)				0,31
DI	3,97 ± 0,81 (0,01)				1,73
T	1,97 ± 0,74 (0,01)				2,36

Taxa de falhas nas leituras dos sensores:				**25%**	
Número do Teste	**Tipos de Erros**				**Tempo**
	ANE	**CI**	**DI**	**T**	
1	0	0	57	57	00:11:53
2	3	0	57	60	00:12:27
3	0	0	56	56	00:11:56
4	1	0	53	54	00:11:52
5	1	0	57	58	00:12:00
6	3	0	54	57	00:11:49
7	1	0	58	59	00:12:10
8	0	1	59	60	00:12:11
9	3	0	52	55	00:12:01
10	0	0	58	58	00:12:19
11	3	0	56	59	00:12:16
12	0	0	58	58	00:12:05
13	1	0	59	60	00:12:25
14	3	0	57	60	00:12:09
15	0	0	52	52	00:11:26
16	2	0	53	55	00:11:49
17	3	0	57	60	00:11:57
18	3	0	56	59	00:12:06
19	0	0	56	56	00:11:38
20	4	0	56	60	00:12:15
21	0	1	57	58	00:12:14
22	2	0	58	60	00:12:26
23	0	0	60	60	00:12:30
24	2	0	58	60	00:12:35
25	1	0	59	60	00:12:41
26	2	0	52	54	00:11:42
27	1	0	53	54	00:11:41
28	2	0	56	58	00:12:03
29	4	0	51	55	00:11:55
30	1	0	58	59	00:12:01
	Tempo Total				06:02:32
	M ± IC (NS)				**DP**
ANE	1,53 ± 0,63 (0,01)				1,33
CI	0,07 ± 0,12 (0,01)				0,25
DI	56,1 ± 1,16 (0,01)				2,47
T	57,7 ± 1,11 (0,01)				2,37

Estação Meteorológica Com Técnicas de Tolerância a Falhas		
Falhas inseridas na região de memória	:	Memória de dados
Quantidade de ciclos em cada teste	:	60
Quantidade de falhas injetadas na memória em cada round	:	**4**

Taxa de falhas nas leituras dos sensores:				**0%**	
Número do Teste	**Tipos de Erros**				**Tempo**
	ANE	**CI**	**DI**	**T**	
1	0	0	0	0	00:09:54
2	0	0	0	0	00:09:54
3	0	0	0	0	00:09:51
4	0	0	0	0	00:09:57
5	0	0	0	0	00:09:53
6	0	0	0	0	00:09:51
7	0	0	0	0	00:09:56
8	0	0	0	0	00:09:51
9	0	0	0	0	00:09:55
10	0	0	0	0	00:09:59
11	0	0	0	0	00:09:55
12	0	0	0	0	00:09:56
13	0	0	0	0	00:09:56
14	0	0	0	0	00:09:53
15	0	0	0	0	00:09:51
16	0	0	0	0	00:09:51
17	1	0	0	1	00:10:09
18	0	0	0	0	00:09:59
19	0	0	0	0	00:09:56
20	0	0	0	0	00:09:57
21	0	0	0	0	00:09:51
22	0	0	1	1	00:10:07
23	0	0	0	0	00:09:52
24	0	0	0	0	00:09:54
25	0	0	0	0	00:09:50
26	0	0	0	0	00:09:52
27	0	0	0	0	00:09:58
28	0	0	0	0	00:09:50
29	0	0	0	0	00:09:57
30	0	0	0	0	00:09:51
	Tempo Total				04:57:26
	M ± IC (NS)				**DP**
ANE	0,03 ± 0,09 (0,01)				0,18
CI	0,00				0,00
DI	0,03 ± 0,09 (0,01)				0,18
T	0,07 ± 0,12 (0,01)				0,25

Taxa de falhas nas leituras dos sensores:				**25%**	
Número do Teste	**Tipos de Erros**				**Tempo**
	ANE	**CI**	**DI**	**T**	
1	0	0	2	2	00:10:18
2	0	0	1	1	00:10:08
3	0	0	3	3	00:10:32
4	0	0	0	0	00:09:54
5	0	0	2	2	00:10:22
6	0	0	4	4	00:10:44
7	0	0	1	1	00:10:10
8	0	0	2	2	00:10:21
9	0	0	2	2	00:10:17
10	0	0	4	4	00:10:52
11	0	0	3	3	00:10:35
12	0	0	2	2	00:10:24
13	0	0	6	6	00:11:21
14	0	0	4	4	00:10:55
15	0	0	3	3	00:10:39
16	0	0	3	3	00:10:35
17	0	0	5	5	00:11:11
18	0	0	2	2	00:10:19
19	0	0	3	3	00:10:37
20	0	0	3	3	00:10:38
21	0	0	1	1	00:10:09
22	0	0	0	0	00:09:58
23	0	0	4	4	00:10:00
24	0	0	0	0	00:09:54
25	0	0	1	1	00:10:03
26	0	0	3	3	00:10:29
27	0	0	3	3	00:10:34
28	0	0	4	4	00:10:41
29	0	0	1	1	00:10:04
30	0	0	2	2	00:10:23
	Tempo Total				05:13:07
	M ± IC (NS)				**DP**
ANE	0,00				0,00
CI	0,00				0,00
DI	2,47 ± 0,7 (0,01)				1,48
T	2,47 ± 0,7 (0,01)				1,48

Estação Meteorológica Sem Técnicas de Tolerância a Falhas		
Falhas inseridas na região de memória	:	Memória de dados e memória interna
Quantidade de ciclos em cada teste	:	60
Quantidade de falhas injetadas na memória em cada round	:	**4**

Taxa de falhas nas leituras dos sensores:				**0%**	
Número do Teste	**Tipos de Erros**				**Tempo**
	ANE	**CI**	**DI**	**T**	
1	3	0	2	5	00:05:37
2	6	0	7	13	00:06:35
3	3	0	1	4	00:05:26
4	3	0	4	7	00:05:55
5	2	0	4	6	00:05:40
6	0	0	5	5	00:05:33
7	9	0	3	12	00:06:29
8	5	0	0	5	00:05:40
9	3	0	2	5	00:05:35
10	3	0	1	4	00:05:25
11	2	0	0	2	00:05:17
12	9	0	2	11	00:06:14
13	1	0	3	4	00:05:29
14	3	0	6	9	00:06:02
15	5	0	7	12	00:06:14
16	1	0	3	4	00:05:30
17	7	1	3	11	00:06:25
18	0	0	8	8	00:05:58
19	4	0	1	5	00:05:35
20	5	0	4	9	00:05:59
21	6	0	1	7	00:05:56
22	6	0	7	13	00:06:33
23	4	0	3	7	00:05:51
24	2	0	4	6	00:05:44
25	4	0	2	6	00:05:43
26	5	0	3	8	00:05:55
27	2	0	5	7	00:05:49
28	2	0	4	6	00:05:47
29	1	0	3	4	00:05:29
30	3	1	0	4	00:05:33
	Tempo Total				02:54:58
	M ± IC (NS)				**DP**
ANE	3,63 ± 1,09 (0,01)				2,33
CI	0,07 ± 0,12 (0,01)				0,25
DI	3,27 ± 1,03 (0,01)				2,20
T	6,97 ± 1,43 (0,01)				3,03

Taxa de falhas nas leituras dos sensores:				**25%**	
Número do Teste	**Tipos de Erros**				**Tempo**
	ANE	**CI**	**DI**	**T**	
1	2	0	58	60	00:12:41
2	5	0	55	60	00:12:28
3	2	0	56	58	00:12:30
4	4	1	55	60	00:12:18
5	4	0	56	60	00:12:27
6	3	0	57	60	00:12:46
7	8	0	52	60	00:12:08
8	4	0	55	59	00:12:03
9	8	0	52	60	00:12:37
10	1	1	55	57	00:11:48
11	8	0	52	60	00:12:09
12	1	0	57	58	00:12:07
13	2	0	54	56	00:12:27
14	4	1	55	60	00:12:49
15	4	0	56	60	00:12:24
16	2	0	57	59	00:12:18
17	6	0	54	60	00:12:13
18	6	0	54	60	00:12:42
19	3	0	55	58	00:12:01
20	4	0	54	58	00:12:15
21	0	0	58	58	00:12:11
22	3	0	56	59	00:12:09
23	2	1	57	60	00:12:33
24	2	0	58	60	00:12:20
25	0	0	59	59	00:12:28
26	2	0	56	58	00:12:05
27	5	0	55	60	00:12:19
28	5	0	55	60	00:12:16
29	3	0	57	60	00:12:30
30	1	1	54	56	00:11:43
	Tempo Total				06:09:45
	M ± IC (NS)				**DP**
ANE	3,47 ± 1,04 (0,01)				2,21
CI	0,17 ± 0,18 (0,01)				0,38
DI	55,47 ± 0,84 (0,01)				1,80
T	59,1 ± 0,58 (0,01)				1,24

Estação Meteorológica Com Técnicas de Tolerância a Falhas		
Falhas inseridas na região de memória	:	Memória de dados e memória interna
Quantidade de ciclos em cada teste	:	60
Quantidade de falhas injetadas na memória em cada round	:	**4**

Taxa de falhas nas leituras dos sensores:				**0%**	
Número do Teste	**Tipos de Erros**				**Tempo**
	ANE	**CI**	**DI**	**T**	
1	3	0	0	3	00:10:47
2	0	0	0	0	00:09:55
3	1	0	0	1	00:10:02
4	7	0	0	7	00:11:39
5	1	0	0	1	00:10:09
6	5	0	0	5	00:11:15
7	6	0	0	6	00:11:23
8	2	0	0	2	00:10:13
9	3	0	0	3	00:10:36
10	0	0	0	0	00:09:58
11	1	0	0	1	00:10:07
12	4	0	0	4	00:11:03
13	4	0	1	5	00:11:25
14	5	0	0	5	00:11:29
15	1	0	0	1	00:10:11
16	1	0	0	1	00:10:04
17	2	0	0	2	00:10:23
18	4	1	0	5	00:11:19
19	2	0	0	2	00:10:17
20	5	0	0	5	00:11:15
21	1	0	0	1	00:10:09
22	2	0	0	2	00:10:27
23	3	0	0	3	00:10:49
24	1	0	0	1	00:10:06
25	4	0	0	4	00:10:56
26	6	0	0	6	00:11:29
27	3	0	0	3	00:10:40
28	1	0	0	1	00:10:12
29	2	0	0	2	00:10:22
30	4	0	0	4	00:11:08
	Tempo Total				05:19:48
	M ± IC (NS)				**DP**
ANE	2,8 ± 0,89 (0,01)				1,90
CI	0,03 ± 0,09 (0,01)				0,18
DI	0,03 ± 0,09 (0,01)				0,18
T	2,87 ± 0,92 (0,01)				1,96

Taxa de falhas nas leituras dos sensores:				**25%**	
Número do Teste	**Tipos de Erros**				**Tempo**
	ANE	**CI**	**DI**	**T**	
1	2	0	2	4	00:10:52
2	4	0	1	5	00:11:07
3	2	0	4	6	00:11:18
4	2	0	0	2	00:10:25
5	1	0	0	1	00:10:15
6	3	0	0	3	00:11:38
7	2	0	2	4	00:10:52
8	4	0	0	4	00:10:53
9	2	0	3	5	00:11:08
10	3	0	2	5	00:11:01
11	2	0	2	4	00:10:47
12	5	0	1	6	00:11:29
13	2	0	5	7	00:11:44
14	3	0	0	3	00:10:34
15	1	0	3	4	00:10:57
16	2	0	1	3	00:10:40
17	4	0	0	4	00:11:07
18	3	0	3	6	00:11:25
19	3	0	1	4	00:10:47
20	5	0	2	7	00:11:46
21	1	0	0	1	00:10:06
22	1	0	4	5	00:11:13
23	3	0	0	3	00:10:36
24	4	0	0	4	00:10:54
25	3	0	1	4	00:10:59
26	4	0	1	5	00:11:06
27	3	0	0	3	00:10:35
28	2	0	1	3	00:10:42
29	3	0	0	3	00:10:41
30	8	0	3	11	00:12:56
	Tempo Total				05:30:33
	M ± IC (NS)				**DP**
ANE	2,9 ± 0,69 (0,01)				1,47
CI	0,00				0,00
DI	1,4 ± 0,68 (0,01)				1,45
T	4,3 ± 0,92 (0,01)				1,95

Estação Meteorológica Sem Técnicas de Tolerância a Falhas		
Falhas inseridas na região de memória	:	Memória de dados
Quantidade de ciclos em cada teste	:	60
Quantidade de falhas injetadas na memória em cada round	:	**8**

Taxa de falhas nas leituras dos sensores:				**0%**	
Número do Teste	**Tipos de Erros**				**Tempo**
	ANE	**CI**	**DI**	**T**	
1	3	0	6	9	00:05:58
2	2	0	7	9	00:06:05
3	1	0	6	7	00:05:44
4	3	0	7	10	00:06:17
5	0	0	6	6	00:05:34
6	1	0	5	6	00:05:50
7	3	1	7	11	00:06:23
8	1	1	5	7	00:05:56
9	0	0	6	6	00:05:44
10	3	1	7	11	00:06:09
11	4	0	5	9	00:06:07
12	2	0	7	9	00:06:08
13	4	0	10	14	00:06:49
14	0	0	5	5	00:05:34
15	1	0	3	4	00:05:28
16	2	0	4	6	00:05:42
17	4	2	6	12	00:06:28
18	3	0	8	11	00:06:18
19	3	0	5	8	00:06:03
20	3	0	8	11	00:06:12
21	0	0	3	3	00:05:18
22	2	0	8	10	00:06:26
23	3	1	5	9	00:06:08
24	3	1	8	12	00:06:32
25	2	0	6	8	00:05:57
26	3	0	8	11	00:06:16
27	4	0	7	11	00:06:29
28	1	0	5	6	00:05:43
29	4	0	9	13	00:06:46
30	2	0	5	7	00:05:44
	Tempo Total				03:01:48
	M ± IC (NS)				**DP**
ANE	2,23 ± 0,61 (0,01)				1,30
CI	0,23 ± 0,24 (0,01)				0,50
DI	6,23 ± 0,78 (0,01)				1,65
T	8,7 ± 1,29 (0,01)				2,74

Taxa de falhas nas leituras dos sensores:				**25%**	
Número do Teste	**Tipos de Erros**				**Tempo**
	ANE	**CI**	**DI**	**T**	
1	2	0	58	60	00:12:30
2	5	0	54	59	00:12:31
3	3	0	56	59	00:12:20
4	2	0	55	57	00:11:55
5	3	0	57	60	00:11:46
6	0	1	58	59	00:12:15
7	4	0	56	60	00:12:32
8	0	0	59	59	00:12:08
9	2	0	58	60	00:11:41
10	3	0	56	59	00:12:10
11	2	1	56	59	00:12:08
12	2	0	57	59	00:12:06
13	1	0	58	59	00:12:15
14	3	0	56	59	00:12:41
15	2	0	58	60	00:12:14
16	3	0	54	57	00:12:11
17	1	0	58	59	00:12:14
18	3	0	55	58	00:12:12
19	2	0	57	59	00:11:58
20	4	1	55	60	00:12:18
21	3	0	57	60	00:12:09
22	0	0	59	59	00:12:09
23	1	0	59	60	00:12:36
24	0	0	58	58	00:12:24
25	2	0	56	58	00:12:24
26	3	0	55	58	00:12:22
27	0	2	55	57	00:12:33
28	3	0	53	56	00:12:02
29	3	0	55	58	00:11:59
30	0	0	59	59	00:12:28
	Tempo Total				06:07:11
	M ± IC (NS)				**DP**
ANE	2,07 ± 0,64 (0,01)				1,36
CI	0,17 ± 0,22 (0,01)				0,46
DI	56,57 ± 0,79 (0,01)				1,68
T	58,8 ± 0,5 (0,01)				1,06

Estação Meteorológica Com Técnicas de Tolerância a Falhas		
Falhas inseridas na região de memória	:	Memória de dados
Quantidade de ciclos em cada teste	:	60
Quantidade de falhas injetadas na memória em cada round	:	**8**

Taxa de falhas nas leituras dos sensores:				**0%**	
Número do Teste	**Tipos de Erros**				**Tempo**
	ANE	**CI**	**DI**	**T**	
1	0	0	0	0	00:09:58
2	0	0	0	0	00:09:55
3	0	0	0	0	00:09:52
4	0	0	0	0	00:09:54
5	0	0	0	0	00:09:59
6	1	0	0	1	00:10:06
7	0	0	0	0	00:09:58
8	1	0	0	1	00:10:10
9	0	0	0	0	00:09:59
10	0	0	0	0	00:09:55
11	0	0	0	0	00:09:51
12	0	0	0	0	00:09:59
13	2	0	0	2	00:10:28
14	0	0	0	0	00:09:59
15	0	0	0	0	00:09:54
16	1	0	0	1	00:10:04
17	0	0	0	0	00:09:50
18	1	0	0	1	00:10:08
19	0	0	0	0	00:09:59
20	1	0	0	1	00:10:05
21	0	0	0	0	00:09:56
22	1	0	0	1	00:10:07
23	0	0	0	0	00:09:55
24	0	0	0	0	00:09:51
25	0	1	0	1	00:10:06
26	0	0	0	0	00:09:53
27	2	0	0	2	00:10:17
28	1	0	0	1	00:10:11
29	0	0	0	0	00:09:54
30	0	0	0	0	00:09:57
	Tempo Total				05:00:10
	M ± IC (NS)				**DP**
ANE	0,37 ± 0,29 (0,01)				0,61
CI	0,03 ± 0,09 (0,01)				0,18
DI	0,00				0,00
T	0,4 ± 0,29 (0,01)				0,62

Taxa de falhas nas leituras dos sensores:				**25%**	
Número do Teste	**Tipos de Erros**				**Tempo**
	ANE	**CI**	**DI**	**T**	
1	0	0	4	4	00:10:57
2	0	0	1	1	00:10:11
3	1	0	2	3	00:10:41
4	0	0	5	5	00:11:12
5	0	0	0	0	00:09:55
6	1	0	1	2	00:10:08
7	0	0	3	3	00:10:42
8	1	0	2	3	00:10:46
9	0	0	3	3	00:10:49
10	0	0	1	1	00:10:19
11	1	0	4	5	00:11:00
12	0	0	1	1	00:10:06
13	0	0	1	1	00:10:11
14	2	0	5	7	00:11:40
15	0	0	4	4	00:10:59
16	3	0	4	7	00:11:46
17	0	0	2	2	00:10:31
18	0	0	2	2	00:10:29
19	1	0	2	3	00:10:45
20	1	0	1	2	00:10:30
21	0	0	2	2	00:10:34
22	1	0	4	5	00:11:13
23	0	0	3	3	00:10:49
24	0	0	2	2	00:10:22
25	1	0	4	5	00:11:36
26	0	0	2	2	00:10:29
27	0	0	6	6	00:11:33
28	1	0	3	4	00:11:03
29	0	0	5	5	00:11:07
30	0	0	3	3	00:10:50
	Tempo Total				05:23:13
	M ± IC (NS)				**DP**
ANE	0,47 ± 0,34 (0,01)				0,73
CI	0,00				0,00
DI	2,73 ± 0,71 (0,01)				1,51
T	3,2 ± 0,85 (0,01)				1,81

Estação Meteorológica Sem Técnicas de Tolerância a Falhas		
Falhas inseridas na região de memória	:	Memória de dados e memória interna
Quantidade de ciclos em cada teste	:	60
Quantidade de falhas injetadas na memória em cada round	:	**8**

Taxa de falhas nas leituras dos sensores:				**0%**	
Número do Teste	**Tipos de Erros**				**Tempo**
	ANE	**CI**	**DI**	**T**	
1	8	0	4	12	00:06:27
2	7	0	7	14	00:06:46
3	2	2	9	13	00:06:28
4	8	1	6	15	00:06:44
5	4	0	2	6	00:05:42
6	4	0	6	10	00:06:20
7	2	0	7	9	00:06:10
8	5	0	1	6	00:05:39
9	9	0	4	13	00:06:34
10	3	0	8	11	00:06:21
11	4	0	7	11	00:06:23
12	8	0	2	10	00:06:18
13	7	2	3	12	00:06:23
14	5	0	4	9	00:05:57
15	4	0	6	10	00:06:11
16	4	1	4	9	00:06:02
17	3	0	8	11	00:06:18
18	7	0	4	11	00:06:21
19	6	0	1	7	00:05:48
20	3	0	6	9	00:06:15
21	3	0	3	6	00:05:42
22	2	1	3	6	00:05:46
23	5	0	9	14	00:06:39
24	3	0	9	12	00:06:30
25	2	0	6	8	00:05:58
26	5	1	8	14	00:06:36
27	3	0	4	7	00:05:51
28	3	1	0	4	00:05:28
29	3	0	4	7	00:05:48
30	5	0	5	10	00:06:10
	Tempo Total				03:05:35
	M ± IC (NS)				**DP**
ANE	4,57 ± 0,97 (0,01)				2,06
CI	0,3 ± 0,28 (0,01)				0,60
DI	5 ± 1,19 (0,01)				2,53
T	9,87 ± 1,36 (0,01)				2,89

Taxa de falhas nas leituras dos sensores:				**25%**	
Número do Teste	**Tipos de Erros**				**Tempo**
	ANE	**CI**	**DI**	**T**	
1	5	0	55	60	00:12:18
2	3	0	56	59	00:11:32
3	9	0	51	60	00:12:26
4	5	0	54	59	00:12:09
5	6	0	54	60	00:12:35
6	4	0	55	59	00:12:22
7	4	0	55	59	00:12:17
8	6	0	54	60	00:12:23
9	3	0	57	60	00:12:25
10	7	0	52	59	00:12:20
11	2	0	57	59	00:12:19
12	4	0	55	59	00:12:24
13	11	0	49	60	00:12:21
14	5	0	55	60	00:11:36
15	4	0	56	60	00:12:33
16	2	0	57	59	00:12:15
17	10	0	50	60	00:12:24
18	7	0	53	60	00:12:25
19	1	0	59	60	00:12:19
20	3	0	57	60	00:12:27
21	2	0	56	58	00:12:12
22	7	0	53	60	00:12:33
23	6	0	54	60	00:12:19
24	5	0	55	60	00:12:12
25	5	0	55	60	00:12:31
26	9	0	51	60	00:12:24
27	8	0	52	60	00:12:26
28	3	0	56	59	00:12:11
29	2	0	55	57	00:11:59
30	4	0	56	60	00:12:17
	Tempo Total				06:08:54
	M ± IC (NS)				**DP**
ANE	5,07 ± 1,2 (0,01)				2,56
CI	0,00				0,00
DI	54,47 ± 1,07 (0,01)				2,29
T	59,53 ± 0,34 (0,01)				0,73

Estação Meteorológica Com Técnicas de Tolerância a Falhas		
Falhas inseridas na região de memória	:	Memória de dados e memória interna
Quantidade de ciclos em cada teste	:	60
Quantidade de falhas injetadas na memória em cada round	:	**8**

Taxa de falhas nas leituras dos sensores:				**0%**	
Número do Teste	**Tipos de Erros**				**Tempo**
	ANE	**CI**	**DI**	**T**	
1	3	0	0	3	00:10:49
2	2	0	0	2	00:10:32
3	1	0	0	1	00:10:15
4	6	0	0	6	00:11:46
5	3	0	0	3	00:10:44
6	1	0	1	2	00:10:30
7	4	0	0	4	00:10:59
8	6	0	2	8	00:12:21
9	0	0	0	0	00:09:57
10	5	0	0	5	00:11:18
11	0	0	0	0	00:09:54
12	1	0	0	1	00:10:11
13	2	0	0	2	00:10:34
14	1	0	0	1	00:10:20
15	0	0	0	0	00:09:55
16	4	0	1	5	00:11:25
17	2	0	0	2	00:10:37
18	9	0	0	9	00:13:26
19	5	0	0	5	00:11:34
20	1	0	0	1	00:10:20
21	4	0	0	4	00:11:04
22	7	0	0	7	00:11:50
23	3	0	0	3	00:10:47
24	0	0	0	0	00:09:54
25	1	0	0	1	00:10:17
26	1	0	0	1	00:10:15
27	8	0	0	8	00:13:01
28	4	0	0	4	00:11:08
29	3	0	1	4	00:11:00
30	5	0	0	5	00:11:21
	Tempo Total				05:28:04
	M ± IC (NS)				**DP**
ANE	3,07 ± 1,16 (0,01)				2,48
CI	0,00				0,00
DI	0,17 ± 0,22 (0,01)				0,46
T	3,23 ± 1,21 (0,01)				2,58

Taxa de falhas nas leituras dos sensores:				**25%**	
Número do Teste	**Tipos de Erros**				**Tempo**
	ANE	**CI**	**DI**	**T**	
1	4	0	2	6	00:11:35
2	1	0	4	5	00:11:31
3	3	0	1	4	00:10:58
4	3	0	5	8	00:13:17
5	0	0	1	1	00:10:14
6	4	0	2	6	00:11:29
7	6	0	2	8	00:12:59
8	1	0	0	1	00:10:19
9	4	0	3	7	00:13:02
10	2	0	4	6	00:11:45
11	3	0	2	5	00:11:24
12	2	0	3	5	00:11:31
13	5	0	5	10	00:14:09
14	4	0	1	5	00:11:29
15	7	0	0	7	00:12:00
16	5	1	0	6	00:11:40
17	1	0	2	3	00:10:51
18	6	0	3	9	00:14:02
19	8	0	4	12	00:14:44
20	4	0	1	5	00:11:15
21	4	0	3	7	00:12:18
22	5	0	2	7	00:12:34
23	2	0	7	9	00:13:51
24	3	0	4	7	00:12:28
25	3	0	2	5	00:11:34
26	7	0	3	10	00:14:20
27	2	0	4	6	00:11:27
28	5	0	5	10	00:14:43
29	5	0	1	6	00:11:33
30	2	0	1	3	00:10:47
	Tempo Total				06:05:49
	M ± IC (NS)				**DP**
ANE	3,7 ± 0,93 (0,01)				1,97
CI	0,03 ± 0,09 (0,01)				0,18
DI	2,57 ± 0,81 (0,01)				1,72
T	6,3 ± 1,2 (0,01)				2,56

Estação Meteorológica Sem Técnicas de Tolerância a Falhas		
Falhas inseridas na região de memória	:	Memória de dados
Quantidade de ciclos em cada teste	:	60
Quantidade de falhas injetadas na memória em cada round	:	**16**

Taxa de falhas nas leituras dos sensores:				**0%**	
Número do Teste	**Tipos de Erros**				**Tempo**
	ANE	**CI**	**DI**	**T**	
1	4	0	10	14	00:06:51
2	3	0	8	11	00:06:22
3	1	0	8	9	00:06:08
4	4	0	7	11	00:06:22
5	2	0	4	6	00:05:41
6	3	1	10	14	00:06:47
7	6	0	4	10	00:06:01
8	5	3	10	18	00:07:21
9	2	0	7	9	00:06:09
10	2	1	9	12	00:06:23
11	5	2	10	17	00:07:15
12	5	1	9	15	00:06:52
13	3	0	8	11	00:06:16
14	2	0	10	12	00:06:27
15	3	1	5	9	00:06:07
16	4	0	7	11	00:06:15
17	1	0	6	7	00:05:55
18	4	0	9	13	00:06:33
19	3	0	5	8	00:05:57
20	4	1	9	14	00:06:40
21	6	0	9	15	00:06:46
22	1	0	7	8	00:05:55
23	6	2	13	21	00:05:34
24	2	0	3	5	00:05:34
25	4	1	9	14	00:06:48
26	2	1	9	12	00:06:23
27	7	1	8	16	00:06:50
28	2	0	7	9	00:06:00
29	3	0	7	10	00:06:15
30	3	0	10	13	00:06:39
	Tempo Total				03:11:06
	M ± IC (NS)				**DP**
ANE	3,4 ± 0,76 (0,01)				1,61
CI	0,5 ± 0,37 (0,01)				0,78
DI	7,9 ± 1,04 (0,01)				2,20
T	11,8 ± 1,71 (0,01)				3,63

Taxa de falhas nas leituras dos sensores:				**25%**	
Número do Teste	**Tipos de Erros**				**Tempo**
	ANE	**CI**	**DI**	**T**	
1	3	0	57	60	00:12:29
2	1	2	56	59	00:12:30
3	2	0	56	58	00:12:14
4	4	0	54	58	00:12:21
5	9	1	50	60	00:12:40
6	2	0	56	58	00:12:13
7	3	2	50	55	00:12:09
8	2	0	58	60	00:12:34
9	2	0	58	60	00:12:02
10	5	1	54	60	00:12:11
11	1	0	59	60	00:12:37
12	3	1	54	58	00:12:03
13	4	1	55	60	00:12:27
14	7	0	53	60	00:12:31
15	3	0	56	59	00:12:28
16	2	1	57	60	00:12:27
17	3	0	54	57	00:12:01
18	2	0	57	59	00:12:22
19	0	1	57	58	00:12:18
20	4	0	56	60	00:12:21
21	0	1	59	60	00:12:45
22	5	0	55	60	00:12:37
23	2	1	57	60	00:12:14
24	7	0	53	60	00:12:44
25	3	1	56	60	00:12:46
26	3	0	56	59	00:12:10
27	3	0	57	60	00:12:24
28	5	0	55	60	00:12:00
29	2	1	57	60	00:12:19
30	1	0	56	57	00:11:55
	Tempo Total				06:10:52
	M ± IC (NS)				**DP**
ANE	3,1 ± 0,96 (0,01)				2,04
CI	0,47 ± 0,3 (0,01)				0,63
DI	55,6 ± 1,02 (0,01)				2,18
T	59,17 ± 0,59 (0,01)				1,26

Estação Meteorológica Com Técnicas de Tolerância a Falhas		
Falhas inseridas na região de memória	:	Memória de dados
Quantidade de ciclos em cada teste	:	60
Quantidade de falhas injetadas na memória em cada round	:	**16**

Taxa de falhas nas leituras dos sensores:				**0%**	
Número do Teste	**Tipos de Erros**				**Tempo**
	ANE	**CI**	**DI**	**T**	
1	3	0	0	3	00:10:57
2	2	0	0	2	00:10:31
3	1	0	0	1	00:10:22
4	0	0	1	1	00:09:59
5	2	1	0	3	00:11:04
6	4	0	1	5	00:11:13
7	1	0	0	1	00:10:12
8	3	0	0	3	00:10:57
9	1	0	1	2	00:10:29
10	2	0	2	4	00:11:31
11	5	0	0	5	00:11:04
12	1	1	0	2	00:10:33
13	0	0	0	0	00:09:53
14	1	0	2	3	00:10:47
15	6	0	0	6	00:11:43
16	0	0	1	1	00:10:21
17	2	0	3	5	00:11:29
18	4	0	0	4	00:11:17
19	0	0	0	0	00:09:56
20	0	0	4	4	00:10:59
21	3	0	1	4	00:11:06
22	3	0	0	3	00:10:51
23	2	0	1	3	00:10:49
24	1	0	0	1	00:10:15
25	0	0	3	3	00:10:52
26	2	1	0	3	00:11:00
27	1	0	0	1	00:10:14
28	0	0	0	0	00:09:59
29	2	0	1	3	00:10:59
30	2	0	2	4	00:11:02
	Tempo Total				05:22:24
	M ± IC (NS)				**DP**
ANE	1,8 ± 0,73 (0,01)				1,56
CI	0,1 ± 0,14 (0,01)				0,31
DI	0,77 ± 0,52 (0,01)				1,10
T	2,67 ± 0,76 (0,01)				1,63

Taxa de falhas nas leituras dos sensores:				**25%**	
Número do Teste	**Tipos de Erros**				**Tempo**
	ANE	**CI**	**DI**	**T**	
1	1	0	3	4	00:11:13
2	4	0	9	13	00:13:40
3	2	0	3	5	00:11:41
4	2	0	3	5	00:11:26
5	0	0	5	5	00:11:30
6	7	0	3	10	00:13:24
7	2	0	7	9	00:13:15
8	1	0	1	2	00:10:36
9	0	1	2	3	00:10:39
10	3	0	3	6	00:11:28
11	3	0	1	4	00:11:01
12	0	0	5	5	00:11:19
13	2	1	3	6	00:11:40
14	0	0	4	4	00:11:18
15	4	0	2	6	00:11:33
16	0	0	5	5	00:11:40
17	2	0	1	3	00:10:55
18	1	0	8	9	00:13:21
19	2	0	3	5	00:11:30
20	6	0	5	11	00:13:19
21	2	0	1	3	00:10:51
22	3	1	3	7	00:12:02
23	3	0	2	5	00:11:36
24	1	1	5	7	00:11:48
25	5	0	2	7	00:12:03
26	0	0	0	0	00:09:56
27	1	0	1	2	00:10:26
28	3	0	4	7	00:11:56
29	4	0	2	6	00:11:43
30	1	0	1	2	00:10:32
	Tempo Total				05:49:21
	M ± IC (NS)				**DP**
ANE	2,17 ± 0,86 (0,01)				1,82
CI	0,13 ± 0,16 (0,01				0,35
DI	3,23 ± 1,02 (0,01)				2,16
T	5,53 ± 1,34 (0,01)				2,85

Estação Meteorológica Sem Técnicas de Tolerância a Falhas		
Falhas inseridas na região de memória	:	Memória de dados e memória interna
Quantidade de ciclos em cada teste	:	60
Quantidade de falhas injetadas na memória em cada round	:	**16**

Taxa de falhas nas leituras dos sensores:				**0%**	
Número do Teste	**Tipos de Erros**				**Tempo**
	ANE	**CI**	**DI**	**T**	
1	6	0	11	17	00:07:05
2	9	0	7	16	00:06:57
3	8	0	5	13	00:06:36
4	10	1	2	13	00:06:31
5	10	0	10	20	00:07:30
6	5	0	10	15	00:07:00
7	12	0	6	18	00:07:08
8	5	1	3	9	00:05:58
9	8	0	7	15	00:06:45
10	12	0	6	18	00:07:12
11	13	0	4	17	00:07:04
12	9	0	7	16	00:06:57
13	7	0	3	10	00:06:12
14	7	0	10	17	00:06:50
15	9	0	9	18	00:07:00
16	9	0	11	20	00:07:33
17	10	0	6	16	00:07:06
18	6	0	6	12	00:06:29
19	7	0	5	12	00:06:34
20	9	0	9	18	00:07:14
21	9	0	2	11	00:06:19
22	6	2	9	17	00:07:18
23	10	0	3	13	00:06:40
24	7	0	9	16	00:07:07
25	9	1	2	12	00:06:25
26	6	0	11	17	00:07:04
27	12	2	3	17	00:07:10
28	5	0	6	11	00:06:13
29	6	1	6	13	00:06:36
30	5	1	5	11	00:06:17
	Tempo Total				03:24:50
	M ± IC (NS)				**DP**
ANE	8,2 ± 1,09 (0,01)				2,31
CI	0,3 ± 0,28 (0,01)				0,60
DI	6,43 ± 1,37 (0,01)				2,92
T	14,93 ± 1,42 (0,01)				3,03

Taxa de falhas nas leituras dos sensores:				**25%**	
Número do Teste	**Tipos de Erros**				**Tempo**
	ANE	**CI**	**DI**	**T**	
1	9	0	51	60	00:12:28
2	5	0	55	60	00:12:32
3	13	0	47	60	00:12:26
4	10	0	50	60	00:12:19
5	9	1	50	60	00:12:45
6	7	0	53	60	00:11:37
7	7	0	53	60	00:12:47
8	14	1	45	60	00:12:32
9	6	0	53	59	00:12:23
10	8	0	52	60	00:12:35
11	7	1	52	60	00:12:18
12	3	0	55	58	00:12:04
13	8	0	52	60	00:12:20
14	11	1	48	60	00:12:26
15	6	0	54	60	00:12:33
16	9	0	51	60	00:12:15
17	16	0	44	60	00:12:44
18	8	0	52	60	00:12:25
19	8	1	51	60	00:12:19
20	11	0	49	60	00:12:17
21	6	0	53	59	00:11:59
22	4	1	55	60	00:12:33
23	7	3	50	60	00:12:29
24	10	0	49	59	00:12:12
25	8	0	52	60	00:12:41
26	7	2	51	60	00:12:26
27	14	0	46	60	00:12:44
28	11	0	49	60	00:12:18
29	5	0	54	59	00:12:22
30	8	1	51	60	00:12:27
	Tempo Total				06:12:16
	M ± IC (NS)				**DP**
ANE	8,5 ± 1,43 (0,01)				3,04
CI	0,4 ± 0,34 (0,01)				0,72
DI	50,9 ± 1,34 (0,01)				2,86
T	59,8 ± 0,23 (0,01)				0,48

Estação Meteorológica Com Técnicas de Tolerância a Falhas		
Falhas inseridas na região de memória	:	Memória de dados e memória interna
Quantidade de ciclos em cada teste	:	60
Quantidade de falhas injetadas na memória em cada round	:	**16**

Taxa de falhas nas leituras dos sensores:				**0%**	
Número do Teste	**Tipos de Erros**				**Tempo**
	ANE	**CI**	**DI**	**T**	
1	4	0	0	4	00:10:57
2	6	0	0	6	00:11:32
3	3	0	0	3	00:10:55
4	8	0	0	8	00:12:29
5	1	0	1	2	00:10:29
6	3	0	0	3	00:10:39
7	3	0	3	6	00:11:20
8	7	1	0	8	00:12:09
9	4	0	0	4	00:11:01
10	9	1	0	10	00:13:28
11	5	0	0	5	00:11:36
12	6	0	0	6	00:11:40
13	2	0	2	4	00:11:03
14	10	0	0	10	00:13:15
15	5	0	0	5	00:11:15
16	3	0	1	4	00:10:53
17	6	2	0	8	00:11:58
18	9	0	0	9	00:12:11
19	4	0	0	4	00:10:59
20	12	0	0	12	00:14:01
21	6	0	0	6	00:11:26
22	3	0	0	3	00:10:44
23	5	0	1	6	00:11:35
24	6	1	0	7	00:11:38
25	8	0	0	8	00:11:50
26	4	0	2	6	00:11:24
27	5	0	2	7	00:11:44
28	6	0	0	6	00:11:23
29	5	0	0	5	00:11:31
30	6	1	0	7	00:11:55
	Tempo Total				05:49:00
	M ± IC (NS)				**DP**
ANE	5,47 ± 1,16 (0,01)				2,47
CI	0,2 ± 0,23 (0,01)				0,48
DI	0,4 ± 0,38 (0,01)				0,81
T	6,07 ± 1,11 (0,01)				2,36

Taxa de falhas nas leituras dos sensores:				**25%**	
Número do Teste	**Tipos de Erros**				**Tempo**
	ANE	**CI**	**DI**	**T**	
1	6	0	4	10	00:13:21
2	2	0	2	4	00:11:18
3	7	1	1	9	00:13:02
4	4	0	2	6	00:12:29
5	10	1	2	13	00:14:48
6	4	1	5	10	00:13:42
7	2	0	2	4	00:11:15
8	5	0	4	9	00:13:21
9	7	0	3	10	00:13:34
10	3	0	3	6	00:12:42
11	4	0	1	5	00:12:25
12	1	0	5	6	00:12:37
13	8	0	2	10	00:13:45
14	6	2	1	9	00:13:30
15	5	0	2	7	00:12:54
16	9	1	1	11	00:13:58
17	5	0	3	8	00:13:16
18	8	0	6	14	00:15:06
19	9	0	0	9	00:13:33
20	2	0	3	5	00:12:34
21	14	0	2	16	00:15:42
22	6	0	1	7	00:12:50
23	6	1	0	7	00:13:18
24	4	0	4	8	00:13:15
25	5	0	3	8	00:13:44
26	5	0	6	11	00:14:11
27	3	0	4	7	00:13:27
28	8	3	5	16	00:15:55
29	8	0	5	13	00:15:08
30	3	0	3	6	00:12:34
	Tempo Total				06:43:14
	M ± IC (NS)				**DP**
ANE	5,63 ± 1,33 (0,01)				2,83
CI	0,33 ± 0,33 (0,01)				0,71
DI	2,83 ± 0,79 (0,01)				1,68
T	8,8 ± 1,52 (0,01)				3,23

Estação Meteorológica Sem Técnicas de Tolerância a Falhas		
Falhas inseridas na região de memória	:	Memória de dados
Quantidade de ciclos em cada teste	:	60
Quantidade de falhas injetadas na memória em cada round	:	**32**

Taxa de falhas nas leituras dos sensores:				**0%**	
Número do Teste	**Tipos de Erros**				**Tempo**
	ANE	**CI**	**DI**	**T**	
1	7	0	8	15	00:06:57
2	10	1	17	28	00:08:20
3	5	1	12	18	00:07:09
4	9	0	24	33	00:08:58
5	6	0	15	21	00:07:27
6	6	0	13	19	00:07:21
7	3	1	19	23	00:07:48
8	14	0	14	28	00:08:29
9	4	2	13	19	00:07:21
10	5	1	18	24	00:08:07
11	4	0	12	16	00:06:55
12	5	0	17	22	00:07:48
13	4	0	11	15	00:06:52
14	8	2	19	29	00:08:36
15	4	0	17	21	00:07:32
16	5	0	10	15	00:06:44
17	10	1	17	28	00:08:19
18	4	0	14	18	00:07:16
19	8	0	15	23	00:07:49
20	5	0	22	27	00:08:13
21	4	0	10	14	00:06:37
22	8	2	12	22	00:07:33
23	4	1	18	23	00:07:49
24	10	0	15	25	00:08:10
25	8	1	21	30	00:08:42
26	17	2	5	24	00:08:01
27	5	0	17	22	00:07:34
28	1	0	12	13	00:06:39
29	8	1	11	20	00:07:29
30	3	0	11	14	00:06:43
	Tempo Total				03:49:18
	M ± IC (NS)				**DP**
ANE	6,47 ± 1,59 (0,01)				3,39
CI	0,53 ± 0,34 (0,01)				0,73
DI	14,63 ± 2 (0,01)				4,25
T	21,63 ± 2,52 (0,01)				5,35

Taxa de falhas nas leituras dos sensores:				**25%**	
Número do Teste	**Tipos de Erros**				**Tempo**
	ANE	**CI**	**DI**	**T**	
1	6	0	54	60	00:12:39
2	2	1	57	60	00:12:40
3	5	1	54	60	00:12:25
4	7	1	52	60	00:12:38
5	10	2	48	60	00:12:46
6	7	1	51	59	00:12:38
7	5	1	54	60	00:12:38
8	8	0	52	60	00:13:09
9	2	1	56	59	00:12:28
10	5	2	53	60	00:12:43
11	2	2	56	60	00:12:32
12	9	0	51	60	00:12:51
13	5	1	54	60	00:12:23
14	5	1	52	58	00:11:58
15	13	0	47	60	00:12:24
16	5	1	54	60	00:12:10
17	7	1	52	60	00:12:12
18	6	2	51	59	00:12:01
19	5	1	54	60	00:12:35
20	3	0	54	57	00:11:59
21	4	0	56	60	00:12:22
22	1	1	58	60	00:12:35
23	7	0	51	58	00:12:18
24	4	0	56	60	00:12:24
25	2	1	57	60	00:12:49
26	6	2	52	60	00:13:09
27	3	0	56	59	00:12:37
28	1	0	59	60	00:12:40
29	11	1	48	60	00:12:34
30	4	0	56	60	00:12:44
	Tempo Total				06:16:01
	M ± IC (NS)				**DP**
ANE	5,33 ± 1,37 (0,01)				2,90
CI	0,8 ± 0,34 (0,01)				0,71
DI	53,5 ± 1,39 (0,01)				2,96
T	59,63 ± 0,36 (0,01)				0,76

Estação Meteorológica Com Técnicas de Tolerância a Falhas		
Falhas inseridas na região de memória	:	Memória de dados
Quantidade de ciclos em cada teste	:	60
Quantidade de falhas injetadas na memória em cada round	:	**32**

Taxa de falhas nas leituras dos sensores:				**0%**	
Número do Teste	**Tipos de Erros**				**Tempo**
	ANE	**CI**	**DI**	**T**	
1	5	0	3	8	00:11:47
2	4	0	3	7	00:11:36
3	1	0	2	3	00:10:42
4	3	1	3	7	00:11:48
5	3	0	1	4	00:11:00
6	2	1	2	5	00:11:17
7	0	0	1	1	00:10:06
8	1	0	1	2	00:10:18
9	7	1	3	11	00:12:42
10	4	0	3	7	00:12:00
11	3	0	1	4	00:10:45
12	6	0	4	10	00:12:25
13	3	0	3	6	00:11:31
14	4	1	1	6	00:11:34
15	2	0	4	6	00:11:24
16	1	0	0	1	00:10:11
17	4	0	5	9	00:11:57
18	5	0	3	8	00:11:56
19	1	1	3	5	00:11:18
20	3	1	3	7	00:12:01
21	0	0	0	0	00:09:52
22	3	0	3	6	00:11:23
23	1	0	0	1	00:10:18
24	1	0	1	2	00:10:26
25	2	0	5	7	00:12:13
26	1	0	5	6	00:11:20
27	5	1	3	9	00:12:24
28	3	0	2	5	00:11:05
29	2	0	1	3	00:10:42
30	6	1	6	13	00:13:18
	Tempo Total				05:29:23
	M ± IC (NS)				**DP**
ANE	2,87 ± 0,87 (0,01)				1,85
CI	0,27 ± 0,21 (0,01)				0,45
DI	2,5 ± 0,75 (0,01)				1,59
T	5,63 ± 1,48 (0,01)				3,16

Taxa de falhas nas leituras dos sensores:				**25%**	
Número do Teste	**Tipos de Erros**				**Tempo**
	ANE	**CI**	**DI**	**T**	
1	4	0	6	10	00:12:29
2	2	0	7	9	00:12:25
3	1	0	3	4	00:11:04
4	5	1	1	7	00:12:15
5	3	0	4	7	00:12:06
6	2	1	9	12	00:13:01
7	3	0	2	5	00:11:24
8	3	0	5	8	00:11:50
9	4	2	4	10	00:12:44
10	2	0	4	6	00:11:28
11	7	0	9	16	00:14:10
12	2	0	5	7	00:12:12
13	6	1	6	13	00:13:25
14	1	0	6	7	00:11:51
15	2	0	8	10	00:12:31
16	4	0	7	11	00:12:48
17	3	0	5	8	00:12:04
18	3	1	2	6	00:11:36
19	5	0	0	5	00:11:26
20	2	0	4	6	00:11:41
21	3	1	10	14	00:13:44
22	1	0	3	4	00:11:42
23	5	2	4	11	00:12:49
24	2	0	7	9	00:12:28
25	3	0	5	8	00:12:25
26	1	0	6	7	00:12:15
27	4	2	6	12	00:13:07
28	4	0	6	10	00:12:38
29	5	0	1	6	00:11:37
30	3	0	4	7	00:11:55
	Tempo Total				06:09:10
	M ± IC (NS)				**DP**
ANE	3,17 ± 0,72 (0,01)				1,53
CI	0,37 ± 0,31 (0,01)				0,67
DI	4,97 ± 1,15 (0,01)				2,44
T	8,5 ± 1,4 (0,01)				2,98

Estação Meteorológica Sem Técnicas de Tolerância a Falhas		
Falhas inseridas na região de memória	:	Memória de dados e memória interna
Quantidade de ciclos em cada teste	:	60
Quantidade de falhas injetadas na memória em cada round	:	**32**

Taxa de falhas nas leituras dos sensores:				**0%**	
Número do Teste	**Tipos de Erros**				**Tempo**
	ANE	**CI**	**DI**	**T**	
1	21	2	11	34	00:09:10
2	19	0	7	26	00:08:11
3	19	3	9	31	00:09:02
4	7	0	9	16	00:07:02
5	15	2	0	17	00:09:41
6	19	1	10	30	00:08:37
7	15	0	12	27	00:08:22
8	17	1	7	25	00:08:06
9	17	0	11	28	00:08:19
10	18	4	11	33	00:09:13
11	24	1	12	37	00:09:38
12	6	0	6	12	00:06:31
13	26	0	10	36	00:09:27
14	19	1	14	34	00:09:06
15	24	3	7	34	00:09:22
16	8	0	4	12	00:06:16
17	23	1	11	35	00:09:15
18	11	0	8	19	00:07:29
19	21	1	12	34	00:09:19
20	24	0	8	32	00:08:47
21	18	1	14	33	00:09:05
22	16	0	18	34	00:09:19
23	20	0	12	32	00:08:54
24	21	0	12	33	00:09:13
25	16	1	10	27	00:08:16
26	18	0	13	31	00:08:45
27	22	0	6	28	00:08:05
28	17	2	16	35	00:09:21
29	27	0	5	32	00:08:54
30	21	0	7	28	00:08:29
	Tempo Total				04:19:14
	M ± IC (NS)				**DP**
ANE	18,3 ± 2,44 (0,01)				5,18
CI	0,8 ± 0,52 (0,01)				1,10
DI	9,73 ± 1,75 (0,01)				3,72
T	28,83 ± 3,29 (0,01)				7,00

Taxa de falhas nas leituras dos sensores:				**25%**	
Número do Teste	**Tipos de Erros**				**Tempo**
	ANE	**CI**	**DI**	**T**	
1	25	0	35	60	00:12:38
2	14	1	45	60	00:12:42
3	19	0	41	60	00:12:36
4	17	0	43	60	00:12:29
5	21	0	39	60	00:12:45
6	18	1	41	60	00:12:27
7	18	0	42	60	00:12:47
8	23	0	37	60	00:12:22
9	28	0	32	60	00:12:35
10	16	2	42	60	00:12:40
11	14	0	46	60	00:12:19
12	15	1	44	60	00:12:32
13	16	0	44	60	00:13:01
14	20	0	40	60	00:12:51
15	8	1	50	59	00:12:13
16	19	0	41	60	00:12:35
17	10	0	50	60	00:12:44
18	13	3	44	60	00:12:25
19	17	0	43	60	00:12:19
20	15	1	44	60	00:12:27
21	21	2	37	60	00:12:43
22	25	0	35	60	00:12:24
23	29	0	31	60	00:12:31
24	22	0	38	60	00:12:31
25	18	1	41	60	00:12:46
26	24	0	36	60	00:12:40
27	9	1	49	59	00:12:06
28	15	0	45	60	00:12:21
29	13	3	44	60	00:12:18
30	16	0	44	60	00:12:29
	Tempo Total				06:16:16
	M ± IC (NS)				**DP**
ANE	17,93 ± 2,45 (0,01)				5,21
CI	0,57 ± 0,42 (0,01)				0,90
DI	41,43 ± 2,25 (0,01)				4,79
T	59,93 ± 0,12 (0,01)				0,25

Estação Meteorológica Com Técnicas de Tolerância a Falhas		
Falhas inseridas na região de memória	:	Memória de dados e memória interna
Quantidade de ciclos em cada teste	:	60
Quantidade de falhas injetadas na memória em cada round	:	**32**

Taxa de falhas nas leituras dos sensores:				**0%**	
Número do Teste	**Tipos de Erros**				**Tempo**
	ANE	**CI**	**DI**	**T**	
1	9	0	1	10	00:12:07
2	13	0	0	13	00:13:23
3	6	0	2	8	00:12:24
4	11	2	1	14	00:13:28
5	15	0	0	15	00:13:57
6	8	0	4	12	00:12:43
7	17	1	0	18	00:14:11
8	5	0	3	8	00:12:31
9	10	0	3	13	00:13:23
10	18	2	1	21	00:15:03
11	12	0	0	12	00:12:53
12	9	0	2	11	00:12:40
13	20	1	0	21	00:15:10
14	13	0	2	15	00:13:45
15	15	0	0	15	00:13:43
16	16	1	3	20	00:13:51
17	22	0	1	23	00:15:49
18	7	0	3	10	00:12:12
19	17	2	0	19	00:14:39
20	15	0	0	15	00:13:50
21	15	1	0	16	00:13:57
22	11	0	1	12	00:13:03
23	20	1	0	21	00:14:53
24	11	0	2	13	00:13:14
25	14	0	0	14	00:13:16
26	16	0	0	16	00:13:54
27	21	1	1	23	00:16:00
28	18	0	0	18	00:14:29
29	20	0	0	20	00:14:41
30	14	0	1	15	00:13:45
	Tempo Total				06:52:54
	M ± IC (NS)				**DP**
ANE	13,93 ± 2,16 (0,01)				4,59
CI	0,4 ± 0,32 (0,01)				0,67
DI	1,03 ± 0,57 (0,01)				1,22
T	15,37 ± 2 (0,01)				4,25

Taxa de falhas nas leituras dos sensores:				**25%**	
Número do Teste	**Tipos de Erros**				**Tempo**
	ANE	**CI**	**DI**	**T**	
1	16	0	5	21	00:14:55
2	8	0	3	11	00:12:51
3	11	2	3	16	00:14:02
4	9	0	4	13	00:13:12
5	10	1	6	17	00:14:15
6	13	1	7	21	00:14:51
7	13	1	3	17	00:14:21
8	20	2	3	25	00:16:10
9	14	0	6	20	00:14:56
10	9	0	5	14	00:13:28
11	19	0	10	29	00:17:24
12	5	0	6	11	00:12:32
13	16	3	4	23	00:15:39
14	17	0	3	20	00:14:41
15	13	1	6	20	00:15:05
16	8	0	4	12	00:12:58
17	14	0	4	18	00:14:34
18	12	0	5	17	00:14:30
19	9	3	8	20	00:15:09
20	16	0	0	16	00:14:01
21	11	0	6	17	00:14:20
22	12	0	4	16	00:14:11
23	8	0	2	10	00:12:32
24	16	0	2	18	00:14:35
25	11	2	5	18	00:14:26
26	15	0	3	18	00:14:35
27	13	0	9	22	00:15:22
28	16	0	4	20	00:14:41
29	10	0	2	12	00:12:58
30	21	1	1	23	00:15:36
	Tempo Total				07:12:50
	M ± IC (NS)				**DP**
ANE	12,83 ± 1,82 (0,01)				3,87
CI	0,57 ± 0,44 (0,01)				0,94
DI	4,43 ± 1,06 (0,01)				2,25
T	17,83 ± 2,07 (0,01)				4,41

Estação Meteorológica Sem Técnicas de Tolerância a Falhas		
Falhas inseridas na região de memória	:	Memória de dados
Quantidade de ciclos em cada teste	:	60
Quantidade de falhas injetadas na memória em cada round	:	**64**

Taxa de falhas nas leituras dos sensores:				**0%**	
Número do Teste	**Tipos de Erros**				**Tempo**
	ANE	**CI**	**DI**	**T**	
1	7	2	31	40	00:09:53
2	5	1	9	15	00:06:56
3	8	0	21	29	00:08:48
4	9	0	27	36	00:09:44
5	8	0	13	21	00:09:35
6	5	1	30	36	00:09:17
7	13	1	29	43	00:10:03
8	6	0	25	31	00:10:31
9	16	0	28	44	00:10:06
10	6	1	25	32	00:09:29
11	11	0	26	37	00:09:36
12	3	0	15	18	00:07:02
13	5	0	39	44	00:10:30
14	8	0	19	27	00:08:15
15	11	0	44	55	00:13:17
16	15	0	35	50	00:11:29
17	9	2	30	41	00:09:59
18	14	0	25	39	00:09:52
19	7	0	37	44	00:10:40
20	8	1	33	42	00:10:12
21	12	2	23	37	00:09:22
22	8	0	21	29	00:08:30
23	15	0	28	43	00:10:34
24	12	0	29	41	00:10:01
25	9	1	27	37	00:09:46
26	14	0	31	45	00:10:31
27	7	0	23	30	00:08:43
28	7	2	29	38	00:09:04
29	18	0	23	41	00:09:56
30	2	0	15	17	00:07:09
	Tempo Total				04:48:50
	M ± IC (NS)				**DP**
ANE	9,27 ± 1,88 (0,01)				4,00
CI	0,47 ± 0,34 (0,01)				0,73
DI	26,33 ± 3,59 (0,01)				7,63
T	36,07 ± 4,52 (0,01)				9,61

Taxa de falhas nas leituras dos sensores:				**25%**	
Número do Teste	**Tipos de Erros**				**Tempo**
	ANE	**CI**	**DI**	**T**	
1	10	0	50	60	00:12:44
2	9	2	49	60	00:12:20
3	13	0	47	60	00:12:41
4	7	1	52	60	00:12:37
5	7	0	53	60	00:12:32
6	12	0	48	60	00:12:34
7	8	1	51	60	00:12:28
8	6	1	53	60	00:12:29
9	17	0	43	60	00:12:38
10	4	1	54	59	00:12:30
11	6	1	53	60	00:12:31
12	4	0	56	60	00:12:47
13	8	2	50	60	00:12:35
14	16	1	43	60	00:12:27
15	5	0	54	59	00:12:12
16	11	2	47	60	00:12:27
17	20	0	40	60	00:12:11
18	10	0	50	60	00:12:07
19	6	1	53	60	00:12:38
20	8	1	51	60	00:12:34
21	15	0	45	60	00:12:34
22	9	1	50	60	00:12:36
23	14	0	46	60	00:12:20
24	6	2	52	60	00:12:23
25	5	2	51	58	00:12:17
26	10	0	50	60	00:12:39
27	6	0	54	60	00:12:19
28	12	0	48	60	00:12:19
29	4	1	55	60	00:12:33
30	8	0	52	60	00:12:17
	Tempo Total				06:14:19
	M ± IC (NS)				**DP**
ANE	9,2 ± 1,95 (0,01)				4,15
CI	0,67 ± 0,36 (0,01)				0,76
DI	50 ± 1,8 (0,01)				3,83
T	59,87 ± 0,2 (0,01)				0,43

Estação Meteorológica Com Técnicas de Tolerância a Falhas		
Falhas inseridas na região de memória	:	Memória de dados
Quantidade de ciclos em cada teste	:	60
Quantidade de falhas injetadas na memória em cada round	:	**64**

Taxa de falhas nas leituras dos sensores:				**0%**	
Número do Teste	**Tipos de Erros**				**Tempo**
	ANE	**CI**	**DI**	**T**	
1	4	1	3	8	00:12:04
2	6	0	3	9	00:12:18
3	5	0	2	7	00:11:47
4	6	0	11	17	00:13:51
5	3	2	1	6	00:11:35
6	5	0	4	9	00:12:02
7	5	0	6	11	00:12:29
8	11	1	1	13	00:13:06
9	6	0	4	10	00:12:19
10	2	1	4	7	00:11:42
11	5	0	7	12	00:12:49
12	7	0	7	14	00:13:22
13	6	2	6	14	00:13:33
14	7	0	1	8	00:12:00
15	13	0	3	16	00:13:54
16	5	0	9	14	00:13:26
17	4	0	5	9	00:12:07
18	13	1	5	19	00:14:22
19	4	1	3	8	00:12:06
20	8	0	2	10	00:12:30
21	3	0	9	12	00:11:01
22	4	0	4	8	00:12:50
23	4	0	7	11	00:12:36
24	7	1	5	13	00:12:59
25	9	0	2	11	00:12:37
26	4	1	4	9	00:12:02
27	7	0	6	13	00:13:24
28	5	0	2	7	00:11:35
29	8	2	5	15	00:13:40
30	9	1	2	12	00:12:54
	Tempo Total				06:19:00
	M ± IC (NS)				**DP**
ANE	6,17 ± 1,28 (0,01)				2,73
CI	0,47 ± 0,32 (0,01)				0,68
DI	4,43 ± 1,2 (0,01)				2,54
T	11,07 ± 1,54 (0,01)				3,27

Taxa de falhas nas leituras dos sensores:				**25%**	
Número do Teste	**Tipos de Erros**				**Tempo**
	ANE	**CI**	**DI**	**T**	
1	10	0	7	17	00:14:31
2	5	0	9	14	00:13:19
3	3	0	5	8	00:12:02
4	7	1	11	19	00:14:50
5	3	0	6	9	00:12:15
6	6	0	6	12	00:12:51
7	6	1	3	10	00:12:11
8	4	0	13	17	00:14:23
9	7	0	5	12	00:12:51
10	11	1	4	16	00:14:09
11	5	0	8	13	00:13:03
12	3	0	4	7	00:11:46
13	6	0	6	12	00:12:54
14	7	2	7	16	00:14:27
15	5	0	5	10	00:12:13
16	14	0	4	18	00:14:26
17	4	0	10	14	00:13:14
18	1	0	14	15	00:13:49
19	9	2	7	18	00:14:38
20	3	0	10	13	00:13:12
21	7	2	4	13	00:13:07
22	8	0	6	14	00:13:35
23	6	0	11	17	00:14:42
24	6	0	9	15	00:13:41
25	9	1	9	19	00:14:31
26	4	1	5	10	00:12:26
27	10	1	5	16	00:14:29
28	3	0	7	10	00:12:05
29	2	0	5	7	00:11:40
30	5	0	8	13	00:13:16
	Tempo Total				06:40:36
	M ± IC (NS)				**DP**
ANE	5,97 ± 1,38 (0,01)				2,93
CI	0,4 ± 0,32 (0,01)				0,67
DI	7,1 ± 1,32 (0,01)				2,81
T	13,47 ± 1,64 (0,01)				3,49

Estação Meteorológica Sem Técnicas de Tolerância a Falhas		
Falhas inseridas na região de memória	:	Memória de dados e memória interna
Quantidade de ciclos em cada teste	:	60
Quantidade de falhas injetadas na memória em cada round	:	**64**

Taxa de falhas nas leituras dos sensores:				**0%**	
Número do Teste	**Tipos de Erros**				**Tempo**
	ANE	**CI**	**DI**	**T**	
1	27	3	16	46	00:10:28
2	21	1	30	52	00:11:11
3	32	1	21	54	00:11:32
4	24	2	18	44	00:10:08
5	21	0	19	40	00:10:02
6	31	1	14	46	00:10:50
7	25	2	21	48	00:10:56
8	33	1	10	44	00:10:35
9	26	0	26	52	00:11:24
10	34	2	12	48	00:10:58
11	26	3	19	48	00:10:59
12	23	0	26	49	00:11:09
13	31	0	15	46	00:10:48
14	30	1	18	49	00:11:08
15	37	1	6	44	00:10:11
16	13	0	31	44	00:10:15
17	30	1	19	50	00:11:06
18	37	0	11	48	00:11:07
19	30	2	15	47	00:10:50
20	15	2	35	52	00:11:50
21	29	1	21	51	00:11:02
22	32	1	12	45	00:10:23
23	27	2	23	52	00:11:20
24	33	2	16	51	00:11:07
25	21	3	28	52	00:11:35
26	18	1	30	49	00:11:07
27	22	0	23	45	00:10:34
28	15	1	33	49	00:10:51
29	37	2	13	52	00:11:28
30	19	0	29	48	00:10:51
	Tempo Total				05:27:45
	M ± IC (NS)				**DP**
ANE	26,63 ± 3,18 (0,01)				6,75
CI	1,2 ± 0,45 (0,01)				0,96
DI	20,33 ± 3,53 (0,01)				7,50
T	48,17 ± 1,55 (0,01)				3,30

Taxa de falhas nas leituras dos sensores:				**25%**	
Número do Teste	**Tipos de Erros**				**Tempo**
	ANE	**CI**	**DI**	**T**	
1	14	0	46	60	00:12:24
2	25	2	33	60	00:12:35
3	13	0	46	59	00:12:19
4	17	1	42	60	00:12:22
5	16	0	44	60	00:12:41
6	31	1	28	60	00:12:30
7	22	2	36	60	00:12:47
8	16	1	43	60	00:12:32
9	25	0	35	60	00:12:23
10	22	3	35	60	00:12:35
11	41	1	18	60	00:12:18
12	12	1	47	60	00:12:24
13	27	2	31	60	00:12:33
14	29	5	26	60	00:12:47
15	20	1	39	60	00:12:40
16	35	1	24	60	00:12:15
17	20	4	36	60	00:12:44
18	31	0	29	60	00:12:31
19	19	2	39	60	00:12:19
20	30	0	30	60	00:12:37
21	28	4	28	60	00:12:30
22	24	0	36	60	00:12:51
23	34	2	24	60	00:12:33
24	20	1	39	60	00:12:23
25	27	0	33	60	00:12:18
26	15	0	45	60	00:12:20
27	19	1	40	60	00:12:06
28	25	3	32	60	00:12:18
29	32	2	26	60	00:12:22
30	38	0	22	60	00:12:17
	Tempo Total				06:14:14
	M ± IC (NS)				**DP**
ANE	24,23 ± 3,59 (0,01)				7,64
CI	1,33 ± 0,65 (0,01)				1,37
DI	34,4 ± 3,7 (0,01)				7,88
T	59,97 ± 0,09 (0,01)				0,18

Estação Meteorológica Com Técnicas de Tolerância a Falhas		
Falhas inseridas na região de memória	:	Memória de dados e memória interna
Quantidade de ciclos em cada teste	:	60
Quantidade de falhas injetadas na memória em cada round	:	**64**

Taxa de falhas nas leituras dos sensores:				**0%**	
Número do Teste	**Tipos de Erros**				**Tempo**
	ANE	**CI**	**DI**	**T**	
1	20	2	3	25	00:15:45
2	21	0	2	23	00:15:43
3	14	1	3	18	00:14:36
4	15	1	4	20	00:15:14
5	7	0	4	11	00:12:38
6	6	0	7	13	00:13:12
7	20	0	0	20	00:15:26
8	23	0	4	27	00:16:19
9	14	0	4	18	00:14:14
10	17	1	8	26	00:16:02
11	21	0	7	28	00:17:03
12	7	1	6	14	00:13:25
13	24	1	0	25	00:15:59
14	14	1	7	22	00:15:29
15	18	1	1	20	00:14:34
16	21	1	1	23	00:15:52
17	12	0	7	19	00:14:37
18	15	1	0	16	00:14:12
19	10	1	8	19	00:14:41
20	23	0	0	23	00:15:45
21	17	0	3	20	00:15:20
22	26	0	3	29	00:17:15
23	19	2	9	30	00:17:11
24	14	0	2	16	00:14:26
25	18	1	0	19	00:14:56
26	12	0	10	22	00:15:26
27	29	1	4	34	00:18:18
28	15	0	9	24	00:15:47
29	16	2	1	19	00:14:41
30	15	0	5	20	00:14:50
	Tempo Total				07:38:56
	M ± IC (NS)				**DP**
ANE	16,77 ± 2,6 (0,01)				5,54
CI	0,6 ± 0,32 (0,01)				0,67
DI	4,07 ± 1,46 (0,01)				3,10
T	21,43 ± 2,43 (0,01)				5,17

Taxa de falhas nas leituras dos sensores:				**25%**	
Número do Teste	**Tipos de Erros**				**Tempo**
	ANE	**CI**	**DI**	**T**	
1	19	2	9	30	00:17:24
2	24	0	7	31	00:17:51
3	18	0	6	24	00:15:48
4	18	0	8	26	00:16:33
5	13	1	11	25	00:16:02
6	16	0	5	21	00:15:09
7	14	0	8	22	00:15:40
8	11	0	10	21	00:15:13
9	16	0	14	30	00:17:29
10	11	2	0	13	00:13:01
11	17	0	7	24	00:15:33
12	15	1	12	28	00:16:49
13	16	0	14	30	00:17:25
14	15	1	9	25	00:16:13
15	23	0	5	28	00:16:48
16	15	1	8	24	00:16:14
17	18	0	8	26	00:16:16
18	20	1	5	26	00:16:34
19	15	0	7	22	00:15:35
20	19	1	9	29	00:16:54
21	25	0	7	32	00:17:41
22	18	0	9	27	00:16:38
23	14	0	5	19	00:14:20
24	14	0	10	24	00:16:05
25	16	0	7	23	00:15:40
26	22	1	5	28	00:16:47
27	20	0	9	29	00:17:01
28	18	0	10	28	00:16:50
29	28	1	6	35	00:18:36
30	21	0	10	31	00:17:42
	Tempo Total				08:11:51
	M ± IC (NS)				**DP**
ANE	17,63 ± 1,88 (0,01)				4,00
CI	0,4 ± 0,29 (0,01)				0,62
DI	8 ± 1,35 (0,01)				2,88
T	26,03 ± 2,1 (0,01)				4,46

Estação Meteorológica Sem Técnicas de Tolerância a Falhas		
Falhas inseridas na região de memória	:	Memória de dados
Quantidade de ciclos em cada teste	:	60
Quantidade de falhas injetadas na memória em cada round	:	**128**

Taxa de falhas nas leituras dos sensores:				**0%**	
Número do Teste	**Tipos de Erros**				**Tempo**
	ANE	**CI**	**DI**	**T**	
1	12	0	40	52	00:11:22
2	13	0	35	48	00:10:57
3	16	3	21	40	00:10:47
4	9	0	41	50	00:11:09
5	10	1	36	47	00:10:51
6	44	0	15	59	00:12:16
7	8	3	47	58	00:12:05
8	14	0	40	54	00:11:46
9	17	0	35	52	00:11:35
10	7	4	41	52	00:11:41
11	9	1	37	47	00:10:33
12	42	0	18	60	00:12:29
13	13	1	40	54	00:11:46
14	16	0	30	46	00:10:35
15	37	2	20	59	00:12:08
16	10	7	38	55	00:11:51
17	9	1	42	52	00:11:40
18	15	0	30	45	00:10:30
19	36	0	20	56	00:11:55
20	9	2	37	48	00:10:36
21	20	1	23	44	00:10:51
22	9	5	42	56	00:11:56
23	14	0	44	58	00:12:06
24	40	3	16	59	00:12:10
25	8	2	29	39	00:09:35
26	37	0	20	57	00:12:03
27	10	0	28	38	00:09:34
28	17	0	36	53	00:11:26
29	14	1	21	36	00:09:35
30	8	0	26	34	00:09:17
	Tempo Total				05:37:05
	M ± IC (NS)				**DP**
ANE	17,43 ± 5,49 (0,01)				11,67
CI	1,23 ± 0,83 (0,01)				1,76
DI	31,6 ± 4,47 (0,01)				9,50
T	50,27 ± 3,48 (0,01)				7,41

Taxa de falhas nas leituras dos sensores:				**25%**	
Número do Teste	**Tipos de Erros**				**Tempo**
	ANE	**CI**	**DI**	**T**	
1	14	1	45	60	00:12:16
2	17	1	42	60	00:12:17
3	8	2	50	60	00:12:17
4	13	1	46	60	00:12:46
5	24	0	36	60	00:12:20
6	44	0	16	60	00:12:33
7	8	1	51	60	00:12:31
8	16	3	41	60	00:12:23
9	16	0	44	60	00:12:39
10	37	0	23	60	00:12:19
11	9	1	50	60	00:12:35
12	49	0	11	60	00:12:28
13	19	0	41	60	00:12:25
14	23	0	37	60	00:12:21
15	14	1	45	60	00:12:26
16	21	1	38	60	00:12:27
17	9	1	50	60	00:12:33
18	26	0	34	60	00:12:29
19	15	4	41	60	00:12:27
20	41	0	19	60	00:12:29
21	11	1	48	60	00:12:23
22	9	1	50	60	00:12:36
23	32	0	28	60	00:12:30
24	16	1	43	60	00:12:30
25	12	1	47	60	00:12:34
26	38	0	22	60	00:12:27
27	18	3	39	60	00:12:18
28	10	1	49	60	00:12:08
29	15	0	45	60	00:12:26
30	20	0	40	60	00:12:34
	Tempo Total				06:13:27
	M ± IC (NS)				**DP**
ANE	20,13 ± 5,39 (0,01)				11,46
CI	0,83 ± 0,48 (0,01)				1,02
DI	39,03 ± 5,17 (0,01)				11,00
T	60				0,00

Estação Meteorológica Com Técnicas de Tolerância a Falhas		
Falhas inseridas na região de memória	:	Memória de dados
Quantidade de ciclos em cada teste	:	60
Quantidade de falhas injetadas na memória em cada round	:	**128**

Taxa de falhas nas leituras dos sensores:				**0%**	
Número do Teste	**Tipos de Erros**				**Tempo**
	ANE	**CI**	**DI**	**T**	
1	11	1	5	17	00:14:30
2	16	0	10	26	00:17:16
3	9	4	15	28	00:15:59
4	12	0	10	22	00:14:10
5	13	1	11	25	00:15:54
6	13	0	13	26	00:15:55
7	15	0	9	24	00:14:41
8	13	2	10	25	00:15:59
9	21	1	12	34	00:16:53
10	10	1	14	25	00:15:00
11	10	0	10	20	00:14:27
12	6	3	15	24	00:14:58
13	17	0	11	28	00:15:48
14	9	3	19	31	00:16:26
15	18	0	9	27	00:16:07
16	21	3	4	28	00:16:53
17	7	3	17	27	00:16:31
18	16	0	6	22	00:14:39
19	13	0	7	20	00:14:21
20	32	1	10	43	00:17:24
21	15	1	11	27	00:16:40
22	12	0	12	24	00:16:20
23	15	0	9	24	00:16:33
24	6	2	12	20	00:15:40
25	19	2	14	35	00:16:00
26	11	0	13	24	00:13:22
27	9	2	1	12	00:14:25
28	12	1	10	23	00:13:37
29	11	0	14	25	00:13:35
30	4	1	16	21	00:13:59
	Tempo Total				07:44:02
	M ± IC (NS)				**DP**
ANE	13,2 ± 2,61 (0,01)				5,54
CI	1,07 ± 0,57 (0,01)				1,20
DI	10,97 ± 1,83 (0,01)				3,89
T	25,23 ± 2,67 (0,01)				5,67

Taxa de falhas nas leituras dos sensores:				**25%**	
Número do Teste	**Tipos de Erros**				**Tempo**
	ANE	**CI**	**DI**	**T**	
1	17	0	12	29	00:17:08
2	9	3	17	29	00:17:21
3	8	1	9	18	00:14:23
4	16	0	15	31	00:17:44
5	7	1	17	25	00:16:01
6	7	2	13	22	00:15:23
7	15	0	10	25	00:16:20
8	14	0	16	30	00:17:25
9	7	1	15	23	00:15:45
10	14	0	20	34	00:18:20
11	13	0	14	27	00:16:40
12	19	2	18	39	00:20:26
13	15	1	18	34	00:18:13
14	11	1	15	27	00:16:44
15	10	0	22	32	00:17:54
16	8	0	23	31	00:17:53
17	12	2	13	27	00:16:17
18	11	1	7	19	00:14:49
19	13	1	10	24	00:15:49
20	14	1	14	29	00:17:22
21	17	0	11	28	00:16:50
22	20	0	9	29	00:17:17
23	9	1	18	28	00:17:18
24	18	0	12	30	00:17:25
25	13	0	16	29	00:17:06
26	11	3	21	35	00:18:29
27	38	1	6	45	00:21:43
28	5	2	16	23	00:15:22
29	6	2	15	23	00:15:10
30	12	2	13	27	00:16:47
	Tempo Total				08:31:24
	M ± IC (NS)				**DP**
ANE	12,97 ± 2,91 (0,01)				6,20
CI	0,93 ± 0,44 (0,01)				0,94
DI	14,5 ± 1,99 (0,01)				4,23
T	28,4 ± 2,62 (0,01)				5,58

Estação Meteorológica Sem Técnicas de Tolerância a Falhas		
Falhas inseridas na região de memória	:	Memória de dados e memória interna
Quantidade de ciclos em cada teste	:	60
Quantidade de falhas injetadas na memória em cada round	:	**128**

Taxa de falhas nas leituras dos sensores:				**0%**	
Número do Teste	**Tipos de Erros**				**Tempo**
	ANE	**CI**	**DI**	**T**	
1	36	0	23	59	00:12:35
2	32	1	17	50	00:11:11
3	21	2	25	48	00:11:06
4	41	1	15	57	00:12:03
5	18	1	37	56	00:12:06
6	40	1	18	59	00:12:08
7	37	0	20	57	00:12:09
8	21	1	34	56	00:11:34
9	22	0	26	48	00:11:00
10	23	0	32	55	00:11:57
11	25	1	32	58	00:12:23
12	19	2	39	60	00:12:17
13	26	0	34	60	00:12:27
14	35	2	17	54	00:11:28
15	25	2	26	53	00:11:38
16	37	0	21	58	00:12:22
17	30	0	25	55	00:11:58
18	35	2	21	58	00:12:10
19	15	2	43	60	00:12:12
20	19	3	28	50	00:11:20
21	34	0	21	55	00:11:56
22	24	1	32	57	00:12:16
23	42	1	15	58	00:12:02
24	36	0	13	49	00:11:07
25	27	1	29	57	00:11:46
26	21	1	31	53	00:11:44
27	37	2	14	53	00:11:18
28	16	0	41	57	00:12:15
29	13	3	36	52	00:11:20
30	36	1	23	60	00:12:26
	Tempo Total				05:56:14
	M ± IC (NS)				**DP**
ANE	28,1 ± 4,04 (0,01)				8,60
CI	1,03 ± 0,44 (0,01)				0,93
DI	26,27 ± 3,99 (0,01)				8,49
T	55,4 ± 1,73 (0,01)				3,67

Taxa de falhas nas leituras dos sensores:				**25%**	
Número do Teste	**Tipos de Erros**				**Tempo**
	ANE	**CI**	**DI**	**T**	
1	42	0	18	60	00:12:13
2	28	3	29	60	00:12:55
3	33	2	25	60	00:12:19
4	17	1	42	60	00:12:23
5	30	1	29	60	00:12:22
6	28	0	32	60	00:12:30
7	24	0	36	60	00:12:28
8	28	2	30	60	00:12:24
9	23	2	35	60	00:12:27
10	29	5	26	60	00:12:11
11	35	1	24	60	00:12:32
12	38	0	22	60	00:12:09
13	27	3	30	60	00:12:18
14	39	0	21	60	00:12:29
15	34	6	20	60	00:12:13
16	19	0	41	60	00:12:31
17	31	0	29	60	00:12:22
18	30	1	29	60	00:12:38
19	41	0	19	60	00:12:18
20	37	0	23	60	00:12:20
21	23	0	37	60	00:12:30
22	26	1	33	60	00:12:24
23	40	3	17	60	00:12:33
24	26	3	31	60	00:12:27
25	44	0	16	60	00:12:48
26	41	1	18	60	00:12:14
27	26	0	34	60	00:12:08
28	34	3	23	60	00:12:30
29	31	2	27	60	00:12:47
30	28	0	32	60	00:12:36
	Tempo Total				06:12:59
	M ± IC (NS)				**DP**
ANE	31,07 ± 3,28 (0,01)				6,97
CI	1,33 ± 0,75 (0,01)				1,60
DI	27,6 ± 3,32 (0,01)				7,05
T	60				0,00

Estação Meteorológica Com Técnicas de Tolerância a Falhas		
Falhas inseridas na região de memória	:	Memória de dados e memória interna
Quantidade de ciclos em cada teste	:	60
Quantidade de falhas injetadas na memória em cada round	:	**128**

Taxa de falhas nas leituras dos sensores:				**0%**	
Número do Teste	**Tipos de Erros**				**Tempo**
	ANE	**CI**	**DI**	**T**	
1	20	0	4	24	00:15:52
2	17	2	5	24	00:15:48
3	24	0	8	32	00:17:54
4	22	2	14	38	00:19:13
5	21	0	9	30	00:17:22
6	36	1	3	40	00:19:54
7	28	1	9	38	00:19:16
8	23	2	6	31	00:17:31
9	34	0	4	38	00:19:23
10	23	1	9	33	00:17:33
11	10	0	10	20	00:14:52
12	29	0	5	34	00:18:23
13	13	3	10	26	00:16:05
14	32	1	5	38	00:19:47
15	32	1	8	41	00:20:17
16	25	2	7	34	00:18:12
17	26	1	5	32	00:17:57
18	10	0	4	14	00:13:27
19	32	0	8	40	00:19:37
20	25	1	7	33	00:18:15
21	28	3	7	38	00:19:48
22	28	2	11	41	00:20:01
23	17	1	16	34	00:18:25
24	12	0	14	26	00:16:20
25	21	4	9	34	00:18:34
26	42	0	4	46	00:21:28
27	15	2	6	23	00:15:40
28	29	2	8	39	00:19:31
29	25	0	13	38	00:19:18
30	23	1	15	39	00:20:01
	Tempo Total				09:05:44
	M ± IC (NS)				**DP**
ANE	24,07 ± 3,64 (0,01)				7,75
CI	1,1 ± 0,51 (0,01)				1,09
DI	8,1 ± 1,67 (0,01)				3,56
T	33,27 ± 3,41 (0,01)				7,25

Taxa de falhas nas leituras dos sensores:				**25%**	
Número do Teste	**Tipos de Erros**				**Tempo**
	ANE	**CI**	**DI**	**T**	
1	33	0	12	45	00:21:00
2	22	0	14	36	00:19:21
3	23	5	14	42	00:20:27
4	20	0	19	39	00:19:38
5	31	1	12	44	00:21:15
6	21	2	17	40	00:19:53
7	18	0	9	27	00:16:32
8	23	0	5	28	00:16:43
9	26	2	10	38	00:19:18
10	23	2	20	45	00:20:58
11	31	2	11	44	00:21:07
12	22	0	15	37	00:19:27
13	17	1	17	35	00:18:58
14	28	0	13	41	00:19:50
15	10	0	13	23	00:15:20
16	18	2	15	35	00:18:55
17	22	0	7	29	00:17:27
18	24	3	22	49	00:22:13
19	19	1	0	20	00:15:04
20	32	0	13	45	00:21:01
21	13	0	6	19	00:14:34
22	30	0	11	41	00:20:00
23	22	1	25	48	00:22:08
24	13	1	4	18	00:14:43
25	21	0	15	36	00:18:45
26	28	1	12	41	00:20:05
27	20	1	17	38	00:19:00
28	20	2	18	40	00:19:09
29	28	0	7	35	00:18:37
30	28	0	12	40	00:19:48
	Tempo Total				09:31:16
	M ± IC (NS)				**DP**
ANE	22,87 ± 2,73 (0,01)				5,81
CI	0,9 ± 0,56 (0,01)				1,18
DI	12,83 ± 2,57 (0,01)				5,47
T	36,6 ± 3,99 (0,01)				8,48

Estação Meteorológica Sem Técnicas de Tolerância a Falhas		
Falhas inseridas na região de memória	:	Memória de dados
Quantidade de ciclos em cada teste	:	60
Quantidade de falhas injetadas na memória em cada round	:	**256**

Taxa de falhas nas leituras dos sensores:				**0%**	
Número do Teste	**Tipos de Erros**				**Tempo**
	ANE	**CI**	**DI**	**T**	
1	42	0	18	60	00:12:13
2	39	0	21	60	00:12:55
3	40	1	19	60	00:12:12
4	46	0	14	60	00:12:40
5	39	0	21	60	00:12:32
6	43	0	17	60	00:12:00
7	26	0	34	60	00:12:23
8	52	0	8	60	00:12:30
9	49	0	11	60	00:12:40
10	35	2	23	60	00:12:29
11	44	0	16	60	00:12:48
12	48	0	12	60	00:12:34
13	45	0	15	60	00:12:42
14	54	0	6	60	00:12:33
15	51	0	9	60	00:12:25
16	49	0	11	60	00:12:25
17	56	0	4	60	00:12:21
18	51	0	9	60	00:12:12
19	34	0	25	59	00:12:34
20	43	0	17	60	00:12:43
21	37	1	22	60	00:12:30
22	37	0	23	60	00:12:28
23	46	0	13	59	00:12:16
24	40	0	20	60	00:12:15
25	50	0	10	60	00:12:39
26	48	0	12	60	00:12:38
27	49	0	11	60	00:12:25
28	33	0	27	60	00:12:23
29	43	0	17	60	00:12:16
30	41	0	19	60	00:12:24
	Tempo Total				06:14:05
	M ± IC (NS)				**DP**
ANE	43,67 ± 3,24 (0,01)				6,89
CI	0,13 ± 0,2 (0,01)				0,43
DI	16,13 ± 3,16 (0,01)				6,73
T	59,93 ± 0,12 (0,01)				0,25

Taxa de falhas nas leituras dos sensores:				**25%**	
Número do Teste	**Tipos de Erros**				**Tempo**
	ANE	**CI**	**DI**	**T**	
1	49	0	11	60	00:12:09
2	45	0	15	60	00:12:18
3	51	0	9	60	00:12:18
4	38	2	20	60	00:12:21
5	42	1	17	60	00:12:20
6	40	1	19	60	00:12:26
7	56	0	4	60	00:12:22
8	48	0	12	60	00:12:20
9	47	0	13	60	00:12:30
10	54	0	6	60	00:12:47
11	51	0	9	60	00:12:32
12	34	2	24	60	00:12:29
13	46	1	13	60	00:12:20
14	51	0	9	60	00:12:54
15	43	3	14	60	00:12:44
16	45	0	15	60	00:12:39
17	50	0	10	60	00:12:18
18	46	0	14	60	00:12:31
19	27	3	30	60	00:12:31
20	50	0	10	60	00:12:43
21	44	1	15	60	00:12:29
22	49	0	11	60	00:12:12
23	46	1	13	60	00:12:39
24	51	0	9	60	00:12:41
25	33	5	22	60	00:12:34
26	46	0	14	60	00:12:37
27	46	1	13	60	00:12:18
28	42	1	17	60	00:12:34
29	51	0	9	60	00:12:31
30	43	0	17	60	00:12:57
	Tempo Total				06:15:04
	M ± IC (NS)				**DP**
ANE	45,47 ± 2,98 (0,01)				6,34
CI	0,73 ± 0,57 (0,01)				1,20
DI	13,8 ± 2,56 (0,01)				5,44
T	60				0,00

Estação Meteorológica Com Técnicas de Tolerância a Falhas		
Falhas inseridas na região de memória	:	Memória de dados
Quantidade de ciclos em cada teste	:	60
Quantidade de falhas injetadas na memória em cada round	:	**256**

Taxa de falhas nas leituras dos sensores:				**0%**	
Número do Teste	**Tipos de Erros**				**Tempo**
	ANE	**CI**	**DI**	**T**	
1	17	1	6	24	00:17:07
2	19	1	12	32	00:19:26
3	16	0	15	31	00:20:26
4	12	1	8	21	00:19:39
5	18	0	16	34	00:19:45
6	34	2	13	49	00:21:52
7	27	1	9	37	00:19:40
8	13	2	16	31	00:19:07
9	20	0	13	33	00:17:52
10	14	1	14	29	00:18:26
11	20	1	22	43	00:17:24
12	39	0	9	48	00:21:51
13	13	0	15	28	00:15:05
14	12	3	19	34	00:18:45
15	20	1	16	37	00:19:34
16	26	0	17	43	00:19:36
17	17	2	11	30	00:17:13
18	15	0	11	26	00:19:03
19	27	0	8	35	00:19:09
20	19	2	9	30	00:20:31
21	17	1	13	31	00:19:50
22	14	2	21	37	00:17:28
23	7	0	12	19	00:17:38
24	19	1	14	34	00:20:02
25	20	1	10	31	00:19:36
26	13	0	10	23	00:18:50
27	18	3	20	41	00:18:44
28	25	1	9	35	00:18:03
29	12	0	17	29	00:15:05
30	12	1	6	19	00:14:37
	Tempo Total				09:21:24
	M ± IC (NS)				**DP**
ANE	18,5 ± 3,23 (0,01)				6,88
CI	0,93 ± 0,43 (0,01)				0,91
DI	13,03 ± 2,03 (0,01)				4,31
T	32,47 ± 3,55 (0,01)				7,55

Taxa de falhas nas leituras dos sensores:				**25%**	
Número do Teste	**Tipos de Erros**				**Tempo**
	ANE	**CI**	**DI**	**T**	
1	16	0	16	32	00:18:28
2	24	1	14	39	00:19:45
3	16	2	17	35	00:19:07
4	17	1	10	28	00:16:52
5	15	2	21	38	00:20:02
6	7	0	19	26	00:16:38
7	20	1	16	37	00:19:13
8	21	0	9	30	00:17:08
9	12	1	9	22	00:15:34
10	25	1	13	39	00:19:55
11	23	0	11	34	00:18:36
12	19	3	17	39	00:20:17
13	15	0	17	32	00:17:15
14	13	0	23	36	00:19:04
15	25	1	13	39	00:19:42
16	12	1	15	28	00:17:06
17	9	0	11	20	00:14:35
18	13	1	19	33	00:18:13
19	16	2	18	36	00:19:03
20	22	0	21	43	00:21:05
21	16	0	21	37	00:19:10
22	5	3	8	16	00:13:49
23	25	1	13	39	00:19:10
24	13	3	20	36	00:18:42
25	14	1	15	30	00:17:26
26	21	0	17	38	00:19:12
27	19	1	18	38	00:19:19
28	27	0	13	40	00:20:07
29	12	0	27	39	00:19:49
30	18	0	18	36	00:18:33
	Tempo Total				09:12:55
	M ± IC (NS)				**DP**
ANE	17 ± 2,63 (0,01)				5,58
CI	0,87 ± 0,46 (0,01)				0,97
DI	15,97 ± 2,11 (0,01)				4,49
T	33,83 ± 3,01 (0,01)				6,41

Estação Meteorológica Sem Técnicas de Tolerância a Falhas		
Falhas inseridas na região de memória	:	Memória de dados e memória interna
Quantidade de ciclos em cada teste	:	60
Quantidade de falhas injetadas na memória em cada round	:	**256**

Taxa de falhas nas leituras dos sensores:				**0%**	
Número do Teste	**Tipos de Erros**				**Tempo**
	ANE	**CI**	**DI**	**T**	
1	47	1	12	60	00:12:19
2	51	0	9	60	00:12:27
3	46	2	12	60	00:12:32
4	58	0	2	60	00:12:18
5	53	0	7	60	00:12:35
6	50	1	9	60	00:12:30
7	51	0	9	60	00:12:48
8	49	0	11	60	00:12:21
9	49	1	10	60	00:12:35
10	54	0	6	60	00:12:42
11	43	2	5	50	00:12:37
12	56	0	4	60	00:12:27
13	52	0	8	60	00:12:24
14	51	0	9	60	00:12:40
15	49	2	9	60	00:12:39
16	54	0	6	60	00:12:26
17	41	4	15	60	00:12:44
18	55	0	5	60	00:12:19
19	48	0	12	60	00:12:42
20	54	0	6	60	00:12:30
21	51	0	9	60	00:12:37
22	50	0	10	60	00:12:31
23	44	1	15	60	00:12:42
24	50	0	10	60	00:12:41
25	51	0	9	60	00:12:24
26	47	1	12	60	00:12:32
27	56	0	4	60	00:12:30
28	48	1	11	60	00:12:43
29	51	0	9	60	00:12:40
30	46	3	11	60	00:12:26
	Tempo Total				06:16:21
	M ± IC (NS)				**DP**
ANE	50,17 ± 1,86 (0,01)				3,96
CI	0,63 ± 0,49 (0,01)				1,03
DI	8,87 ± 1,48 (0,01)				3,14
T	59,67 ± 0,86 (0,01)				1,83

Taxa de falhas nas leituras dos sensores:				**25%**	
Número do Teste	**Tipos de Erros**				**Tempo**
	ANE	**CI**	**DI**	**T**	
1	51	0	9	60	00:12:28
2	49	0	11	60	00:12:32
3	46	2	12	60	00:12:26
4	46	1	13	60	00:12:55
5	54	0	6	60	00:12:45
6	50	0	10	60	00:12:37
7	56	0	4	60	00:12:47
8	48	1	11	60	00:12:31
9	53	0	7	60	00:12:23
10	51	1	8	60	00:12:35
11	51	0	9	60	00:12:29
12	50	0	10	60	00:12:52
13	47	3	10	60	00:12:21
14	56	0	4	60	00:12:54
15	54	0	6	60	00:12:23
16	55	0	5	60	00:12:25
17	42	0	18	60	00:12:44
18	53	0	7	60	00:12:35
19	57	0	3	60	00:12:19
20	50	0	10	60	00:12:38
21	51	0	9	60	00:12:46
22	52	0	8	60	00:12:33
23	46	2	12	60	00:12:49
24	44	3	13	60	00:12:22
25	49	0	11	60	00:12:41
26	52	0	8	60	00:12:34
27	45	2	13	60	00:12:26
28	54	0	6	60	00:12:18
29	58	0	2	60	00:12:22
30	54	0	6	60	00:12:17
	Tempo Total				06:16:47
	M ± IC (NS)				**DP**
ANE	50,8 ± 1,9 (0,01)				4,04
CI	0,5 ± 0,44 (0,01)				0,94
DI	8,7 ± 1,65 (0,01)				3,51
T	60				0,00

Estação Meteorológica Com Técnicas de Tolerância a Falhas		
Falhas inseridas na região de memória	:	Memória de dados e memória interna
Quantidade de ciclos em cada teste	:	60
Quantidade de falhas injetadas na memória em cada round	:	**256**

Taxa de falhas nas leituras dos sensores:				**0%**	
Número do Teste	**Tipos de Erros**				**Tempo**
	ANE	**CI**	**DI**	**T**	
1	34	0	8	42	00:19:53
2	22	1	2	25	00:16:14
3	29	1	8	38	00:19:34
4	39	0	4	43	00:20:49
5	51	0	2	53	00:22:52
6	30	1	9	40	00:19:54
7	33	2	7	42	00:19:37
8	25	1	2	28	00:16:48
9	24	4	14	42	00:19:14
10	35	2	18	55	00:23:26
11	42	0	13	55	00:23:24
12	28	1	4	33	00:18:13
13	30	0	6	36	00:18:53
14	33	1	7	41	00:19:59
15	33	2	10	45	00:21:24
16	34	0	5	39	00:19:09
17	26	3	7	36	00:18:41
18	38	1	6	45	00:20:45
19	31	2	8	41	00:20:30
20	33	2	3	38	00:19:04
21	35	2	7	44	00:20:50
22	30	0	13	43	00:20:32
23	34	1	10	45	00:21:37
24	31	0	6	37	00:19:01
25	28	6	12	46	00:21:26
26	27	1	15	43	00:20:53
27	25	2	17	44	00:20:34
28	29	0	15	44	00:20:53
29	28	0	10	38	00:19:07
30	40	2	8	50	00:22:09
	Tempo Total				10:05:25
	M ± IC (NS)				**DP**
ANE	13,9 ± 2,83 (0,01)				6,01
CI	1,27 ± 0,64 (0,01)				1,36
DI	8,53 ± 2,09 (0,01)				4,45
T	41,7 ± 3,15 (0,01)				6,71

Taxa de falhas nas leituras dos sensores:				**25%**	
Número do Teste	**Tipos de Erros**				**Tempo**
	ANE	**CI**	**DI**	**T**	
1	30	1	9	40	00:19:56
2	33	0	16	49	00:22:48
3	38	3	4	45	00:21:02
4	33	4	15	52	00:22:36
5	19	0	26	45	00:20:34
6	27	2	12	41	00:20:10
7	35	1	15	51	00:22:04
8	26	6	14	46	00:21:22
9	23	2	16	41	00:19:56
10	19	3	7	29	00:17:19
11	30	1	17	48	00:21:56
12	38	1	14	53	00:23:01
13	29	0	13	42	00:20:14
14	31	2	13	46	00:21:03
15	25	1	6	32	00:18:14
16	22	3	12	37	00:18:55
17	32	0	9	41	00:20:25
18	19	3	12	34	00:18:09
19	29	2	13	44	00:21:26
20	41	0	15	56	00:24:28
21	14	4	14	32	00:17:12
22	24	1	16	41	00:20:14
23	33	3	9	45	00:21:05
24	23	1	14	38	00:19:29
25	27	2	9	38	00:19:35
26	31	0	10	41	00:19:38
27	28	2	11	41	00:20:04
28	33	1	6	40	00:20:31
29	25	3	17	45	00:21:25
30	25	3	13	41	00:20:19
	Tempo Total				10:15:10
	M ± IC (NS)				**DP**
ANE	28,07 ± 2,94 (0,01)				6,26
CI	1,83 ± 0,69 (0,01)				1,46
DI	12,57 ± 2,02 (0,01)				4,29
T	42,47 ± 2,97 (0,01)				6,31