Pedro Lobo

Desenho e Teste de Sistemas Resilientes

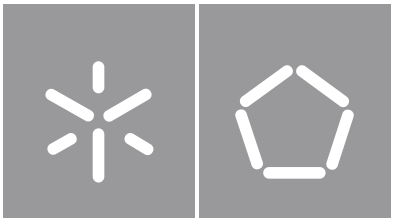
UMinho | 2021

Universidade do Minho
Escola de Engenharia

Pedro Alexandre Costa Lobo

Desenho e Teste de Sistemas Resilientes

fevereiro de 2021



Universidade do Minho
Escola de Engenharia

Pedro Alexandre Costa Lobo

Desenho e Teste de Sistemas Resilientes

Dissertação de Mestrado
Mestrado em Engenharia Eletrónica Industrial e Computadores
Sistemas Embebidos e Computadores

Trabalho efetuado sob a orientação do
Professor Doutor Jorge Cabral

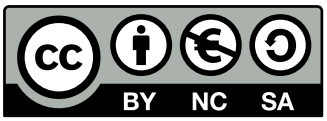
fevereiro de 2021

DIREITOS DE AUTOR E CONDIÇÕES DE UTILIZAÇÃO DO TRABALHO POR TERCEIROS

Este é um trabalho académico que pode ser utilizado por terceiros desde que respeitadas as regras e boas práticas internacionalmente aceites, no que concerne aos direitos de autor e direitos conexos.

Assim, o presente trabalho pode ser utilizado nos termos previstos na licença abaixo indicada.

Caso o utilizador necessite de permissão para poder fazer um uso do trabalho em condições não previstas no licenciamento indicado, deverá contactar o autor, através do RepositóriUM da Universidade do Minho.



Atribuição-NãoComercial-CompartilhaIgual

CC BY-NC-SA

https://creativecommons.org/licenses/by-nc-sa/4.0/

Agradecimentos

”Em primeiro lugar, gostaria de agradecer ao meu orientador, doutor Jorge Cabral, por todo o conhecimento transmitido ao longo deste projeto. Um especial agradecimento aos Engenheiros Luís Novais e Nelson Naia por todo o apoio prestado e pela enorme dísponibilidade. Ainda no âmbito académico, gostaria de agradecer ao João Carvalho por toda a ajuda e suporte prestado neste esforço de engenharia.

Não poderia deixar de agradecer aos meus colegas de laboratório, Rui Almeida e Luís Vale, assim como aos meus amigos mais chegados, que me acompanharam desde o ínicio desta longa caminhada, agradeço pelo suporte, alegria e por todos os momento de descontração.

Por fim, endereço um especial agradecimento aos meus pais por todo o suporte emocional e financeiro que me permitiu frequentar e finalizar um curso superior.

A todos que me ajudaram neste percurso, o meu maior obrigado.”

This work is supported by: European Structural and Investment Funds in the FEDER component, through the Operational Competitiveness and Internationalization Programme (COMPETE 2020) [Project nº 037902; Funding Reference: POCI-01-0247-FEDER-037902].

DECLARAÇÃO DE INTEGRIDADE

Declaro ter atuado com integridade na elaboração do presente trabalho académico e confirmo que não recorri à prática de plágio nem a qualquer forma de utilização indevida ou falsificação de informações ou resultados em nenhuma das etapas conducente à sua elaboração.

Mais declaro que conheço e que respeitei o Código de Conduta Ética da Universidade do Minho.

Resumo

Desenho e Teste de Sistemas Resilientes

A introdução da condução autónoma no setor automóvel, aumentou a necessidade de sistemas elétricos/eletrónicos (E/E). Como a falha destes sistemas pode resultar na perda do controlo do veículo, a necessidade por sistemas resilientes têm aumentado neste setor. Sistemas resilientes garantem uma baixa probabilidade de falha ao longo da sua vida. Esta baixa probabilidade de falha é alcançada através do uso de redundância e de boas práticas. Apesar de redundância aumentar a resiliência de um sistema, este aumento não é objetivamente quantificável. Tradicionalmente, modelação de resiliência e testes acelerados são utilizados para estimar a probabilidade de falha do sistema ao longo da sua vida.

Esta dissertação visou estudar a eficiência dos métodos tradicionais de estimação de resiliência no contexto dos atuais sistemas automóvel. Para alcançar este objetivo, inicialmente foi desenvolvida uma arquitetura resiliente de hardware para um sensor automóvel. Para estimar a resiliência desta arquitetura e auxiliar o seu desenho, foi desenvolvido um modelo de resiliência com base em Cadeias de Markov, o qual foi simulado recorrendo a métodos de Monte Carlo. Após a implementação da arquitetura desenvolvida, uma amostra com 10 sistemas foi submetida a testes acelerados, os quais recorreram a temperatura como fator de aceleração. Para realizar estes testes, foi previamente desenvolvido um *setup* de testes *online*.

A partir da aplicação dos métodos tradicionais de estimação foi possível concluir que, atualmente, estes métodos não são eficientes para estimar sistemas resilientes no contexto de sistemas automóvel. Por um lado, os fornecedores de componentes não fornecem informação suficiente para realizar estimativas de resiliência fiáveis através de modelos de resiliência. Por outro lado, os fatores de aceleração alcançados durante os testes acelerados não são suficiente para alcançar a fase final da vida de um sistema e assim obter a sua distribuição de falhas, num período que não comprometa o *time-to-market*.

Palavras-chave: desenvolvimento de hardware resiliente, redundância, modelação de resiliência, testes acelerados, simulação de Monte Carlo

Abstract

Design and Testing for Reliability

The introduction of autonomous driving in the automotive sector has increased the necessity of electrical/electronic (E/E) systems. Given that the failure of these systems can result in the loss of control of a vehicle, the necessity for resilient systems has also increased over the years. Resilient systems guarantee the low probability of failure over its lifespan. This low probability of failure is achieved through redundancy and best practices. Although, the redundancy of a given system can promote the system's resilience, it is not objectively quantifiable. Traditionally, the modulation of resiliency and empirical methods are used to estimate the probability of failure of a system during its lifespan.

This dissertation addresses the study of efficiency of these traditional methods used for the estimate of the resiliency but focused on contemporary automotive systems. In order achieve this objective, a development of a hardware resilient architecture was initially made for an automotive sensor. To estimate the resiliency of the architecture and its design, a model of resiliency was developed, based in Markov chain, and simulated using a Monte Carlo method. After the implementation of the developed architecture, a sample of 10 systems were submitted to accelerated tests, which resource to temperature as a factor of acceleration. To accomplish these tests, a pre-emptive development was made of setup for online tests.

The application of traditional methods of estimation, it was possible to conclude that at this moment, these methods are insufficient to estimate resilient systems for an automotive application. On the hand, manufactures of components do not provide sufficient information to make appraisals of their resiliency using the mentioned models of resiliency. Still, the acceleration factors achieved during the accelerated tests were not sufficient to reach wear-out and, thus, obtain a distributed failure of a given system, at least within a timeline that does not compromise the time-to-market.

Keywords: resilient hardware development, redundancy, resiliency modulation, accelerated testing, Monte Carlo simulations

Índice

Lista de Figuras

Lista de Tabelas

Lista de Acrónimos

λ Taxa De Falha

ADC Conversor Analógico-Digital

AEC *Automotive Electronics Council*

API Interface De Programação De Aplicações

AQEC *Aerospace Qualified Electronic Component*

ASH *Averaged Shifted Histogram*

ASIL *Automotive Safety Integrity Level*

BbW *Break-by-Wire*

CAN *Controller Area Network*

CDF Função Distribuição De Probabilidade Acumulada

COTS *Commercial off-the-Shelf*

CRC *Cyclic Redundancy Check*

E/E Elétricos/eletrónicos

Ea Energia De Ativação

EMC Compatibilidade Eletromagnética

EMI Interferência Eletromagnética

ESD Descarga Electrostática

FSM Máquina De Estados Finita

GPIO *General Purpouse Input/ output*

HALT *Highly Accelerated Life Test*

IC Circuito Integrado

LDO Regulador De Tensão

MIL-PRF *Military Performance Specification*

MTBF Tempo Médio Entre Falhas

MTTF Tempo Médio Até À Falha

NMR *N-Modular Redundancy*

OPAMP Amplificador Operacional

PC Computador Pessoal

PCB Placas De Circuito Impresso

PDF Função Densidade De Probabilidade

PDO *Physical Device Object*

R Resiliência

RF Radiofrequência

SAS *Steering Angle Sensor*

SbW *Steer-by-Wire*

SMT *Surface-Mount Technology*

TbW *Throttle-by-Wire*

UART *Universal Asynchronous Receiver Transmitter*

XbW *X-by-Wire*

Capítulo 1

Introdução

Sistemas resilientes tornaram-se populares em aplicações militares em meados do século XX [1]. Desde então, estes sistemas são aplicados nos setores nuclear, aeroespacial, aeronáutico, suporte de vida, e mais recentemente, na indústria automóvel. A resiliência é uma propriedade que está diretamente relacionada com a probabilidade de falha do sistema. Desta forma, um sistema que apresente uma elevada resiliência possui uma baixa probabilidade de falha. Sistemas resilientes são pretendidos em aplicações onde uma falha pode ter consequências severas.

A baixa probabilidade de falha de um sistema pode ser alcançada através da utilização de um conjunto de boas práticas e do uso de técnicas de tolerância a faltas. Nesta dissertação, o foco será no desenvolvimento e teste de uma solução de hardware resiliente.

A utilização de boas práticas no desenvolvimento destes sistemas minimiza a introdução de faltas de desenvolvimento. O conjunto de boas práticas utilizadas no desenvolvimento destes sistemas, estão reunidas em dispendiosos *standards* de diferentes setores como o ISO:26262 no caso do setor automóvel, ou em documentos privados de empresas com experiência no desenvolvimento destes sistemas. Contudo, existem alguns autores que fornecem algumas diretrizes para o desenho, modelação e estimação destes sistemas [1], [2].

Relativamente às técnicas de tolerância a faltas, a mais utilizada para aumentar a resiliência de sistemas é a redundância. Redundância pode ser aplicada tanto a hardware como a software e consiste em fornecer múltiplas alternativas a um sistema para o cumprimento de uma função, reduzindo assim a probabilidade de falha do sistema. É um facto que a utilização de redundância aumenta drasticamente a resiliência de um sistema. Contudo, a mesma também aumenta: os custos, o tempo de desenvolvimento, a complexidade, os custos de manutenção e o seu tamanho [3]. Na literatura de setores onde o fator custo não é uma limitação, existem um conjunto de arquiteturas altamente sofisticadas e resilientes [4] .

Apesar da utilização de técnicas de tolerância a faltas e de boas práticas de desenho aumentar a resiliência dos sistemas, este aumento não é objetivamente quantificável. Tradicionalmente, técnicas de modelação de resiliência e testes acelerados são utilizados para quantificar a resiliência de um sistema [1], [2], [5], [6].

A modelação de resiliência é uma técnica analítica que permite estimar a probabilidade de falha de um sistema, com base na probabilidade de falha dos seus componentes e das suas interligações. Esta, é usualmente utilizada em paralelo com o desenho, possibilitando a comparação entre arquiteturas e auxiliando a escolha de componentes.

Testes acelerados são técnicas empíricas, os quais, através do envelhecimento acelerado e da recolha de dados de N sistemas, permitem estimar a resiliência e obter uma distribuição de falhas de um sistema. O envelhecimento acelerado é alcançado à custa do uso de fatores de aceleração, fatores estes que consistem na utilização de condições de operação, como temperatura e humidade, mais exigentes do que aquelas a que o sistema está sujeito durante a sua operação normal.

Atualmente, arquiteturas cada vez mais complexas e resilientes, bem como componentes com maior qualidade e resiliência, têm sido utilizados para aumentar a longevidade dos sistemas. Apesar disto, testes acelerados são utilizados à décadas para estimar e testar a resiliência dos sistemas [7], [8].

1.1 Motivação e Objetivos

A introdução da condução autónoma no setor automóvel, aumentou a necessidade de sistemas E/E que a suportem. Neste sentido, sistemas tipicamente mecânicos têm vindo a ser substituídos por sistemas E/E, os sistemas *x-by-Wire* (XbW) como o *steer-by-wire* (SbW), o *throttle-by-wire* (TbW) e o *break-by-wire* (BbW) são alguns exemplos desta tendência. Uma vez que a falha destes sistemas pode ter como consequência a perda do controlo do veículo, e consequentemente perdas monetárias, ou até mesmo a perda de vidas humanas, a necessidade por sistemas e subsistemas resilientes têm aumentado nos últimos anos neste setor. Sensores automóvel como o *steering angle sensor* (SAS), atuadores e sistemas de controlo com requisitos de resiliência são alguns exemplos de sistemas necessários neste setor [9].

A literatura de sistemas resilientes para setores, onde o custo do sistema não é uma limitação, apresenta um conjunto de arquiteturas com baixa probabilidade de falha [4]. Porém, no setor automóvel o baixo custo é um dos requisitos dos sistemas. Assim, a utilização de arquiteturas sugeridas pela literatura no contexto do setor automóvel apresenta um desafio.

Outro aspeto interessante e abordado nesta dissertação, são os métodos de estimação de resiliência, os quais são utilizados à décadas, apesar da evolução da resiliência e complexidade das arquiteturas.

Modelações de resiliência necessitam de uma distribuição que caraterize as falhas dos componentes de um sistema e, deste modo, realizar projeções fiáveis sobre a resiliência de um sistema.

Testes acelerados utilizam condições de operação mais exigentes que aquelas a que o sistema está sujeito durante a sua normal operação, de modo a reduzir o tempo de teste. Estes testes são muito utilizados para determinar a distribuição que carateriza a vida de um sistema durante o período de garantia. Porém, é desconhecida a eficiência da sua utilização para estimar sistemas com uma vida superior a 10 ou 15 anos, num período que não comprometa o *time-to-market* do projeto.

Esta dissertação tem como principal objetivo estudar a eficiência dos métodos tradicionais de estimação de resiliência. Para alcançar este objetivo, estes métodos serão utilizados para estimar a resiliência do hardware de um sensor automóvel, o SAS, o qual será utilizado como caso de estudo desta dissertação.

Inicialmente, será desenvolvida uma arquitetura de hardware resiliente para o SAS com base nas técnicas descritas no estado da arte. Em paralelo com o desenho do caso de estudo, será construído um modelo de resiliência do mesmo, a simulação deste modelo possibilitará uma estimativa da resiliência do desenho de hardware. Para avaliar a fiabilidade da estimativa obtida com a abordagem analítica, será posteriormente obtida uma nova estimativa utilizando uma abordagem empírica, abordagem esta que recorrerá a testes acelerados para mais rapidamente obter a resiliência do caso de estudo.

Para possibilitar a realização dos testes acelerados ao caso de estudo, será desenvolvido um *setup* para testes *online*. Este *setup* irá permitir realizar testes acelerados, recorrendo a temperatura como fator de aceleração, e também recolher as falhas de protótipos ao longo do teste.

1.2 Estrutura da Dissertação

Este documento começa por apresentar uma breve introdução sobre sistemas resilientes e pela definição dos objetivos deste trabalho, ambos são realizados neste capítulo. De seguida, o capítulo 2 apresenta uma introdução aos fundamentos teóricos e aos conceitos que suportam este trabalho. Este capítulo começa por introduzir sistemas embebidos e sistemas confiáveis, passando depois a abordar uma das propriedades de sistemas confiáveis, a resiliência. Dentro de sistemas resilientes, são apresentadas algumas técnicas utilizadas para alcançar uma baixa probabilidade de falha e são introduzidos os métodos tradicionais para estimar a resiliência de um sistema.

O capítulo 3 começa por apresentar os métodos tradicionais que serão utilizados para estimar a resiliência do caso de estudo, bem como estes métodos irão ser utilizados ao longo do desenvolvimento do caso de estudo. Ainda dentro deste capítulo, são apresentados o modelo de resiliência do caso de estudo e o *setup* que irá suportar a realização dos testes acelerados.

O capítulo 4 apresenta o desenvolvimento do caso de estudo. Este capítulo começa por introduzir a problemática relacionada com o caso de estudo, passando depois para os seus requisitos e o seu desenvolvimento. No final deste capítulo são apresentados os resultados da simulação do modelo de resiliência do caso de estudo e dos testes acelerados.

Finalmente, o capítulo 5 apresenta as conclusões retiradas da aplicação dos métodos tradicionais de estimação de resiliência e sugestões de melhoria ao trabalho desenvolvido.

Capítulo 2

Estado da Arte

2.1 Sistemas Embebidos

Atualmente, sistemas embebidos apresentam uma vasta variedade de aplicações em diferentes setores, tais como: consumidor final, indústria aeroespacial, indústria médica, militar, aeronáutica e automação industrial. No caso do consumidor final, estes sistemas estão completamente diluídos no seu quotidiano, assumindo diversos tamanhos e formatos. Os *smartphones*, *smartwatches*, MP3, detetores de incêndio e os alarmes, são exemplos de sistemas embebidos que são utilizados diariamente por todos nós.

Apesar de não existir uma definição clara de sistemas embebidos, usualmente, estes são descritos como sendo sistemas constituídos por hardware, software, e por vezes partes mecânicas, os quais são desenhados com o propósito de uma aplicação específica [10]. Sistemas embebidos interagem com o mundo através de sensores e atuadores, e possuem interfaces dedicadas com o utilizador. Dependendo da aplicação, estes sistemas podem possuir requisitos como pouco peso, baixo consumo energético, pequenas dimensões, tempo real, elevado desempenho e confiabilidade. Devido à grande diversidade de aplicações, o hardware e software destes sistemas é desenhado em função da necessidade de cada aplicação. Normalmente, o desenho de hardware e software ocorre em paralelo e é realizado por equipas diferentes.

Sistemas embebidos também podem possuir requisitos de tempo real, os quais representam uma classe diferente de sistemas embebidos. Estes possuem restrições temporais, ou seja, o correto cumprimento da sua funcionalidade está também relacionado com o tempo que demoram a responder a um estímulo. Esta classe de sistemas pode ser dividida em *soft real-time systems* e *hard real-time systems*. Os segundos distinguem-se dos primeiros, pois apresentam consequências muito mais severas, no caso do incumprimento das suas restrições temporais.

2.1.1 Fases do Desenvolvimento de um Sistema Embebido

Usualmente, o desenvolvimento de um novo sistema embebido passa por diversas fases até ser alcançado um produto final. Este desenvolvimento nem sempre é um processo direto, sendo por vezes necessárias várias reiterações até que seja alcançado um protótipo. A figura 2.1 apresenta as diferentes fases do desenvolvimento de um sistema embebido. Para gerir as diversas fases do desenvolvimento de um sistema embebido, são utilizadas metodologias como o *Waterfall*, *Agile* ou *Spider*.

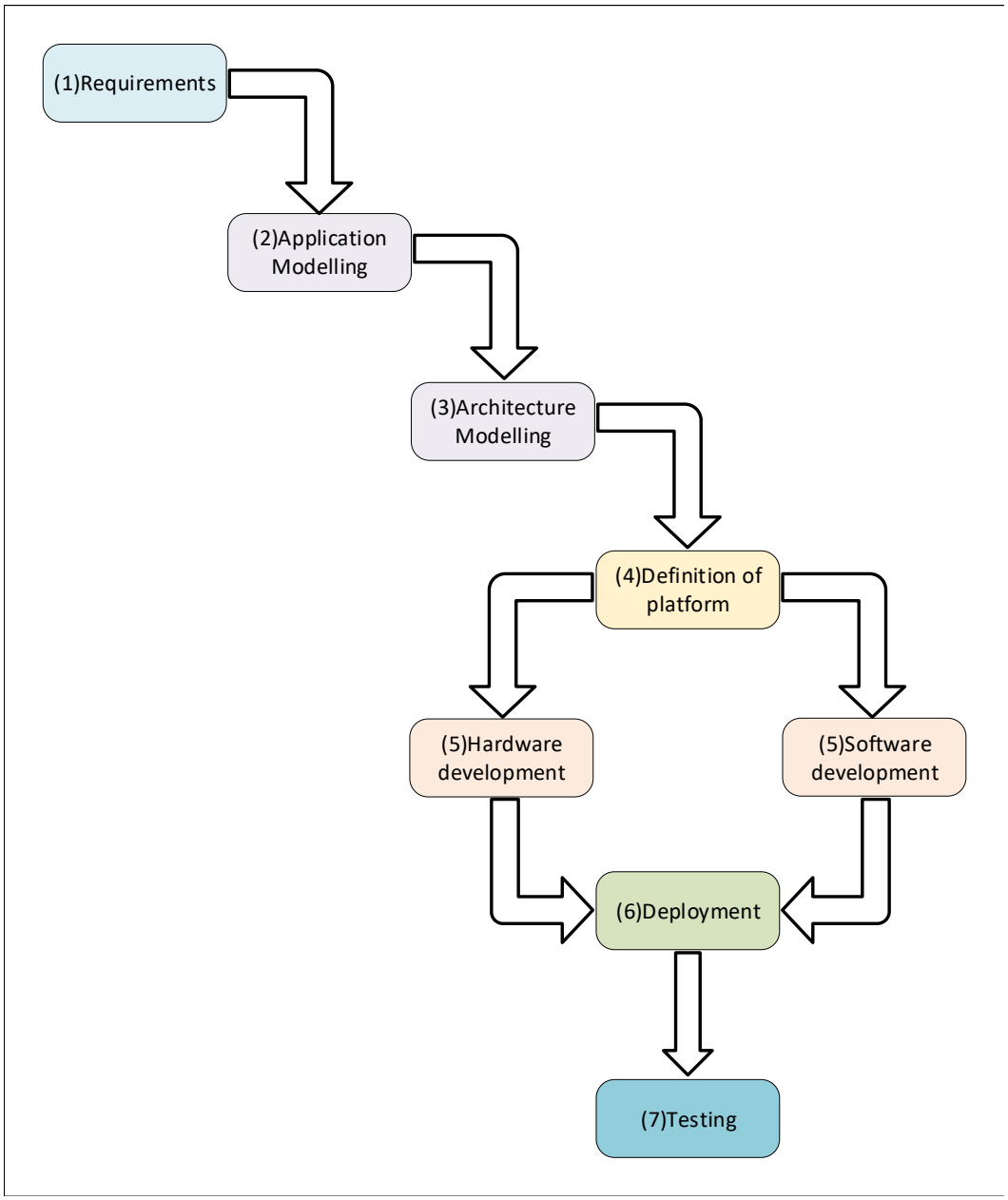


Figura 2.1: Fases do desenvolvimento de um sistema embebido.

No desenvolvimento de um sistema embebido, inicialmente, são recolhidos os requisitos funcionais e não funcionais, e as restrições do sistema a desenvolver. Após uma redefinição destes requisitos, é desenhado um conceito do sistema. Para avaliar como o conceito desenhado irá cumprir os requisitos do sistema, as suas tarefas são conceptualizadas em software, sendo este software utilizado validar o conceito desenvolvido.

Depois de validado o conceito do sistema, é iniciada a modelação da arquitetura do sistema. Nesta etapa, são identificados os diversos subsistemas que irão auxiliar a arquitetura a alcançar os seus requisitos, por exemplo: microcontrolador, conversor analógico-digital (ADC), CAN *transceiver*, etc. Durante a definição da arquitetura ainda não são conhecidos os constituintes de cada um dos subsistemas utilizados na arquitetura. O comportamento da arquitetura conceptualizada, pode ser modulado através de máquina de estados finitas (FSM), as quais podem ser simuladas recorrendo por exemplo a uma abordagem *software-only*. Nesta abordagem, as FSMs são implementadas recorrendo a software, tendo os seus estados e as suas transições validadas. O software utilizado nesta abordagem, pode posteriormente ser utilizado na versão final.

Após a modelação e validação da arquitetura do sistema, é iniciada a alocação dos recursos em software e hardware, e escolhida a plataforma alvo. A decisão de desenhar uma nova plataforma, especificamente moldada para o sistema, ou a utilização de recursos *commercial off-the-shelf* (COTS) é um compromisso entre o tempo de desenvolvimento, e os recursos financeiros disponibilizados para o projeto. Dependendo da decisão, diferentes recursos serão alocados em hardware e software. Com a alocação dos recursos concluída, é iniciado o desenvolvimento desacoplado de hardware e software, significando isto que tanto software como hardware serão desenvolvidos em paralelo. No desenho de software, é definida a arquitetura, as interfaces, as classes e os módulos que suportam a aplicação, sendo esta aplicação ser validada através do uso de *development kit*.

No desenvolvimento de hardware, inicialmente, é definida uma arquitetura e são escolhidos os seus componentes. Dependendo da complexidade da arquitetura e do tempo de desenvolvimento, pode ser optado por: (1) Inicialmente, apenas desenhar uma extensão para uma plataforma já existente, a qual permita validar a arquitetura conceptualizada para o hardware; (2) Desenhar a plataforma de hardware específica para o sistema. Após esta decisão é iniciado o desenho dos circuitos.

No desenho de circuitos, simuladores de circuitos como o LTspice ou TINA podem ser utilizados para validar os circuitos desenhados. Após ser alcançada uma versão estável do esquemáticos, é iniciado o desenho do *layout* do circuito/sistema. Depois da implementação e fabrico do hardware, o mesmo deve ser testado, antes do *deployment* da versão final de software e dos testes globais do sistema.

2.1.2 Desenho de PCBs

Inicialmente, as primeiras *boards* eram similares às *protoboards*, nas quais eram utilizados apenas componentes *through-hole*. Estes componentes eram interligados recorrendo a pistas de solda, ou então, quando o número de conexões tornava o uso de pistas impossível, através de fios. Apesar de alguns problemas relacionados com *cross talk* entre sinais e com *overshoot*, esta abordagem funcionava devido às baixas frequências utilizadas pelos componentes. Em consequência da evolução da tecnologia do processo de fabrico de circuitos integrados (IC), aumento das frequências de *clock* e do número de pinos, esta abordagem tornou-se obsoleta, surgindo assim as *multilayer boards* [11].

Uma *multilayer board* é constituída por uma camada de dielétrico (core), por diversas camadas condutoras (*layers*), e por camadas de um material conhecido como *prepreg* [11]. A figura 2.2, retirada de [11], ilustra as diferentes camadas de uma *board* com quatro *layers*. As *layers*, permitem gravar caminhos de cobre (*traces*), os quais interligam os componentes. Além disto, para interligar as *traces* e os pinos dos componentes (*surface-mount technology* (SMT)), através de soldadura, são gravados *pads* nas *layers*. As camadas de *prepreg*, quando submetidas a temperatura e pressão, "colam"às camadas condutoras, mantendo assim um espaçamento entre *layers* e evitando curto-circuitos.

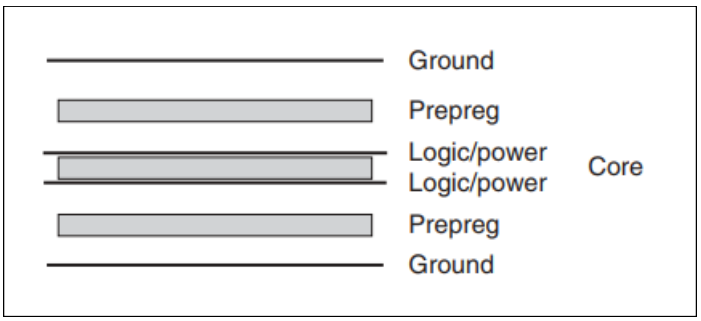


Figura 2.2: PCB com quatro *layers*.

Para interligar as diferentes *layers*, são utilizadas *vias*, as quais são furos preenchidos por cobre (condutor). *Through-hole vias*, podem ser utilizadas para atravessar todas as *layers* da *printed* PCB e conectar diversas *layers*. Apesar desta abordagem poder à primeira vista parecer a mais lógica, por vezes, é apenas necessário conectar duas ou três *layers*, podendo ser utilizadas *buried vias* ou *blind vias*. *Buried vias* são utilizadas para conectar uma ou mais *layers* interiores, enquanto *blind vias* são utilizadas para conectar a *bottom layer* ou *top layer* a uma camada interior. A utilização de *buried vias* ou *blind vias* ao invés de *through-hole vias*, reduz o acoplamento entre os campos gerados pelos sinais que percorrem as *vias* e as *layers* da PCB. A figura 2.3, retirada de [11], ilustra os diferentes tipos de *vias*.

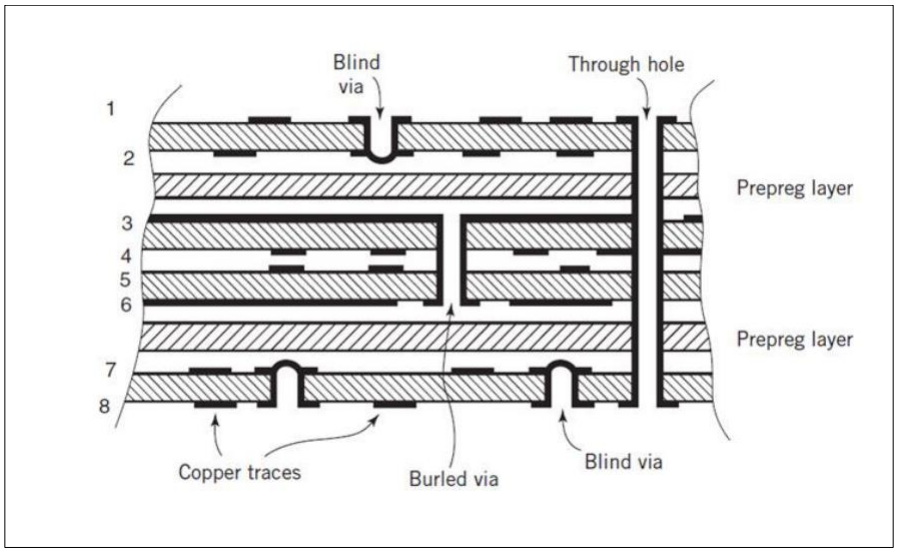


Figura 2.3: *Multilayer* PCB.

Durante os últimos anos, o tamanho do transístor tem vindo a reduzir, o que deixou espaço para aumentar o número de funcionalidades que cada circuito integrado (IC) pode realizar, consequentemente levando ao aumento do número de transístores e comutações nos ICs [11]. O aumento do número de funcionalidades, levou a um aumento do número de pinos de cada *IC*, o que do ponto de vista da PCB, significa um aumento do número de *traces* e dos possíveis cruzamentos entre *traces*. Em resposta ao aumento do número de *traces*, a largura das mesmas tem vindo a reduzir, bem como o espaço entre elas, e o número de *layers* têm vindo a aumentar, sendo possível recorrer a PCBs com 70 *layers* [11], [12].

O aumento do número de transístores levou a um aumento do número de comutações, e o aumento da frequência de comutação levou a uma redução do *rise* e *fall time* do sinais. Isto, juntamente com a redução na distância entre *traces*, levaram a um aumento do *cross talk* e da

radiação eletromagnética emitida pelas *traces* que interconectam estes sinais [11]. Além disto, o maior número de comutações em consequência do maior número de transístores e o maior declive na comutação dos sinais levaram a uma maior necessidade de energia. Se esta energia não estiver imediatamente disponível, provoca oscilações nos *traces* da tensões de alimentação dos ICs.

A introdução de um plano de massa, para frequências acima dos 100KHz, demonstrou-se uma solução muito útil para confinar campos elétricos e reduzir problemas de impedâncias [11]. O uso destes planos para intercalar *layers* de sinal ou *layers* de alimentação e sinal, é uma das principais técnicas para assegurar a compatibilidade eletromagnética (EMC) no desenho de PCBs. Apesar disto, a utilização do plano de massa para PCBs de duas *layers* pode acarretar problemas. Quando nestas PCBs existe *routing* em ambas as *layers*, não é possível desenhar um plano de massa sem descontinuidades, levando a variações na impedância do plano de massa, ao longo da PCB. Desta forma, só é possível tirar total partido da utilização de um plano de massa para PCBs com um número de *layers* superior a duas.

Tal como o plano de massa, a utilização de "planos de tensão de alimentação"também apresenta mais valias para a PCB. Esta abordagem em conjunto com o plano de massa, funciona como um condensador de *decouple*, impedindo que ruído se propague entre circuitos através das *traces* de tensões de alimentação dos ICs. Além disto, a utilização de "planos de tensão de alimentação"encurta o caminho de retorno, que por sua vez aumenta a EMC da PCB. Quando não existe a possibilidade de utilizar planos de alimentação em conjunto com planos de massa, é preferível a utilização de um plano de massa [13].

Como anteriormente referido, a utilização de planos de massa sem descontinuidades em PCBs de duas *layers* nem sempre é possível. Apesar disto, existem algumas boas práticas que podem ser utilizadas nestas PCBs para aumentar a sua EMC [11].

Para minimizar o acoplamento entre *traces*, deve ser evitado o uso de *traces* adjacentes em paralelo. Caso não seja possível, estas devem existir *traces* de massa intercaladas. Outra boa prática, é a colocação de condensadores de *decouple* e de ilhas de massa e de *power* junto dos ICs. Esta prática ajuda a minimizar o efeito das comutações dos transístores nas linhas de alimentação.

Além do já mencionado, normalmente PCBs trocam energia com outras PCBs. Estes sinais que viajam entre PCBs, estão sujeitos a EMIs provocadas por campos oriundos de fontes de

alimentação e transmissores radiofrequência (RF) ou televisão. Estes campos provocam tanto o aparecimento de ruído de modo comum, como de ruído normal. Caso este ruído não seja removido à entrada, o mesmo irá afetar a EMC da PCB. Ruído de modo comum acontece quando um campo afeta de igual forma um par de sinais balanceados (V_1 e G_1, S_1^+ e S_1^-), enquanto ruído normal acontece quando campos afetam de forma diferente os pares de sinais [11]. Uma boa estratégia para resolver estes problemas é a utilização de isolamento galvânico. Acoplamento capacitivo, ótico ou indutivo podem ser utilizados para isolar sinais digitais, enquanto o acoplamento indutivo pode ser utilizado para isolar alimentações. Além disto, o uso de filtros também é uma boa solução. Filtros passo-baixo e passa-banda podem ser utilizados para remover ruído normal, enquanto *common mode chokes* e amplificadores diferenciais podem ser utilizados para remover ruído de modo comum.

2.2 Sistemas Resilientes

Nos últimos anos, o rápido crescimento da complexidade dos sistemas e o aumento de custos associados com a sua perda de funcionalidade, revelaram a importância de aspetos ligados à resiliência dos sistemas [2]. Usualmente, a resiliência de um sistema é expressada através da probabilidade de um sistema executar a sua função (R), em determinadas condições de operação, e durante um período específico.

A resiliência de um sistema é um atributo expetável em qualquer aplicação, sendo que, o consumidor final espera que um produto não deixe de cumprir a sua função, durante o tempo garantido pelo fabricante. Todavia, em alguns casos o não funcionamento de um sistema pode ter consequências mais severas. Um exemplo é a falha de um subsistema de uma sonda espacial, que ao deixar de cumprir a sua funcionalidade está a comprometer o sucesso da missão desta sonda. O incumprimento da missão por parte da sonda, compromete longos períodos de desenvolvimento e elevados custos monetários. Outro exemplo é a falha de um subsistema de um avião, o qual por não ser capaz de cumprir a sua funcionalidade pode colocar em risco a vida dos passageiros. Os dois últimos exemplos apresentados, são classificados como sistemas críticos.

Sistemas críticos podem ser divididos em duas classes, *mission critical systems* e *safety critical systems* [14], [15]. *Mission critical systems* são sistemas onde o seu não funcionamento

ou o seu mau funcionamento podem comprometer o sucesso de uma tarefa que é crucial para uma missão. *Safety critical systems* são sistemas onde o seu não funcionamento, ou o seu mau funcionamento podem causar danos físicos ao utilizador, ou a quem o rodeia.

No contexto do setor automóvel, a classificação *Automotive Safety Integrity Level* (ASIL) é fornecida pelo *standard* ISO 26262 para classificar a criticidade de um sistema automóvel [15]. Esta classificação tem por base uma análise de riscos e eventos de perigo para o utilizador, definida pela ISO 26262, a qual assenta em três pilares [16]:

- *Severity* (S) - Define a gravidade dos danos para a vida das pessoas, em que S1 diz respeito ao grau mais leve de lesões e o S3 ao pior grau.
- *Exposure* (E) - Determina o risco de exposição a uma situação perigosa, em que E1 representa a probabilidade de exposição mais baixa e E4 representa a probabilidade de exposição mais alta.
- *Controllability* (C) - Explora o grau de controlo do condutor sobre o veículo em caso de falha de algum sistema, a menor perda de controlo sobre o veículo é dado por C1 e a maior perda por C3.

A classificação ASIL tem quatro estágios, ASIL A, ASIL B, ASIL C e ASIL D, sendo que o ASIL D representa o nível mais alto de criticidade.

2.2.1 Faltas, Erros e Falhas

A falta é uma condição anormal que pode causar a falha de um elemento [17]. Tipicamente faltas causam erros, os quais correspondem à diferença entre o valor medido e o valor real [17]. A propagação do erro leva à falha do sistema. A falha de um sistema é a não execução, ou a execução errada, da sua funcionalidade [17]. Dependendo do ponto de vista, faltas, erros e falhas podem possuir significados diferentes. Por exemplo, do ponto de vista do sistema, uma falta é uma falha num componente, sendo ele hardware ou software, o qual leva ao aparecimento dum erro no sistema, este erro por sua vez pode causar a falha do sistema.

Faltas podem ser classificadas de acordo com diferentes critérios. Segundo [17], a origem das faltas podem ser resumidas em três grandes grupos: faltas de desenvolvimento, faltas físicas e faltas de interação. Faltas de desenvolvimento englobam todas as faltas introduzidas

durante o desenvolvimento do sistema, por exemplo: faltas introduzidas na especificação, faltas de desenho, etc. Faltas físicas englobam todas as faltas que afetam o hardware, por exemplo: defeitos introduzidos no fabrico, envelhecimento dos componentes, etc. Faltas de interação englobam todas as faltas causadas por fatores externos, por exemplo: condições de operação, utilizador,etc.

Relativamente às faltas que afetam o hardware, as mesmas podem apresentar diferentes durações. Dependendo da sua duração, as faltas de hardware podem ser classificadas como faltas permanentes, temporárias, ou intermitentes [18]. Faltas temporárias ficam ativas durante pequenos períodos de tempo. Estas faltas tornam-se intermitentes quando apresentam um caráter repetitivo. Faltas permanentes mantêm-se ativas até que ações sejam tomadas.

2.2.2 Distribuição de Falhas

Usualmente em sistemas resilientes, são utilizadas funções para descrever as probabilidades de falha de uma amostra de sistemas durante um determinado período de funcionamento. A

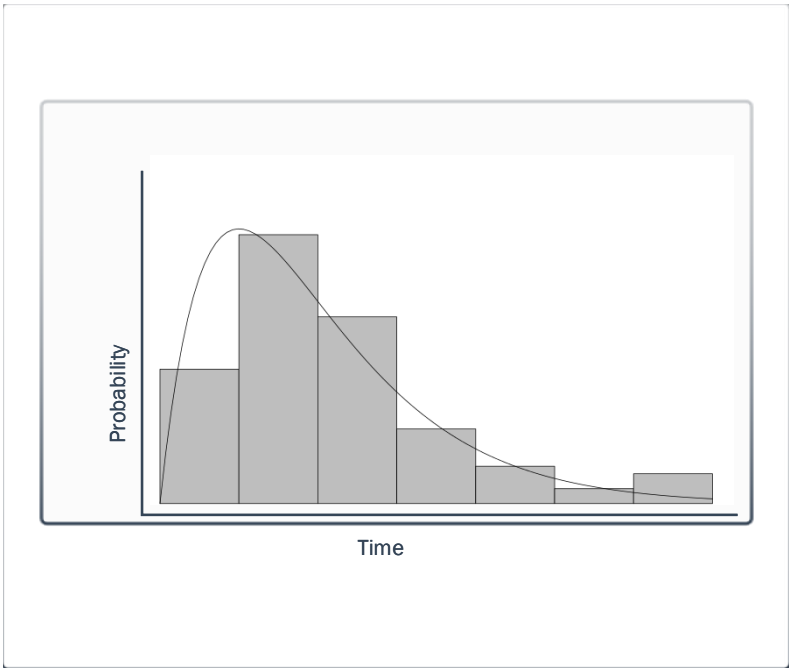


Figura 2.4: Exemplo de uma função PDF.

função densidade de probabilidade (PDF) (t) é uma representação contínua, que demonstra como a probabilidade de falha de uma amostra de sistemas ou componentes está distribuída no tempo [19]. Esta função pode ser obtida através dos resultados de testes acelerados, ou de simulações

de uma dada população. A figura 2.4 ilustra um exemplo de uma função PDF. A função PDF tem de respeitar a seguinte condição:

$$\int_0^\infty PDF(t)dt = 1 \tag{2.1}$$

A partir da função PDF é possível calcular as funções distribuição de probabilidade acumulada (CDF) (t) e a função resiliência (R) (t). A partir das funções PDF (t) e R (t) é possível calcular a função λ (t) [19]. A figura 2.5 ilustra as relações entre estas funções.

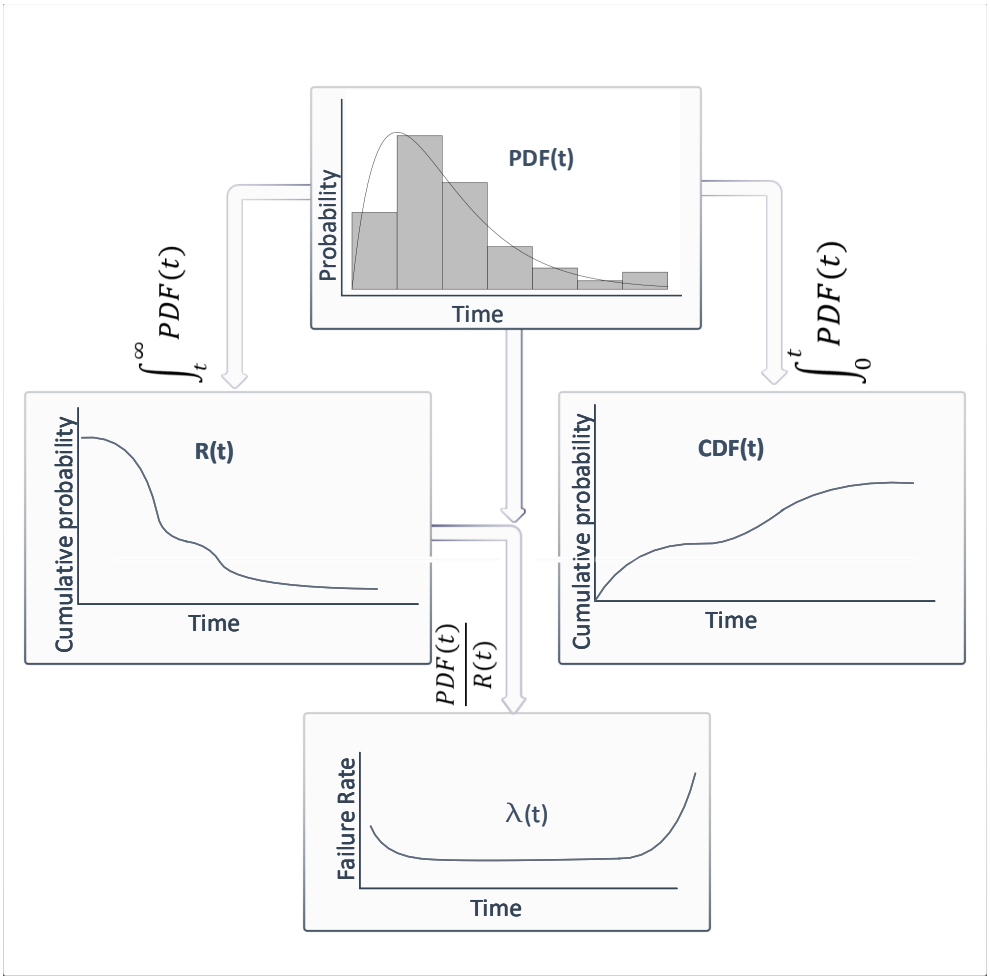


Figura 2.5: Relação entre as funções PDF (t), CDF (t), R (t) e λ (t).

A função CDF (t) exprime a não resiliência ou a probabilidade de falha acumulada de uma amostra de sistemas ou componentes ao longo do tempo [19]. Enquanto, a função R (t) exprime a resiliência ou a probabilidade acumulada de sobrevivência, de uma amostra de sistemas ou componentes ao longo do tempo [19]. Por último, a função taxa de falha (λ) (t), também conhecida como curva da banheira, exprime a evolução da taxa de falha de uma amostra de sistemas

ou componentes ao longo do tempo [19].

Curva da Banheira

Usualmente a função $\lambda(t)$ dos componentes ou sistemas, durante a sua vida, segue o formato de uma curva da banheira. Esta curva, tipicamente apresenta 3 fases : mortalidade infantil, vida útil e desgaste ou mortalidade senil [1], [2]. Um exemplo desta curva pode ser visualizado na figura 2.6.

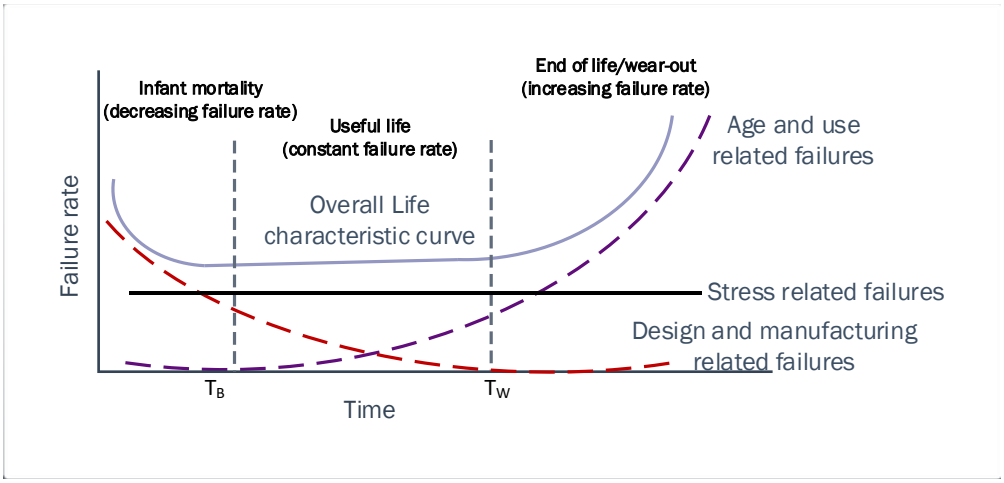


Figura 2.6: Curva da banheira.

A primeira fase é caraterizada por uma elevado λ, decrescendo à medida que o tempo avança. O elevado número de falhas nesta fase (falhas prematuras) deve-se principalmente a faltas de desenho, imperfeições nos componentes e a defeitos de fabrico.

Para mitigar as imperfeições nos componentes, uma boa prática é a escolha de componentes certificados. Componentes com certificados, por exemplo *Automotive Electronics Council* (AEC), *Military Performance Specification* (MIL-PRF) e *Aerospace Qualified Electronic Component* (AQEC) são testados de acordo com *standards*, garantindo um determinado nível de resiliência e qualidade. Alguns exemplos de *standards* são [20], [21], [22]: (1) AEC-Q10x e AEC-Q200-00x, no caso da indústria automóvel; (2) MIL-PRF 38535, no caso do setor militar; (3) AQEC, no caso do setor aeroespacial.

Outra forma de reduzir as imperfeições nos componentes, defeitos de fabrico, e as faltas de desenvolvimento, é através testes acelerados. A utilização de testes qualitativos pretende avaliar de que forma o sistema falha, com o intuito de determinar as possíveis causas destas falhas e corrigi-las através de uma nova reiteração do desenho ou através da melhoria do processo de

fabrico. Quando já não é possível eliminar faltas através do redesenho, estes testes podem ser utilizados para envelhecer o sistema, fazendo com que este avance para a fase seguinte da curva da banheira [1], [23].

A segunda fase é caraterizada por um λ constante. Nesta fase, usualmente os sistemas falham de forma aleatória, sendo estas falhas causadas, principalmente, por faltas de iteração. Condições de operação como temperatura, vibração, ou radiação causam *stress* nos componentes, o que excessivamente pode levar ao aparecimento de faltas aleatórias. Para prevenir este tipo de faltas, uma boa seleção de componentes é fundamental. Na escolha dos componentes, é necessário tomar em consideração se os limites de operação dos componentes vão de encontro aos limites de operação do sistema. Uma técnica utilizada nesta escolha é o *derating*, a qual consiste em escolher componentes com os limites de operação superiores aos limites do sistema, aumentando assim a sua tolerância ao *stress* causado por estes elementos [2]. Outro fator relacionado com as condições de operação são as descarga electrostática (ESD). Estas podem surgir no sistema com diferentes tempos e com diferentes intensidades causando a falha do hardware. Para prevenir o seu efeito, técnicas como isolamento galvânico podem ser utilizadas para isolar os *inputs* do sistema, ou, para isolar subsistemas, prevenindo assim, a propagação de uma ESD [2].

A terceira fase é caracterizada por um rápido crescimento do λ. Este rápido crescimento deve-se ao desgaste ou envelhecimento dos componentes do sistema. Esta fase de vida do sistema não pode ser evitada, apenas pode ser prevenida pela manutenção. Para prever quando um componente ou sistema irá atingir o seu fim de vida, testes acelerados (quantitativos) ou simulações podem ser utilizadas para determinar a distribuição de falhas do sistema e consequentemente a sua longevidade. Neste sentido, a manutenção deve ser aplicada antes dos componentes ou sistemas atingirem a fase de mortalidade senil [1].

Para descrever o comportamento das falhas das diferentes fases da curva da banheira, são utilizadas diferentes distribuições [1], [24], [25]. Para a mortalidade infantil, são sugeridas as distribuições de gamma, Weibull ou log-normal. Para a vida útil são sugeridas a distribuição exponencial ou de Weibull. Para o desgaste ou mortalidade senil são sugeridas as distribuições de Weibull, gama ou Gauss. Por vezes, para descrever os padrões de falhas de alguns componentes ou sistemas, poderá ser necessário sobrepor distribuições.

2.2.3 Taxa de Falha, MTBF and MTTF

A taxa de falha (λ), o tempo médio entre falhas (MTBF) e o tempo médio até à falha (MTTF), são métricas utilizadas em resiliência. O λ representa a frequência de falhas de uma amostra de sistemas ou componentes, por unidade de tempo, usualmente expresso em horas [1]. O MTBF representa o tempo médio entre falhas de uma amostra de sistemas ou componentes, sendo esta métrica usualmente utilizada para sistemas reparáveis [1]. O MTTF representa o tempo médio até à falha de uma amostra de sistemas ou componentes. Em oposição ao MTBF, o MTTF é utilizado para sistemas não reparáveis [1].

O MTBF e MTTF podem ser obtidos a partir da função PDF do sistema pela equação [24]:

$$MTTF(t) = \int_0^\infty t * PDF(t)dt \tag{2.2}$$

Na distribuição exponencial o MTBF pode ser calculado a partir do λ, utilizando a equação (2.3) [1]. O valor de MTBF é muito útil para realizar comparações entre componentes, porém, o mesmo não fornece informação suficiente para determinar a longevidade de um componente [26], [27]. Um bom exemplo das limitações apresentadas por este dado estatístico é apresentado em [28]. Neste exemplo, é apresentada uma amostra de 500000 seres humanos com 25 anos de idade. Durante um ano, foram recolhidos dados sobre as falhas (morte) destes seres humanos. Ao longo de um ano, morreram 625 pessoas, o que corresponde a um λ de 0.125%. Como visto anteriormente na distribuição exponencial, o MTBF pode ser obtido a partir do inverso do λ, o que neste caso corresponde a um MTBF de 800 anos. Como é possível constatar, 800 anos não correspondem ao tempo de vida expetável para um ser humano. Tal como neste exemplo, o valor de MTBF não exprime o tempo de vida expetável para um competente, porém componentes com maior MTBF possuem maior resiliência.

$$MTTF = 1/\lambda \tag{2.3}$$

O λ é o valor da função λ (t) para a vida útil do componente ou sistema, o que corresponde ao valor da função λ (t) quando esta é aproximadamente constante [24].

2.2.4 Redundância

Sistemas recorrem a redundância para aumentar a sua probabilidade de sucesso e alcançar a tolerância a faltas. A redundância consiste em fornecer mais do que um meio ao sistema para alcançar uma determinada função [29]. É um facto que a utilização de redundância aumenta drasticamente a resiliência de um sistema. Contudo, a mesma também aumenta: os custos, o tempo de desenvolvimento, a complexidade, os custos de manutenção e o seu tamanho [3]. A utilização de redundância ao nível do sistema pode ser dividido nas 4 seguintes categorias [18], [30]: (1) redundância de hardware; (2) redundância de software; (3) redundância de informação; (4) redundância temporal.

Redundância de Hardware

Redundância de hardware consiste na replicação de componentes ou subsistemas, aos quais, o sistema recorre quando as suas contrapartes falham. Esta categoria de redundância tem 3 tipos: estática, dinâmica e híbrida.

Na redundância estática, quando um componente falha, este é substituído pela sua contraparte, sem que o sistema fique ciente da sua falha (*fault-masking*). As arquiteturas *n-modular redundancy* (NMR), figura 2.7, são exemplos da utilização deste tipo de redundância [31]. Neste exemplo, uma arquitetura é constituída por N módulos redundantes, e por um único *voter*. O *voter* arbitra a saída dos módulos redundantes. Como esta arquitetura apresenta apenas um *voter*, o mesmo representa um ponto único de falha. Porém, a complexidade do *voter* é menor do que a dos restantes módulos, possibilitando que o ponto único de falha seja tolerado. O número de faltas que uma arquitetura NMR pode tolerar segue a equação $N = 2k + 1$, em que N representa o número de módulos redundantes e K o número de faltas.

Na redundância ativa ou dinâmica, inicialmente, as faltas são identificadas e localizadas, e posteriormente o sistema é reconfigurado de forma a retomar a sua operacionalidade. Exemplos de arquiteturas que exploram o uso de redundância ativa temos: *duplication with comparison* e *standby*.

A figura 2.8 apresenta uma arquitetura *duplication with comparison* [18]. Esta arquitetura é constituída por dois módulos redundantes. O resultado de ambos os módulos é avaliado por um comparador, e em caso de discrepâncias é gerado um sinal de erro. Esta arquitetura possibilita a deteção de erros, porém, a mesma não permite a recuperação de faltas.

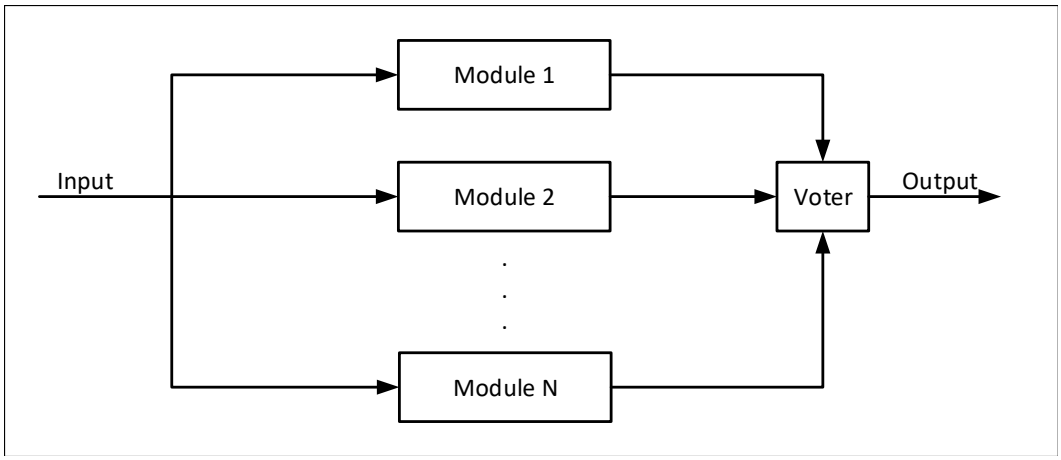


Figura 2.7: Arquitetura NMR.

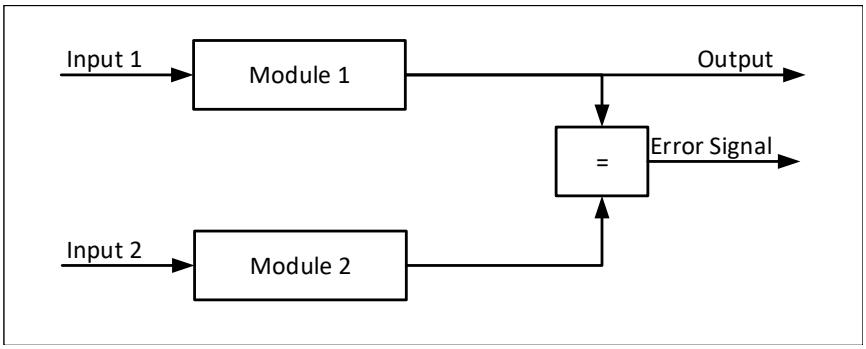


Figura 2.8: Arquitetura *Duplication with comparison*.

Relativamente às arquiteturas *standby*, as mesmas utilizam os módulos redundantes como *backup*. Na figura 2.9, retirada de [31], é possível visualizar dois exemplos [31]. Cada uma das arquiteturas dos exemplos é constituída por um módulo primário e um de *backup*. Ambos os módulos são monitorizados pelos módulos de diagnóstico de faltas. Caso surja uma falta, os módulos de diagnóstico reportam ao módulo de lógica arbitral.

No exemplo a), ambos os módulos, primário e *backup*, encontram-se em operação, em caso de falha de um módulo, o módulo de lógica arbitral desativa o módulo em falha (*hot standby*). No exemplo b), apenas o módulo primário encontra-se em funcionamento, em caso de falha, este módulo é desativado pelo módulo de lógica arbitral, sendo substituído pelo módulo de *backup* (*cold standby*). Pelo facto da arquitetura (*cold standby*) apenas ativar o módulo de *backup* em caso de falha do primário, a mesma reduz o desgaste causado ao módulo de *backup*, e o consumo energético do sistema. Contudo, para ativar o módulo de *backup* é necessário reconfigurar o sistema, sendo que durante esse período o mesmo está inoperacional. Estas arquiteturas podem tolerar até N-1 faltas, em que N é o número de módulos redundantes.

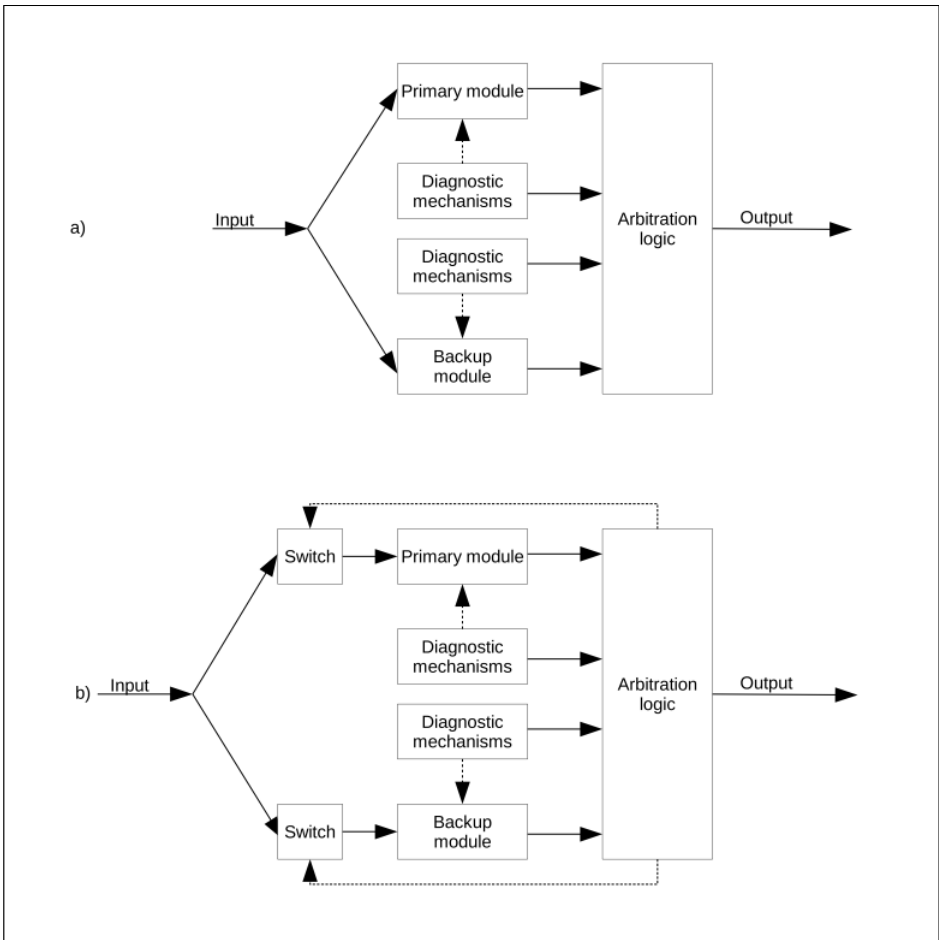


Figura 2.9: Exemplos de arquiteturas que exploram redundância dinâmica, a) *hot standby*, b) *cold standby*

A redundância híbrida combina a utilização de redundância estática e dinâmica, tirando partido das vantagens de cada uma. Na ocorrência de uma falta, para prevenir erros momentâneos, técnicas de redundância passiva são utilizadas para mascarar faltas. Recorrendo a técnicas de redundância ativa, as faltas são localizadas e removidas, e o sistema é reconfigurado, retomando assim a sua operacionalidade. As arquiteturas *self-purging* e *duo-duplex* são exemplos de arquiteturas que exploram o uso deste tipo de redundância, figura 2.10 ([31]).

A arquitetura *self-purging* é uma arquitetura NMR, à qual são acrescentados interruptores. Estes permitem desativar os módulo redundantes. Dependendo do resultado da votação, o *voter* desativa o módulo em falha, e reconfigura-se diminuindo o número de entradas. Esta arquitetura pode tolerar até N-2 faltas, sendo que N representa o número de módulos redundantes.

A arquitetura *duo-duplex* é baseada na arquitetura *hot standby*, porém os mecanismos de deteção de faltas são substituídos por dois *duo-duplex*. Cada *duo-duplex* é constituído por dois

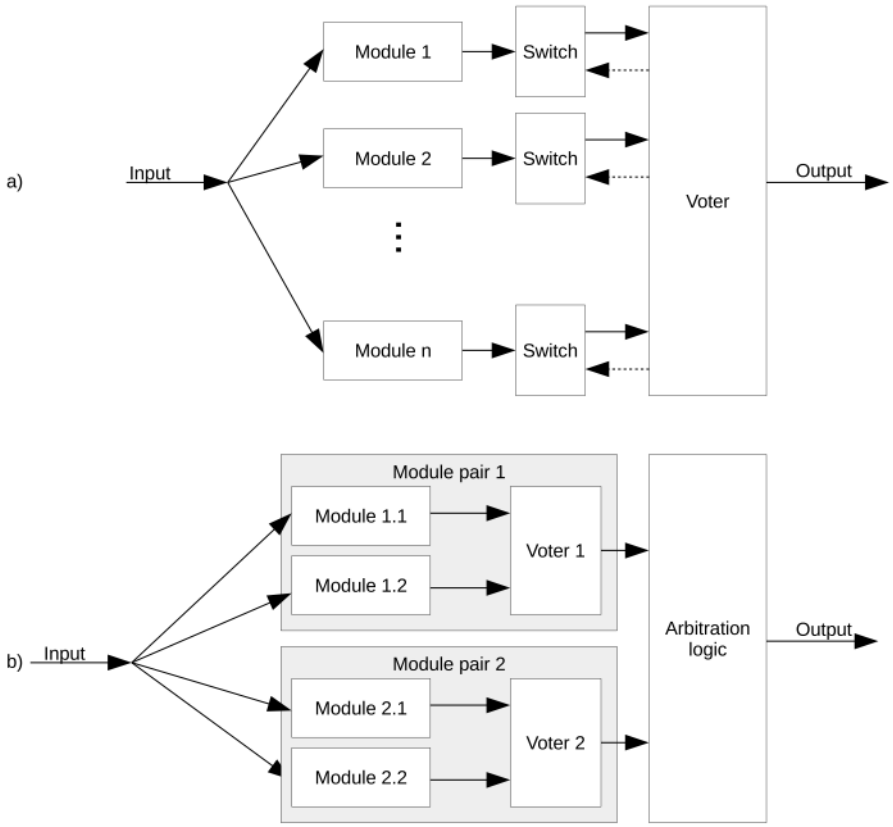


Figura 2.10: Exemplos de arquiteturas que exploram redundância híbrida, a) *self-purging*, b) *duo-duplex*

módulos redundantes, e um *voter*. A comparação das saídas de ambos os módulos redundantes, permite ao *voter* identificar faltas. Inicialmente, tal como na arquitetura *hot standby*, apenas, o módulo primário está ativo. No caso de ser detetada uma falta, o mó dulo de lógica arbitral desativa o *duo-duplex* primário, sendo substituído pelo *duo-duplex* de *backup*. Comparando esta arquitetura com a arquitetura *hot standby*, esta arquitetura apresenta mecanismos de deteção de faltas mais simples, contudo, a mesma necessita de um número maior de módulos. Arquiteturas *duo-duplex* podem apenas tolerar uma falta.

Redundância de Software

Redundância de software apresenta dois tipos de técnicas: *single version* ou *multi version*. As técnicas *single version* pretendem aumentar a tolerância a faltas de uma versão de software. As técnicas *multi version* recorrem a diferentes versões do mesmo software para alcançar a tolerância a faltas. É importante salientar que ambas as técnicas podem ser aplicadas aos diferentes

níveis do sistema de software [32]: procedimento, processo, aplicação, sistema operativo, etc.

As técnicas *single version* introduzem mecanismos que permitem a deteção, contenção e remoção de erros originados por faltas de desenho. Alguns exemplos de técnicas de tolerância a faltas são [18], [32]: *timing checks*, *coding checks*, *partitioning*, *modularization*, *error detection*, *error handling*, *checkpoint and restart*, *process pairs* e *data diversity*.

As técnicas *multi version* recorrem a múltiplas versões do mesmo software, executadas em paralelo ou sequência. O princípio de utilização destas técnicas assenta no princípio da diversidade, onde diferentes componentes (diferentes algoritmos), produzidos por diferentes equipas (diferentes ferramentas, diferentes desenhos), falham de forma diferente [32]. As múltiplas versões de software podem ser estruturalmente utilizadas como: (1) alternativas, permitindo meios alternativos para deteção de faltas; (2) pares, permitindo a deteção de faltas através da comparação; (3) em grupos, permitindo mascarar faltas através de votação. Alguns exemplos destas técnicas são [18], [32]: *recovery blocks*, *N-Version Programming* e *N Self-Checking Programming*.

Redundância de Informação

A redundância de informação é utilizada como meio para alcançar tolerância a faltas nos dados a transmitir, ou a armazenar. A forma mais comum deste tipo de redundância é a codificação. Esta, através da introdução de *check bits* aos dados possibilita a deteção de faltas e em certos casos a correção de alguns *data bits*. Alguns exemplos de esquemas de codificação usados em redundância de informação são [31]: *parity codes*, *double-inverted coding*, *cyclic redundancy check* (CRC)s e *AN-codes*.

Redundância Temporal

Redundância temporal consiste na repetição de uma determinada computação, a comparação entre os resultados das diferentes computações permite detetar discrepâncias. A discrepância entre resultados, indica a existência de faltas temporárias. Em relação à redundância de hardware e à redundância de informação, a redundância temporal não necessita da adição de recursos extras, quer lógicos ou físicos. Contudo, a mesma aumenta o tempo necessário para realizar o processamento.

Redundância temporal apenas permite a deteção de faltas temporárias [5], [31]. Na ocorrência de faltas permanentes, as diferentes repetições irão produzir resultados iguais, o que impossibilita a deteção de discrepâncias. Porém, através da combinação de redundância temporal com redundância de informação é possível detetar faltas permanentes. Exemplos de técnicas que combinam redundância de informação com redundância temporal são [18]: *alternating logic*, *recomputing with shifted operands*, *recomputing with swapped operands* e *recomputing with duplication with comparison*.

Redundância e Diversidade

A utilização do mesmo hardware e software, redundância homogénea, desenvolvido pela mesma equipa, pode resultar em faltas de modo comum. Faltas de modo comum são faltas que podem ocorrer em dois ou mais componentes/subsistemas ao mesmo tempo [18]. Para colmatar esta lacuna, surge o termo diversidade que consiste na utilização de diferentes meios, técnicas ou tecnologias, na execução da mesma tarefa [31]. A utilização de diversidade irá prevenir a falha comum dos diversos componentes.

A diversidade pode ser usada em todas as formas de redundância, como redundância de hardware, software, etc, podendo ser classificada como redundância heterogénea. Exemplos de técnicas que fornecem diversidade ao nível do software são [15]: *N-version (Multiversion) programming* e *recovery blocks*. Relativamente à diversidade ao nível do hardware, um possível exemplo é a utilização de sensores diferentes para adquirir a mesma variável, por exemplo diferentes sensores de posição angular (magneto-resistivos e óticos) colocados na mesma localização física.

2.2.5 Métodos para Estimar a Resiliência de um Sistema

Para estimar a resiliência de um sistema existem duas possíveis abordagens [1], [2], [5], [6]: experimental ou analítica. A figura 2.11 ilustra ambas as opções.

Na abordagem empírica ou experimental, é utilizada uma amostra com N sistemas para recolher dados sobre as falhas destes sistemas. A percentagem de falhas em relação ao número de sistemas permite obter uma indicação sobre a resiliência dum sistema, com uma determinada precisão. Por exemplo, 1000 protótipos de um sistema foram testados durante N horas, onde

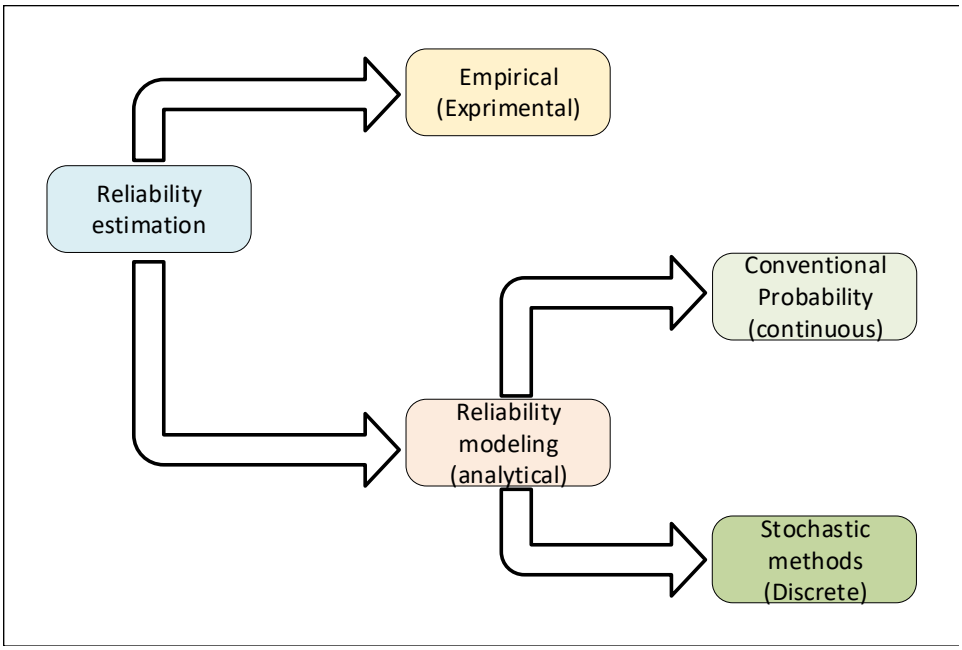


Figura 2.11: Modelação e estimação de resiliência.

ocorreram 100 falhas. Deste exemplo é possível concluir que durante N horas de funcionamento, o sistema testado apresenta uma resiliência de 90%.

As desvantagens das abordagens experimentais consistem no tamanho da amostra, na logística necessária para realizar experiências com este tipo de amostras, e na duração que estas experiências podem alcançar. Por exemplo, para estimar a resiliência de um isolador capacitivo a Texas Instruments utilizou uma amostra com 66423 componentes [33]. Nestas experiências, o elevado tamanho das amostras está interligado com a precisão que é pretendida no resultado desta abordagem. Relativamente à duração da experiência, componentes resilientes podem demorar longos períodos de tempo até falharem. Utilizando o exemplo anterior, a amostra foi testada durante 1000 horas sem que nenhuma falha ocorresse. Uma técnica muito utilizada para reduzir o tempo do processo experimental são os testes acelerados, esta técnica é descrita no capítulo 2 (2.2.7).

A abordagem analítica consiste na modelação de um sistema, e no uso de probabilidade de falha de cada um dos seus componente para determinar a probabilidade de falha do sistema. Nesta alternativa, a probabilidade de falha de um sistema depende também da forma como os seus componentes estão interligados. Segundo [6], esta alternativa é útil para auxiliar a escolha de componentes e para fornecer valores quantitativos para comparação entre arquiteturas do estado da arte. Este autor também evidencia que a estimativa obtida a partir desta alternativa

é pouco exata, sendo isto devido à dificuldade de incorporar no modelo do sistema, todos os fatores que levam ao aparecimento de faltas no sistema.

2.2.6 Métodos Analíticos: Modelação de Resiliência

Nesta abordagem, um modelo matemático de resiliência de um sistema é deduzido a partir de um modelo de resiliência de um sistema. O modelo de resiliência de um sistema é um diagrama de blocos, que demonstra as diferentes alternativas que o sistema possui para alcançar a sua função. Cada uma destas alternativas corresponde a uma cadeia de componentes. A figura 2.12 demonstra um exemplo de um modelo de resiliência de um sistema. O modelo matemático de resiliência de um sistema, permite calcular a probabilidade de falha de um sistema, com base nas probabilidades de falha dos seus componentes e das interligação entre estes componentes.

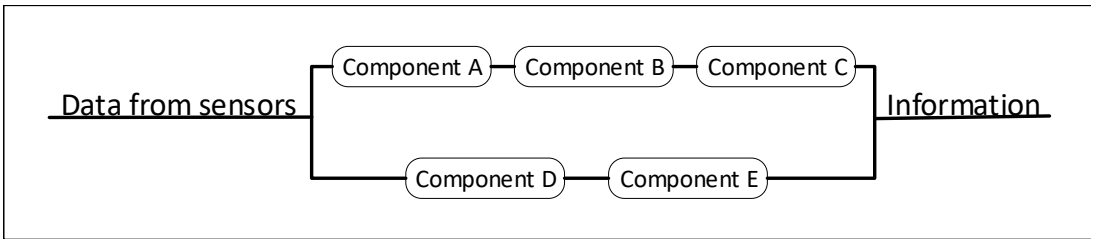


Figura 2.12: Exemplo de um modelo de resiliência do sistema.

Para deduzir o modelo matemático de resiliência de um sistema existem duas alternativas, uma que recorre aos teoremas convencionais de probabilidade e uma que recorre a métodos estocásticos. Relativamente à primeira alternativa, existem 3 métodos que podem ser utilizados para a dedução de um modelo matemático, o primeiro consiste na aplicação direta dos teoremas de probabilidade convencional, a segunda consiste na utilização de tabelas de verdade e a terceira consiste na utilização de diagramas lógicos [1], [29]. Na alternativa que recorre a métodos estocásticos, podem ser utilizados métodos como por exemplo cadeias de Markov ou processos regenerativos, para deduzir um modelo matemático do sistema [2]. Nesta dissertação, apenas serão abordadas as cadeias de Markov e o método de Monte Carlo. Os métodos de Markov podem ser utilizados para sistemas onde a probabilidade de falha dos seus componentes é independente do tempo, ou seja, a mesma é constante. Para sistemas onde a probabilidade de falha dos componentes é independente do tempo, ou a sua complexidade não permite a dedução de um modelo matemático, usualmente, são utilizados métodos de Monte Carlo, também conhecidos como simulações de Monte Carlo.

Nesta secção, inicialmente, serão apresentados os três métodos utilizados para deduzir um modelo matemático recorrendo aos teoremas convencionais de probabilidade, e posteriormente, as cadeias de Markov e o método de Monte Carlo. Os exemplos apresentados para os métodos utilizados para deduzir um modelo matemático recorrendo aos teoremas convencionais de probabilidade, foram retirados de [29].

Probabilidade Convencional

Neste método, os teoremas convencionais de probabilidade são aplicados diretamente às diferentes configurações dos componentes (séries, paralelos ou séries-paralelos), de forma a sintetizar o modelo matemático de resiliência de um sistema.

Para um paralelo de dois componentes, figura 2.13, retirada de [29], com probabilidades de falha Q_A e Q_B, o sistema falha quando ambos os componentes falharem. A probabilidade de falha do sistema ($Q_{Sistema}$) pode ser calculada a partir da probabilidade de dois eventos arbitrários ocorrerem em simultâneo, equação 2.4.

$$Q_{Sistema} = Q_A * Q_B \tag{2.4}$$

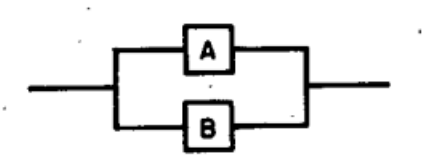


Figura 2.13: Exemplo de um modelo com dois componentes em Paralelo.

Num sistema composto por dois componentes em série com probabilidades de sucesso P_A e P_B, figura 2.14, retirada de [29], o sistema falha quando um dos componentes falhar. A probabilidade de falha do sistema ($Q_{Sistema}$), pode ser calculada a partir da probabilidade de um dos eventos ocorrer, equação 2.5.

$$Q_{Sistema} = 1 - P_{Sistema} = \ = 1 - P_A * P_B \tag{2.5}$$

Aumentando a complexidade, para um sistema composto por um conjunto de componentes em série e paralelo, figura 2.15, retirada de [29]. Neste sistema existem 4 alternativas para o sistema atingir o sucesso ($P_{Sistema}$), sendo elas: (1) B_1 e C_1; (2) B_2 e C_2; (3) A_1 e C_1; (4)

Figura 2.14: Exemplo de um modelo com dois componentes em série.

A_1 e C_2. Considerando a probabilidade de sucesso dos componentes P_{C1}, P_{C2}, P_{B1}, P_{B2}, P_A e sendo $P_{C1} = P_{C2}$ e $P_{B1} = P_{B2}$, a probabilidade de falha do sistema ($Q_{Sistema}$) pode ser calculada a partir das equações 2.6.

$$
\begin{aligned}
Q_{Sistema} &= 1 - P_{Sistema} \\
P_{Sistema} &= P(sucesso\ do\ sistema\ com\ A\ a\ funcionar) * P_A \\
&+P(sucesso\ do\ sistema\ com\ a\ falha\ do\ componente\ A) * (1 - P_A) \\
P(sucesso\ do\ sistema\ com\ A\ a\ funcionar) &= (2P_C - P_C^2) \\
P(sucesso\ do\ sistema\ com\ a\ falha\ do\ componente\ A) &= 2P_B * P_C - (P_B * P_C)^2
\end{aligned}
\tag{2.6}
$$

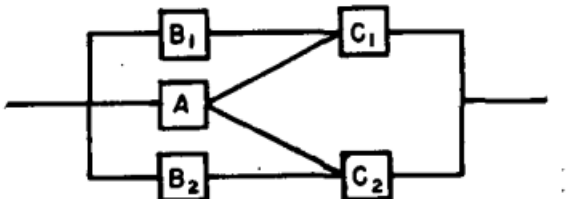


Figura 2.15: Exemplo de um modelo com configurações série e paralelo.

Tabela de Verdade

Neste método, uma tabela de verdade é utilizada para sintetizar a resiliência de um sistema. Na aplicação deste método, tal como em qualquer tabela de verdade, inicialmente são preenchidas as combinações possíveis para o número máximo de entradas da tabela de verdade. Neste método, o número máximo de entradas possíveis corresponde a $2^{NComponentes}$. A figura 2.16, retirada de [29], representa a tabela de verdade para o exemplo do modelo de resiliência com combinações série e paralelo, apresentado anteriormente. Como é possível visualizar na figura 2.16, cada um das preposições da tabela de verdade corresponde a um componente. O estado de um componente, para uma determinada entrada, corresponde ao valor lógico da preposição para essa entrada.

Entry No.	B_1	B_2	C_1	C_2	A	Success or Failure	P_S
1	0	0	0	0	0	F	-
2	0	0	0	0	1	F	-
3	0	0	0	1	0	F	-
4	0	0	0	1	1	S	.03888
5	0	0	1	0	0	F	-
6	0	0	1	0	1	S	.03888
7	0	0	1	1	0	F	-
8	0	0	1	1	1	S	.00972
9	0	1	0	0	0	F	-
10	0	1	0	0	1	F	-
11	0	1	0	1	0	S	.01008
12	0	1	0	1	1	S	.00432
13	0	1	1	0	0	F	-
14	0	1	1	0	1	S	.00432
15	0	1	1	1	0	S	.00252
16	0	1	1	1	1	S	.00108
17	1	0	0	0	0	F	-
18	1	0	0	0	1	F	-
19	1	0	0	1	0	F	-
20	1	0	0	1	1	S	.00432
21	1	0	1	0	0	S	.01008
22	1	0	1	0	1	S	.00432
23	1	0	1	1	0	S	.00252
24	1	0	1	1	1	S	.00108
25	1	1	0	0	0	F	-
26	1	1	0	0	1	F	-
27	1	1	0	1	0	S	.00112
28	1	1	0	1	1	S	.00048
29	1	1	1	0	0	S	.00112
30	1	1	1	0	1	S	.00048
31	1	1	1	1	0	S	.00028
32	1	1	1	1	1	S	.00012
						Σ All success paths =	.13572

Figura 2.16: Tabela de verdade para o modelo com configurações série e paralelo.

Com todas as combinações preenchidas, para cada uma das entradas da tabela de verdade, é analisado se um sistema obtém sucesso ou não. Esta análise é realizada com base no estado dos componentes para uma determinada entrada, e no modelo de resiliência do sistema. Para cada entrada da tabela de verdade em que um sistema obteve sucesso, é calculada a probabilidade de sucesso através da multiplicação da probabilidade de sucesso ou insucesso, de cada um dos seus componentes, dependendo do estado destes componentes.

A probabilidade de sucesso de um sistema, é calculada a partir da soma das probabilidades de sucesso, para cada uma das entradas em que um sistema foi bem sucedido. A equação 2.7 representa o modelo matemático de resiliência, obtido a partir da tabela de verdade apresentada para o exemplo do modelo de resiliência com combinações série e paralelo.

$$\begin{aligned} P_S = {} & B_1 * C_2 + \overline{B_1} * B_2 * C_2 + A * \overline{B_1} * B_2 * C_2 + A * \overline{B_1} * C_1 * \overline{C_2} \\ & + B_1 * B_2 * \overline{C_1} * C_2 + B_1 * \overline{B_2} * \overline{C_1} * C_2 * A \end{aligned} \tag{2.7}$$

Diagramas lógicos

Neste método, são utilizados diagramas lógicos para sintetizar a resiliência de um sistema. Como é possível visualizar na figura 2.17, retirada de [29], o modelo das configurações séries paralelo foi substituído por uma rede de interruptores. Neste novo modelo, cada interruptor fechado representa o sucesso de um componente, em oposição, cada interruptor aberto representa a falha de um componente. Cada um dos diferentes caminhos representa uma alternativa que o sistema possui para obter sucesso. A probabilidade de sucesso de cada um destes caminhos, é calculada através da multiplicação da probabilidade de sucesso ou insucesso, de cada componente presente no caminho, dependendo do estado do interruptor que representa o componente. Para calcular a probabilidade de sucesso do sistema, são somadas as diferentes probabilidades de sucesso de cada caminho. A equação 2.9 representa o modelo matemático de resiliência do sistema.

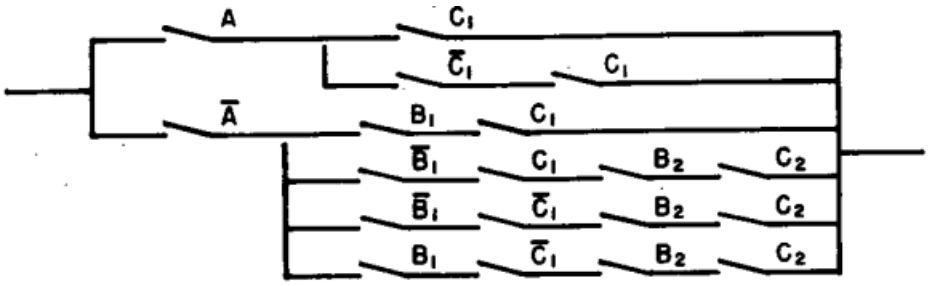


Figura 2.17: Diagramas lógicos para o modelo com configurações série e paralelo.

$$P_S = P_A[P_{C1} + Q_{C1} * P_{C2}] + Q_{A1}[P_{B1} * P_{C1} + Q_{B1} * P_{C1} * P_{B2} * P_{C2} + Q_{B1} * Q_{C1} * P_{B2} * P_{C2} + Q_{B1} * P_{C1} * P_{B2} * P_{C2}] \tag{2.8}$$

Cadeias de Markov

Ao contrário de processos determinísticos, os quais apenas apresentam um único modo de evoluir, processos estocásticos podem apresentar várias possibilidades, ou até mesmo infinitas possibilidades. Segundo [34], a definição de processos estocásticos é algo vago, pois qualquer conjunto de variáveis aleatórias pode ser considerada um processo estocástico.

As cadeias de Markov são modelos analíticos utilizados para descrever um processo Markoviano. Um processo Markoviano é uma classe de processos estocásticos que pode ser descrita como processos sem memória, ou seja, o estado futuro de um sistema só depende do estado atual [35]. Este processo foi assim designado em homenagem ao matemático Russo Andrey Markov [36]. As cadeias de Markov apresentam diversas aplicações em áreas como a biologia, física e ciências da computação [37], [38]. Na área dos sistemas resilientes, modelos discretos de Markov são usualmente utilizados para modelar sistemas em que a probabilidade de falha se mantém constante ao longo do tempo [1].

Simplificando, os modelos discretos de Markov são baseados no conceito de estados e transições de estados [5]. Considerando um exemplo composto apenas por um componente, este componente pode ser modelado recorrendo a uma máquina de estados. Esta máquina de estados contém apenas 2 estados: o estado S1 representa o componente em modo operacional e o estado S2 que representa a falha do componente. Esta máquina de estados é apresentada na figura 2.18. As transições entre estados podem ser efetuadas a cada instante $\Delta(t)$, o λ ,como anteriormente mencionado, representa a taxa de falha do componente.

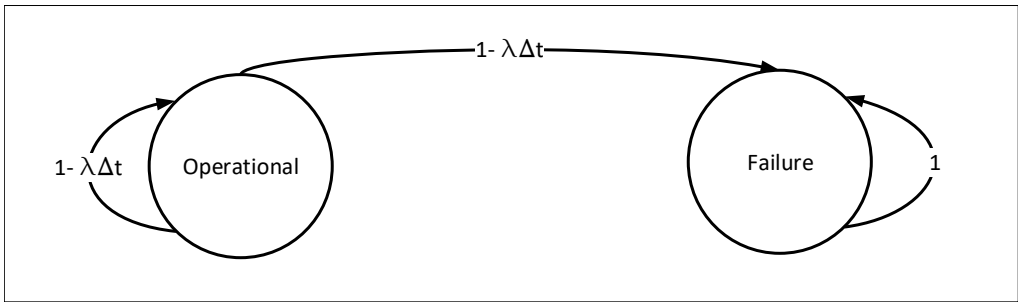


Figura 2.18: Exemplo de um sistema composto apenas por um componente.

A partir da máquina de estados desenhada, é possível escrever as equações diferenciais que permitem calcular a probabilidade do componente estar operacional ou em falha.

$$\begin{aligned} P_{S1}(t+\delta t) &= (1-\Delta t) * P_{S1}(t) + P_{S2}t \\ P_{S2}(t+\delta t) &= \lambda\Delta t * P_{S1}(t) + P_{S2}t \end{aligned} \tag{2.9}$$

Uma das grandes desvantagens das cadeias Markovianas é que com o aumento da complexidade de um modelo, o número de equações diferenciais a resolver aumenta exponencialmente.

Por exemplo, para um sistema composto por 10 componentes, em que cada componente possuí 2 estados, é necessário resolver 2^{10} equações diferenciais [39].

Simulação de Monte Carlo

Simulação de Monte Carlo é uma método utilizado para modelar e simular sistemas complexos. Este método também pode ser descrito como uma ferramenta para estimar a solução de problemas matemáticos, tais como modelos matemáticos de sistemas, através de números aleatórios. O nome deste método deriva da sua utilização de números aleatórios, os quais, por exemplo, podem ser obtidos a partir de uma roleta de um casino de Monte Carlo.

As origens deste método remontam à agulha de Buffon, experiência esta que mais tarde foi utilizada por Laplace para estimar o valor da constante π [40], [41]. Este método foi também utilizado por Kelvin para estimar uma solução para os integrais relacionadas com a sua teoria da cinética dos gases. No último século, durante cerca de 30 anos, este método foi utilizado maioritariamente pelo ramo da ciências nuclear. Esta utilização quase exclusiva, deveu-se sobretudo à falta de poder computacional, pois este método requer uma elevada quantidade de memória e um tempo extensivo. Um exemplo da utilização deste método no ramo da ciência nuclear, é o trabalho realizado por Fermi, von Neumann e Ulam, os quais utilizaram métodos de Monte Carlo para resolver integrais de seis dimensões no contexto do projeto Manhattan, durante a segunda guerra mundial [42].

Atualmente, devido ao avanço das ciências computacionais, este método apresenta um vasto conjunto de aplicações no cálculo do risco em gestão de projetos, inteligência artificial, etc [43]. Outro possível exemplo de aplicação deste método, é a estimação da resiliência de sistemas.

O método de Monte Carlo consiste em amostrar um conjunto de números (X1,X2,X3...Xn) de uma distribuição de números aleatórios, dentro da gama do domínio do sistema. Para cada *input* aleatório, é calculada e guardada a resposta do sistema (Y(X1),Y(X2)...Y(Xn)). O cálculo da resposta de um sistema a um *input* aleatório, constituí um *trial* de Monte Carlo. O números de *trials* a executar depende da precisão que é pretendida para a distribuição do sistema. Na base deste método está a lei dos grandes números, a qual enuncia que quanto maior a amostra, mais a média das amostras será uma boa estimativa da população [29].

Em comparação com as cadeias de Markov, o método de Monte Carlo é mais simples de

resolver, porém, este apresenta como grande desvantagem o elevado número de *trials* e consequentemente, o tempo de cálculo [39].

2.2.7 Testes Acelerados

Como anteriormente referido, é possível estimar a resiliência de um sistema através de métodos empíricos. Inicialmente, esta abordagem era realizada utilizando condições normais de operação. Contudo, a mesma demorava longos períodos de tempo podendo comprometer o *time-to-market* de um produto. Para reduzir a duração destas experiências, foram criados os testes acelerados.

Testes acelerados utilizam condições de operação como temperatura, vibração, pressão, humidade, etc, mais exigentes que aquelas sentidas durante a normal operação de um sistema. Estas condições de operação mais exigentes, são denominadas fatores de aceleração ou *stress*, e são aplicadas com o intuito de acelerar o envelhecimento e o aparecimento de faltas no sistema.

Durante a realização dos testes acelerados, é importante que a intensidade dos fatores de aceleração aplicados, não ultrapassem os limites do sistema. Caso estes limites sejam ultrapassados, podem ser desencadeadas faltas irrealistas, ou seja, faltas que nunca seriam sentidas durante a normal operação do sistema [1].

Testes Quantitativos versus Testes Qualitativos

Testes acelerados podem ser realizados com dois propósitos, o de quantificar a longevidade de um sistema, ou o de avaliar a qualidade do sistema [1].

Testes qualitativos pretendem desencadear a falha do sistema o mais rápido possível. Para isso, nestes testes são utilizadas amostras mais pequenas e níveis de *stress* mais intensos, o que incluí múltiplos fatores de *stress* ou fatores de *stress* que variam no tempo. Alguns exemplos destes testes são: *elephant tests*, *torture tests*, *highly accelerated life test* (HALT) e *shake & bake tests* [23].

Testes quantitativos pretendem estimar a resiliência ou probabilidade de falha de um sistema ao longo da sua vida. Para isto, estes testes aplicam fatores de aceleração com o intuito de acelerar o aparecimento de faltas e assim obter a distribuição de falhas que carateriza um sistema. Com a distribuição que carateriza a vida acelerada do sistema, é possível determinar a

distribuição para a vida normal do sistema, utilizando um modelo que relacione as duas. Alguns exemplos destes modelos são: modelo de Arrhenius e modelo de Eyring.

Relações Entre Vida Acelerada e Vida Normal

Como mencionado anteriormente, a aplicação de fatores de aceleração irá desencadear o envelhecimento acelerado e o aparecimento de faltas. Estas faltas poderão dar origem à falha dos sistemas em teste. A falha dos sistemas em teste, e o momento em que estas ocorreram, irão permitir deduzir a função PDF do sistema. Assim como apresentado no capítulo 2.2.2, para descrever a função PDF, também podem ser utilizadas distribuições como por exemplo: Weibull, gama e exponencial. Cada uma destas distribuições possuí um parâmetro de vida, por exemplo a Weibull possuí o η, e a exponencial o MTBF. Através da utilização de um modelo que relacione a vida acelerada e a vida normal, para o fator de aceleração utilizado, é possível determinar a aceleração sofrido pelo sistema, e assim relacionar o parâmetro de vida acelerada com o parâmetro de vida normal. Contudo, antes de aplicar qualquer um destes modelos é necessário determinar experimentalmente os seus parâmetros, os quais diferem entre sistemas.

Para deduzir os parâmetros de cada modelo, é necessário obter diferentes funções PDF do sistema, para diferentes intensidades do mesmo fator de aceleração. Relacionado as diferentes funções, é possível deduzir os parâmetros dos modelos que relacionam a vida acelerada e a vida normal. A figura 2.19, retirada de [23], apresenta um exemplo da utilização de três funções PDF de um sistema, obtidas experimentalmente, para determinar os parâmetros do modelo de Arrhenius. Neste exemplo, cada uma das funções PDF foi descrita utilizando a distribuição de Weibull, e os parâmetros B e C representam, respetivamente, os parâmetros energia de ativação (Ea) e $v0$ do modelo de Arrhenius. No capítulo 2 (2.2.7) é apresentado em mais detalhe o modelo de Arrhenius e a determinação dos seus parâmetros.

Modelo de Arrhenius

O modelo de Arrhenius para testes acelerados consiste na equação proposta pelo físico sueco Svante August Arrhenius em 1884, a qual foi baseada no trabalho do químico Jacobus Henricus van 't Hoff [44], [45]. A equação de Arrhenius, equação (2.10), permite calcular a variação da constante de velocidade e também a energia de ativação (E_a) de uma reação química. O parâmetro Ea, é a energia necessária para uma molécula participar numa reação, ou seja, a

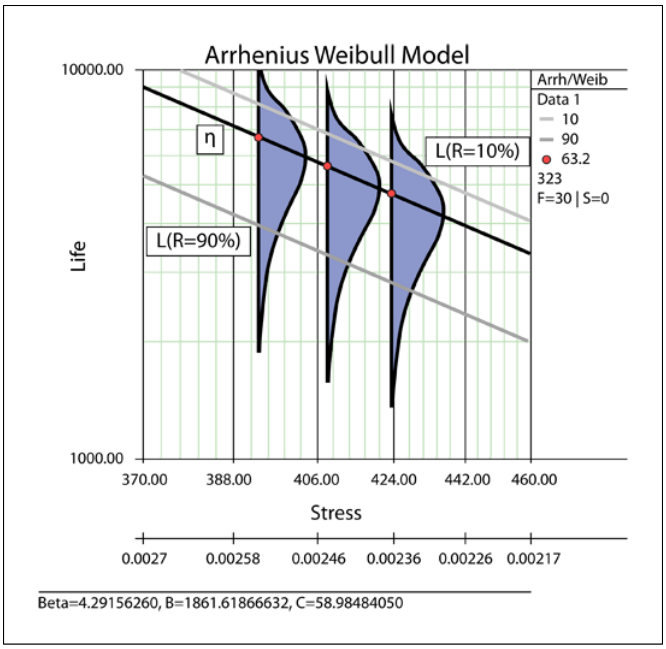


Figura 2.19: Exemplo de um sistema composto apenas por um componente.

mesma representa o efeito da temperatura numa reação química. A relação de Arrhenius foi obtida empiricamente, porém, o físico forneceu uma justificação física e uma interpretação para a fórmula [46].

$$v(T) = v_0 e^{-\frac{E_a}{k*T}} \tag{2.10}$$

onde:

$T =$ Temperatura absoluta(K).

$v =$ Velocidade da reação.

$v_0 =$ Constante não térmica desconhecida.

$k =$ Constante de Boltzmann$(8.6173303{\times}10^{-5} eV K^{-1})$.

$E_a =$ Energia de ativação(eV).

Em testes acelerados, este modelo é o mais utilizado para relacionar a vida acelerada e a vida normal, quando a temperatura é utilizada como fator de aceleração [23]. No caso de componentes eletrónicos, o modelo de Arrhenius pode ser utilizado para determinar o fator de

aceleração produzido por temperaturas entre 0° e 150° C [2]. O fator de aceleração (A) sofrido por um sistema para uma temperatura constante durante os testes acelerados (T2), temperatura esta que é superior à temperatura normal de operação do sistema (T1), pode ser calculado da seguinte forma:

$$A = \frac{v(T1)}{v(T2)} = \frac{v_0 e^{\frac{E_a}{k}(\frac{1}{T1})}}{v_0 e^{\frac{E_a}{k}(\frac{1}{T2})}} = e^{\frac{E_a}{k}(\frac{1}{T1}-\frac{1}{T2})} \tag{2.11}$$

Para determinar o parâmetro Ea, são necessárias 3 funções PDF do sistema, obtidas experimentalmente a partir de 3 testes com diferentes intensidades de temperatura. Duas das funções PDF são utilizadas para o cálculo do fator de aceleração e as três para verificar o modelo. Por exemplo, supondo que as três funções PDF foram descritas utilizando uma distribuição exponencial, o fator de aceleração sofrido pelo sistema pode ser calculado a partir do quociente dos MTBFs de duas funções PDF, como é possível observar na equação 2.12. Com o fator de aceleração calculado, através da utilização da equação (2.11), é possível determinar o parâmetro Ea. Para componentes eletrónicos, o Ea apresenta valores entre 0.3 e 0.7 [2].

$$A = \frac{MTBF_1}{MTBF_2}. \tag{2.12}$$

onde:

$A =$ Fator de aceleração sofrido pelo sistema.

$MTBF_1 =$ parâmetro de vida da resposta n°1.

$MTBF_2 =$ parâmetro de vida da resposta n°2.

Capítulo 3

Simulação e Testes Acelerados para Hardware Resiliente

No desenvolvimento de hardware é difícil quantificar o nível de resiliência de um desenho. Como previamente mencionado, uma técnica utilizada para aumentar a resiliência de um desenho é a redundância. Esta, baseia-se na replicação de componentes, aos quais o sistema recorre quando as suas contrapartes falham, todavia, nem sempre é claro quais os componentes que devem ser replicados. Existe um conjunto de abordagens que auxiliam na estimação da resiliência de um sistema.

Uma possível abordagem analítica é a utilização de equações diferenciais que descrevem os princípios físicos de envelhecimento de um componente. Contudo, com o aumento da complexidade do sistema e nível de integração, o uso destas equações para determinar a resiliência do sistema à custa das equações de cada componente individual, torna-se quase impossível.

Outra alternativa é uma abordagem experimental. Esta recorre a testes acelerados, onde protótipos são testados durante longos períodos de tempo, com condições de operação mais exigentes que as sentidas durante a operação normal. Através da inferência dos resultados destes testes, é possível determinar a probabilidade de falha do sistema ou de um componente. Apesar de tal abordagem ser altamente confiável, a mesma envolve custos e longos períodos de tempo para ser realizada, com a consequência dos seus resultados poderem demonstrar a necessidade de um redesenho e posterior teste.

Uma outra alternativa é o uso de simulações. Estas, baseando-se na probabilidade de falha de cada um dos componentes, recorrem a métodos estocásticos para estimar a resiliência de sistemas. As probabilidades de falha usadas nestas simulações são obtidas através de testes acelerados, ou através de outras simulações.

Cada uma das abordagens sugeridas tem as suas vantagens e desvantagens, não sendo sempre claro quais as mais indicadas a utilizar no desenvolvimento do projeto. Com o intuito de potenciar o uso das abordagens mencionadas, é sugerida a sua utilização em determinados momentos do desenvolvimento de sistemas embebidos resilientes. No decorrer desta dissertação, o foco recaiu sobre o desenvolvimento de modelos de simulação e de um *setup* de testes.

O desenvolvimento de sistemas embebidos resilientes é na sua natureza um processo iterativo. Apesar disso, a utilização das abordagens sugeridas pretende reduzir o número de iterações requerido. A figura 3.1 representa uma das primeiras fases do desenvolvimento destes sistemas.

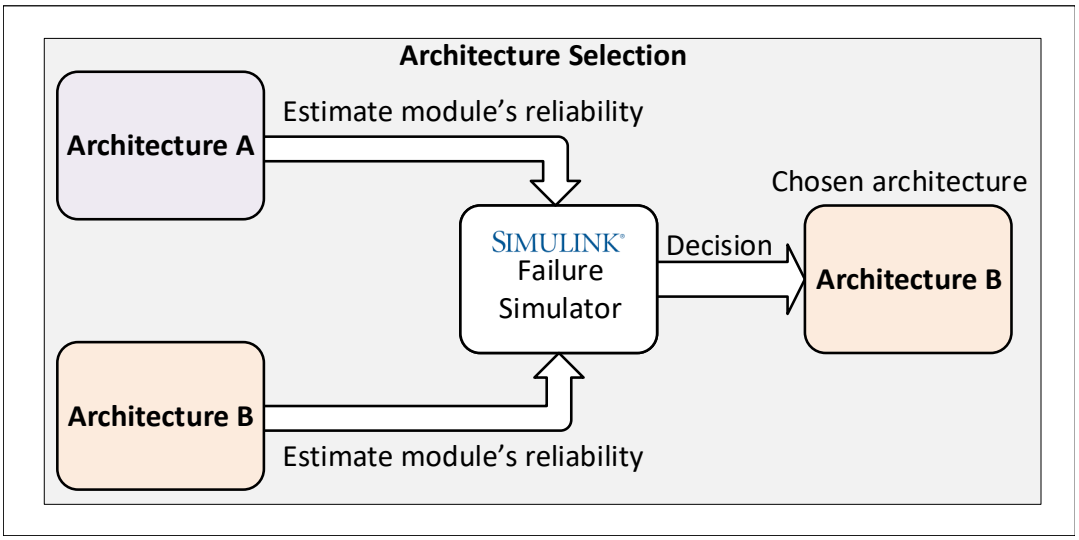


Figura 3.1: Fase1: seleção da arquitetura do sistema.

Relembrando o *workflow* mencionado no capítulo 2, numa das primeiras fases do desenvolvimento de um sistema embebido, é selecionada uma arquitetura. Usualmente, esta escolha é baseada em fatores como o custo, tempo de desenvolvimento e resiliência desejada. É um facto que a utilização de redundância aumenta a resiliência de um sistema, porém, a sua utilização também resulta num aumento da complexidade, do custo e do tempo de desenvolvimento do sistema. Nesta fase ainda não são conhecidos os componentes que constituem cada um dos subsistemas da arquitetura, sendo que não é possível obter uma estimativa fiável da resiliência do sistema. Porém, é possível comparar arquiteturas através de simulações que recorrem a valores de resiliência, que por sua vez tem como base conhecimento adquirido em projetos passados. Após ser alcançado um consenso sobre a arquitetura do sistema, é iniciado o desenvolvimento de hardware e software, como representado na figura 3.2.

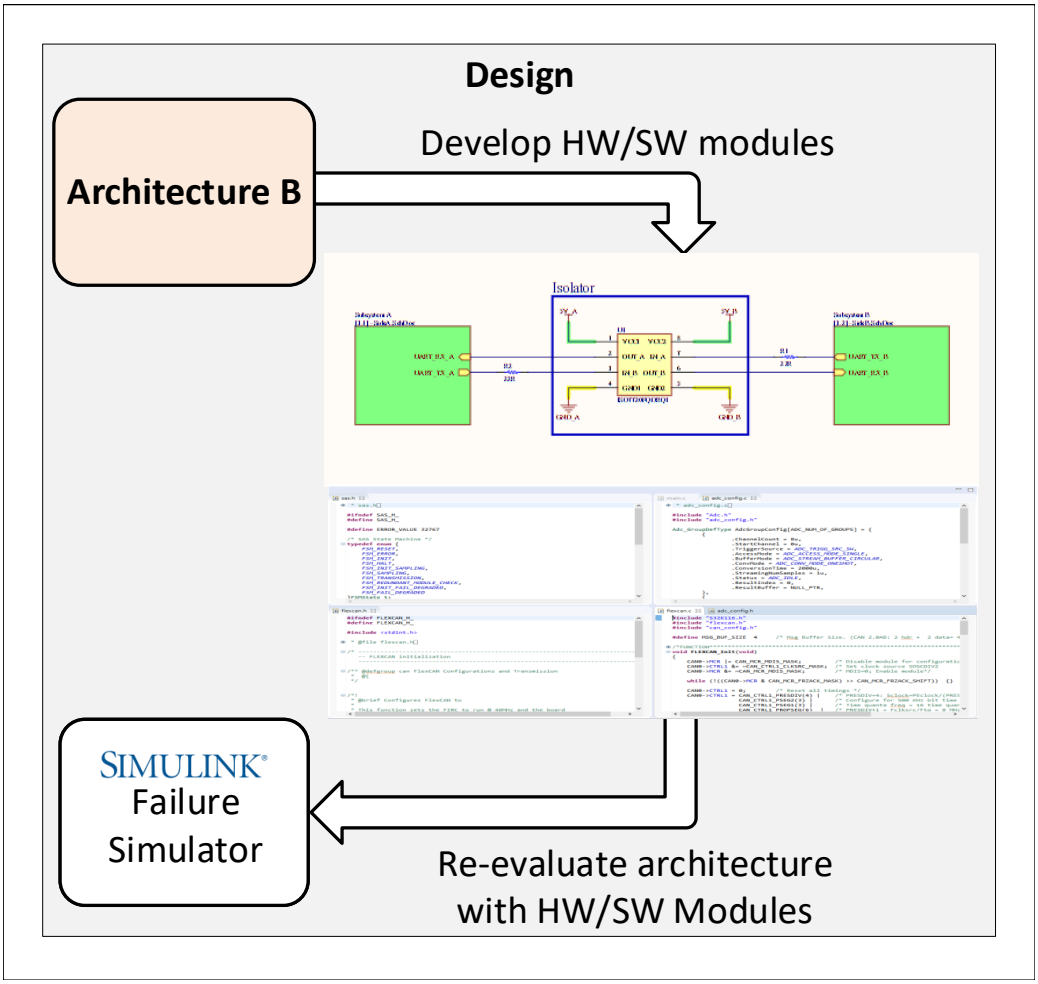


Figura 3.2: Fase 2: desenho.

Usualmente, durante o desenho de sistemas embebidos o hardware e software são desenhados em paralelo, e, por equipas diferentes. No desenvolvimento de hardware resiliente, o MTBF, e o custo são usados como elementos chave na escolha dos componentes que compõem os subsistemas da arquitetura. O uso de simulações aquando desta escolha, auxilia na perceção do impacto que a probabilidade de falha de cada componente tem na resiliência do sistema. Com a escolha dos componentes que constituem os subsistemas da arquitetura é possível refinar o modelo de simulação, utilizando agora, a probabilidade de falha de cada componente. Esta simulação resulta numa estimativa mais próxima do quanto robusto o sistema poderá ser, quando em funcionamento. A comparação desta estimativa, com a estimativa realizada com base nas probabilidades de falha arbitradas, poderá corroborar a escolha da arquitetura, ou em oposição, demonstrar que a estimativa inicial era demasiado otimista. Caso a arquitetura proposta seja demonstrada como inadequada, os valores obtidos nesta fase podem ser reutilizados numa

nova comparação arquitetural, fornecendo assim, estimativas mais realistas às comparações da arquitetura atual com outras arquiteturas alternativas. Após a implementação de uma arquitetura satisfatória, é sugerida a utilização de métodos experimentais para avaliar a resiliência do sistema, como representado na figura 3.3.

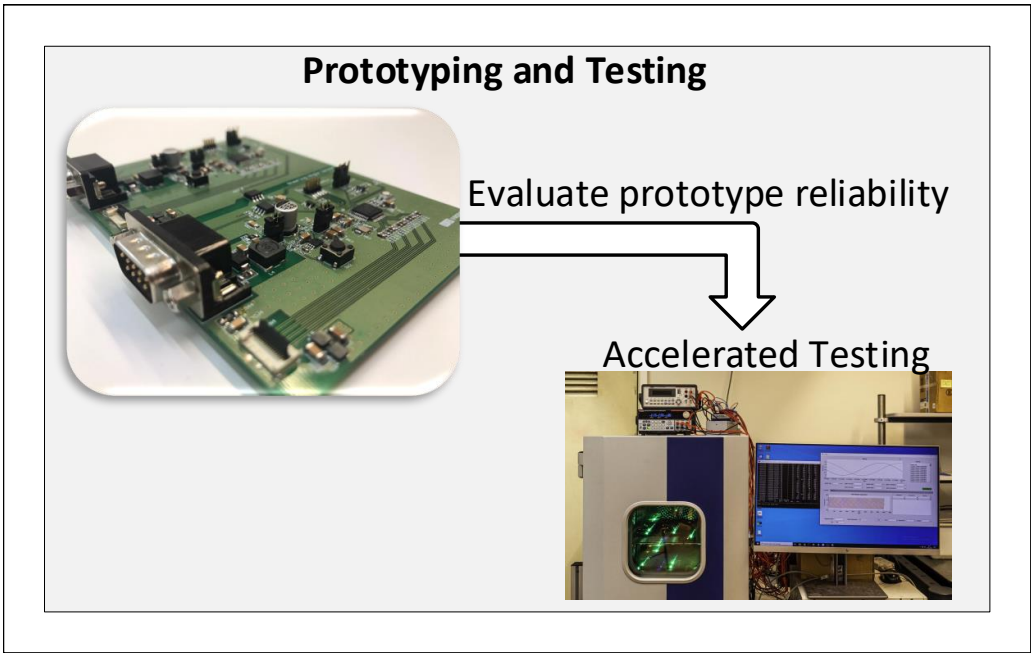


Figura 3.3: Fase 3:Prototipagem e teste.

A necessidade de reavaliar o sistema, advém de faltas de desenho que possam não ter sido detetadas durante o processo de simulação. Para além disso, o processo de fabrico pode não ser ideal, havendo a possibilidade de introdução de novas faltas. Desta forma, o uso de testes de vida acelerada vem trazer uma nova estimativa, a qual é ainda mais fiável que a obtida através das simulações. Tais testes recorrem a condições de operação mais exigentes que aquelas a que o sistema está sujeito durante o seu normal funcionamento, as quais são aplicadas com o intuito de simular o envelhecimento do sistema e provocar faltas. Caso a estimativa de resiliência obtida durante este processo não satisfaça as métricas estipuladas, existe a necessidade de identificar a origem das suas falhas. Se na origem das falhas estiver o processo de fabrico, e caso este não possa ser melhorado, existe a necessidade de um redesenho, ou mesmo, do recurso a uma nova arquitetura que mitigue estas deficiências na produção. Se na origem das falhas estiverem faltas introduzidos durante a fase de desenho, os mesmos devem ser corrigidos. A figura 3.4 representa a visão global da utilização das abordagens sugeridas, bem como as diversas iterações entre as fases de desenvolvimento.

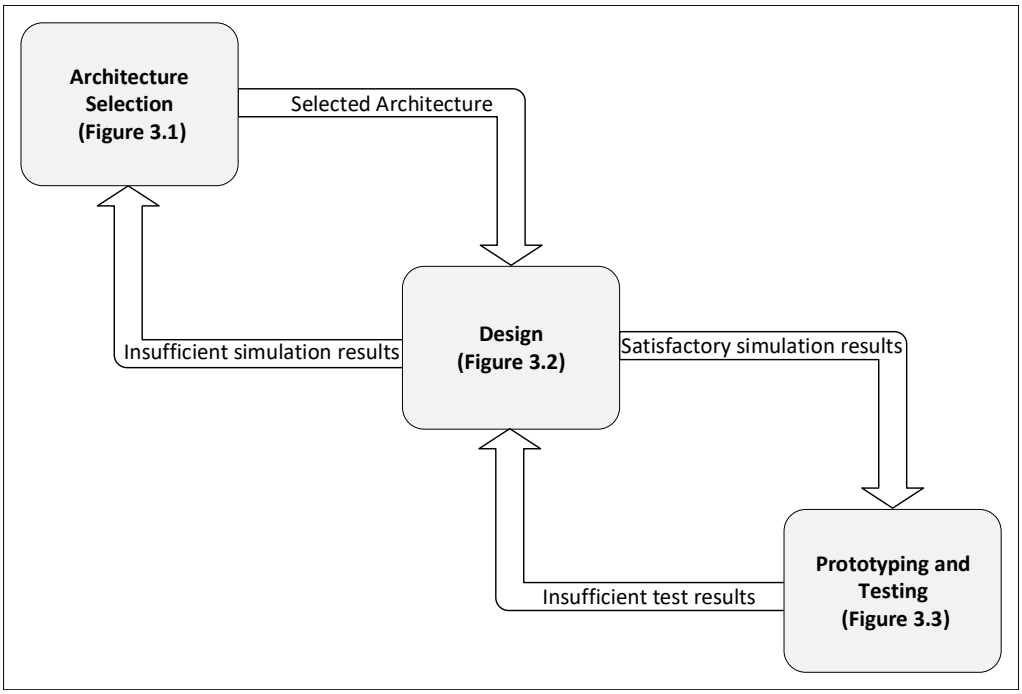


Figura 3.4: Interações entre as diversas fases do desenvolvimento do sistema.

3.1 Simulações de Resiliência do Sistema

Como mencionado anteriormente, as simulações de resiliência pretendem auxiliar a escolha da arquitetura e o desenho do sistema. Estas simulações utilizam métodos de Monte Carlo com o propósito de avaliar se o sistema consegue manter a sua funcionalidade, mesmo quando os seus componentes/subsistemas falham. No modelo de simulação desenvolvido, o Simulink é utilizado como ferramenta de modelação do sistema, mais precisamente, o Stateflow, o qual permite que sejam simulados fluxogramas ou FSMs. Em cada iteração de simulação, é decidido se vai ser injetada uma falha num componente, tendo por base aleatoriedade e a sua probabilidade de falha.

A figura 3.5 representa um exemplo conceptual de uma simulação de resiliência do sistema. Os diferentes módulos da simulação podem representar componentes eletrónicos, ou blocos de alto nível, incluindo camadas de software, permitindo assim, diferentes níveis de granularidade. Esta flexibilidade possibilita que a simulação possa auxiliar as diversas fases do desenvolvimento do projeto. No exemplo apresentado, uma arquitetura que apresenta redundância de ADC, quer a nível de hardware, quer a nível de software, é simulada com um alto nível de abstração. Em cada iteração da simulação, com base em números aleatórios e na probabilidade de falha, podem ser injetadas falhas nos componentes. Através da análise das alternativas que o sistema possui

para cumprir a sua funcionalidade, é possível determinar se o sistema sobrevive à falha dos seus componentes. No caso do sistema sobreviver, a simulação avança para a próxima iteração.

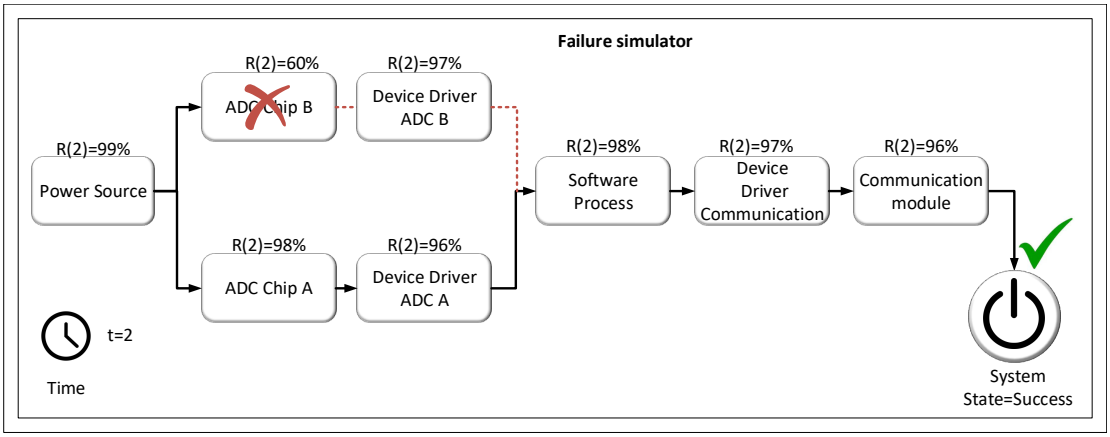


Figura 3.5: Simulações de resiliência, *overview* conceptual.

3.1.1 Ecossistema do Failure Simulator

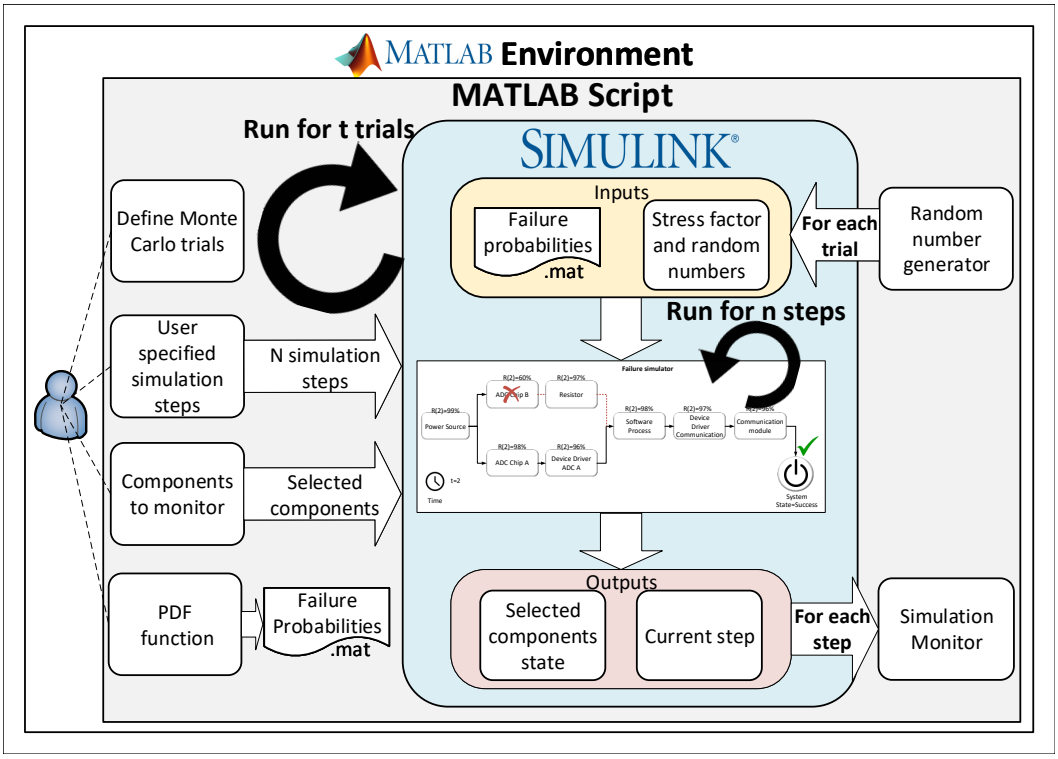


Figura 3.6: Ambiente de Simulação.

A figura 3.6 apresenta um diagrama do ambiente desenvolvido para estas simulações de Monte Carlo. Cada *step* da simulação é conceptualizado como uma hora de funcionamento

do sistema, o número total de *steps* de cada simulação, corresponde a um tempo de vida do sistema. Cada simulação é conceptualizada como um *trial* de Monte Carlo. Através do conjunto de tempos de vida do sistema é desenhada a função PDF do sistema.

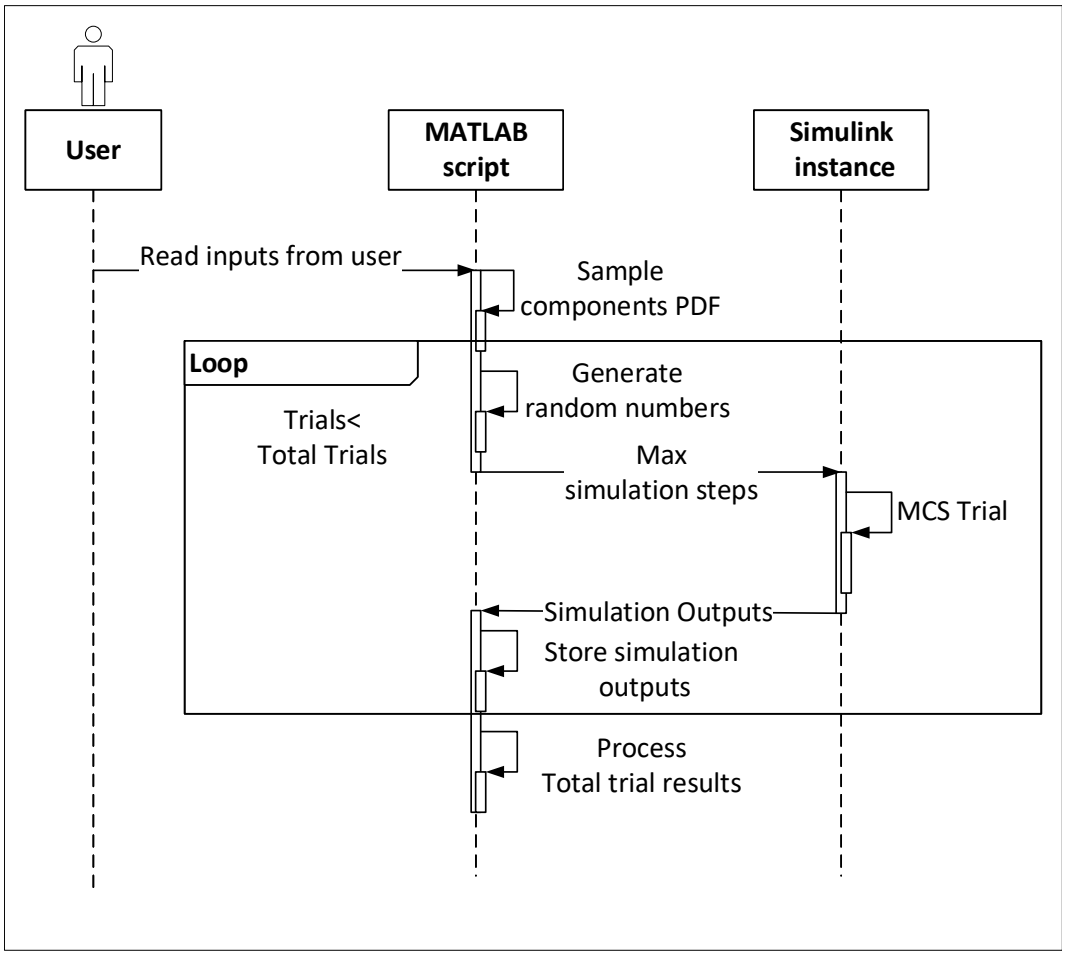


Figura 3.7: Diagrama de sequência das simulações.

O diagrama de sequência da figura 3.7 detalha o procedimento para um conjunto de *trials*. Um *script* Matlab executa um número de simulações igual ao número de Monte Carlo *trials* definido pelo utilizador. O utilizador também específica o número máximo de *steps* que cada simulação pode atingir, sem que ocorra a falha do sistema, bem como a função PDF de cada componente, e os componentes a monitorizar. Inicialmente, os valores da função PDF a serem utilizados em simulação são amostrados e guardados num ficheiro para serem posteriormente carregados durante cada *trial*. Em cada *trial*, as probabilidades de falha aleatórias são geradas e após a execução da simulação, são guardados num ficheiro os dados dos componentes monitorizados, assim como, o número de *steps* de simulação executados antes da falha do sistema.

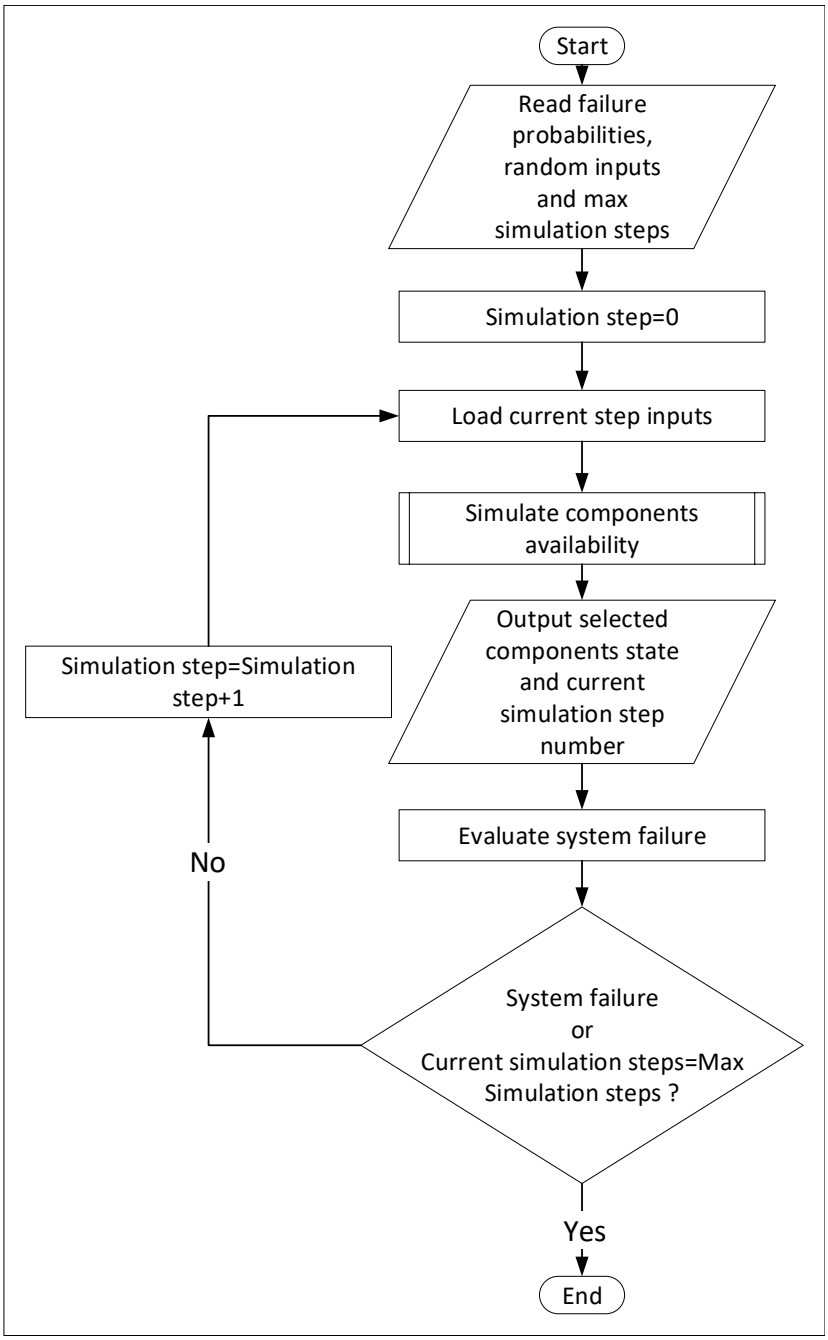


Figura 3.8: Fluxograma de um Monte Carlo *trial*.

O fluxograma da figura 3.8 descreve o procedimento para cada *trial*. Em cada *trial*, inicialmente, são lidos os números aleatórios gerados e carregados os dados de probabilidade de falha de cada componente. Tanto os números aleatórios, como a probabilidade de falha são *inputs* dos componentes, sendo estes atualizados no ínicio de cada *step* com o valor correspondente. Além destes dois, também a temperatura de operação é um *input* dos componentes, a qual se mantém constante ao longo da simulação. Para cada *step* é calculada a disponibilidade de cada

componente, o *output* dos componentes selecionados é guardado no *workspace* do Matlab, e a funcionalidade do sistema avaliada. Este procedimento é repetido até à falha do sistema, ou até a simulação atingir o número máximo de *steps* pretendidos pelo utilizador.

3.1.2 Modelação e Interligação dos Componentes

Na modelação de componentes, apenas é utilizada a disponibilidade da sua funcionalidade, ou a ausência da mesma. Isto pode ser aplicado em cenários onde os componentes da simulação representam componentes eletrónicos, bem como, para níveis mais altos de abstração. Desta forma, cada elemento da simulação é modelado recorrendo a uma máquina de estados finita (FSM) com dois estados, *operational* ou *failure*. Cada um destes estados representa a disponibilidade do componente. A máquina de estados conceptualizada pode ser encontrada na figura 3.9.

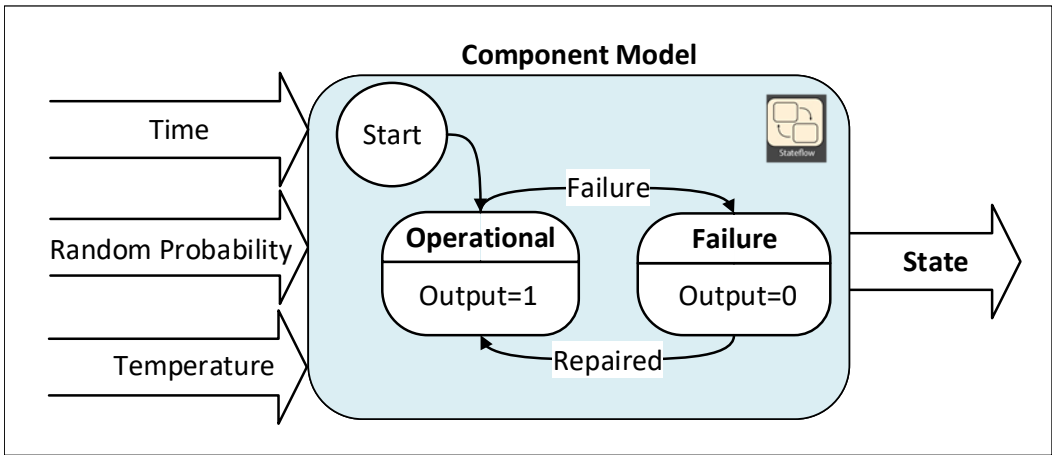


Figura 3.9: Máquina de estados de um componente.

Em cada iteração da simulação, o estado de cada elemento é recalculado utilizando os 3 *inputs* globais da simulação, sendo estes: temperatura, tempo e números aleatórios. O fluxograma da figura 3.10 detalha o procedimento do *operational state*. O modelo de Arrhenius é utilizado para calcular o fator de *stress* exercido pela temperatura. O tempo é utilizado para ler o correto valor da função PDF, sendo que a este valor é aplicado o fator de *stress* exercido pela temperatura, obtendo assim a probabilidade de falha. Esta probabilidade de falha é então comparada com o *input* aleatório. Caso a probabilidade de falha seja menor que o número aleatório, o elemento mantém-se no estado *Operational*, caso contrário, existe uma transição para o estado de *Failure*. Desta forma, baixos valores de probabilidade de falha significam menor ocorrência

de falhas e vice-versa. No caso de na simulação ser contemplada a manutenção do sistema, existe ainda a hipótese de o elemento voltar a transitar para o estado *operational*. Nesta caso, a transição acontece no tempo de manutenção, definido pelo utilizador.

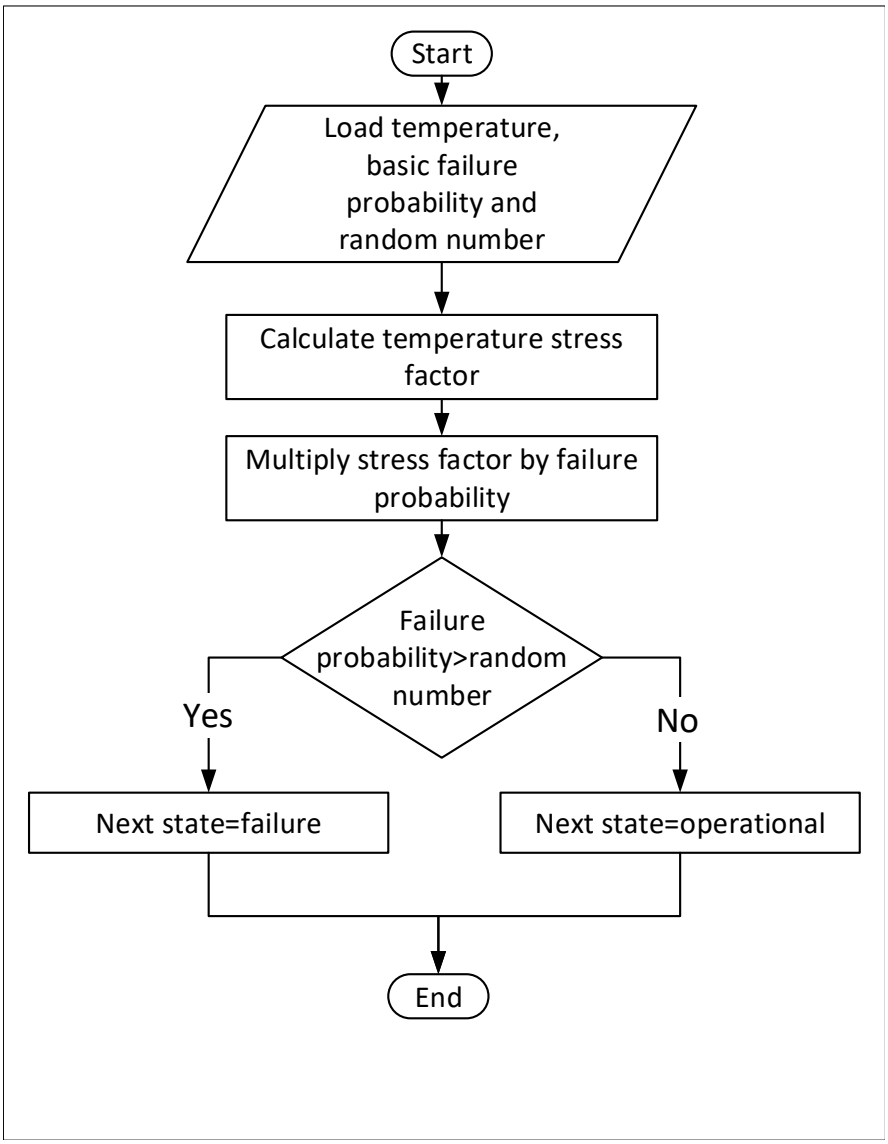


Figura 3.10: Fluxograma do estado *operational*.

A figura 3.11 representa como blocos de lógica combinacional são utilizados para interligar os vários elementos da simulação. Através da sua utilização, é possível modelar as interdependências dos componentes dentro de uma arquitetura. Esta técnica é também utilizada para formar subsistemas, permitindo assim, uma abordagem modular com vários níveis de abstração. Uma abordagem modular permite também a reutilização de módulos para outras simulações.

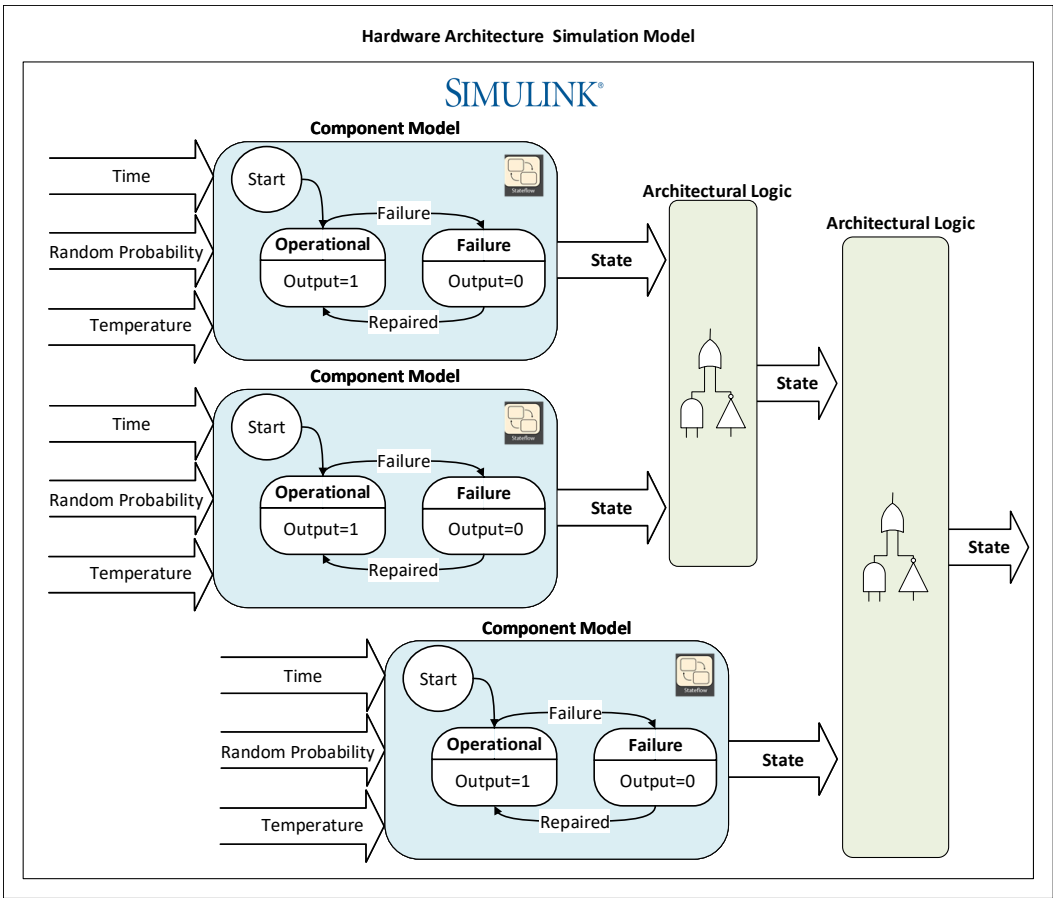


Figura 3.11: *Overview* de um módulo.

3.2 Setup para Testes Acelerados

A proposta de simulação anterior permite estimar a função PDF do sistema, auxiliando assim, a comparação entre as possíveis arquiteturas. No entanto, os modelos desenvolvidos apresentam um conjunto de pressupostos que reduzem a exatidão da estimação, nomeadamente, estando relacionadas com as abstrações utilizadas na modelação do comportamento dos componentes e das suas conexões. Na modelação dos componentes foi considerado um comportamento binário, sendo que apenas foi considerada a sua disponibilidade de funcionamento. Apesar de um componente estar operacional a sua funcionalidade pode estar degradada, podendo assim sobrecarregar os componentes adjacentes. Além disto, não foram considerados os desgastes nem as falhas das ligações elétricas dos componentes. De forma a poder refinar a estimativa da função PDF, são introduzidos testes de vida acelerada para complementar o ciclo de desenvolvimento utilizado.

3.2.1 Overview do Setup para Testes Acelerados

Como previamente apresentado no capítulo 2.2.7, testes de vida acelerada recorrem a fatores de aceleração, como temperatura ou vibração, para acelerar o envelhecimento dos componentes ou sistemas. Este envelhecimento acelerado permite detetar falhas, que possivelmente, só apareceriam após um longo período de funcionamento. Para provocar o envelhecimento é necessário que a intensidade dos fatores aplicados seja superior aos exercidos durante a operação normal do sistema. Contudo, esta intensidade não deve ser superior aos limites para os quais o mesmo foi desenhado. Caso a intensidade dos fatores aplicados seja superior aos limites do sistema, podem ser introduzidos mecanismos de falha que nunca seriam sentidos durante o funcionamento normal do sistema. Por esta razão, é necessário controlar a intensidade dos fatores aplicados durante a execução dos testes.

No âmbito desta dissertação foi desenvolvido um *setup* para testes acelerados, cujo *overview* é apresentado na figura 3.12. A interface gráfica permite ao utilizador monitorizar e controlar a temperatura da câmara climática, bem como gerir e obter *feedback* dos sinais gerados pelo módulo do NI-cDAQ-9178. O NI-cDAQ-9178 ([47]) é um chassi que permite incorporar o gerador de sinais NI-9269 ([48]), o qual permite gerar até quatro sinais independentes, e o módulo de aquisição de sinais NI-9229 ([49]), o qual permite adquirir até quatro sinais independentes. O módulo NI-9269 é utilizado pela interface gráfica para gerar um conjunto de sinais independentes, que são posteriormente partilhados pelos protótipos em teste, através de um DUT Interface. O NI-9229 é utilizado pela interface gráfica para recolher *feedback* dos sinais gerados pelo módulo NI-9269. Para monitorização dos protótipos é utilizado um computador pessoal (PC), o qual recebe mensagens CAN através da interface PCAN-USB Pro ([50]) para armazenamento e posterior processamento de dados.

3.2.2 DUT Interface PCB

Esta interface permite que os sinais gerados pelo NI-9269 sejam partilhados por 10 protótipos. Para preservar a integridade dos sinais é necessário que as impedâncias de entrada e de saída, sejam isoladas. Além disto, para que um curto-circuito num dos protótipos não comprometa a geração de sinais, é necessário isolar eletricamente os sinais. A solução mais coerente para ambos estes problemas é a utilização de isolamento galvânico, contudo, optou-se por uma

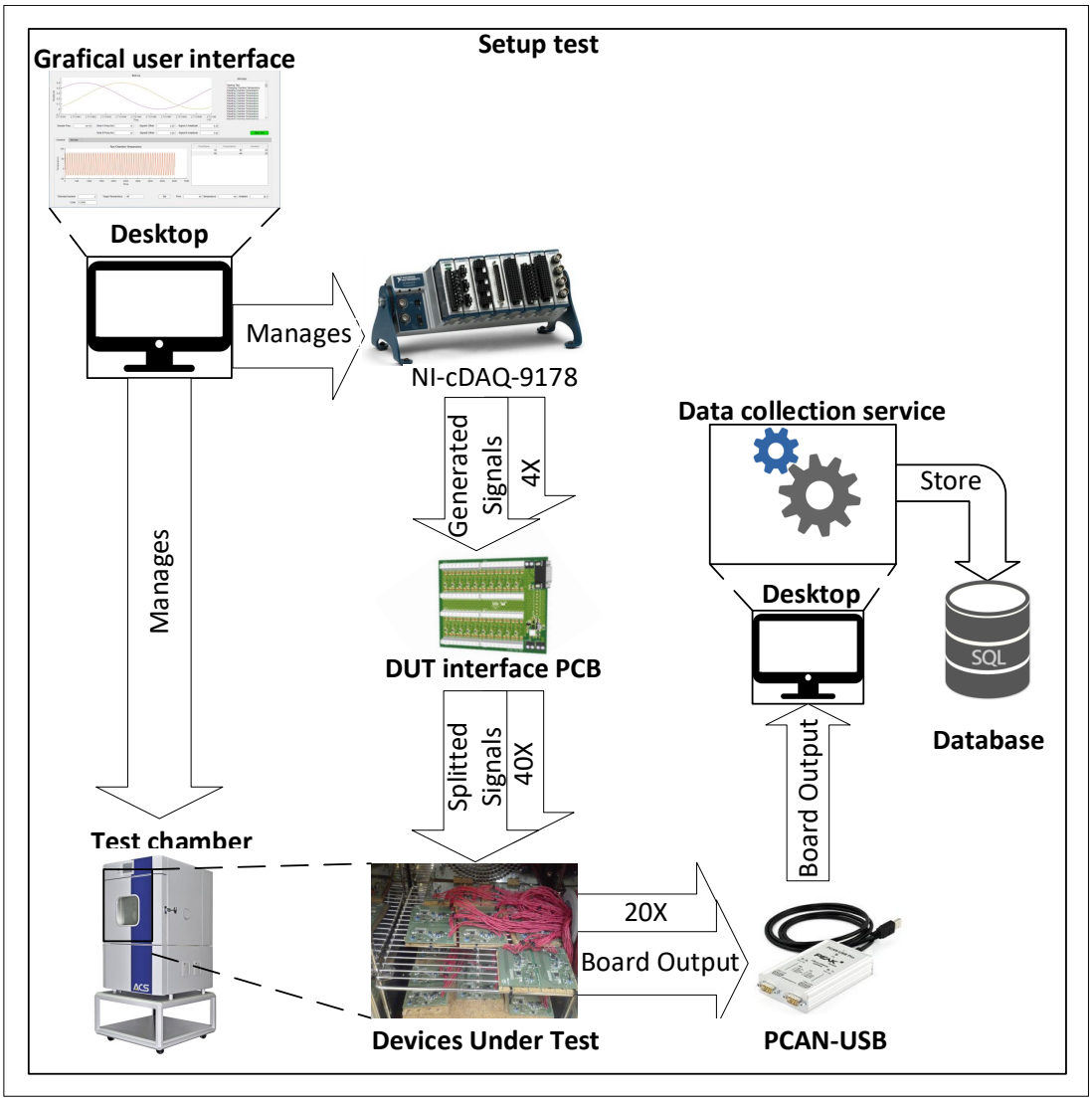


Figura 3.12: *Overview* do *setup* para testes acelerados.

alternativa mais económica. A utilização de amplificadores operacionais para isolamento de impedâncias revelou ser uma solução mais barata, porém, não permite o isolamento elétrico. A tabela 3.1 demonstra a relação de custos entre estes integrados. No anexo A (A.1) podem ser visualizados os esquemáticos e *layouts* desta PCB.

A *interface* PCB faz *signal split* dos 4 sinais de entrada de 1 para 10, resultando em 40 sinais de saída. Para reduzir o ruído foram utilizados sinais diferenciais. É possível fazer um *daisy chain* destas *boards* para alcançar um número de saídas superior, tirando proveito do isolamento de impedâncias providenciado pelas mesmas. A figura 3.13 ilustra as ligações a efetuar para realizar *daisy chain* com 2 *DUT interface boards*.

Componentes	MCP6004T	TSV734IPT	TS1874AI	78601/2C	ADUM3190ARQZ
Device	OP Amp	OP Amp	OP Amp	Pulse Transformer	Isolated Amplifier
Alimentação(V)	1.8-6	1.5-5.5	1.8-6	-	-
Preço (euros)	0.311 [51]	0.872 [51]	1.06 [51]	1.08 [51]	2.33 [51]
Outras caraterísticas	Rail-to-Rail	Rail-to-Rail	Rail-to-Rail	-	-
Número de OPAMPs	4	4	4	1	1

Tabela 3.1: Comparação entre componentes.

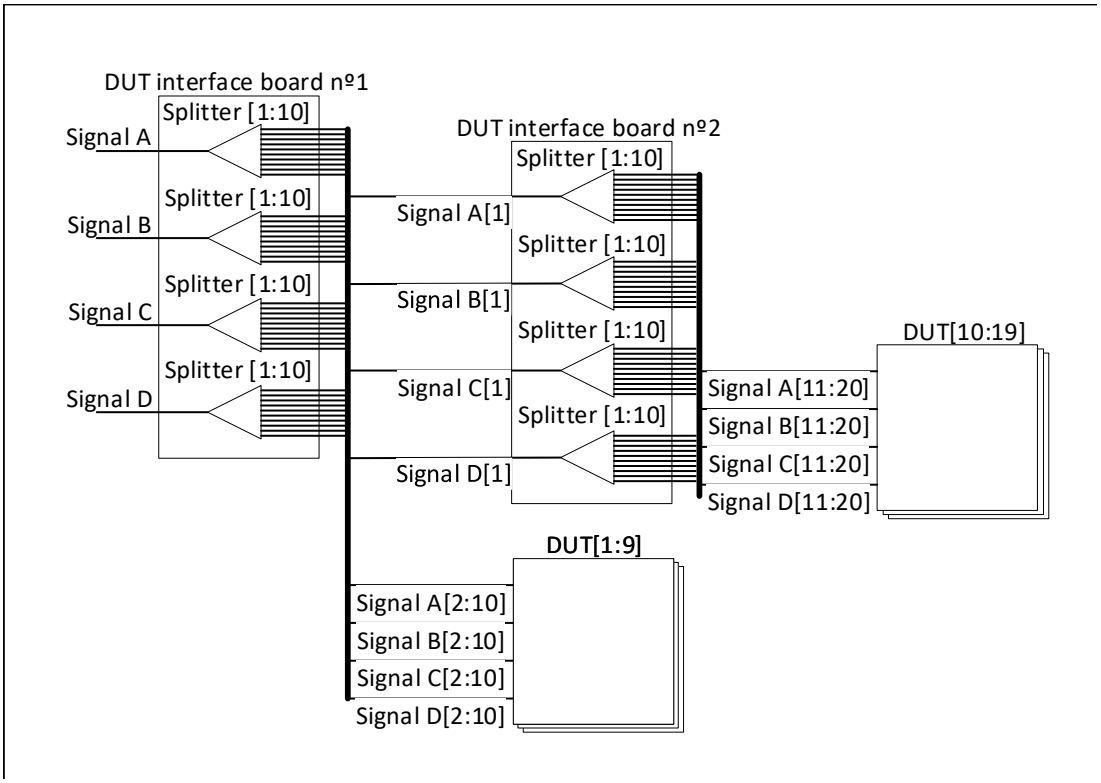


Figura 3.13: *Daisy chaining* com 2 *boards*.

3.2.3 Serviços de Software

A software *stack* da figura 3.14 detalha as diversas interfaces de programação de aplicações (interface de programação de aplicações (API)) e *drivers* utilizadas por cada uma das aplicações desenvolvidas.

O *data collection service* é uma aplicação que executa em *background*. Para cada interface PCAN este serviço cria uma nova conexão. Por sua vez, cada conexão é responsável por ler as mensagens CAN, processá-las e guardá-las numa base de dados.

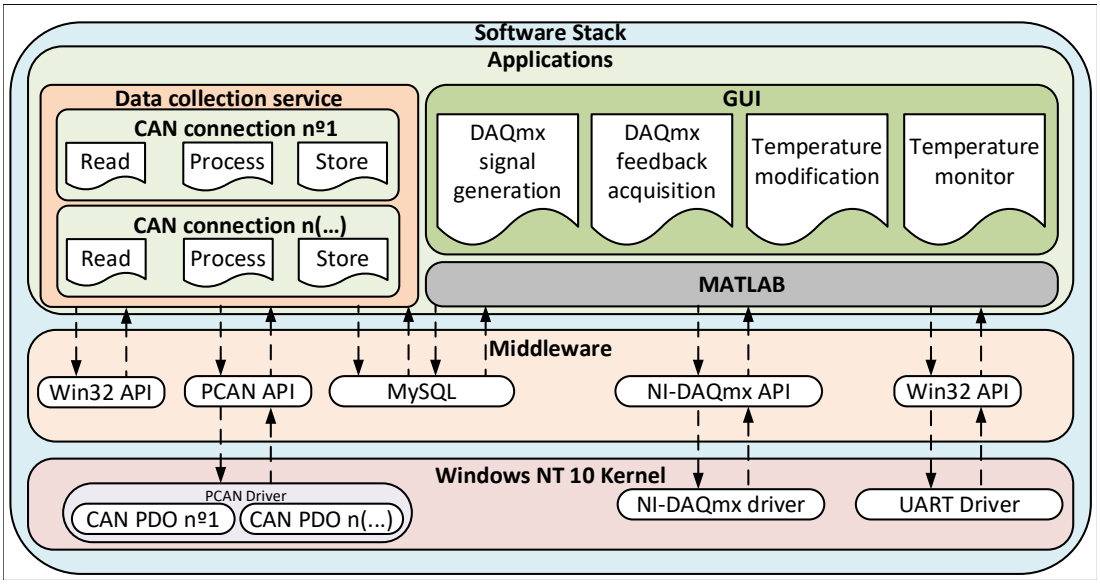


Figura 3.14: Software *stack* do *setup* para testes acelerados.

A aplicação gráfica permite ao utilizador controlar e gerir tanto a câmara climática como os módulos do NI-cDAQ-9178. Esta aplicação foi desenvolvida para ambiente Matlab, o qual possuí uma camada de abstração que fornece uma versão simplificada das APIs que permitem gerir estes equipamentos. Por sua vez, é esta camada de abstração que interage com os *device drivers* dos equipamentos, através das APIs fornecidas pelos fabricantes.

Data Collection Service

Este serviço interage com os dispositivos PCAN e com uma base de dados. O PCAN é um adaptador que permite a aquisição de mensagens CAN em tempo real pelo PC. Cada adaptador PCAN possui uma ou mais interfaces de hardware. A figura 3.15 representa as interações deste serviço com cada interface PCAN.

Cada interface CAN conectada ao PC está associada a um *physical device object* (PDO), o qual permite que o *data collection service* efetue leituras na interface associada. Cada conexão é responsável por realizar tarefas de leitura, processamento e armazenamento dos dados extraídos de cada mensagem adquirida pela interface associada. Desta forma, o número de protótipos que este serviço pode atender torna-se escalável, atendendo ao número de interfaces físicas disponíveis e às limitações físicas do barramento CAN. A figura 3.16 detalha o procedimento da tarefa responsável por avaliar a disponibilidade de novas interfaces. Esta tarefa também tem

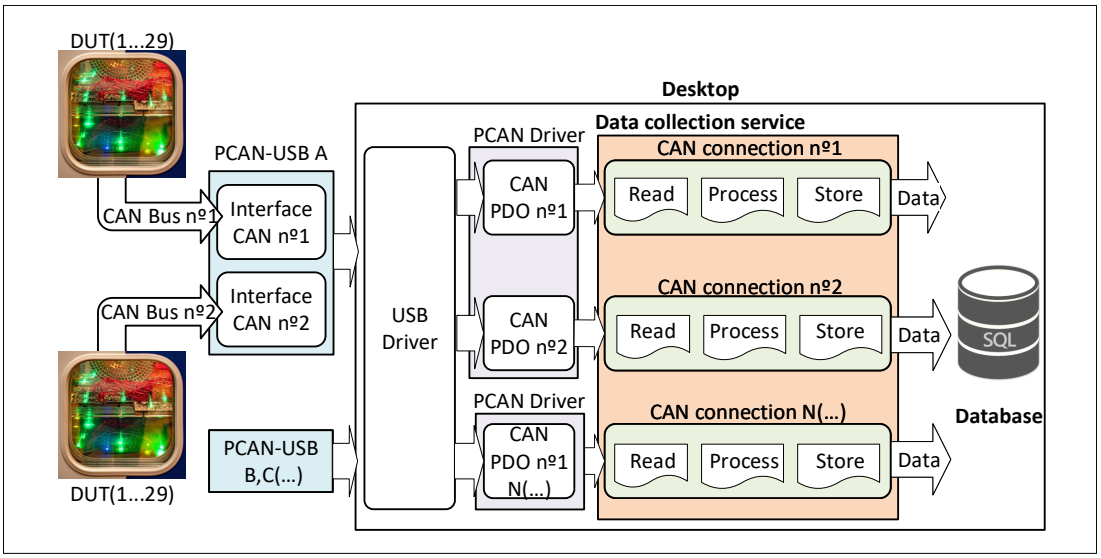


Figura 3.15: *Overview* do *data collection service*.

como responsabilidade avaliar o estado das interfaces conectadas, reiniciando-as em caso de erro.

A divisão de tarefas dentro de cada conexão utiliza o mecanismo de *threads*. A figura 3.17 ilustra as interações entre as *threads* de cada conexão.

Sempre que uma nova mensagem chega ao PCAN, esta é etiquetada pelo hardware com o tempo de chegada, obtendo-se assim tempo real na amostragem das mensagens. Aquando a receção de uma nova mensagem, o PCAN *driver* alerta a conexão correspondente, sendo a mensagem lida pela *read thread*. Caso não exista nenhum erro de leitura, a mensagem é inserida numa *queue*, a qual será posteriormente recolhida pela *process thread*. Caso seja detetado um erro no hardware, a conexão entrará em estado de erro. A *process thread* é responsável por interpretar as mensagens CAN, extraindo o ID da mensagem, a interface CAN que a recebeu, assim como os dados contidos na mesma. Com o intuito de diminuir a frequência de pedidos à base de dados, os dados extraídos de cada mensagem são armazenados numa *queue* e periodicamente descarregados, pela *store thread*, para uma base de dados. As figuras presentes no anexo A (A.2) detalham o procedimento de cada uma destas *threads*.

A figura A.13 presente em anexo mostra o diagrama de classes para o objeto *CANConnection* e para as suas estruturas de dados auxiliares. A classe *CANConnection* contém o procedimento usado em cada uma das *threads*, bem como, a identificação de cada uma delas, os mecanismos de sincronização utilizados e a identificação hardware associado à instância. Os restantes objetos são estruturas de dados auxiliares usadas na extração dos dados das mensagens.

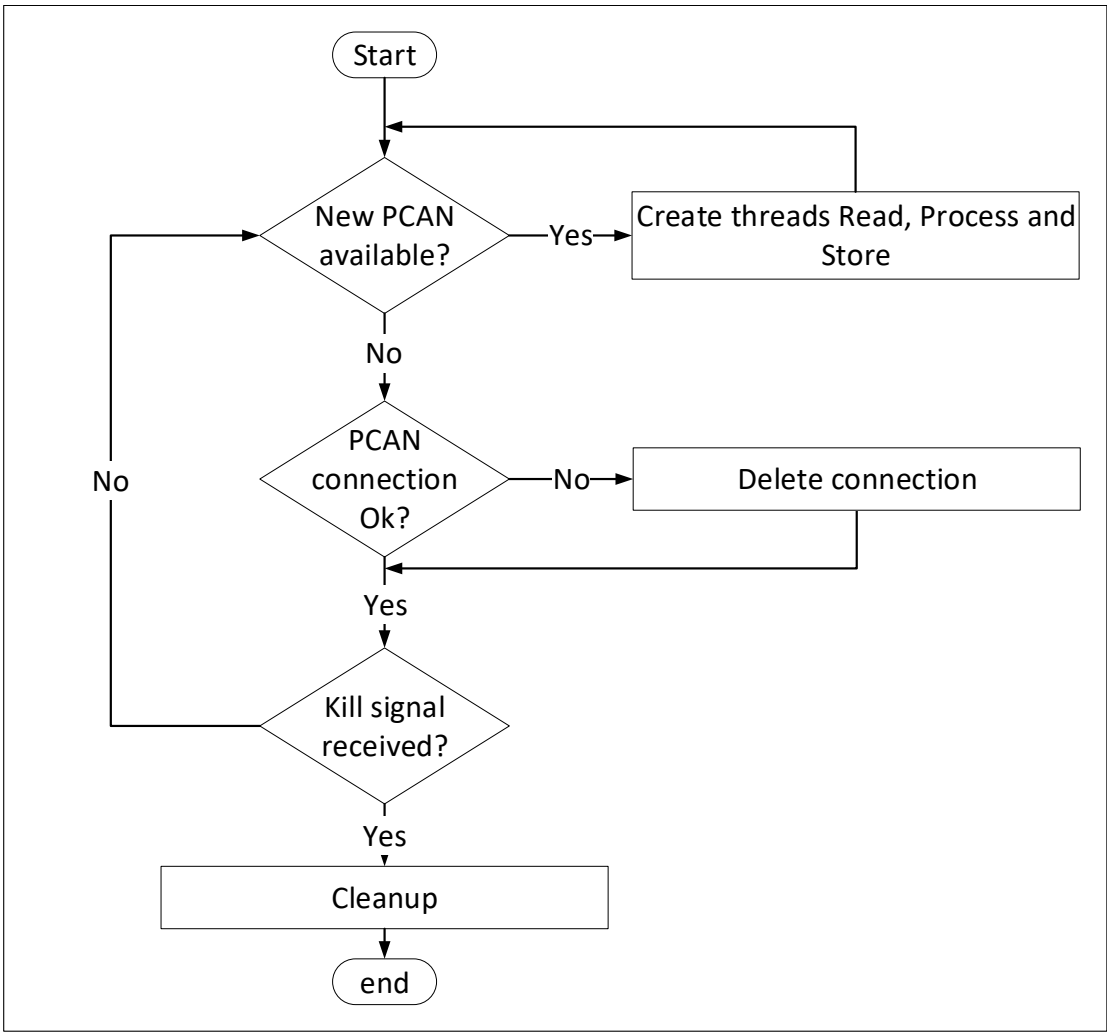


Figura 3.16: Fluxograma da *manage connection task*.

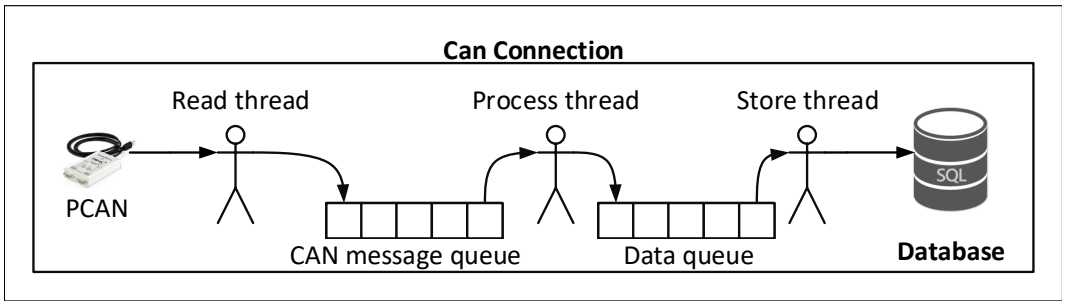


Figura 3.17: Interações entre as *threads* de cada conexão CAN.

Aplicação Gráfica

A aplicação gráfica tem como objetivos a configuração, monitorização e gestão dos equipamentos utilizados durante o teste. O *frontend* permite tanto a configuração dos sinais gerados pelo NI-9269, bem como, a configuração das temperaturas da câmara climática e suas durações.

A figura 3.18 representa o *frontend* da aplicação gráfica.

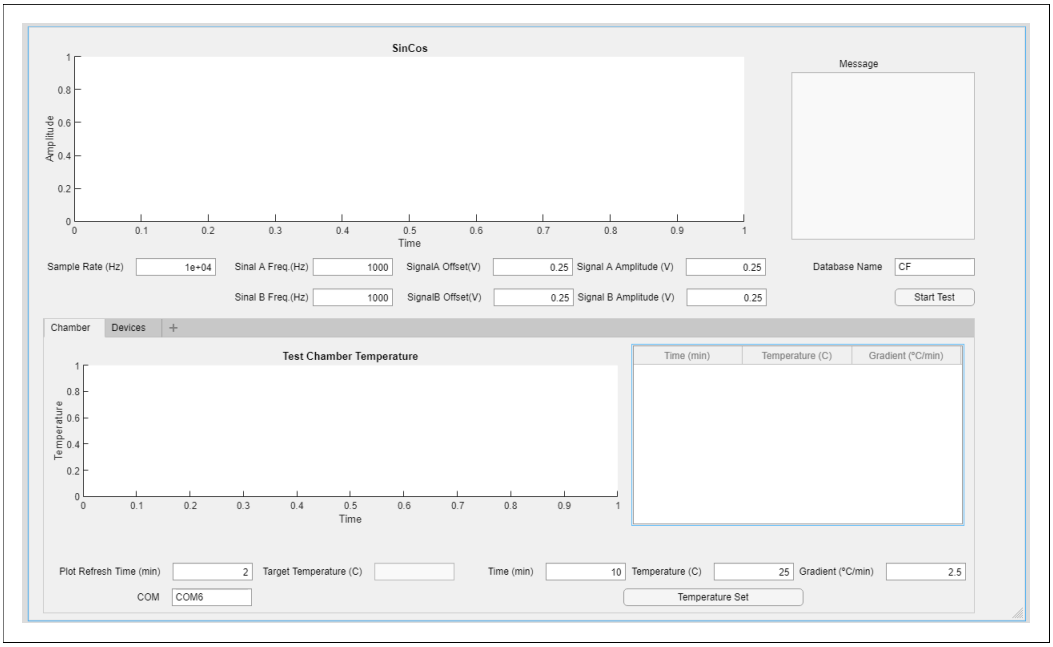


Figura 3.18: *Frontend* da aplicação gráfica.

A figura 3.19 apresenta a máquina de estados para o *backend* da aplicação. Durante a inicialização da aplicação gráfica são configurados os dispositivos a serem usados durante o teste, nomeadamente, a câmara climática e os módulos do NI-cDAQ-9178. Além destes, é também configurado o acesso à base de dados, o qual é usado para guardar as temperaturas da câmara climática ao longo do teste. Caso exista algum erro de inicialização, a aplicação entra em estado de erro, fazendo *display* do *feedback* do erro em causa. Caso não existam erros de inicialização, a aplicação gráfica avança para o estado de *idle* até que o botão de inicialização de teste seja premido. Na inicialização do teste, são configurados os períodos utilizados na atualização da temperatura e na sua leitura, bem como, lido e configurado o estado inicial da temperatura e configurados os sinais a serem gerados. Caso não exista nenhum erro de inicialização, é registado o *timestamp* de inicialização do teste na base de dados especificada pelo utilizador, sendo assim dado início ao teste. O *timestamp* desde a inicialização do sistema operativo é usado como referência temporal para marcar o início do teste na base de dados. O mesmo é também utilizado pelo PCAN para marcar a chegada de novas mensagens. A utilização de uma referência em comum permite sincronizar a amostragem realizada pela aplicação gráfica e pelo *data collection service*. Em caso de ser detetado um erro de inicialização, ou durante o teste, a

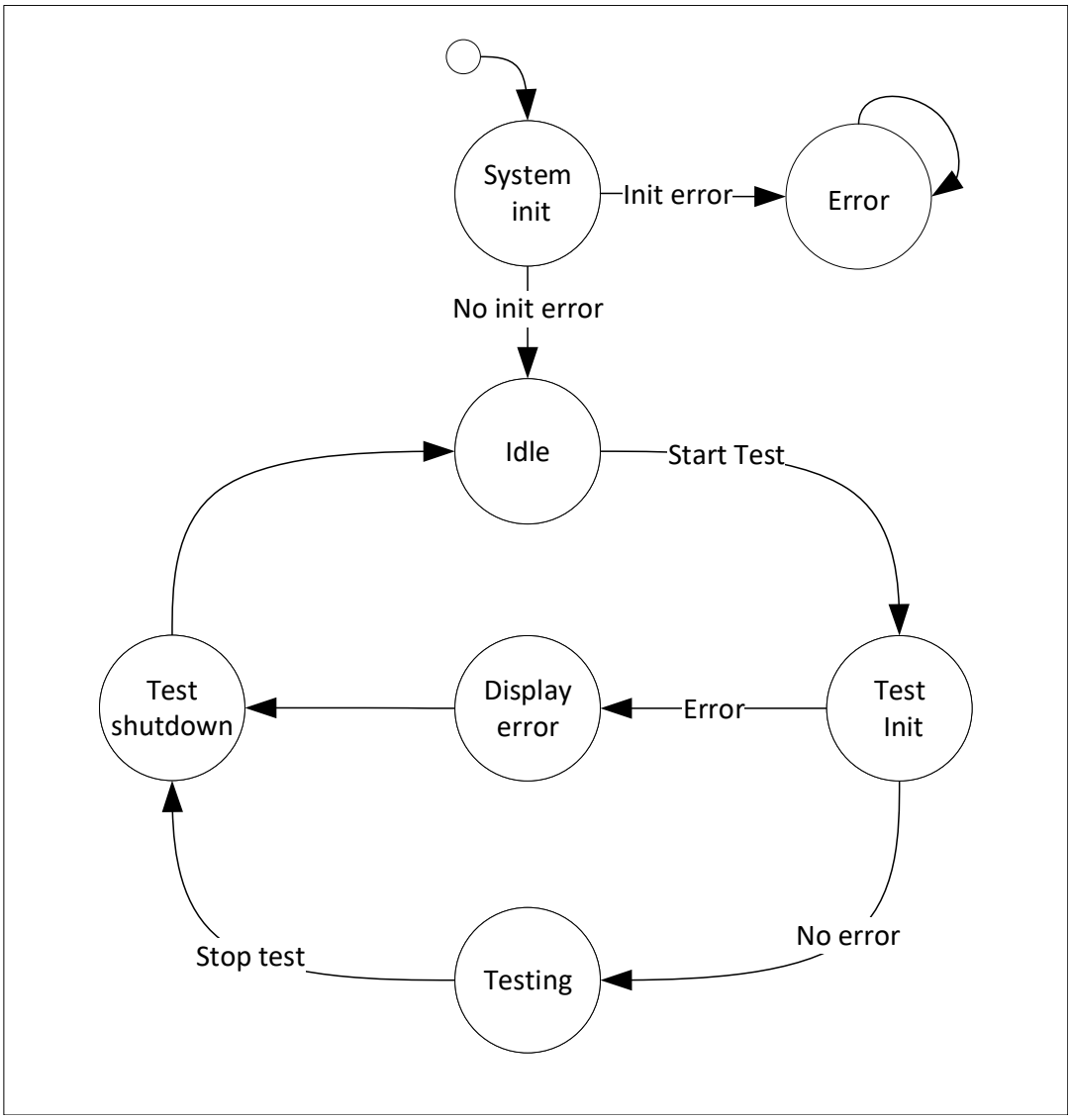


Figura 3.19: Máquina de estados do *backend*.

aplicação gráfica aborta o teste, parando assim a geração de sinais e a câmara climática. Além disso, esta faz *display* da mensagem de erro, voltando depois ao estado de *idle*.

Durante a realização do teste, a aplicação gráfica tem como tarefas a gestão da câmara climática e dos módulos de geração e aquisição de sinais. O fluxograma da figura 3.20 demonstra a gestão destas tarefas no *backend* da aplicação gráfica.

Durante o teste, o software atende a quatro possíveis eventos. O primeiro acontece quando são requisitados mais dados pelo módulo de geração de sinais. O segundo acontece quando o módulo de aquisição de sinais requisita a leitura dos dados adquiridos. O terceiro acontece quando termina o *timeout* de leitura da temperatura da câmara climática. O quarto acontece quando termina o *timeout* de atualização da temperatura da câmara climática. Caso o botão de

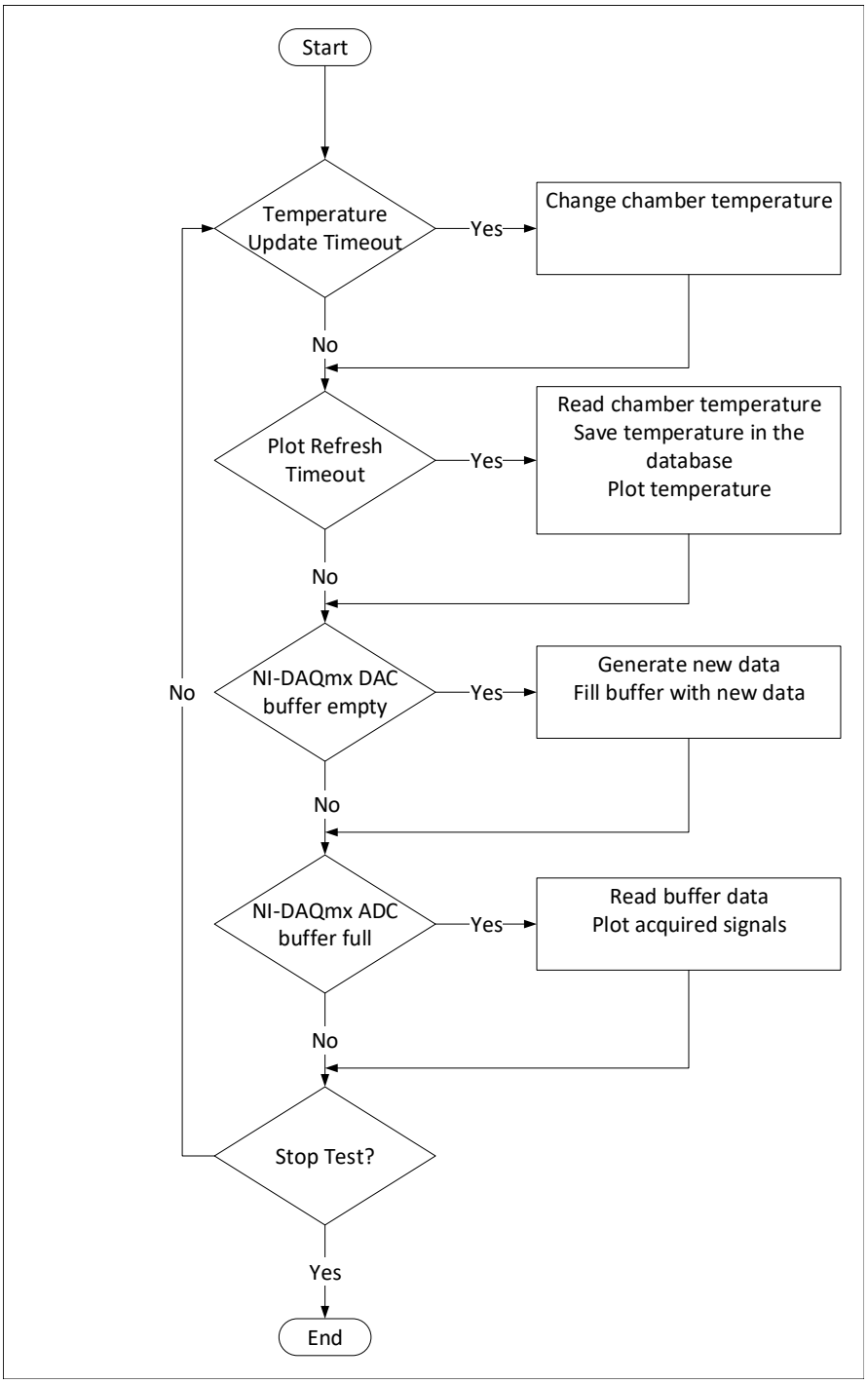


Figura 3.20: Fluxograma do estado *testing*.

stop test seja premido, o teste é encerrado e a aplicação volta para o estado de *idle*. Durante o encerramento do teste, a aplicação guarda o *timestamp* de encerramento do teste na base de dados, termina a geração de sinais, e desativa a câmara climática. No anexo A (A.3) é possível encontrar os fluxogramas para os estados de *error*, *aborting test*, *test init* e *test shutdown*.

3.3 Sumário

Neste capítulo, foram explorados dois métodos que auxiliam o desenvolvimento de hardware de um sistema resiliente: simulações de resiliência e testes acelerados.

As simulações de resiliência foram exploradas como método para auxiliar a comparação de arquiteturas e para estimar a resiliência do desenho de um sistema. Neste sentido, no capítulo 3 (3.1) foram apresentados os princípios utilizados no desenvolvimento de um modelo de resiliência.

Os testes acelerados foram explorados como método para avaliar a resiliência de protótipos de um sistema, complementando assim as estimativas realizadas pelas simulações. Neste sentido, no capítulo 3 (3.2) foi apresentado o desenvolvimento de um setup que pretende auxiliar a execução de testes acelerados a um sistema automóvel. Este *setup* permite a realização de testes acelerados a uma amostra de N sistemas, com até quatro entradas analógicas e com output CAN. O desenvolvimento deste *setup* foi realizado, tendo já em vista, a validação de um caso de estudo no contexto do setor automóvel, onde é comum a utilização do protocolo CAN.

De forma a validar o trabalho realizado ao longo deste capítulo, no capítulo 4 os modelos de resiliência e o setup para testes são utilizados para auxiliar e validar o desenvolvimento de um sensor automóvel.

Capítulo 4

Caso de Estudo

De forma a estimular o desenvolvimento das abordagens sugeridas no capítulo 3, foi utilizado um caso de estudo no contexto de sistemas críticos. O SbW, seguindo as tendências de mercado, visa substituir a direção mecânica de um veículo por um sistema elétrico/ eletrónico. Dentro do sector automóvel, o SbW foi classificado, com o maior nível de risco de lesão (ASIL D), o que significa que a falha deste sistema pode comprometer a vida dos ocupantes do veículo, ou dos que o rodeiam. O SAS é o subsistema do SbW responsável por adquirir e processar o ângulo de direção. Tendo este sido adotado como o caso de estudo no contexto desta dissertação.

O SAS é um projeto que resulta de uma parceria entre a Bosch e a Universidade do Minho. Este projeto conta com múltiplas fases, nesta primeira fase, a Bosch foi responsável por enunciar os requisitos e o conceito inicial, e a Universidade do Minho por auxiliar nas fases de desenho e implementação do sistema. No contexto desta dissertação, é apresentado o desenho e implementação do hardware da primeira fase do SAS, o qual foi desenvolvido utilizando as abordagens descritas no capítulo 3.

4.1 Steering Angle Sensor

O SAS é um sensor automóvel responsável por adquirir e calcular o ângulo de direção. Este sistema é constituído por partes mecânicas e eletrónicas, sendo colocado na coluna de direção para efetuar as medições da posição do volante.

Na solução atual, as partes mecânicas deste sensor possuem 3 rodas dentadas, a maior, o *HUB*, e as duas mais pequenas, as *gears*, figura 4.1. O *HUB* transmite a rotação do volante para as duas *gears*. Cada uma das *gears* possui um íman permanente, o qual possibilita a leitura do seu ângulo, através de um sensor magnético fixado por baixo de cada uma delas.

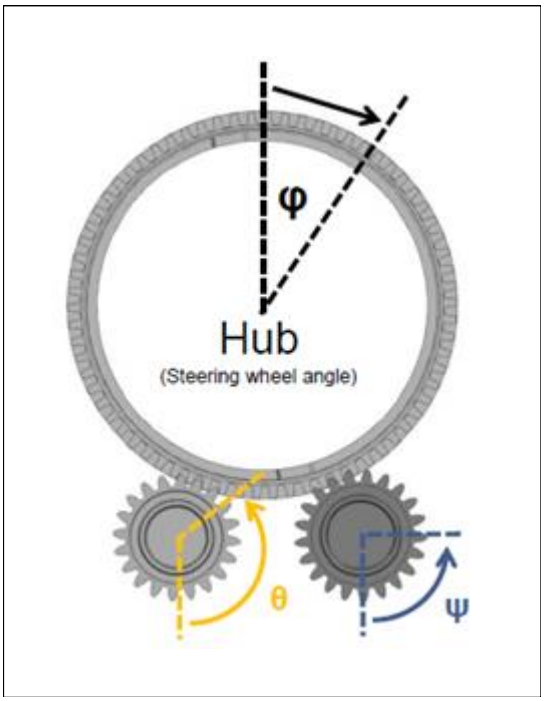


Figura 4.1: Transmissão do movimento do volante (HUB + Gears).

As duas *gears* apresentam um número de dentes diferente, o que leva a que as mesmas girem a uma velocidade diferente. Devido ao desfasamento entre *gears*, existe uma diferença entre os ângulos de cada uma delas. A diferença entre os ângulos das *gears* permite para cada instante, o cálculo do ângulo absoluto do volante, figura 4.2.

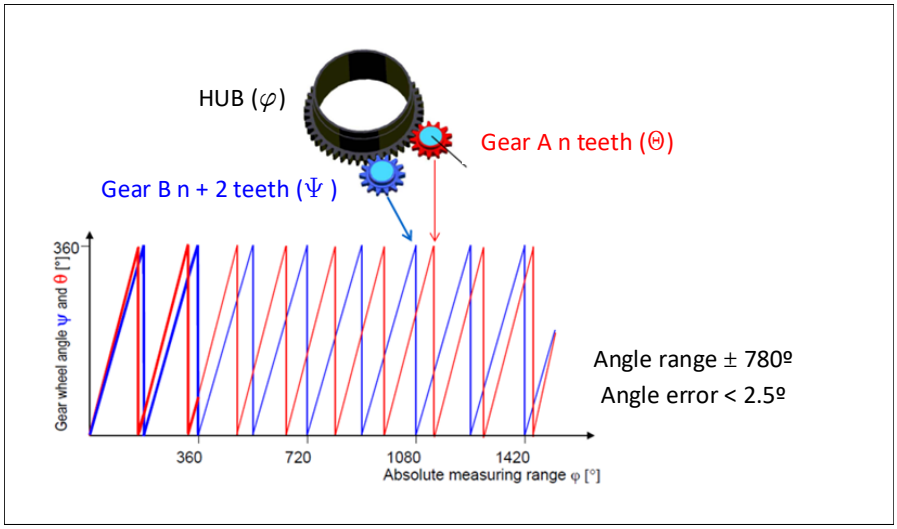


Figura 4.2: Relação entre as *gears* e o ângulo absoluto.

As partes eletrónicas do sensor são o foco desta dissertação. Estas necessitam de ser constituídas pelos sensores de posição angular, que interpretem o campo magnético gerado

pelas *gears*, e por um sistema que permita adquirir o ângulo produzido por estes sensores e calcular o ângulo absoluto do volante.

Como anteriormente referido a falha do SAS pode ter consequências severas. No contexto deste projeto, isto significa que o uso de redundância deve ser explorado de forma a torná-lo *fail-operational*, ou seja, que o mesmo seja capaz de cumprir a sua funcionalidade apesar da falha de um componente ou módulo.

Os requisitos e restrições deste sistema foram enunciados pela Bosch. Relativamente aos requisitos funcionais do SAS, este necessita adquirir o ângulo de cada uma das *gears*, calcular o ângulo absoluto do volante e transmiti-lo via CAN a cada 10 ms. A não transmissão do ângulo ou a não transmissão atempadamente, constituem a falha do sistema. Quanto aos requisitos não funcionais, pelo sistema se encontrar na coluna de direção, o mesmo necessita de operar na gama de temperaturas entre -40°C e 85°C(*automotive grade 3*). Além disto, o mesmo necessita de manter a sua operação por 800 horas/ano durante 12 anos, resultando em 12000 horas de operação. Quanto às restrições do sistema, nesta primeira fase, a Bosch determinou que apenas deve ser utilizada redundância homogénea (capítulo 2 (2.2.4)) para aumentar a resiliência do sistema.

4.1.1 Conceito para o SAS

O primeiro conceito proposto pela Bosch, visa explorar o uso de redundância homogénea. Este conceito apresenta dois módulos redundantes, os quais possuem alimentações diferentes, e as suas saídas estão conectadas ao mesmo barramento CAN. Estes módulos estão isolados eletricamente, contudo, o uso de mecanismos de isolamento galvânico entre eles permite a comunicação. De forma a garantir o isolamento elétrico, cada um dos módulos recebe a informação de dois sensores angulares diferentes. A imagem da figura 4.3 ilustra o conceito idealizado para este sistema.

Cada um dos módulos transmite, alternadamente, o ângulo a cada 20 ms, com um *delay* de 10 ms entre as transmissões dos módulos redundantes. Desta forma é possível cumprir o requisito temporal do sistema, apesar, de cada um dos módulos, apenas transmitir o ângulo a cada 20 ms, figura 4.4.

Sempre que um dos módulos transmite o ângulo de direção, o mesmo também informa o módulo adjacente da sua transmissão, o qual se encontra à escuta. Desta forma, a falha

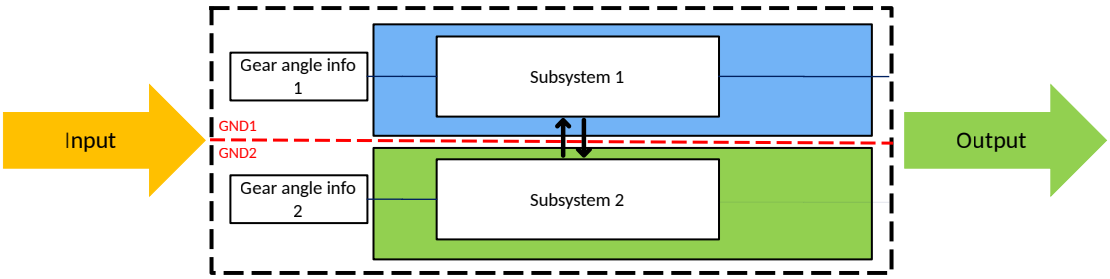


Figura 4.3: Diagrama de blocos do conceito do SAS.

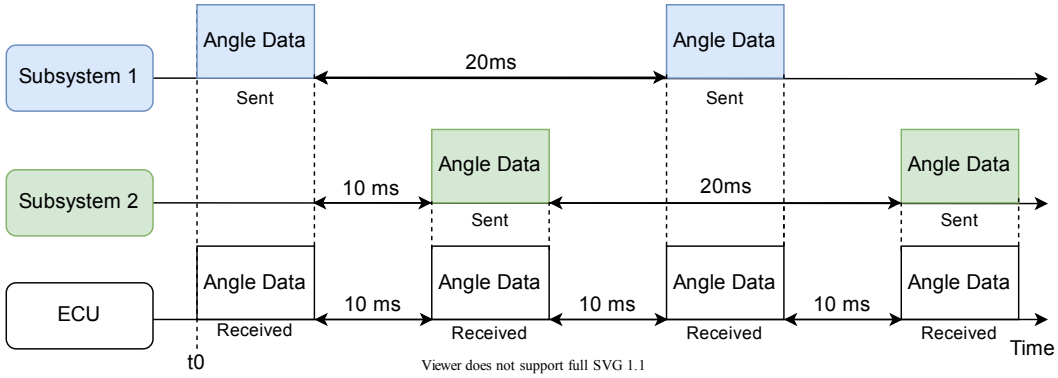


Figura 4.4: Diagrama temporal do SAS.

de um dos módulos redundantes é detetada pelo adjacente. Caso um dos módulos falhe, o adjacente é reconfigurado entrando no estado *fail-degraded*. Quando um dos módulos entra em *fail-degraded*, o intervalo de tempo entre transmissões de ângulos de direção é reconfigurado para 10 ms, de tal forma que este módulo compense a falha do adjacente, figura 4.5.

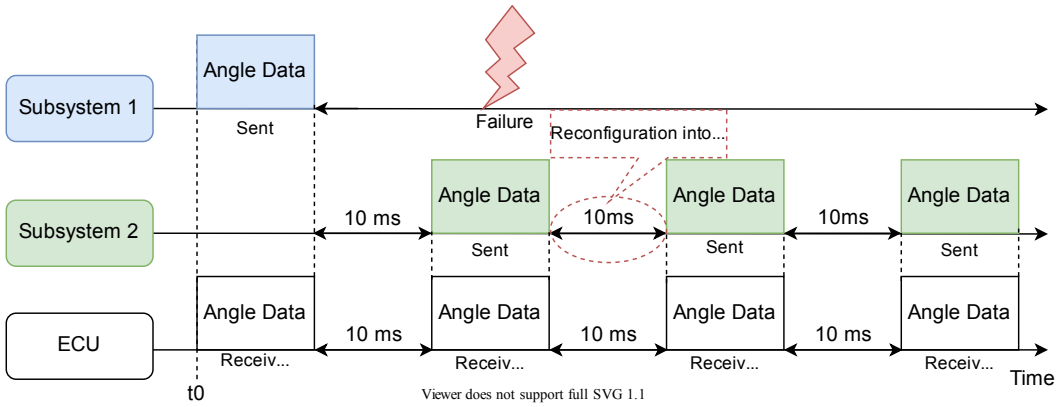


Figura 4.5: Diagrama temporal do SAS após a falha de um dos módulos.

4.1.2 Arquitetura do Sistema

Após a validação da aplicação com recurso a uma modelação *software-only*, iniciou-se a modelação da arquitetura do sistema. A arquitetura inicial deste sistema foi proposta pela Bosch, porém, foram sugeridas alterações por parte da Universidade do Minho.

Inicialmente, a arquitetura apresentada pela Bosch propunha uma configuração *master-slave* com duas linhas de *feedback* entre os módulos redundantes. A configuração *master-slave* permitiria ao lado A, através de lógica combinacional, desativar o *transceiver* do lado B. Esta configuração proporcionaria que uma falta do lado A comprometesse o sucesso do lado B, representando assim um possível *single-point-failure*. As linhas de feedback permitiriam a cada um dos módulos obter *feedback* do estado do *transceiver* CAN do módulo adjacente.

A equipa da Universidade do Minho, sugeriu uma alternativa à configuração *master-slave*, a qual consistiu em substituir as duas linhas de *feedback*, por duas linhas conectadas diretamente aos microcontroladores de cada um dos módulos. Desta forma, estas linhas permitiam a sincronização entre módulos e, ainda, obter o *feedback* da transmissão do módulo adjacente. Além disto, estas linhas foram conectadas às interfaces *universal asynchronous receiver transmitter* (UART) de cada microcontrolador, proporcionado futuramente, a utilização do protocolo UART para comunicação entre os módulos redundantes, sem ser necessário modificar o hardware.

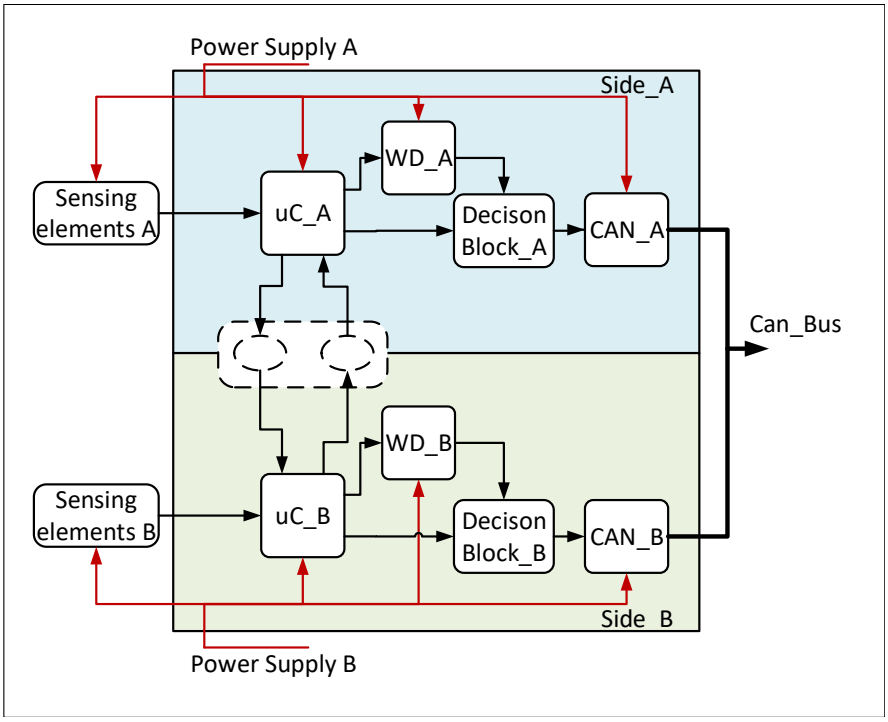


Figura 4.6: Arquitetura do sistema.

A figura 4.6, representa a arquitetura resultante das alterações propostas pela Universidade do Minho. Cada um dos módulos redundantes possui então dois sensores de posição angular, um microcontrolador, um *watchdog* externo, um bloco de decisão e um *transceiver* CAN. Entre os módulos redundantes existe um isolador, o qual permite a comunicação entre os módulos. Os sensores angulares, o microcontrolar e o *transceiver* CAN são utilizados na aquisição, cálculo e transmissão do ângulo do volante. A redundância de *watchdogs*, interno e externo, é utilizada com o intuito de prevenir uma falta de modo comum. Além disto, um bloco de decisão permite ao *watchdog* externo ou ao microcontrolador desativar o *transceiver* CAN. A possibilidade de o *watchdog* externo poder desativar o *transceiver* CAN tem como propósito evitar a propagação de faltas para o exterior.

Conceptualmente o funcionamento da arquitetura foi modelado usando uma máquina de estados. Esta máquina de estados reflete o funcionamento de cada um dos módulos redundantes. Durante a explicação desta máquina de estados será seguido o fluxo de um dos módulos, podendo o módulo adjacente encontrar-se num estado diferente.

Na inicialização, o módulo encontra-se no *reset*, o qual pode ser visualizado na figura 4.7. Neste estado, o módulo inicializa o microcontrolador, habilita o *transceiver* CAN e analisa a possível existência de erros. No caso de ser detetado um erro de inicialização ou um *reset* despertado pelo *watchdog*, o mesmo avança para o estado de erro, ficando permanentemente neste estado. Caso o sistema tenha sido reconfigurado para *fail-degraded*, o módulo avança para a máquina de estados presente na figura 4.13. Caso contrário, o mesmo irá prosseguir a sua operação para o estado *synchronization*.

Como o barramento CAN é partilhado por ambos os módulos redundantes e ambos possuem o mesmo CAN ID, pode ocorrer uma *race condition*. Para evitar a sua ocorrência, após a inicialização do sistema é realizada a sincronização entre módulos, figura 4.8. Durante a sincronização, os números de série dos microcontroladores de ambos os módulos redundantes são comparados, sendo que o módulo com maior numero de série é quem toma a dianteira. Para possibilitar a comparação entre números de série, os mesmos são trocados entre os microcontroladores, uma única vez, durante o processo de calibração (em fim de linha de produção). Com o intuito de desfasar a operação dos módulos, aquele que ficou para trás aguarda que o módulo que tomou a dianteira, o informe da ocorrência da primeira transmissão do ângulo via CAN.

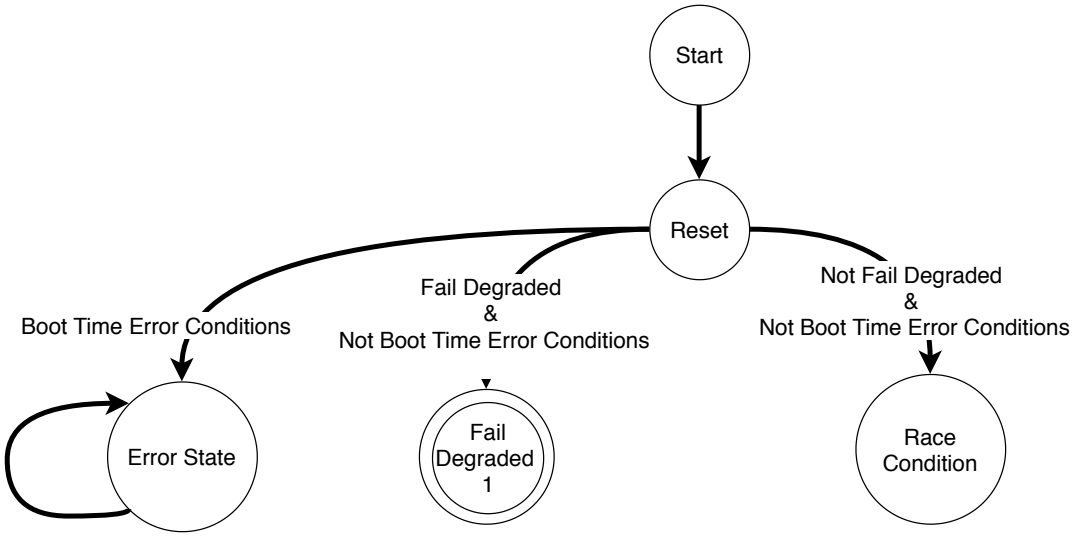


Figura 4.7: Máquina de estados (1).

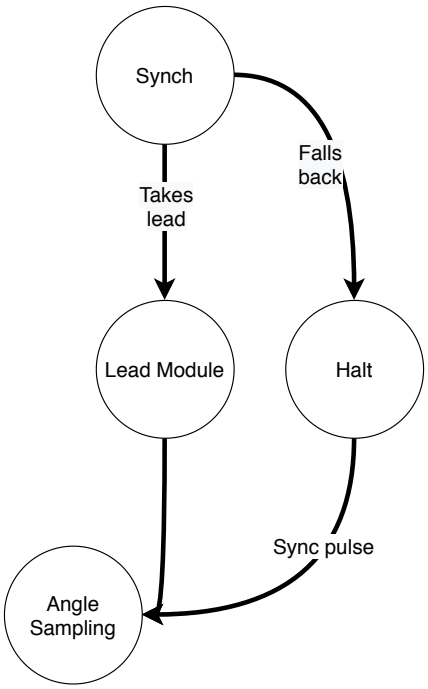


Figura 4.8: Máquina de estados (2).

Durante a primeira iteração, se anteriormente este módulo foi aquele que tomou a dianteira, este amostra e calcula o ângulo, transmitindo-o de seguida. Caso contrário este módulo aguarda um *timeout* de 10 ms, o qual foi inicializado na amostragem, figura 4.9. Desta forma, as execuções entre os módulos redundantes são desfasadas conforme a figura 4.4.

Após a primeira iteração, ambos os módulos inicializam, na amostragem, um *timeout* de 10 ms. Cada um dos módulo, após terminar a amostragem e cálculo do ângulo, aguarda o

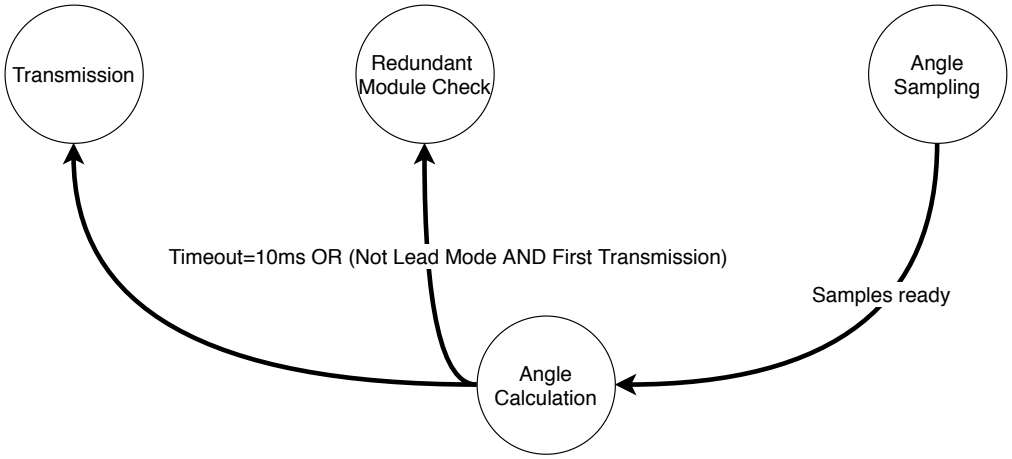


Figura 4.9: Máquina de estados (3).

timeout inicializado na amostragem. Terminado o *timeout*, o módulo verifica se, de facto, o módulo adjacente transmitiu o ângulo, figura 4.10. Caso o módulo adjacente tenha transmitido é inicializado um novo *timeout* de 10 ms, sendo inicializada a transmissão após este *timeout*. Caso o módulo adjacente falhe, o módulo reconfigura-se entrando em *fail-degraded*, figura 4.13.

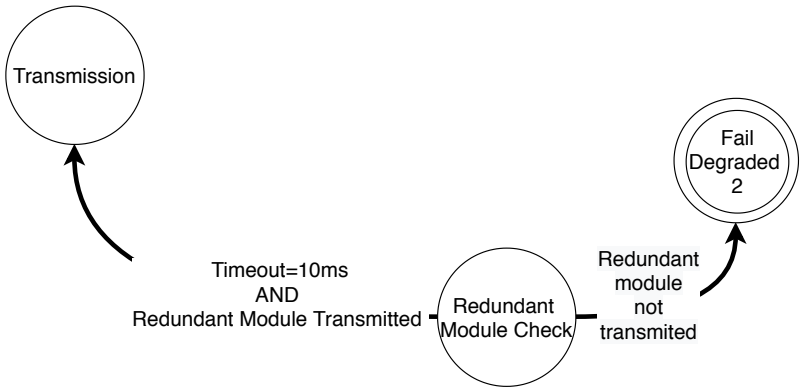


Figura 4.10: Máquina de estados (4).

Durante a normal operação de cada módulo é verificada a existência de possíveis erros no cálculo do ângulo e antes da transmissão. Esta verificação de erros tem como propósito, no primeiro cenário, verificar se existiu algum erro durante a última transmissão e no segundo cenário, verificar se o módulo adjacente está a usar o recurso partilhado. Na ocorrência de erros, ambas as situações representam condições anómalas, às quais, o módulo responde entrando permanentemente no estado de erro.

Quando um módulo entra em *fail-degraded*, este é reconfigurado, passando a transmitir o ângulo em intervalos de 10 ms, em vez de 20 ms, figura 4.5. Caso o módulo entre em *fail-degraded* proveniente do *reset*, este resume a sua operação começando pela amostragem do

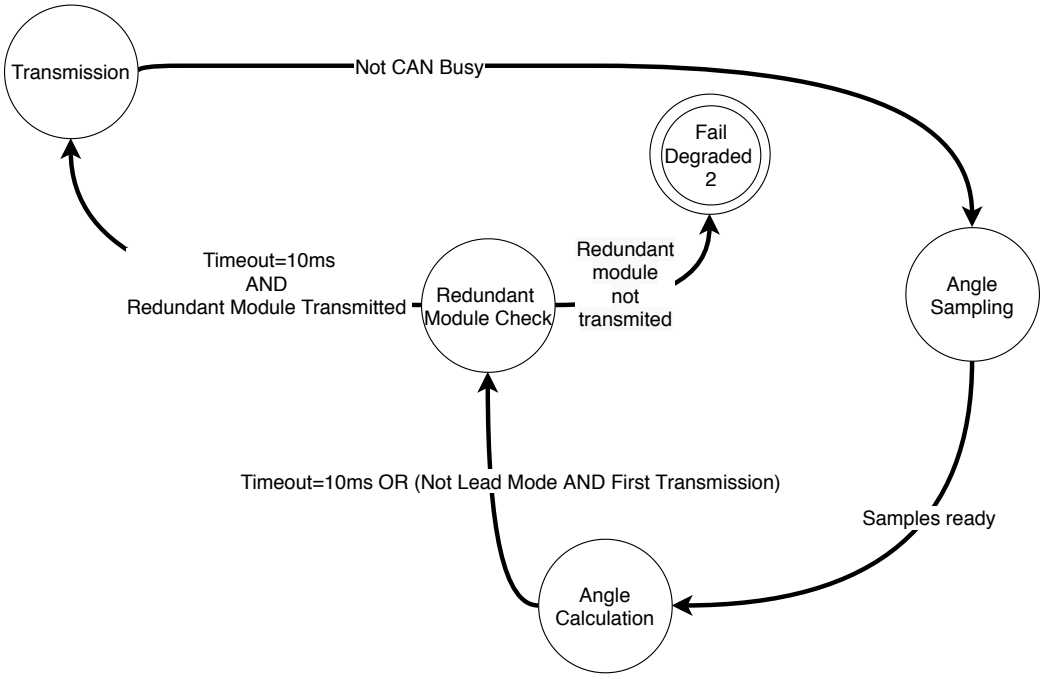


Figura 4.11: Máquina de estados da operação normal.

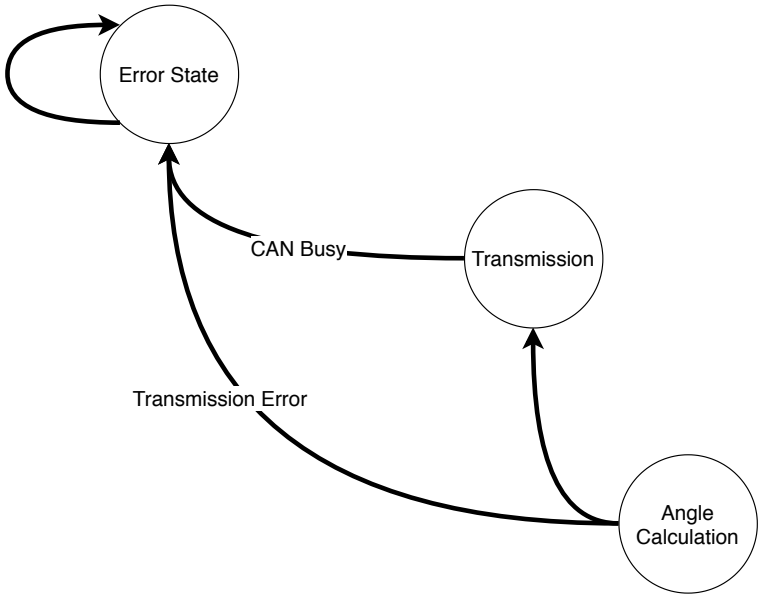


Figura 4.12: Máquina de estados (5).

ângulo. De outro modo, o módulo entra em *fail-degraded* quando é detetada a falha do módulo adjacente. Visto que nesta situação, já existe um ângulo calculado, o módulo resume a sua operação começando pela transmissão. A máquina de estados completa pode ser analisada no anexo B (B.1).

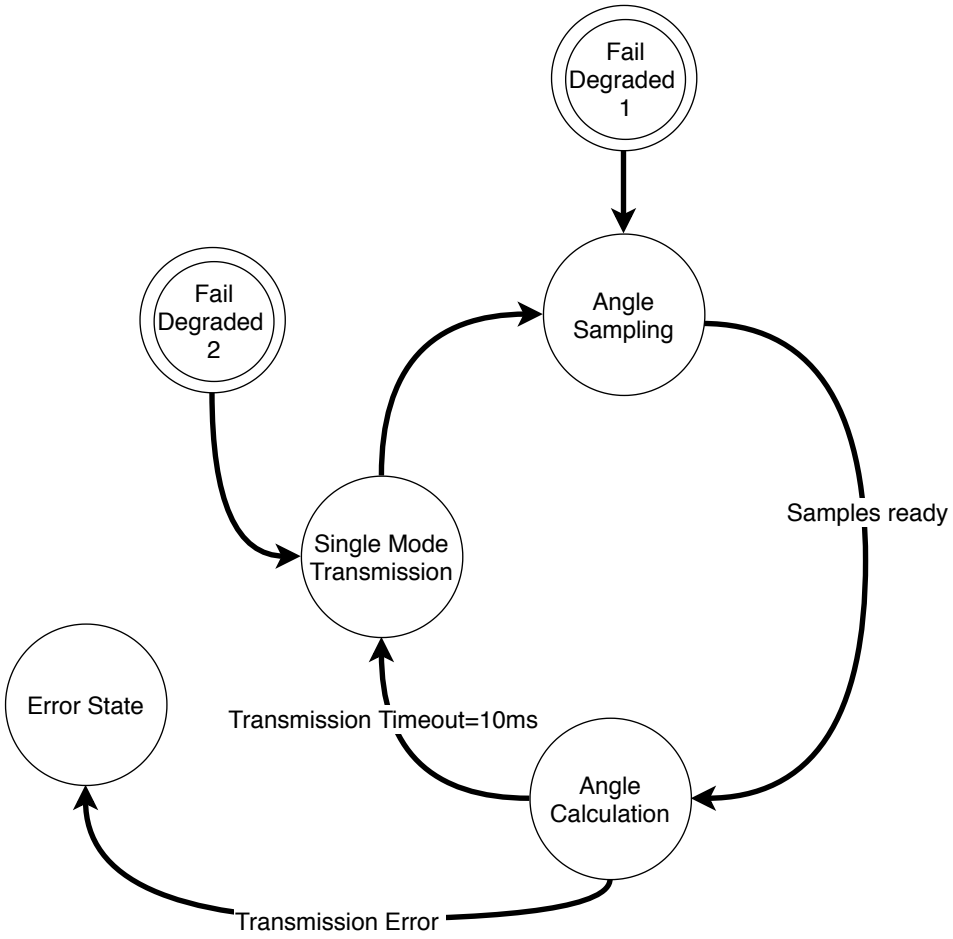


Figura 4.13: Máquina de estados da operação em *fail-degraded*.

4.1.3 Escolha da Plataforma

A escolha da arquitetura S32K2xx da NXP como plataforma alvo foi uma decisão da Bosch. Na base desta escolha, estiveram os requisitos de segurança do sistema, ASIL-D, e a mesma possuir agendamento para o suporte para o sistema operativo AUTOSAR. Como durante esta primeira iteração do projeto esta plataforma não esteve disponível, a Bosch decidiu usar uma plataforma alternativa, a qual pertence à família S32K1xx. Desta forma, durante esta iteração do projeto é utilizado o microcontrolador S32K116 da NXP.

4.1.4 Seleção de Componentes

Na escolha dos componentes utilizados para implementar a arquitetura do SAS, figura 4.6, maioritariamente foi tido em conta a conformidade dos componentes com o certificado automóvel, AEC, e com os limites de temperatura de operação do sistema. A qualificação AEC acrescenta

uma mais valia à resiliência do sistema, pois a mesma garante que os componentes foram testados de acordo com os *standards* automóvel, e que estes possuem um nível de qualidade e resiliência suficiente para serem utilizados nesta indústria.

Na base da escolha do sensor de posição angular, do *watchdog* e do *transceiver* de CAN estiveram decisões da Bosch.

O sensor de posição angular selecionado foi o TAS4141 da TDK, o qual contém 2 sensores magneto-resistivos isolados (*dual-die*). Esta técnologia permite obter informação redundante a partir de 2 sensores presentes no mesmo *package*, devido à sua proximidade física. Cada um dos sensores magnéticos presente no *dual-die*, contém internamente duas pontes magneto-resistivas que produzem uma tensão proporcional ao seno e ao cosseno, em função da orientação do campo magnético em relação ao sensor. As tensões produzidas por cada sensor estão em quadratura, o que permite, através do arco tangente, calcular o valor do ângulo sensorizado, como pode ser visualizado na figura 4.14.

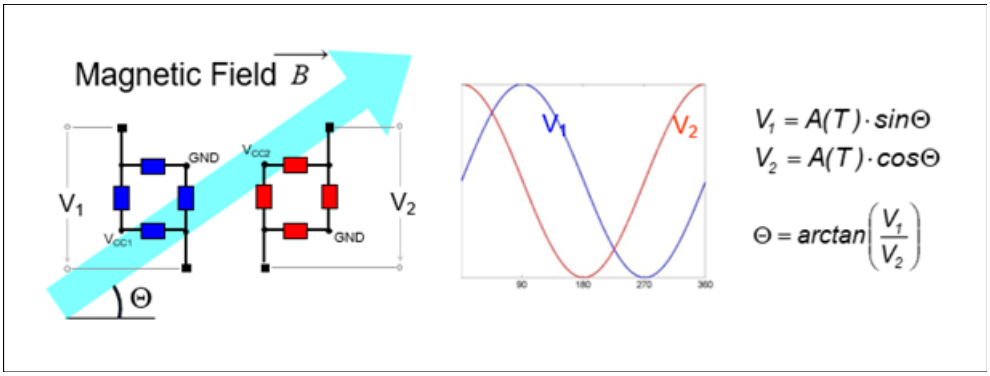


Figura 4.14: Resposta dos sensores magnéticos.

Para o cálculo do ângulo absoluto de direção, é necessário adquirir o ângulo das duas *gears* que estão conectadas ao *HUB* da direção. Desta forma, foi colocado um TAS4141 por baixo de cada uma das *gear*, figura 4.15.

O *watchdog* escolhido foi o STWD100YNPWY3F da STMicroelectronics [52], com um tempo de atuação de 3,6 ms. Para o *kicking*, o pino WD_in do *watchdog* foi conectado a um pino *general purpouse input/ output* (GPIO) do microcontrolador, este pino GPIO também pode ser configurado como a saída de um *timer*. Relativamente ao *transceiver* CAN, o escolhido foi o TJA1044GTJ da NXP Semiconductors [53] .

Para implementar isolamento galvânico entre os módulos redundantes, foi utilizado o isolador capacitivo, ISO7721QDRQ1 da Texas Instruments [54]. Esta decisão baseou-se na velocidade

e durabilidade que a tecnologia de isolamento capacitivo apresenta em relação à ótica e por apresentar um custo inferior à magnética, tabela 4.1.

Componentes	**ISO7721QDRQ1**	**ACPL-M71T-000E**	**ADUM121**
Tecnologia	Capacitiva	Ótica	Magnética
Data rate(Mbps)	100	x	150
MTBF(h)	$2.5x10^9$	$1.55x10^9$	x
Propagation delay(ns)	11	26	7.2
Preço(euros)	1.79 [51]	3.61 [51]	2.23 [51]
Isolamento(V)	5000/3000	4000	3000
Canais	2	1	2

Tabela 4.1: Comparação entre isoladores que recorrem a diferentes tecnologias.

O *decision block* permite ao *watchdog* ou ao microcontrolador desativar o *transceiver CAN*, a tabela 4.2 apresenta a tabela de verdade para o *decision block*. Para implementar a funcionalidade do *decision block* foi utilizada uma porta NAND, NLVVHC1G132DTT1G da ON Semiconductors [55].

uC_GPIO	WD_Out	CAN_STB
0	x	1
x	0	1
1	1	0

Tabela 4.2: Tabela de verdade do *decision block*.

Para alimentação dos integrados foi utilizada uma tensão de 5V, a qual será fornecida pelo regulador de tensão (LDO), TPS7B82-Q1 da Texas Instruments [56]. A figura 4.15 ilustra a arquitetura do sistema após a seleção dos componentes.

4.1.5 Desenho dos Circuitos

Para implementar a arquitetura conceptualizada para SAS foi desenhada uma PCB, a qual contém exclusivamente as funcionalidades pretendidas para o SAS. No anexo B (B.2) é possível visualizar o *overview* dos esquemáticos desta PCB.

Durante o desenho dos esquemáticos foram adicionados os condensadores de *decouple* sugeridos pelos fabricantes e um conjunto de circuitos que pretendem auxiliar a funcionalidade da PCB. De entre eles, foram adicionados à fonte de alimentação proteções de correntes inversas

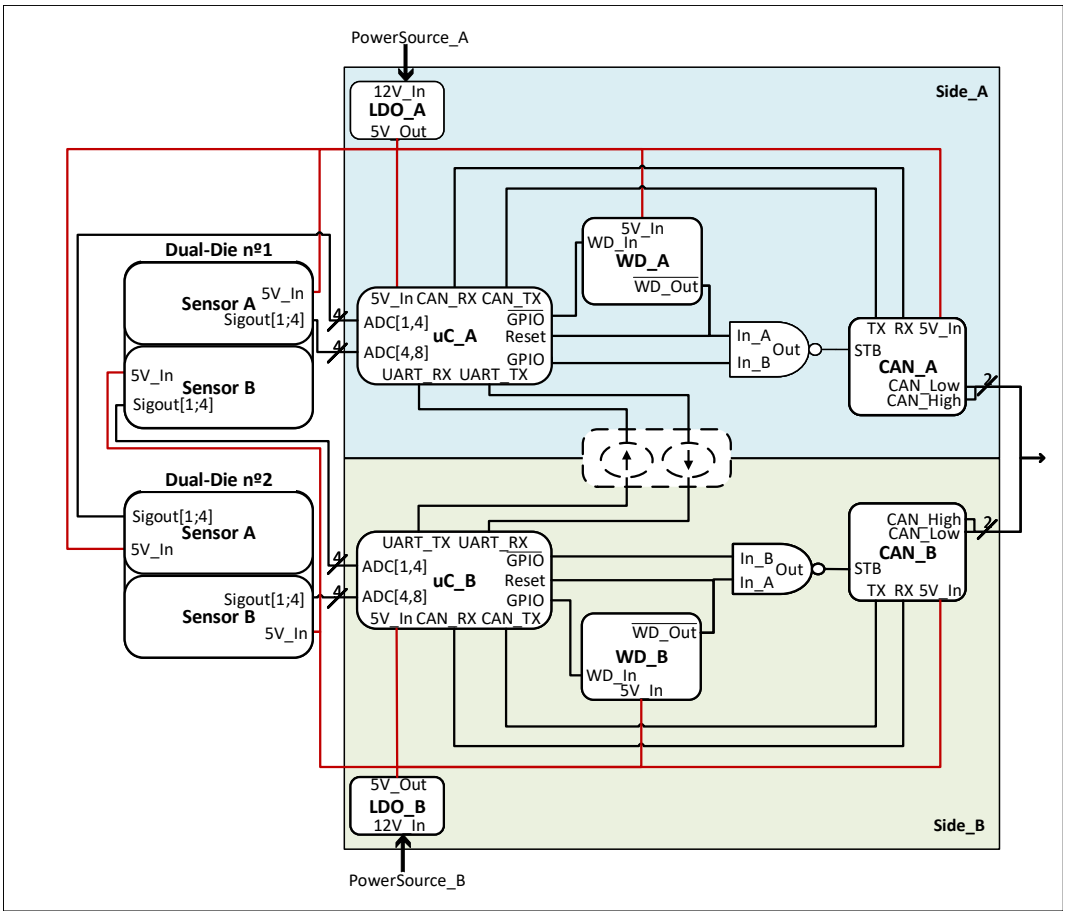


Figura 4.15: Arquitetura do sistema refinada.

e sobre-tensões, bem como um filtro PI com uma frequência de corte de 15,9 KHz. Este filtro foi adicionado com o intuito de remover ruído AC de alta frequência, que possa estar presente na linha de alimentação.

Como o protocolo CAN requer um *timing* com elevada precisão foi adicionado um cristal externo, a frequência do cristal foi sugerida pela Bosch.

Para possibilitar a programação e o *debug* do microcontrolador foi adicionado um conector que permite o interface com o *standard* JTAG. Durante a programação e *debug* a interface JTAG requer acesso ao pino de *reset* do microcontrolador, pelo que o *watchdog* não pode ser conectado ao pino de *reset* do microcontrolador nesse momento. Desta forma, foi adicionado um *jumper* que permite alternar a conexão do *reset* do microcontrolador, entre o *watchdog* e o JTAG, para funcionamento em operação normal ou em programação e *debug*, respetivamente. De forma a realizar *debug* do *watchdog* externo aquando da depuração por JTAG, ao desconectar o *watchdog* do pino de *reset* do microcontrolador, este é simultaneamente conectado a um pino GPIO do microcontrolador.

Por sugestão do fabricante do sensor de posição angular, foram adicionados filtros passa-baixo (RC passivos), com uma frequência de corte de 1575Hz aos sinais provenientes do sensor.

4.1.6 Layout

Durante o desenho do *layout* foram utilizadas boas práticas de desenho. De entre os cuidados tidos aquando o *layout*, é salientada a utilização de uma distância de 150 mils entre os planos de massa dos dois módulos redundantes. Esta distância garante o isolamento proporcionado pelo isolador capacitivo, tendo sido calculada através do critério 40V/mil especificado no *standard* UL 60950-1. A *clearance* utilizada é possível de ser visualizada na figura 4.16.

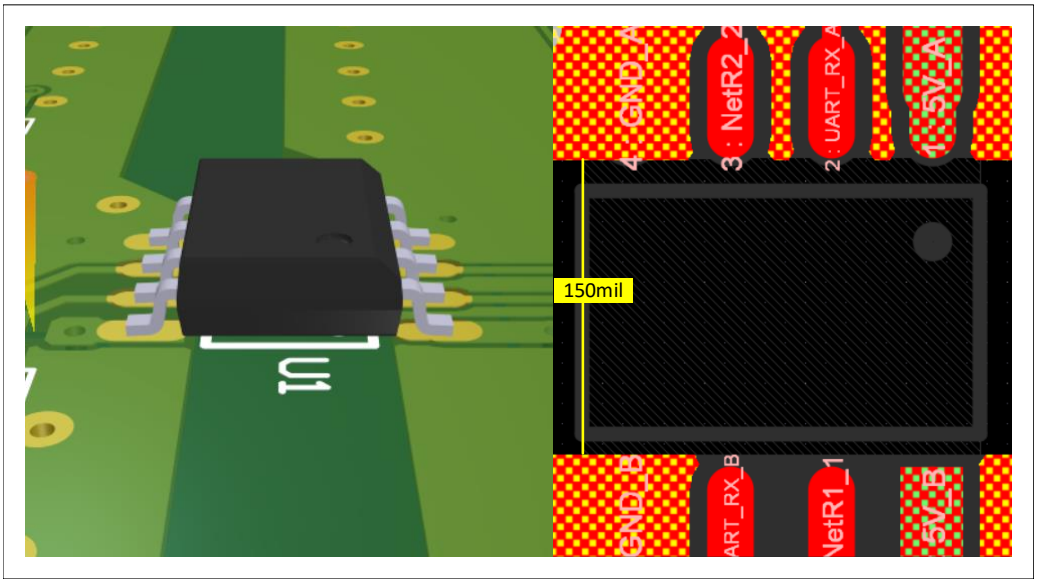


Figura 4.16: *Clearance* entre os planos de massa dos dois módulos redundantes.

4.2 Estimação da Resiliência do SAS

Para obter uma apreciação da resiliência do SAS, foram aplicadas as abordagens sugeridas no capítulo 3. Uma iteração do SAS foi realizada antes da execução das simulações de resiliência, este passo foi tomado para que os testes acelerados começassem o mais cedo possível e, assim, a sua duração não comprometesse os prazos de entrega desta dissertação.

4.2.1 Simulações de Resiliência da Arquitetura do SAS

Na arquitetura do SAS, a falha do sistema resulta da falha de ambos os módulos redundantes ou do isolador capacitivo. A figura 4.17, baseada no ecossistema de simulação apresentado no capítulo 3 (3.1.1), representa o *overview* do modelo de simulação conceptualizado para o SAS.

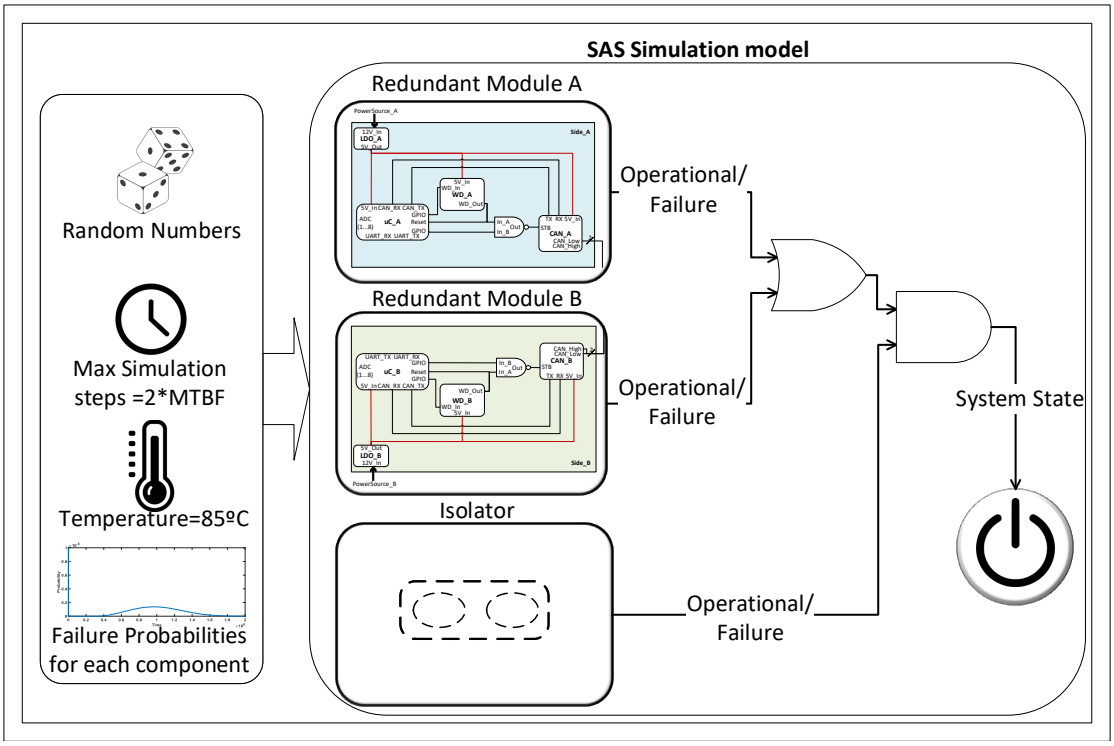


Figura 4.17: *Overview* do modelo de simulação do SAS.

Relembrando os modelos de simulação apresentados no capítulo 3, um dos *inputs* destes modelos é a função PDF de cada componente. A informação fornecida pelos fabricantes acerca da resiliência de cada componente, o MTBF, não é suficiente para obter a função PDF de cada componente. Por esta razão, foi assumido que o valor máximo da função CDF para as primeiras 100 horas de funcionamento (falhas iniciais) seria de 0.001%. Nas restantes horas, foi assumido que a função PDF seguiria uma distribuição de Weibull, tal que, o valor de MTBF corresponderia a 50% das falhas.

Para desenhar a função PDF de cada componente, foi desenvolvido um *script* em Matlab que lê os MTBFs de cada um dos componentes de um ficheiro excel e ajusta os parâmetros da distribuição de Weibull em função do MTBF e das assunções realizadas. A figura 4.18 é um exemplo de uma função PDF, a qual foi obtida a partir do MTBF do *watchdog*, de valor 10^6, e das assunções realizadas.

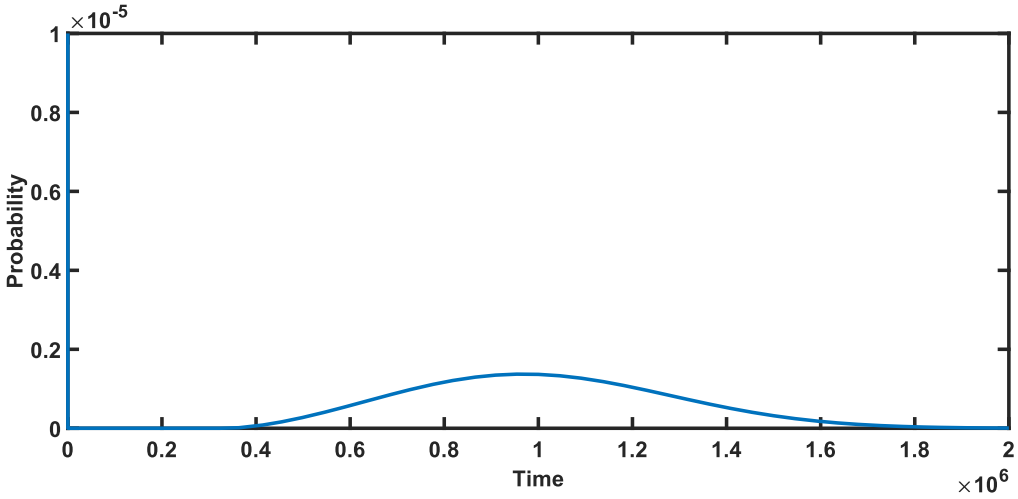


Figura 4.18: Exemplo de uma função PDF, gerada a partir do MTBF do *watchdog*, de valor 10^6, e das assunções realizadas.

Os MTBFs utilizados para cada um dos componentes passivos (condensadores, bobines, fusíveis) e para os díodos, foram obtidos a partir do *standard* militar [6]. No caso dos ICs NAND, isolador capacitivo e LDO, os mesmos foram fornecidos pelos seus fabricantes. Relativamente aos ICs *watchdog*, *transceiver* CAN e microcontrolador, os seus MTBFs foram obtidos através de ICs similares, para os quais, foi possível adquirir esta informação. Os MTBFs utilizados para cada um dos componentes do SAS encontram-se presentes na tabela B.1.

Através da análise da tabela B.1, é possível observar que o componente com o valor mais baixo de MTBF é o cristal externo. Apesar disso, este valor de MTBF corresponde aproximadamente a 3764 anos. Devido à ordem de grandeza dos MTBFs dos componentes utilizados, os mesmos foram redimensionados pelo fator 10^{-4} para reduzir a duração das simulações, com a consequência de perda de granularidade nestas simulações. Os MTBFs redimensionados encontram-se também presentes na tabela B.1.

Resultados das Simulações de Resiliência

Para o modelo desenvolvido, foram executadas 800 *trials* de Monte Carlo. Cada *trial* correu até duração máxima de 6.66 x 10^6 horas ou até à falha do sistema. A figura 4.19 apresenta a função PDF do SAS obtida a partir dos resultados de cada *trial*. Como mencionado no capítulo 2 (2.2.2), a partir da função PDF obteve-se a função CDF, a qual pode ser visualizada na figura 4.20. A partir das funções CDF e PDF obteve-se a função λ, a qual pode ser visualizada na figura 4.21.

Com base na formula mencionada no capítulo 2 (2.2.3) e na função PDF obtida, foi possível chegar à estimativa de 7,69 x 10^3 horas para o MTBF do SAS. Uma vez que os MTBFs dos componentes foram redimensionados utilizando o fator 10^{-4}, o MTBF estimado para o sistema terá de ser redimensionado pelo fator 10^4, resultando numa estimativa de 7,69 x 10^7 horas.

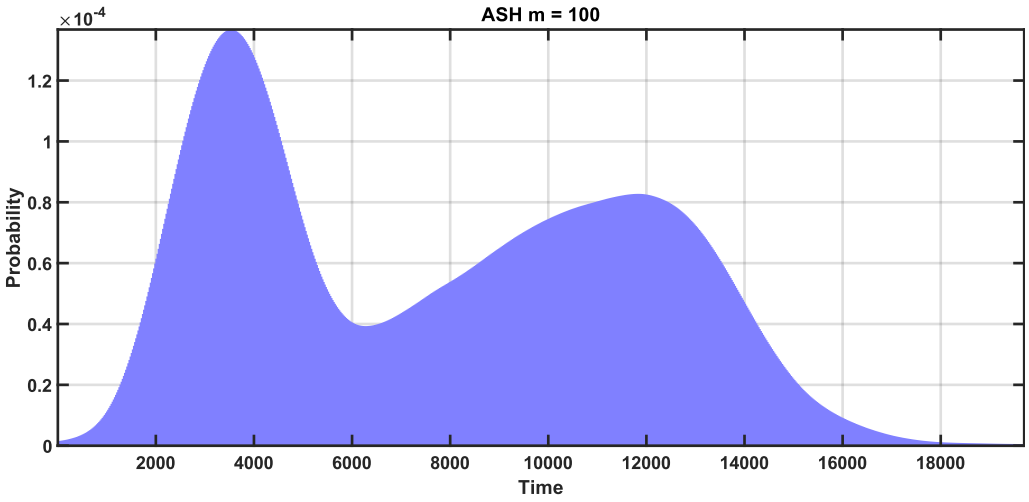


Figura 4.19: Função PDF resultante das simulações da arquitetura de hardware do SAS.

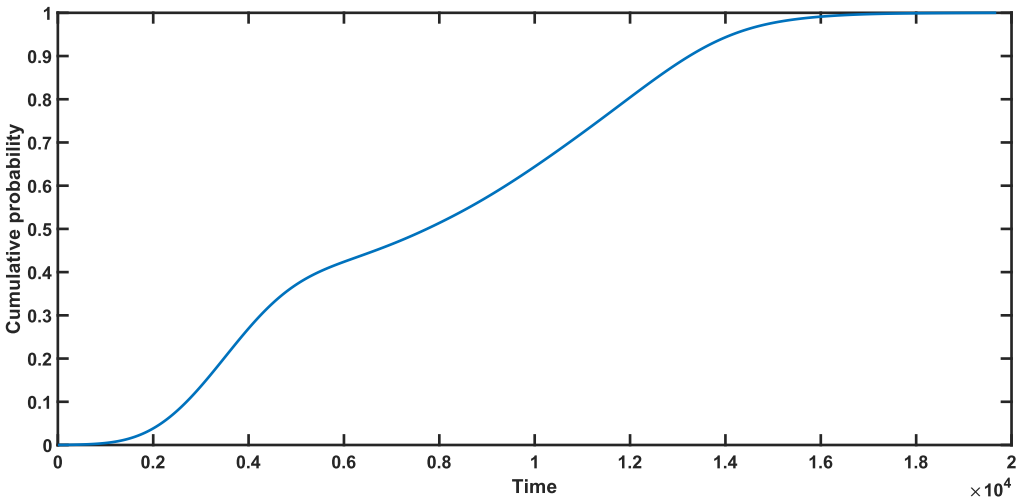


Figura 4.20: Função CDF resultante das simulações da arquitetura de hardware do SAS.

A partir da análise da função PDF obtida, é possível visualizar duas curvas gaussianas. Cada uma destas curvas é centrada em torno dos valores de MTBF mais baixos, $3.3\text{x}10^3$ e $1\text{x}10^4$, os quais correspondem aos valores redimensionados do MTBF do cristal e do fusível, respetivamente. Este facto demonstra que tanto o cristal externo como o fusível escolhido representam o *bottleneck* da arquitetura, e a sua escolha deve ser revista.

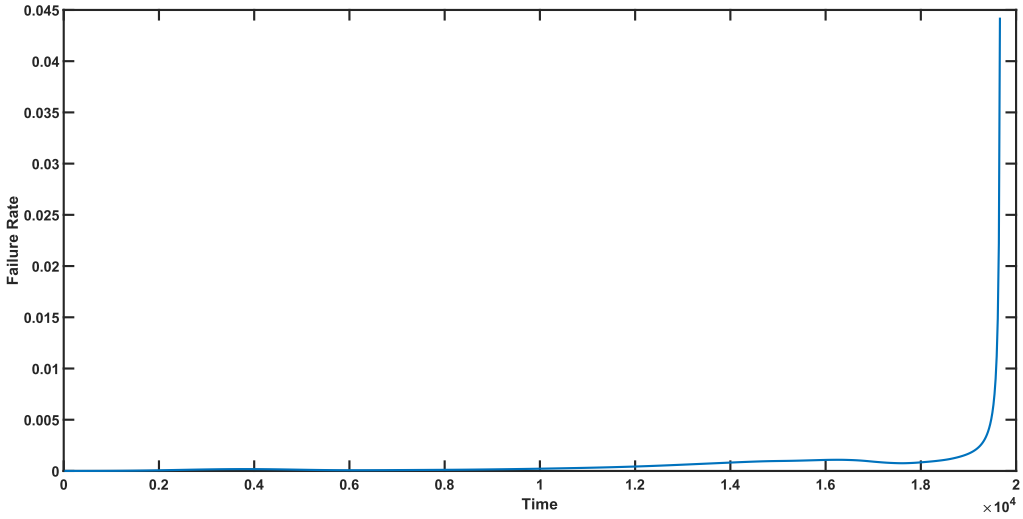


Figura 4.21: Função λ resultante das simulações da arquitetura de hardware do SAS.

O facto de ambos os módulos redundantes apresentarem funções PDF similares, como é possível visualizar nas figuras 4.22 e 4.23, sugere que pelo facto de não ter sido explorado o uso redundância heterogénea, faltas introduzidas durante o desenho irão ter repercussões em ambos os módulo redundantes, causando assim a falha do sistema.

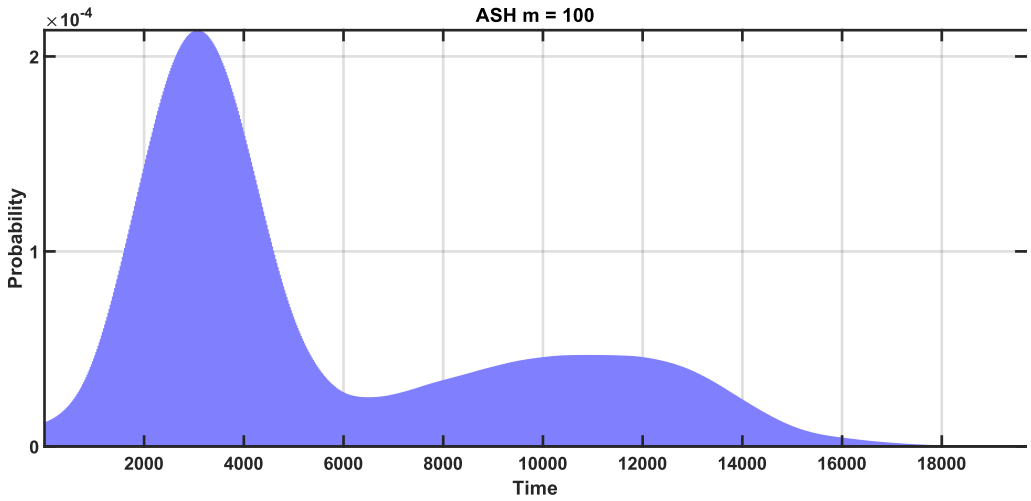


Figura 4.22: Função PDF do módulo redundante A.

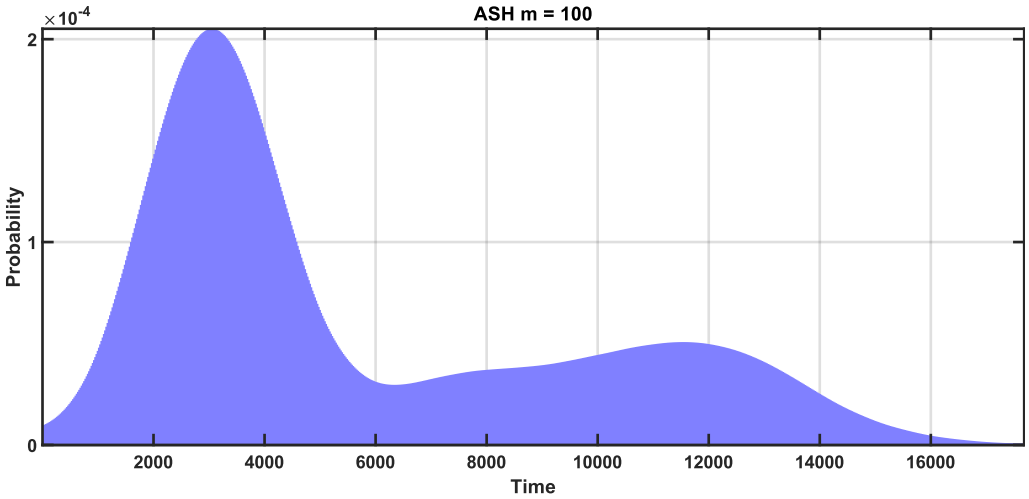


Figura 4.23: Função PDF do módulo redundante B.

4.2.2 Testes Acelerados ao SAS

Para avaliação do SAS com recurso a testes acelerados, foi utilizada uma amostra de 10 protótipos. Estes protótipos foram submetidos a diversos testes, que recorreram à temperatura como fator de aceleração/*stress*. A figura 4.24 apresenta o *overview* do *setup* utilizado para a realização destes testes, *setup* este que teve o seu desenvolvimento apresentado no capítulo 3 (3.2).

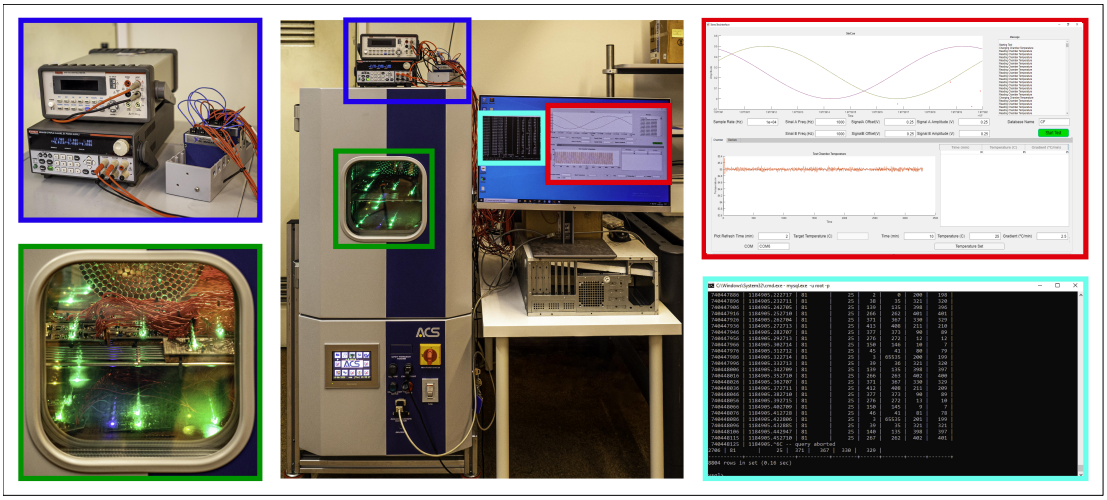


Figura 4.24: Overview do *setup* para os testes acelerados: (1) Azul escuro: NI-cDAQ-9178 e fontes de alimentação; (2) Azul claro: dados armazenados na base de dados; (3) Verde: protótipos na câmara climática; (4) Vermelho: Aplicação Gráfica;

Durante a realização dos testes, os sinais provenientes dos elementos sensitivos foram simulados recorrendo a senos e cossenos em quadratura com uma frequência de 10 Hz. Estes sinais foram gerados com recurso ao módulo de geração de sinais do NI-cDAQ-9178, o qual é controlado pela interface gráfica (capítulo 3 (3.2)). Como o algoritmo de cálculo do ângulo de direção é propriedade intelectual da Bosch, foram utilizados como *outputs* os sinais amostrados pelo SAS. A figura 4.25 ilustra o *overview* de um dos módulos de um protótipo.

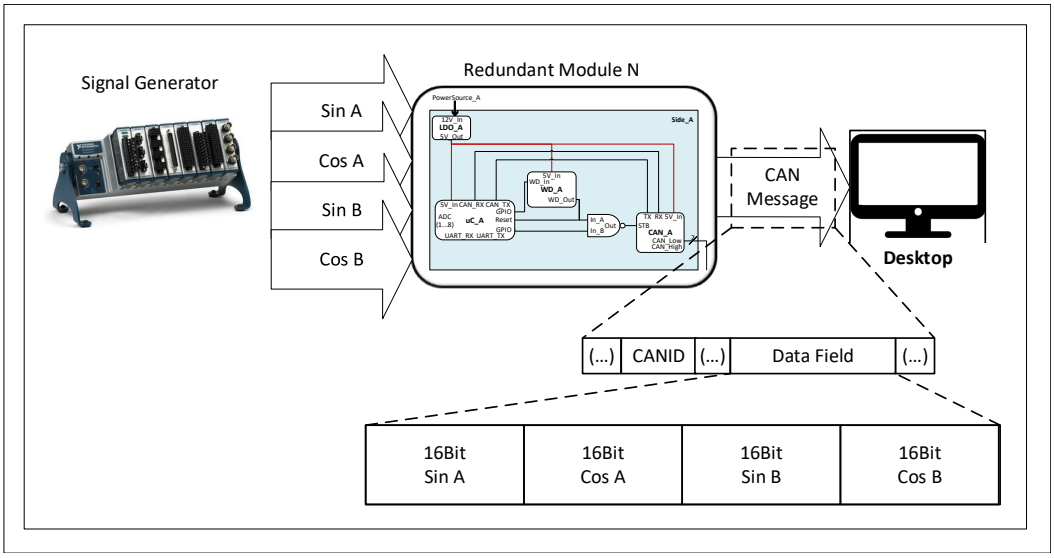


Figura 4.25: Overview do *setup* para os testes acelerados de um módulo de um protótipo.

Durante os testes, os dados de cada protótipo, dados transmitidos e períodos entre transmissões, foram armazenados e posteriormente analisados. A análise a estes dados permitiu, respetivamente, avaliar a qualidade de amostragem dos protótipos e verificar se estes protótipos cumpriam o seu requisito temporal.

Para analisar os períodos entre transmissões dos protótipos, os mesmos foram dispostos num gráfico por horas de funcionamento. O gráfico b) da figura 4.26 apresenta um *averaged shifted histogram* (ASH) com os tempos de amostragem de um dos protótipos durante uma hora.

Para analisar a qualidade de amostragem dos protótipos do SAS, os dados transmitidos por cada um dos protótipos foram processados e dispostos em gráficos por horas de funcionamento. O processamento destes dados consistiu em calcular um sinal médio e compará-lo com os sinais amostrados pelo SAS. Os dados transmitidos pelos protótipos consistiram nas amostras realizadas aos dois pares de sinais, senos e cossenos (S1/C1 e S2/C2), através da tangente inversa é possível calcular o ângulo de direção. O ângulo calculado a partir de cada par seno e

cosseno foi utilizado para calcular os senos SC1 e SC2. Com os senos amostrados (S1 e S2) e os senos calculados (SC1 e SC2) foi calculado um seno médio, o qual foi subtraído aos senos mencionados (SC1,SC2,S1 e S2). O gráfico a) da figura 4.26 representa o desvio de cada um dos senos (SC1,SC2, S1 e S2) ao seno médio, para os dados obtidos de um protótipo durante uma hora.

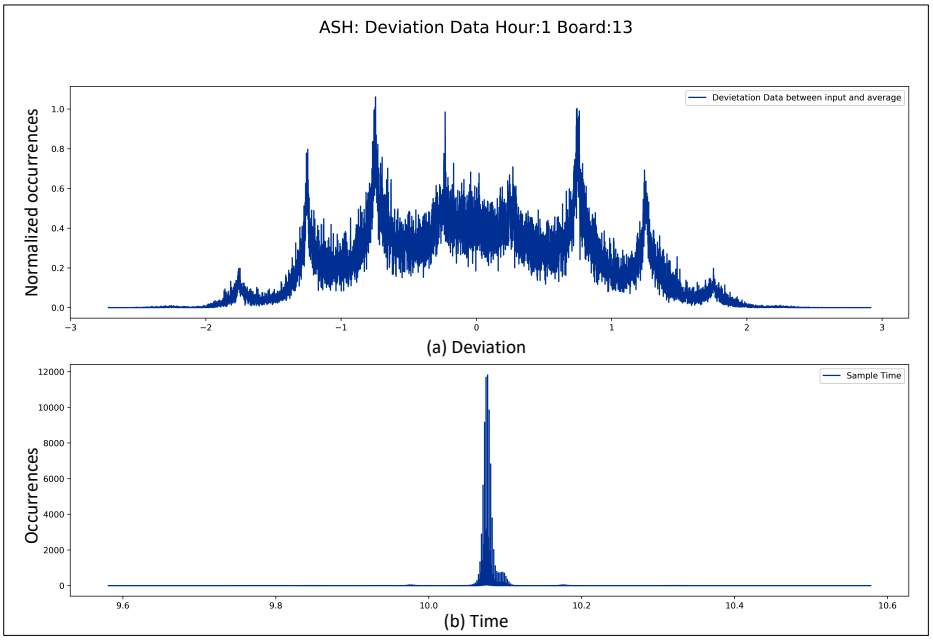


Figura 4.26: ASH com os desvios entre os sinais amostrados e um sinal médio (a), e períodos entre transmissão (b) de um protótipo em teste, durante 1 hora.

No SAS quando um dos módulos falha, o módulo adjacente entra em *fail-degraded* sendo re-configurado de forma a compensar a ausência do módulo em falha. Esta operação internamente realizada pelo SAS, não é transparente, pelo que, não é visível no barramento CAN. Durante os testes, para ser possível identificar a falha de um dos módulos redundantes, a aplicação do SAS foi alterada passando a transmitir uma mensagem identificadora, cada vez que o SAS transitasse para o estado *fail-degraded*. Esta mensagem identificadora consiste em transmitir o máximo de um inteiro de 16 *bits* com sinal, em vez do valor amostrado pelo SAS. A figura 4.27 apresenta um caso em que foi detetada a mensagem identificadora da falha de um módulo de um protótipo. O desvio de cerca de 16000, presente no gráfico b) da figura 4.27 é caraterístico da presença de uma mensagem identificadora nos dados recolhidos dos protótipos do SAS.

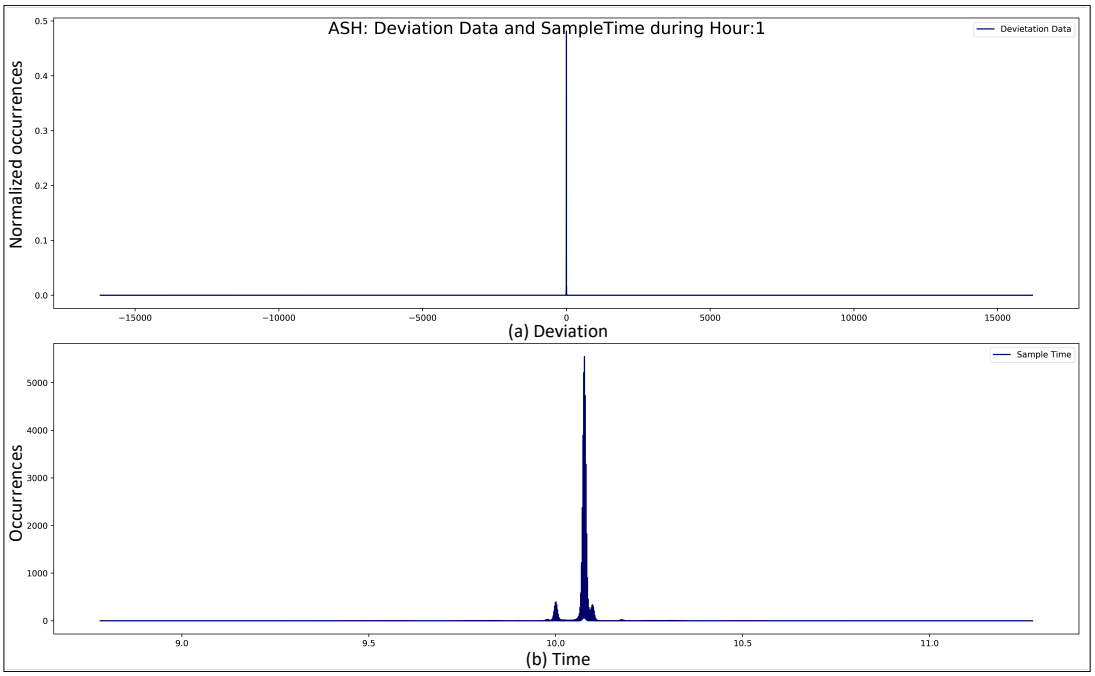


Figura 4.27: ASH com os desvios entre sinais amostrados e um sinal médio (a), e períodos de transmissão (b) de todos os protótipo em teste, durante 1 hora.

Testes Realizados

Aos 10 protótipos do SAS foram realizados 3 teste, 2 com o propósito de avaliar a qualidade do sistema e um de forma a acelerar o envelhecimento e estimar uma distribuição de falhas. Durante estes testes a temperatura foi utilizada como mecanismo para estimular o aparecimento de faltas no sistema.

O primeiro teste qualitativo pretendeu aplicar *stress* mecânico sobre a PCB e avaliar a qualidade da soldadura industrial. Para efetuar este teste, foram aplicados ciclos de temperatura dentro dos limites de operação do sistema, -40°C e 85°C, durante 230 horas.

O segundo teste qualitativo teve o intuito de aplicar *stress* sobre os condensadores eletrolíticos, e avaliar a sua *performance* nos limites inferiores da temperatura de operação do sistema. Para isso, durante este teste os protótipos foram submetidos, durante 65 horas, a uma temperatura no seu limite inferior de operação.

O último teste (quantitativo) foi realizado apenas com o propósito de acelerar o envelhecimento dos componentes e obter uma distribuição de falhas. Durante este teste, os protótipos foram submetidos a uma temperatura no seu limite superior de operação. Este teste ainda está em realização, contando já com 4776 horas de teste.

Resultados dos Testes Acelerados ao SAS

Ao longo das 5071 horas totais de teste, foram recolhidas aproximadamente 18 mil milhões de mensagens provenientes dos 10 protótipos em teste. O armazenamento desta mensagens resultou em cerca de 32 *terabytes* de dados.

Da análise destes dados, foi possível detetar a ocorrência de falhas prematuras nos protótipos do SAS. A partir de uma análise ao hardware do SAS, foi possível determinar que na origem destas falhas estavam imperfeições na soldadura industrial dos cristais de relógio. Além disto, estas imperfeições (faltas) intensificaram o desfasamento entre os cristais de relógio dos módulos redundantes, revelando uma falta de desenho nos mecanismos de sincronização entre módulos. Após a correção da falta de desenho e do defeito de fabrico, retomaram-se os testes.

Até ao momento não ocorreram mais falhas, para além das já mencionadas. Relembrando o capítulo 2 (2.2.7), componentes eletrónicos apresentam uma Ea entre 0.3 e 0.7. Segundo a equação de Arrhenius e considerando a temperatura de operação do sistema como 25°C, durante o teste quantitativo, o sistema sofreu um fator de aceleração por temperatura de pelo menos 7 vezes. Da aplicação do fator de aceleração mencionado, às 4776 horas de duração do último teste, resultam pelo menos 33432 horas de funcionamento acelerado sem ocorrência de mais falhas.

4.3 Sumário

Neste capítulo, foi apresentado o desenvolvimento de uma arquitetura de hardware resiliente para o SAS.

A utilização do modelo de simulação explorado no capítulo 3 (3.1), permitiu obter uma estimativa da função PDF do SAS, o que consequentemente permitiu obter as funções CDF e λ da arquitetura desenvolvida. Além disto, a partir da função PDF foi obtida uma estimativa de valor 7,69 x 10^7 horas para o MTBF da arquitetura.

A utilização do *setup* para testes acelerados, cujo o desenvolvimento foi apresentado no capítulo 4 (4.2.2), permitiu realizar testes acelerados a 10 protótipos do SAS. Estes testes ainda se encontram em realização, contando já com um total de 5071 horas. Durante as primeiras horas dos testes acelerados ao sistema, surgiram falhas prematuras, oriundas de defeitos introduzidos na produção e de faltas de design. Assim como sugerido no capítulo 2 (2.2.2), estas falhas

enquadram-se na primeira fase da curva da banheira (mortalidade infantil). Uma vez que não surgiram mais falhas, é possível concluir que o sistema avançou para a próxima fase da curva da banheira, encontrando-se na sua vida útil.

Capítulo 5

Conclusões

Os objetivos desta dissertação consistiam em desenvolver uma arquitetura resiliente para o sensor automóvel SAS e em estimar a sua resiliência, recorrendo a modelação de resiliência e a testes acelerados. Para alcançar estes objetivos, era necessário desenvolver um *setup* de testes, que permitisse realizar os testes acelerados aos protótipos do SAS.

Relativamente ao desenvolvimento da arquitetura, foram utilizadas técnicas de tolerância a faltas, mais precisamente, redundância homogénea. E, também, boas práticas de desenho, nomeadamente escolha de componentes certificados e utilização de isolamento galvânico. Estas foram aplicadas com o intuito de desenhar uma arquitetura de hardware resiliente para o SAS.

Para estimar a resiliência desta arquitetura, foram desenvolvidos com sucesso um modelo de resiliência e um *setup* para testes acelerados. O *setup* desenvolvido já permitiu realizar 5071 horas de teste, nas quais foram recolhidas 18 mil milhões de mensagens provenientes de uma amostra de 10 protótipos.

A simulação do modelo de resiliência do SAS permitiu estimar a função PDF do sistema. A partir da função PDF foram obtidas as funções CDF e λ, e uma estimativa de 7,69 x 10^7 horas para o MTBF do sistema. A análise da função PDF, permitiu identificar dois *bottlenecks* na arquitetura. O primeiro foi relativo à escolha do cristal externo e do fusível, sendo que as suas falhas levaram à falha da arquitetura. O segundo foi referente ao uso de redundância homogénea, o que resultou em longevidades similares para ambos os módulos redundantes.

Aos 10 protótipos do SAS foram realizados 3 testes (2 qualitativos e 1 quantitativo), os quais utilizaram a temperatura como fator de aceleração. Durante os testes, foram detetadas falhas prematuras oriundas de defeitos na produção e de uma falta de desenho.

Como mencionado no capítulo 4 (4.2.2), ao longo das 4776 horas do teste quantitativo, o sistema sofreu um fator de aceleração de pelo menos 7 vezes, resultando em 33432 horas de

funcionamento acelerado. Apesar das 5071 horas totais de teste não serem suficientes para alcançar a terceira fase da curva da banheira do sistema (mortalidade senil ou desgaste), foi possível cobrir, e até ultrapassar, as 12000 horas de funcionamento pretendidas pela Bosch.

Uma vez que através de testes acelerados ainda não foi possível alcançar a terceira fase da curva da banheira do sistema (mortalidade senil ou desgaste), também ainda não foi possível estimar a função PDF do SAS através destes testes. Desta forma, não é possível realizar comparações entre as estimativas realizadas pela simulação e pelos testes acelerados.

5.1 Conclusões Sobre o Desenho de Sistemas Resilientes

A partir do trabalho realizado ao longo desta dissertação, é possível retirar três conclusões sobre o desenho de sistemas resilientes.

Como primeira conclusão, não é possível transpor diretamente as soluções encontradas em setores como o militar, aeroespacial e aeronáutico para sistemas automóvel. Nos setores onde o custo não é uma limitação, existe uma grande variedade de arquiteturas com uma baixa probabilidade de falha. Porém, no setor automóvel o baixo custo representa um dos principais requisitos.

Como segunda conclusão, estimativas obtidas a partir da simulação dos modelos de resiliência não são fiáveis, pois os fabricantes não fornecem informação suficiente. Os fabricantes fornecem como estimativa de resiliência para os seus componentes o MTBF. Porém, este valor possuí pouca informação acerca da distribuição de falhas de componentes ou sistemas, sendo útil apenas para realizar comparações.

Como terceira conclusão, com as acelerações de envelhecimento alcançadas durante os testes acelerados, não é possível estimar a longevidade do sistema num período que não comprometa o *time-to-market* do produto. Testes acelerados que recorrem a temperatura como fator de aceleração, para sistemas automóvel que possuem o limite superior de temperatura de operação de 85°C, permitem alcançar acelerações entre cerca de 7 a 100 vezes, segundo o modelo de Arrhenius e considerando a temperatura normal de operação como 25°C. Isto significa, que pode levar cerca de 2 anos até alcançar a mortalidade senil ou desgaste de um sistema, o qual possua uma longevidade de aproximadamente 15 anos.

Se por um lado os fabricantes não fornecem informação suficiente para realizar projeções de resiliência, por outro lado, para realizar testes acelerados são necessários longos períodos de tempo, o que pode comprometer o *time-to-marker* do produto a desenvolver. Desta forma, tanto testes acelerados como modelos de resiliência, atualmente, são insuficientes para estimar a longevidade de um sistema.

5.2 Trabalho Futuro

Relativamente ao trabalho futuro, existem um conjunto de melhorias que poderiam ser realizadas, tanto nas simulações como nos testes acelerados.

Em relação às simulações, em primeiro, melhorar a fiabilidade das suas estimativas. Para alcançar este objetivo, os fabricantes teriam de ser contactados, por forma a fornecerem melhor qualidade de dados relativos à resiliência dos seus componentes. Caso isto não seja possível, é possível a recolha de dados sobre a longevidade e as falhas destes componentes através da realização de testes acelerados. Em segundo, aumentar a exatidão das simulações através da melhoria do modelo de resiliência do sistema. A inclusão das conexões e a inclusão da degradação da funcionalidade dos componentes aumentariam a exatidão das estimativas realizadas pelas simulações.

Relativamente aos testes acelerados, em primeiro lugar realizar outros testes com diferentes fatores de aceleração como vibração, humidade, alimentação, pressão, etc, de forma a despertar o aparecimento de faltas diferentes. Em segundo, a combinação de vários fatores de aceleração ao mesmo tempo, poderia aumentar a aceleração do envelhecimento do sistema, reduzindo assim a duração dos testes.

Quanto ao caso de estudo, apesar dos requisitos impostos pela Bosch, algumas melhorias poderiam ser levadas a cabo. Uma delas diz respeito ao mecanismo de comunicação entre módulos redundantes, o qual é utilizado para a sincronização entre módulos redundantes, e apenas consiste num *feedback* digital. A alteração deste *feedback* digital para um protocolo de comunicação mais complexo possibilitaria uma melhor sincronização e decisões entre módulos. Além disto a utilização de diversidade, através da utilização de componentes diferentes de diferentes fabricantes, aumentaria a resistência do sistema à falta de modo comum.

Referências

[1] US Department of Defense, *MILITARY HANDBOOK Electronic Reliability Design Handbook*, 1 October. 1998. doi: MIL-HDBK-338B.

[2] A. Birolini, *Reliability Engineering: Theory and Practice*, sér. Engineering online library. Springer Berlin Heidelberg, 2003, isbn: 9783540402879. URL: https://books.google.pt/books?id=FhcxBJZeMqUC.

[3] M. Q. Khairuzzaman, *Handbook of Reliability, Availability, Maintainability and Safety in Engineering Design*, 1. London: Springer London, 2009, vol. 4, isbn: 978-1-84800-174-9. doi: 10.1007/978-1-84800-175-6. URL: http://link.springer.com/10.1007/978-1-84800-175-6.

[4] Y. C. Yeh, «Triple-triple redundant 777 primary flight computer,» *IEEE Aerospace Applications Conference Proceedings*, vol. 1, pp. 293–307, 1996. doi: 10.1109/aero.1996.495891.

[5] M. Abd-El-Barr, *Design and Analysis of Reliable and Fault-Tolerant Computer Systems*. PUBLISHED BY IMPERIAL COLLEGE PRESS e DISTRIBUTED BY WORLD SCIENTIFIC PUBLISHING CO., dez. de 2006, pp. 1–441, isbn: 978-1-86094-668-4. doi: 10.1142/p457. URL: https://www.worldscientific.com/worldscibooks/10.1142/p457.

[6] US Department of Defense, *MILITARY HANDBOOK RELIABILITY PREDICTION OF ELECTRONIC EQUIPMENT*. 1995.

[7] S. Limon, O. P. Yadav e H. Liao, «A literature review on planning and analysis of accelerated testing for reliability assessment,» *Quality and Reliability Engineering International*, vol. 33, n.º 8, pp. 2361–2383, 2017, issn: 10991638. doi: 10.1002/qre.2195.

[8] L. A. Escobar e W. Q. Meeker, «A Review of Accelerated Test Models,» *Statistical Science*, vol. 21, n.º 4, pp. 552–577, 2006. doi: 10.1214/088342306000000321. arXiv: 0708.0369v1.

[9] C. Huang, F. Naghdy, H. Du e H. Huang, «Fault tolerant steer-by-wire systems: An overview,» *Annual Reviews in Control*, 2019, issn: 13675788. doi: 10.1016/j.arcontrol.2019.04.001.

[10] E. White, *Making Embedded Systems: Design Patterns for Great Software*. 2011, vol. 2011, isbn: 1449320589. URL: http://books.google.com/books?id=VCOTy1xWZmQC%7B%5C&%7Dpgis=1.

[11] R. Morrison, *Digital Circuit Boards*. Hoboken, NJ, USA: John Wiley & Sons, Inc., 2012, isbn: 9781118278123. doi: 10.1002/9781118278123. URL: https://libproxy.singaporetech.edu.sg/login?url=http://search.ebscohost.com/login.aspx?direct=true%7B%5C&%7DAuthType=cookie,ip,uid%7B%5C&%7Ddb=cat04396a%7B%5C&%7DAN=sit.EF013021%7B%5C&%7Dsite=eds-live%7B%5C%7D0Ahttp://libproxy.singaporetech.edu.sg/login?url=http://onlinelibrary.wiley.com/book/10.10%20http://doi.wiley.com/10.1002/9781118278123.

[12] R. Morrison, *Grounding and Shielding*. Hoboken, NJ, USA: John Wiley & Sons, Inc, 2016, vol. 66, isbn: 9781119183723. doi: 10.1002/9781119183723. URL: http://doi.wiley.com/10.1002/9781119183723.

[13] Cadence PCB solutions, *Power Plane PCB: Best Practices for Power Planes in Multi-board Design | Advanced PCB Design Blog* |. URL: https://resources.pcb.cadence.com/blog/2019-best-practices-for-power-planes-in-multi-board-pcb-design (acedido em 07/02/2021).

[14] A. Grasset, «Design of critical embedded systems: from early specifications to prototypes,» em *2015 International Symposium on Rapid System Prototyping (RSP)*, 2015, pp. 38–38. doi: 10.1109/RSP.2015.7416544.

[15] K. Becker, «Software Deployment Analysis for Mixed Reliability Automotive Systems,» tese de doutoramento, 2017. URL: https://pdfs.semanticscholar.org/e873/92e0f31114ff1070ee51ad111fecf417dd07.pdf.

[16] K. J. Lee, Y. H. Ki, J. S. Cheon, G. Hwang e H. S. Ahn, «Approach to functional safety-compliant ECU design for electro-mechanical brake systems,» *International Journal of Automotive Technology*, vol. 15, n.º 2, pp. 325–332, mar. de 2014, issn: 12299138. doi: 10.1007/s12239-014-0033-7.

[17] A. Algirdas, L. Jean-Claude, R. Brian e L. Carl, «Basic Concepts and Taxonomy of Dependable and Secure Computing,» *IEEE Transactions on Dependable and Secure Computing*, vol. 1, n.º 1, pp. 11–33, 2004, issn: 1545-5971. doi: 10.1109/TDSC.2004.2. URL: http://ieeexplore.ieee.org/xpls/abs%7B%5C_%7Dall.jsp?arnumber=1335465%7B%5C&%7Dtag=1.

[18] E. Dubrova, *Fault-tolerant design*. Springer New York, jan. de 2013, isbn: 9781461421139. doi: 10.1007/978-1-4614-2113-9.

[19] ReliaSoft Corporation, «Life Data Analysis Reference,» *Tools to Empower: The Reliability Professional*, pp. 103–154, 2015.

[20] Texas Instruments, «Understanding quality levels for high reliability-rated components,» rel. téc. URL: www.ti.com2018 (acedido em 07/02/2021).

[21] AECTechnicalCommittee, *AEC Documents*. URL: http://www.aecouncil.com/AECDocuments.html (acedido em 07/02/2021).

[22] golledge, *The AEC-Q200 Standard, what does it really mean?* URL: https://www.golledge.com/news/the-aec-q200-standard-what-does-it-really-mean/ (acedido em 07/02/2021).

[23] R. Corporation, *Arrhenius Relationship*, fev. de 2019. URL: http://reliawiki.org/index.php/Arrhenius_Relationship (acedido em 19/01/2021).

[24] T. I. Biljenescu e M. I. Bazu, *Reliability of Electronic Components - A Practical Guide to Electronic Systems Manufacturing*, 9. 2013, vol. 53, isbn: 9788578110796. doi: 10.1017/CBO9781107415324.004. arXiv: arXiv:1011.1669v3.

[25] K. Hindi, *Practical Reliability Engineering*. 1981, vol. 27, isbn: 9780470979822. doi: 10.1049/ep.1981.0371.

[26] ReliaSoft Corporation, *The Risks of Using Failure Rate to Calculate Reliability Metrics*, jun. de 2017. URL: https://weibull.com/hotwire/issue196/hottopics196.htm.

[27] ReliaSoft Corporation, *The Limitations of Using the MTTF as a Reliability Specification*, 2020. URL: https://www.reliasoft.com/resources/resource-center/the-limitations-of-using-the-mttf-as-a-reliability-specification.

[28] B. W. Torell e V. Avelar, «Mean Time Between Failure: Explanation and Standards,» *Power*, vol. 78, n.º January, 2004. URL: http://support.casit.net/Portals/0/NTForums%7B%5C_%7DAttach/VAVR-5WGTSB%7B%5C_%7DR0%7B%5C_%7DEN.pdf.

[29] US Department of Defense, *Reliability Modeling and Prediction*. 1981.

[30] K. Naruse e T. Nakagawa, «Optimal Checkpoint Intervals, Schemes and Structures for Computing Modules,» em. mar. de 2020, pp. 265–287, isbn: 978-3-030-43411-3. doi: 10.1007/978-3-030-43412-0_16.

[31] A. Schnellbach, «Fail-operational automotive systems,» n.º November, 2016.

[32] T. Wilfredo, «Software Fault Tolerance: A Tutorial,» n.º October, 2000.

[33] Texas Instruments, *MTBF and FIT Rate Estimator*. URL: https://www.ti.com/quality/docs/estimator.tsp?partType=tiPartNumber%7B%5C&%7DpartNumber=ISO7721QDRQ1%7B%5C#%7Dresultstable (acedido em 06/01/2021).

[34] N. Privault, *Understanding Markov Chains*. 2013, isbn: 978-981-4451-50-5. doi: 10.1007/978-981-4451-51-2. URL: http://link.springer.com/10.1007/978-981-4451-51-2.

[35] E. Possan e J. J. De Oliveira Andrade, «Markov chains and reliability analysis for reinforced concrete structure service life,» *Materials Research*, vol. 17, 2014, issn: 15161439. doi: 10.1590/S1516-14392014005000074.

[36] A. Scedrov, «A. A. Markov and N. M. Nagorny. The theory of algorithms. English translation by M. Greendlinger of Teoriya algorifmov . Mathematics and its applications (Soviet series). Kluwer Academic Publishers, Dordrecht, Boston, and London, 1988, xxiv + 369 pp.,» *Journal of Symbolic Logic*, vol. 56, n.º 1, mar. de 1991, issn: 0022-4812. doi: 10.2307/

2274930. URL: https://www.cambridge.org/core/product/identifier/S0022481200025160/type/journal%7B%5C_%7Darticle.

[37] I. Kecerdasan e P. Ikep, *Markov Chains: Theory and Applications*, isbn: 9781848214934.

[38] K. balaji rao, «Markov Chain Modelling for Reliability Estimation of Engineering Systems at Different Scales - Some Considerations,» set. de 2007.

[39] E. Zio, *The Monte Carlo Simulation Method for System Reliability and Risk Analysis*, 145. 2013, vol. 4, isbn: 978-1-4471-4587-5. doi: 10.1007/978-1-4471-4588-2. URL: http://link.springer.com/10.1007/978-1-4471-4588-2.

[40] E. W. Weisstein, «Buffon's Needle Problem,» nov. de 2021. URL: https://mathworld.wolfram.com/BuffonsNeedleProblem.html.

[41] Right. Hon. Lord Kelvin G.C.V.O. D.C.L. LL.D. F.R.S. M.R.I., «I. Nineteenth century clouds over the dynamical theory of heat and light,» *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, vol. 2, n.º 7, pp. 1–40, 1901. doi: 10.1080/14786440109462664.

[42] C. Andrieu, «Monte carlo methods for absolute beginners,» *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 3176, pp. 113–145, 2004, issn: 16113349. doi: 10.1007/978-3-540-28650-9_6. URL: https://link.springer.com/chapter/10.1007/978-3-540-28650-9%7B%5C_%7D6.

[43] IBM Cloud Education, *What is Monte Carlo Simulation?* Ago. de 2020. URL: https://www.ibm.com/cloud/learn/monte-carlo-simulation (acedido em 19/01/2021).

[44] S. Arrhenius, «Über die Dissociationswärme und den Einfluss der Temperatur auf den Dissociationsgrad der Elektrolyte,» *Zeitschrift für Physikalische Chemie*, vol. 4U, n.º 1, jan. de 1889, issn: 2196-7156. doi: 10.1515/zpch-1889-0408. URL: http://www.degruyter.com/view/j/zpch.1889.4.issue-1/zpch-1889-0408/zpch-1889-0408.xml.

[45] K. J. Laidler, *Chemical kinetics*, 3rd ed. New York : Harper & Row, c1987.

[46] K. Connors, *Chemical Kinetics: The Study of Reaction Rates in Solution*. VCH, 1990, isbn: 9781560810537. URL: https://books.google.pt/books?id=nHux3YED1HsC.

[47] NATIONAL INSTRUMENTS CORP., *cDAQ-9178 - NI*. URL: https://www.ni.com/pt-pt/support/model.cdaq-9178.html (acedido em 08/02/2021).

[48] NATIONAL INSTRUMENTS CORP., *NI-9269 - NI*. URL: https://www.ni.com/pt-pt/support/model.ni-9269.html (acedido em 08/02/2021).

[49] NATIONAL INSTRUMENTS CORP., *NI-9229 - NI*. URL: https://www.ni.com/pt-pt/support/model.ni-9229.html (acedido em 08/02/2021).

[50] PEAK-System, *PCAN-USB Pro: PEAK-System*. URL: https://www.peak-system.com/PCAN-USB-Pro.200.0.html?%7B%5C&%7DL=1 (acedido em 08/02/2021).

[51] *Mouser Portugal*. URL: https://pt.mouser.com/ (acedido em 06/01/2021).

[52] STMicroelectronics, *STWD100 - Watchdog timer circuit - STMicroelectronics*. URL: https://www.st.com/content/st%7B%5C_%7Dcom/en/products/reset-and-supervisor-ics/watchdog-timers/stwd100.html (acedido em 08/02/2021).

[53] NXP Semiconductors, *TJA1044GT Product Information | NXP*. URL: https://www.nxp.com/part/TJA1044GT%7B%5C#%7D/ (acedido em 08/02/2021).

[54] Texas Instruments, *ISO7720-Q1 data sheet, product information and support | TI.com*. URL: https://www.ti.com/product/ISO7720-Q1 (acedido em 08/02/2021).

[55] ON Semiconductor, *MC74VHC1G132: Single 2-Input NAND Gate with Schmitt Trigger Input*. URL: https://www.onsemi.com/products/standard-logic/logic-gates/nlvvhc1g132dtt1g (acedido em 08/02/2021).

[56] Texas Instruments, *TPS7B82-Q1 data sheet, product information and support | TI.com*. URL: https://www.ti.com/product/TPS7B82-Q1 (acedido em 08/02/2021).

[57] Texas Instruments, *MTBF and FIT Rate Estimator*. URL: https://www.ti.com/quality/docs/estimator.tsp (acedido em 06/01/2021).

[58] ON Semiconductor, *Reliability Data*. URL: https://www.onsemi.com/PowerSolutions/reliability.do (acedido em 06/01/2021).

Anexo A

Setup para Testes Acelerados Material Auxiliar

A.1 Esquemáticos e Layout da DUT Interface PCB

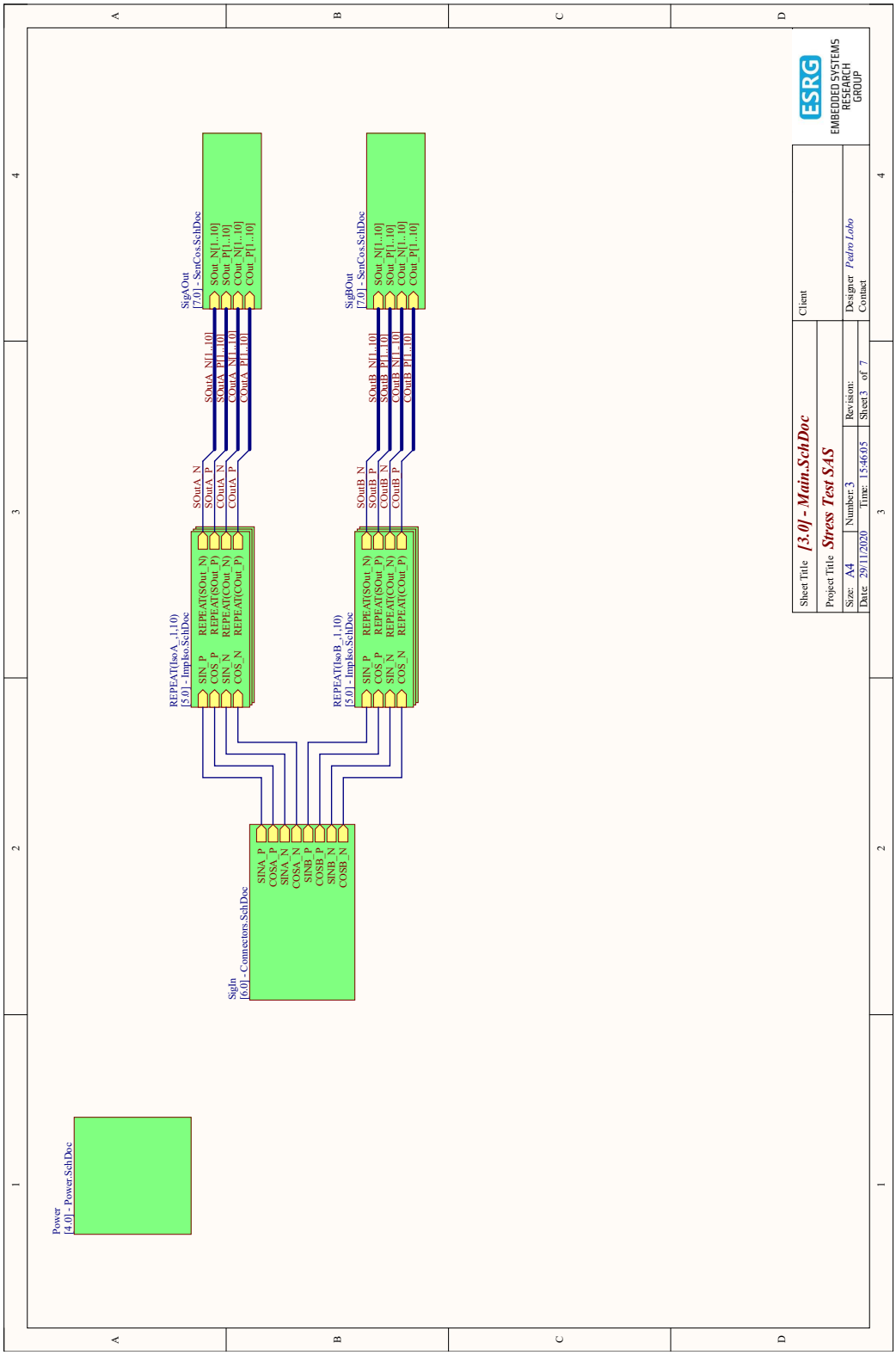


Figura A.1: *Overview* dos esquemáticos da *DUT interface*.

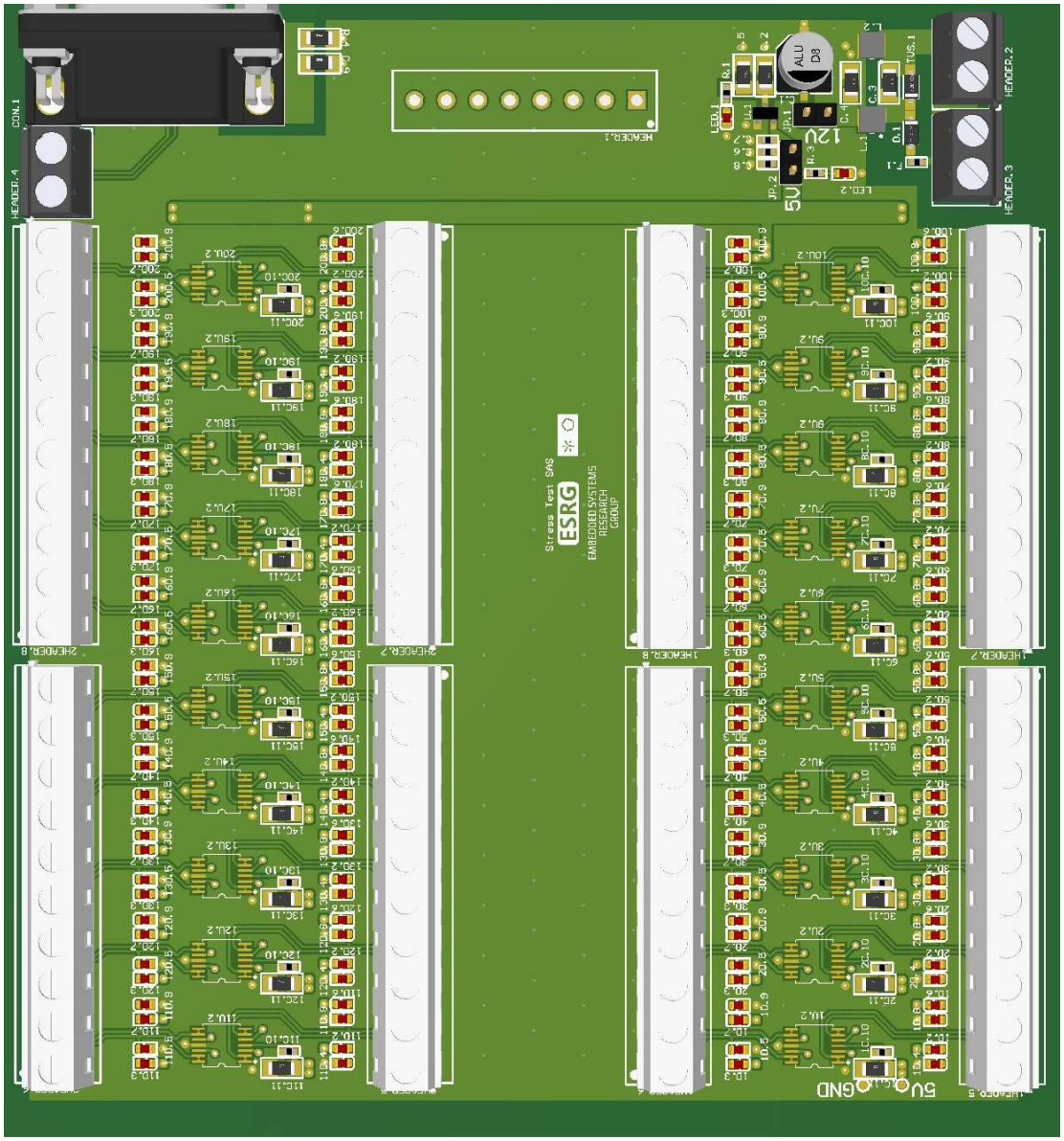


Figura A.2: PCB da *DUT interface* vista de cima.

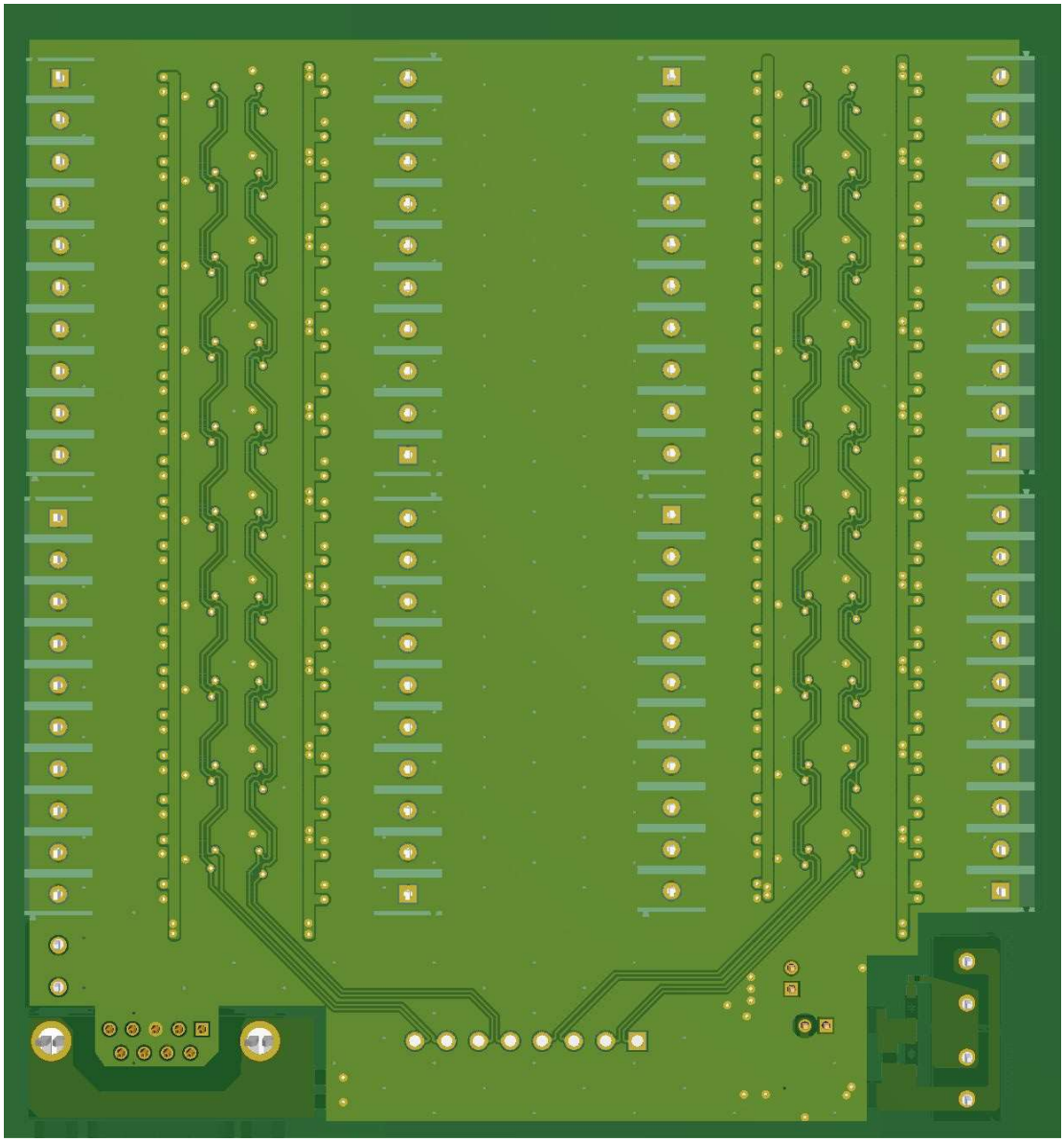


Figura A.3: PCB da *DUT interface* vista de baixo.

A.2 Diagramas Auxiliares do Data Collection Service

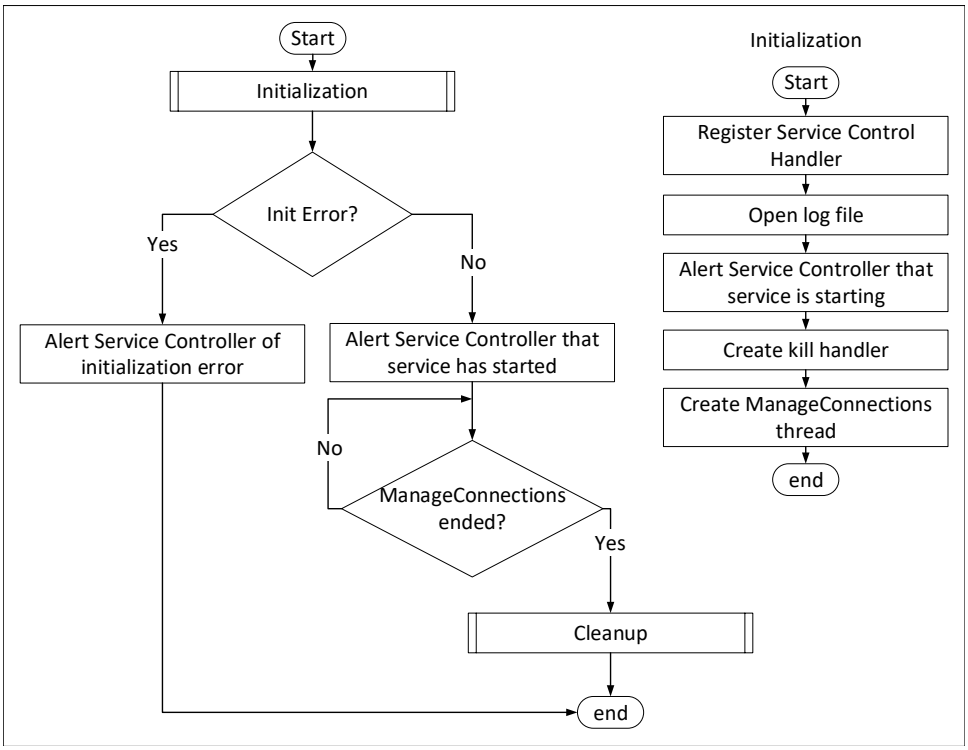


Figura A.4: Fluxograma da *main task*.

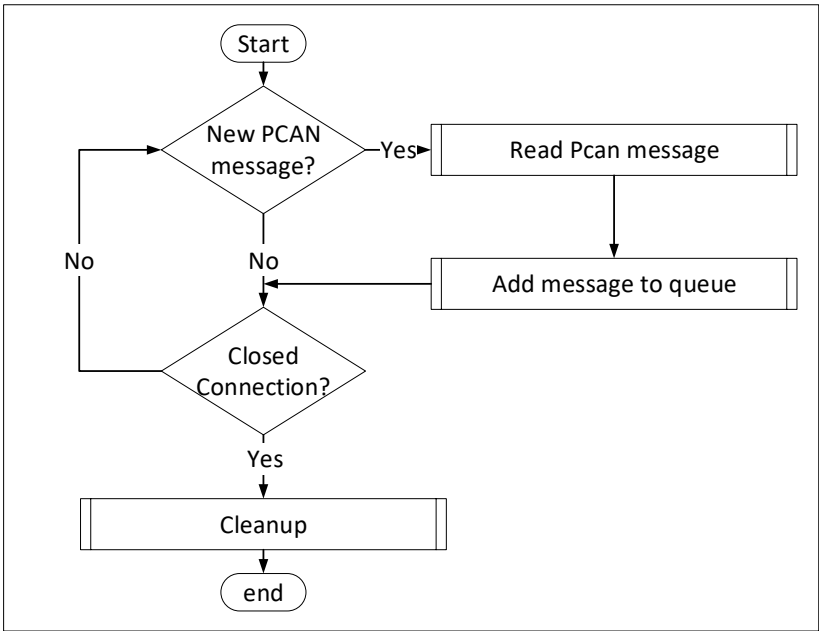


Figura A.5: Fluxograma da *read task*.

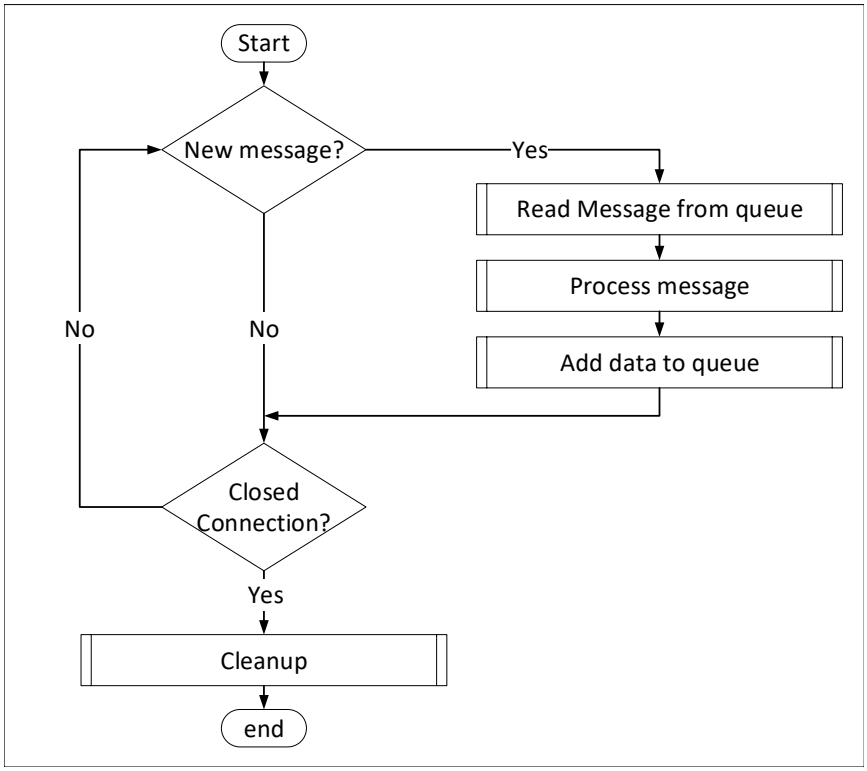


Figura A.6: Fluxograma da *process task*.

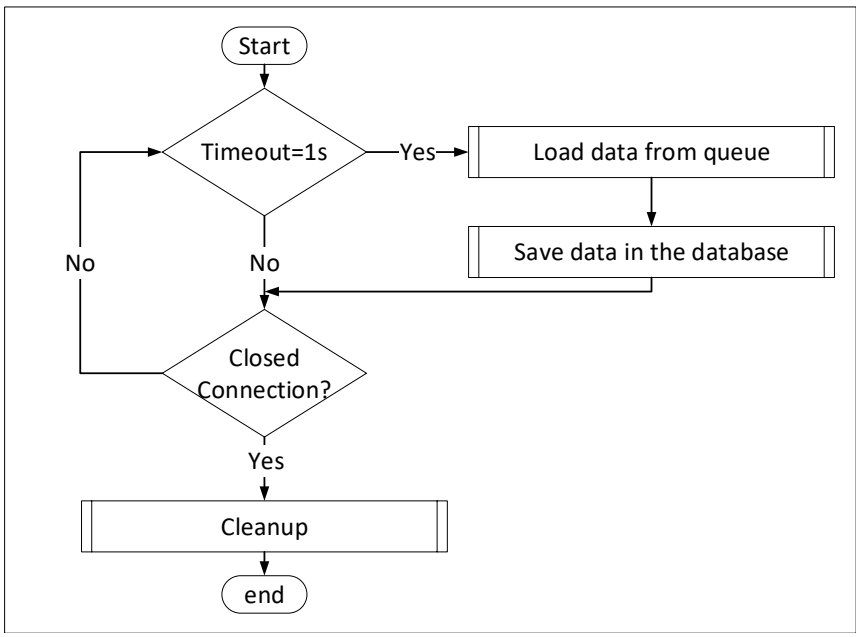


Figura A.7: Fluxograma da *store task*.

A.3 Diagramas Auxiliares da Aplicação Gráfica

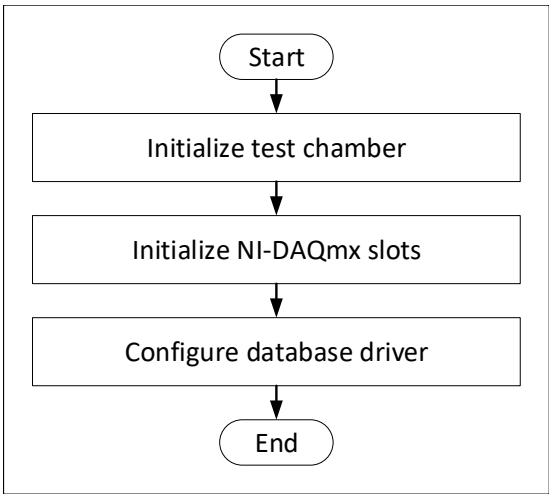


Figura A.8: Fluxograma do estado *initialization*.

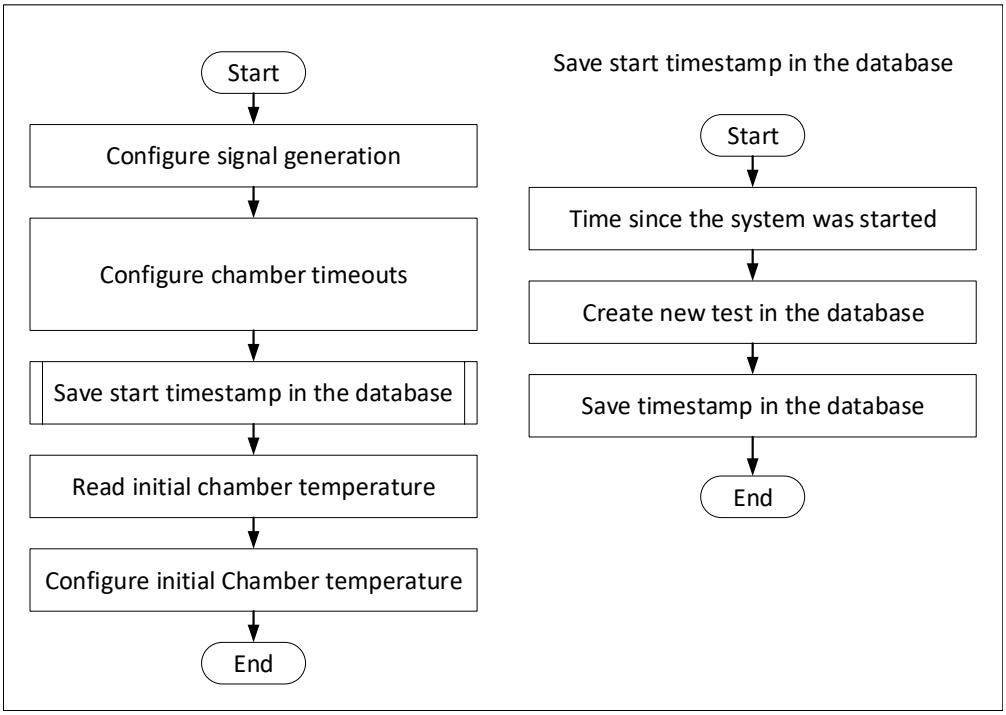


Figura A.9: Fluxograma do estado *test initialization*.

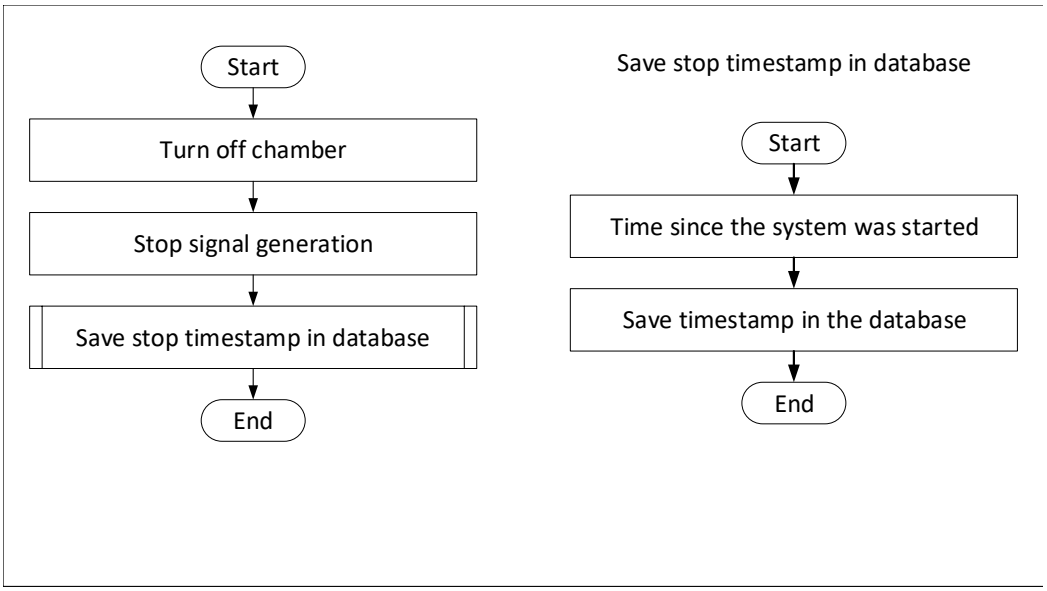


Figura A.10: Fluxograma do estado *test shutdown*.

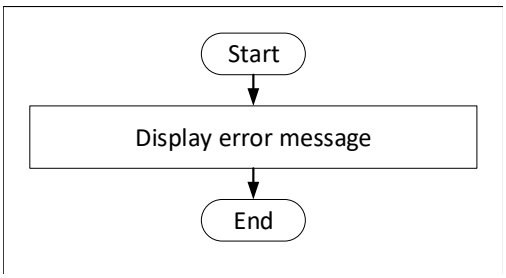


Figura A.11: Fluxograma do estado *aborting test*.

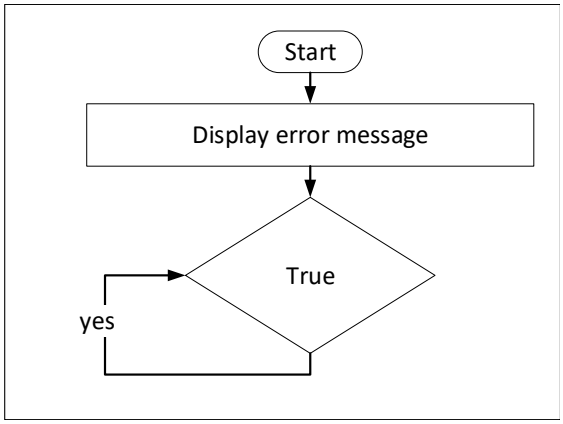


Figura A.12: Fluxograma do estado *initialization error*

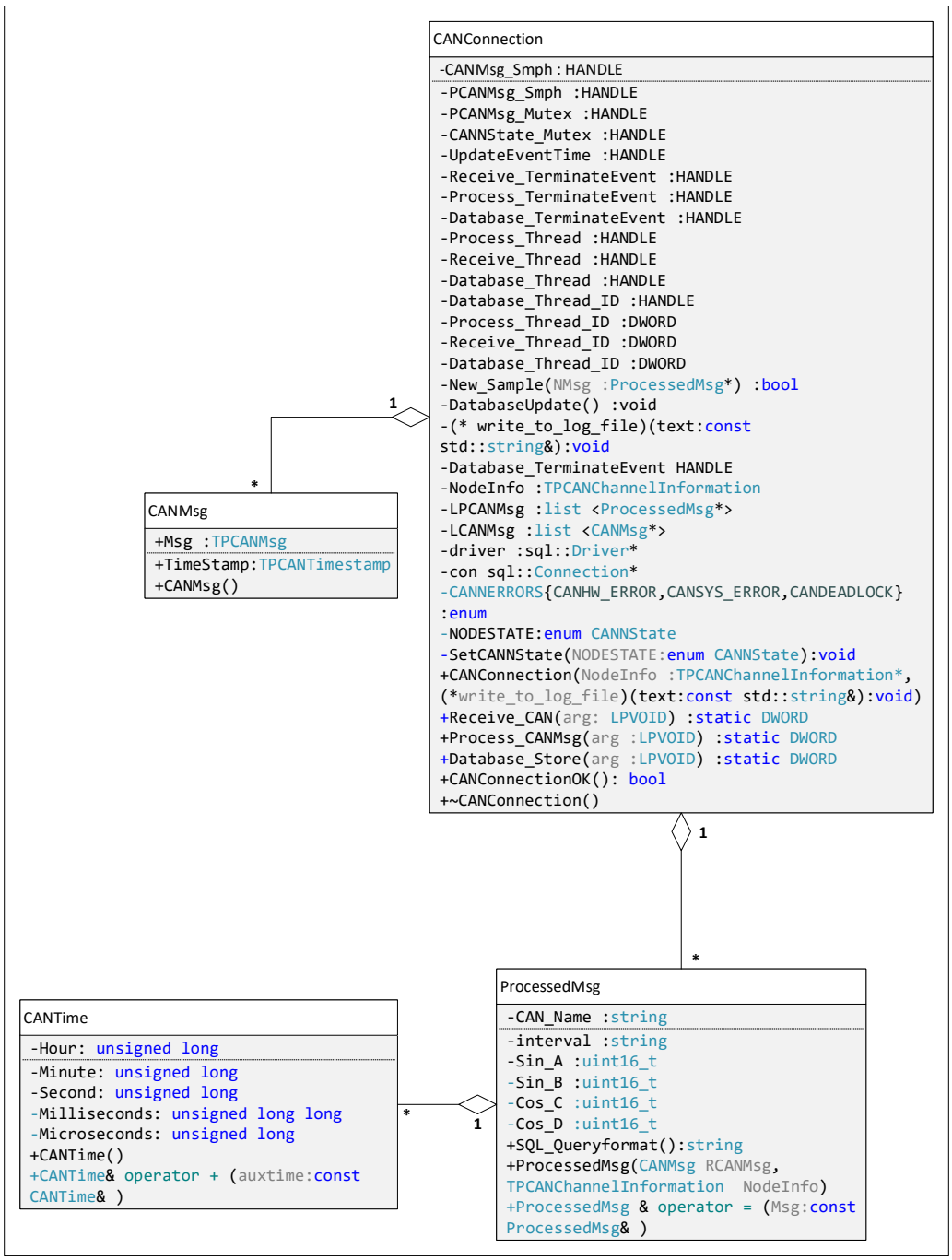


Figura A.13: Diagrama de classes de uma conexão CAN.

Anexo B

Desenvolvimento do Steering Angle Sensor Material Auxiliar

B.1 Diagrama de Máquina de Estados

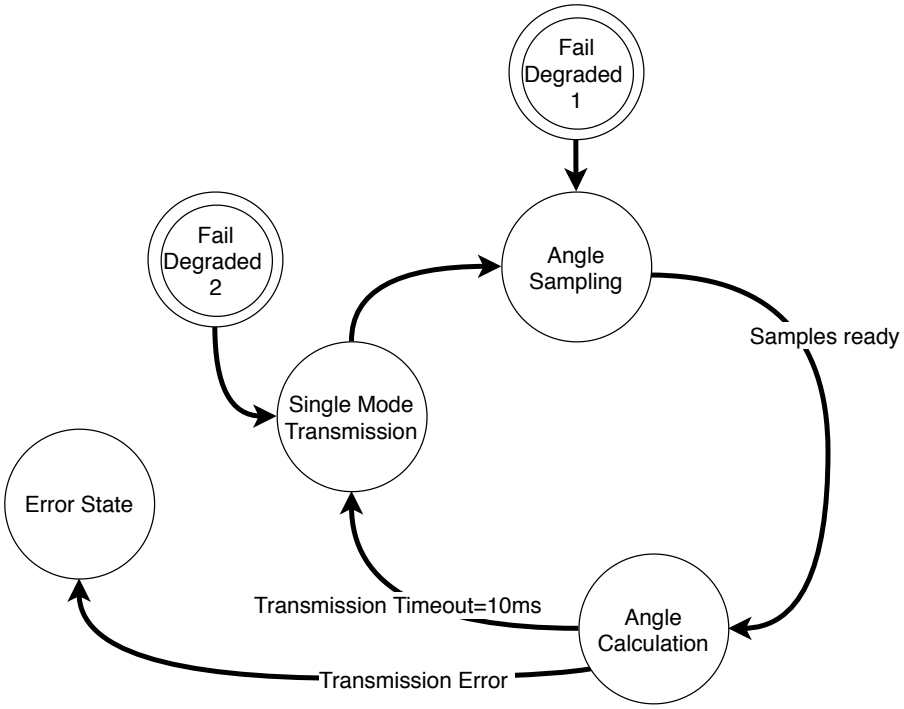


Figura B.1: Máquina de estados do SAS operação em *fail-degraded*.

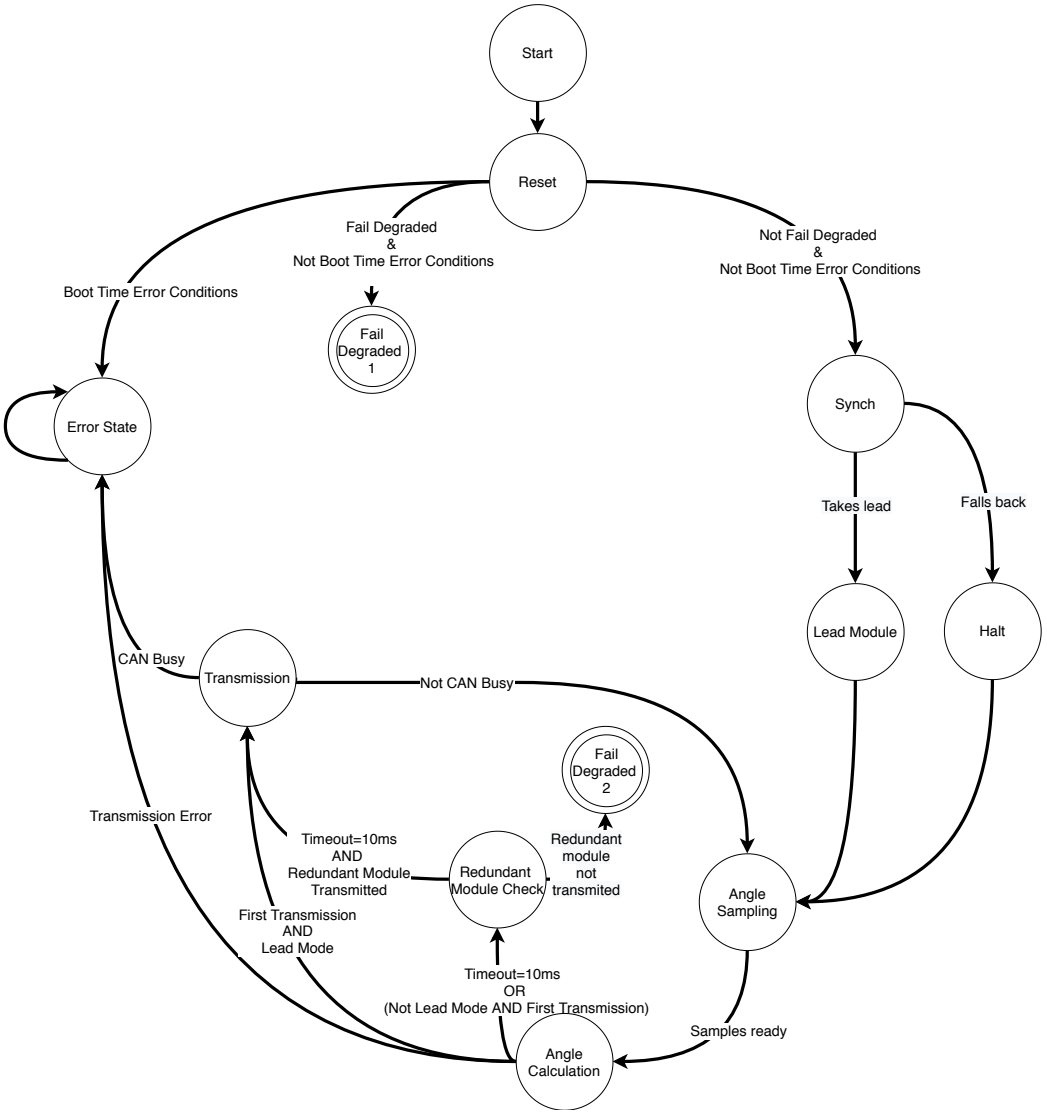


Figura B.2: Máquina de estados do SAS operação normal.

B.2 SAS Esquemáticos e Layout

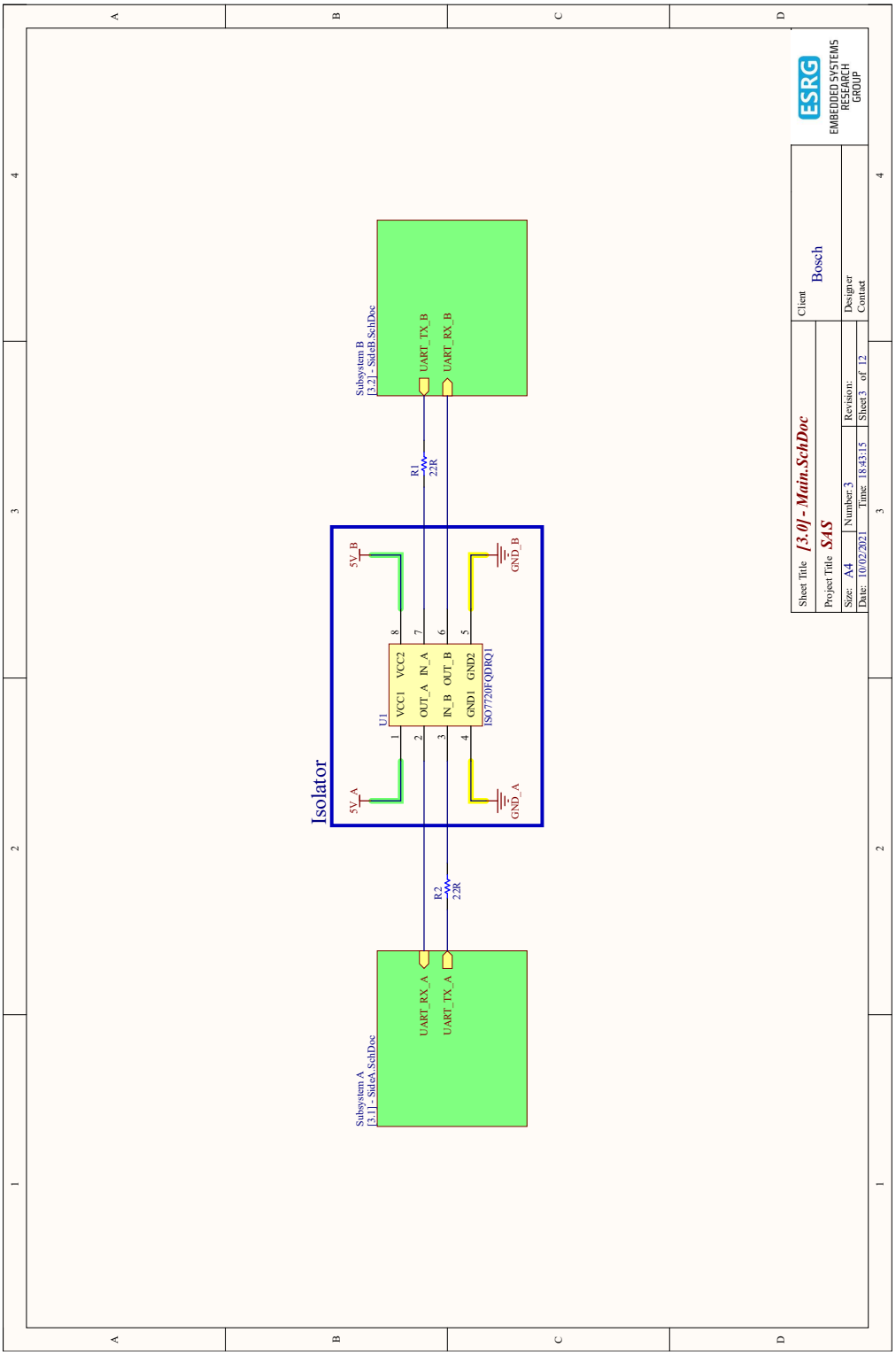
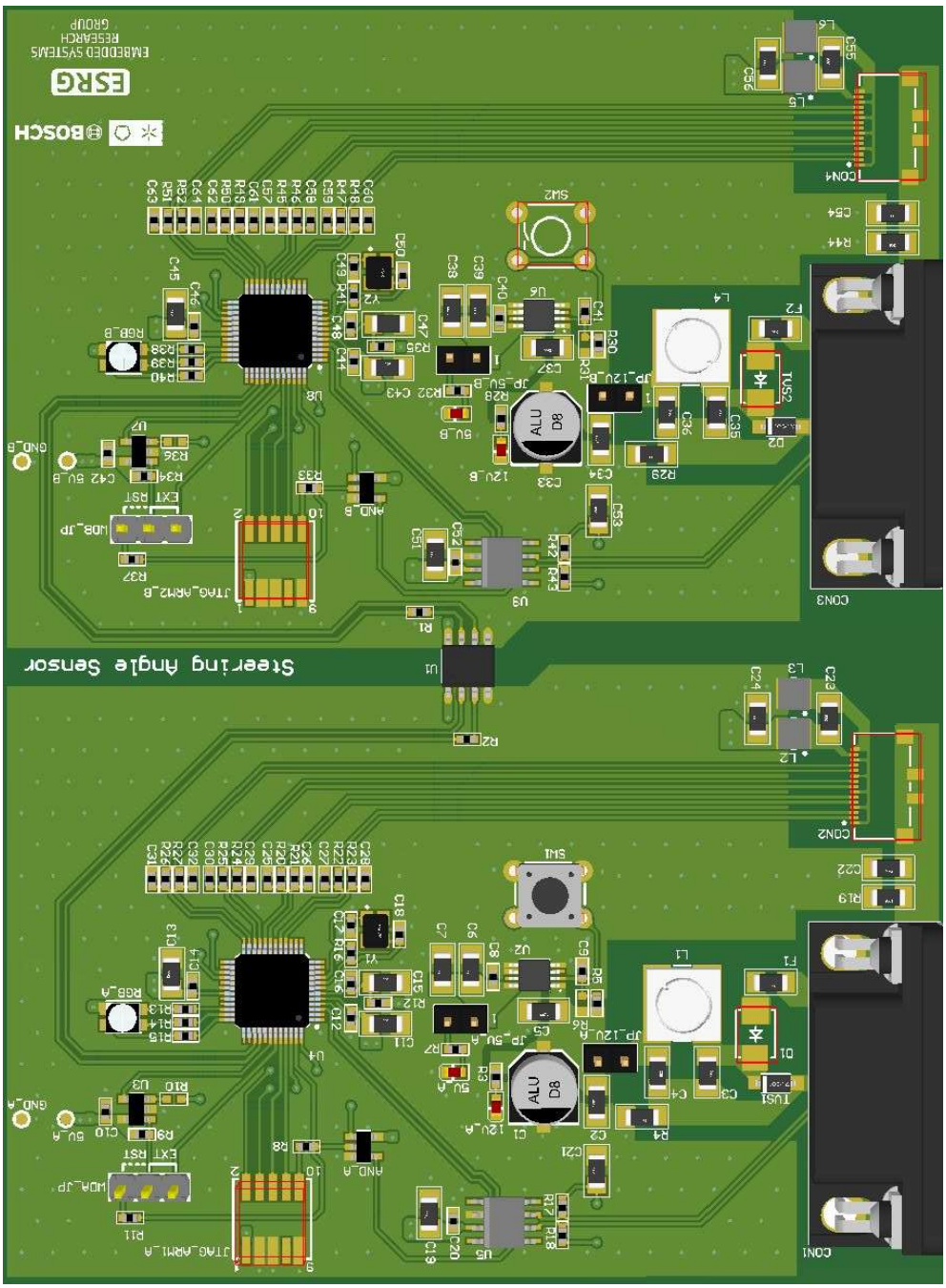
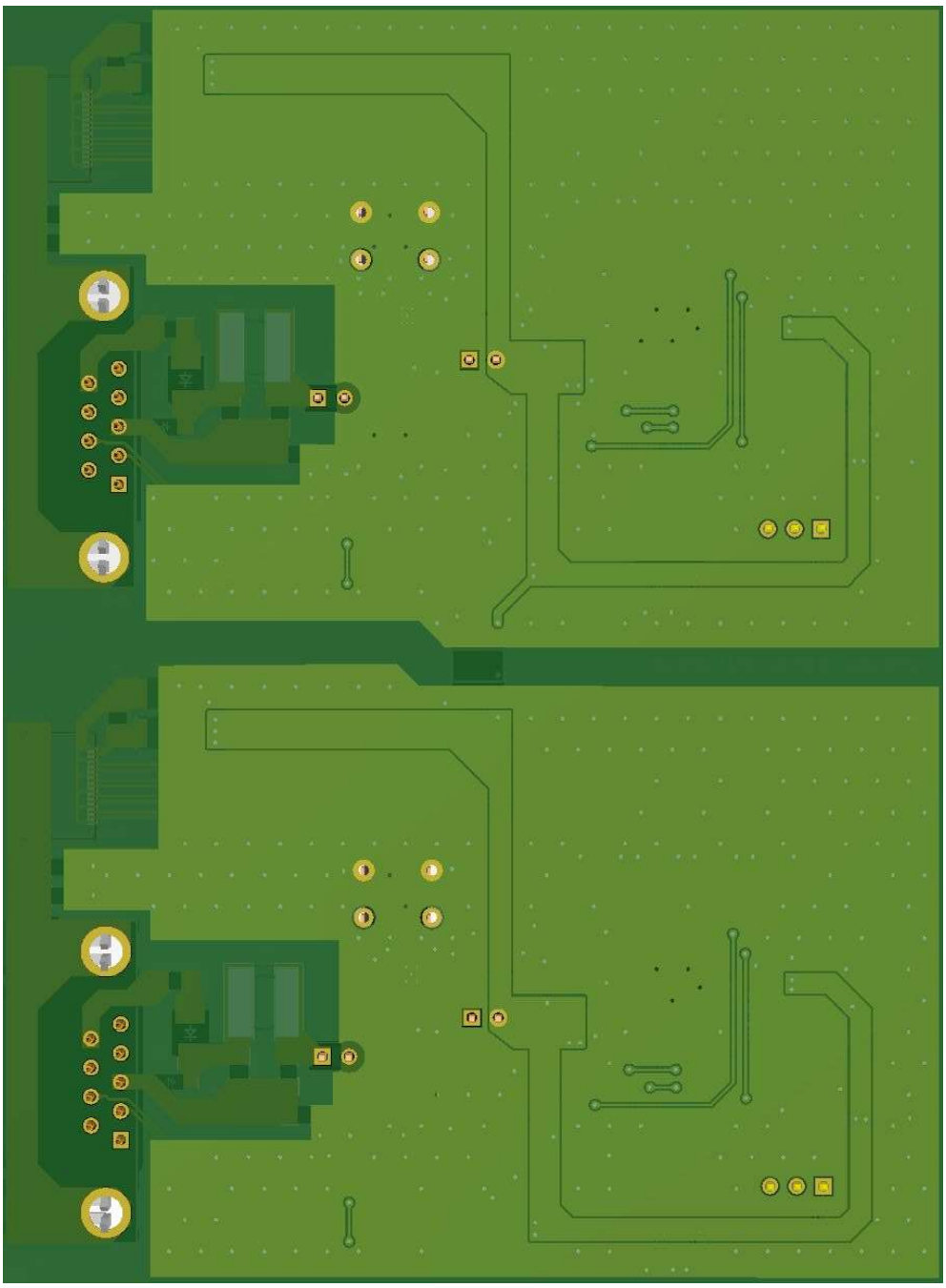


Figura B.3: *Overview* dos esquemáticos do SAS.


Steering Angle Sensor
ESRG
BOSCH
EMBEDDED SYSTEMS RESEARCH GROUP
CON1
CON2
CON3
CON4
JTAG_ARM1_A
JTAG_ARM2_B
U1
U2
U3
U4
U5
U6
U7
U8
ALU
D8



B.3 Modelo de Simulação do SAS

Component	FailureRate	MTBF	MTBFs redimensionados
Fusível [6]	$1{,}00\text{x}10^{-08}$	$1{,}00\text{x}10^{08}$	$1{,}00\text{x}10^{04}$
Diodo [6]	$3{,}00\text{x}10^{-09}$	$3{,}33\text{x}10^{08}$	$3{,}33\text{x}10^{04}$
TVS [6]	$1{,}30\text{x}10^{-09}$	$7{,}69\text{x}10^{08}$	$7{,}69\text{x}10^{04}$
Condensador [6]	$9{,}90\text{x}10^{-10}$	$1{,}01\text{x}10^{09}$	$1{,}01\text{x}10^{05}$
Bobine [6]	$3{,}00\text{x}10^{-11}$	$3{,}33\text{x}10^{10}$	$3{,}33\text{x}10^{06}$
Condensador eletrólitico [6]	$1{,}20\text{x}10^{-10}$	$8{,}33\text{x}10^{09}$	$8{,}33\text{x}10^{05}$
LDO [57]	$3{,}88\text{x}10^{-10}$	$2{,}58\text{x}10^{09}$	$2{,}58\text{x}10^{05}$
Resistência [6]	$3{,}79\text{x}10^{-09}$	$2{,}64\text{x}10^{08}$	$2{,}64\text{x}10^{04}$
Watchdog [57]	$1{,}00\text{x}10^{-10}$	$1{,}00\text{x}10^{10}$	$1{,}00\text{x}10^{06}$
Microcontrolador [57]	$2{,}41\text{x}10^{-09}$	$4{,}15\text{x}10^{08}$	$4{,}1\text{x}510^{04}$
Cristal externo [57]	$3{,}00\text{x}10^{-08}$	$3{,}33\text{x}10^{07}$	$3{,}33\text{x}10^{03}$
NAND [58]	$1{,}54\text{x}10^{-09}$	$6.48\text{x}10^{08}$	$6.48\text{x}10^{04}$
Transceiver CAN [57]	$1{,}81\text{x}10^{-09}$	$5{,}51\text{x}10^{08}$	$5{,}51\text{x}10^{04}$
Isolador capacitivo [57]	$4{,}0\text{x}10^{-10}$	$2{,}50\text{x}10^{09}$	$2{,}50\text{x}10^{05}$

Tabela B.1: MTBFs dos componentes usados para o modelo do SAS.

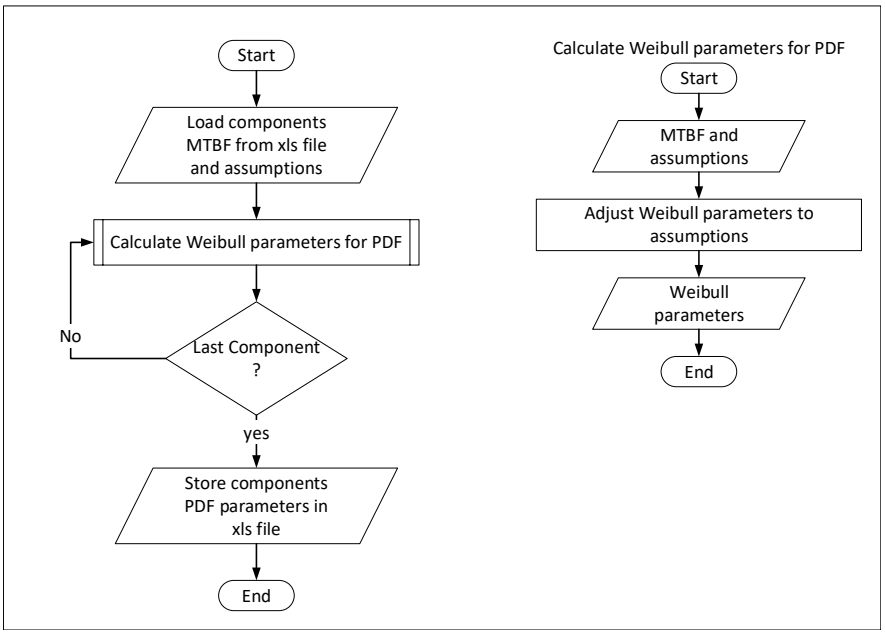


Figura B.6: Fluxograma do cálculo da função PDF.

B.4 Testes acelerados do SAS

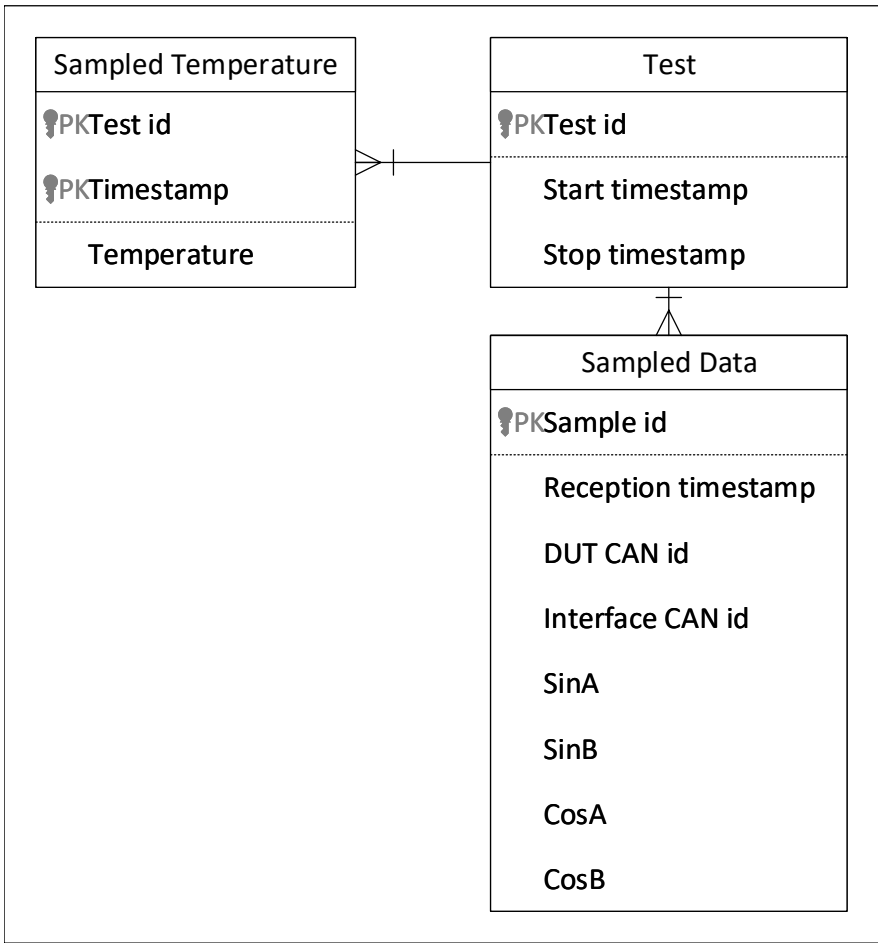


Figura B.7: Diagrama entidade relacionamento da base de dados utilizada para armazenamento dos dados recolhidos durante o teste aos 10 protótipos do SAS.

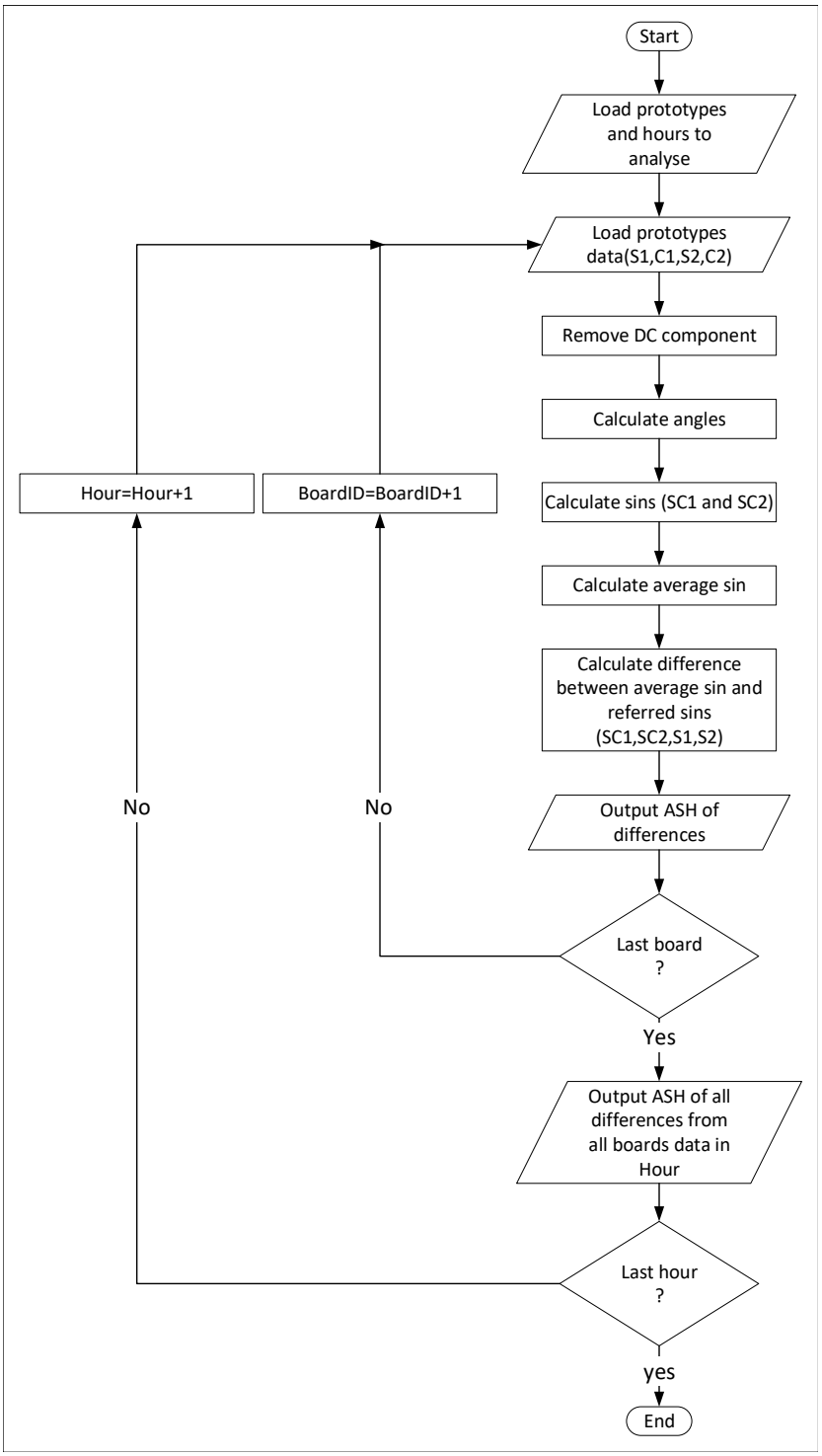


Figura B.8: Processamento dos dados dos testes acelerados.