UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
ESCOLA DE ENGENHARIA
DEPARTAMENTO DE ENGENHARIA ELÉTRICA

João Guilherme Metinger

MITIGAÇÃO DE ERROS EM CADEIAS DE MEDIÇÕES DE TEMPERATURA UTILIZANDO INTERFACES ANALÓGICO-DIGITAIS COM REDUNDÂNCIA E DIVERSIDADE

Porto Alegre

2018

João Guilherme Metinger

Mitigação de Erros em Cadeias de Medições de Temperatura Utilizando Interfaces Analógico-Digitais com Redundância e Diversidade

Trabalho de conclusão de curso, apresentado como requisito parcial para a obtenção de grau de Bacharel em Engenharia Elétrica pela Universidade Federal do Rio Grande do Sul.

Orientador: Tiago Roberto Balen

Porto Alegre

2018

João Guilherme Metinger

Mitigação de Erros em Cadeias de Medições de Temperatura Utilizando Interfaces Analógico-Digitais com Redundância e Diversidade

Este Projeto de Diplomação foi analisado e julgado adequado para a obtenção do título de Bacharel em Engenharia Elétrica e aprovado em sua forma final pelo Orientador e pela Banca Examinadora, designada pelo Departamento de Engenharia Elétrica da Universidade Federal do Rio Grande do Sul.

_____

Prof. Dr. Tiago Roberto Balen

_____

Prof. Dr. Aly Ferreira Flores Filho

BANCA EXAMINADORA:

Prof. Dr. Gilso Inácio Wirth – UFRGS _____

Prof. Dr. Ivan Muller – UFRGS _____

Resumo

Medições de temperatura são uma função crítica para desempenho e manutenção de motores utilizados na aviação. Dentro destes sistemas de medições de temperatura, existem duas principais responsáveis pela identificação da saúde do motor: A temperatura do gás de exaustão (EGT) e a temperatura da cabeça do cilindro (CHT). Entretanto, estas medições realizadas através de termopares necessitam condicionamento e conversões analógico-digitais em sua cadeia de medição e instrumentação, para posterior leitura das informações pelo computador de vôo. Os componentes integrados de tecnologia CMOS, comumente usados em aplicações eletrônicas nos dias de hoje e comumente presente nos conversores ADs, podem apresentar uma quantidade significativa de defeitos, geralmente associadas as condições ambientais e ao envelhecimento do componente. Algumas fontes de falhas são a quebra do óxido fino, eletromigração, transientes causados por radiação, interferência eletromagnética, bem como outros problemas causados por uma série de fatores como estresse mecânico ou corrosão, por exemplo. Uma das técnicas em nível de sistema amplamente utilizada para mitigação de tais efeitos é a Redundância Modular Tripla (TMR*, Triple Modular Redundancy*), que pode ser melhorada através de técnicas de diversidade e também com a aquisição de dados redundante do sensor no sistema de medição. Este trabalho aborda um sistema de medição de temperatura com um sensor termopar tipo K, apresenta um sistema de amplificação de instrumentação, filtro ativo para remoção de ruídos, e um sistema de aquisição de dados com TMR a nível de conversão analógica-digital. Além disto, foi utilizado um SoC (System-on-Chip) da Cypress Semiconductor, chamado PSoC 5LP, para emulação de falhas através do mascaramento da informação lida dos A/Ds. O resultado da emulação de falhas foi a observância de um baixo fator de erros, mostrando a eficácia do sistema baseado em TMR e diversidade espacial-temporal. Todas as falhas simples foram toleradas pelo sistema, e além disto, também mostraram que a adição de diversidade temporal gera, em comparação ao TMR clássico, um ganho significativo na tolerância a falhas duplas e múltiplas. Um baixo percentual das falhas é detectado na forma de erros, tanto em função da resiliência a falhas pela arquitetura do sistema proposto, como pela tolerância a baixas variações de temperatura na aplicação, não alterando as condições de leitura sobre a saúde do sistema monitorado.

Palavras-chave: Instrumentação; *Triple Modular Redundancy* (TMR); Reundância e diversidade; Conversores analógico-digitais; *Exhaust Gas Temperature* (EGT).

Abstract

Temperature measurements are a critical function for performance and maintenance of engines used in aviation. Within these temperature measurement systems, there are two main factors responsible for identifying the health of the engine: The exhaust gas temperature (EGT) and cylinder head temperature (CHT). However, these measurements using thermocouples require conditioning and analog-digital conversions in their measurement and instrumentation chain, for later reading of the information by the flight computer. Integrated components of CMOS technology, commonly used in today's electronic applications and commonly present in ADs, can exhibit a significant quantity of defects, generally associated with environmental conditions and aging of the component. Some sources of failures are thin oxide, electromigration, transients caused by radiation, electromagnetic interference, as well as other problems caused by a number of factors such as mechanical stress or corrosion, for example. One of the system-wide techniques widely used to mitigate such effects is TMR (Triple Modular Redundancy), which can be improved through diversity techniques and also with the acquisition of redundant sensor data in the measurement system . This work addresses a temperature measurement system with a K-type thermocouple sensor, features an instrumentation amplification system, active noise removal filter, and a TMR data acquisition system at the analog-to-digital conversion level. In addition, a Cypress Semiconductor System-on-Chip, called PSoC 5LP, was used for emulation of faults by masking the information read from the A / Ds. The result of the emulation of failures was the observance of a low error factor, showing the effectiveness of the system based on TMR and spatial-temporal diversity. All simple failures were tolerated by the system, and in addition, they also showed that the addition of temporal diversity generates, in comparison to the classic TMR, a significant gain in the tolerance of double and multiple faults. A low percentage of failures is detected in the form of errors, both due to the resilience to failures by the proposed system architecture, and also by the tolerance to low temperature variations in the application, without altering the reading conditions on the health of the monitored system.

Keywords: Instrumentation; *Triple Modular Redundancy* (TMR); Redundancy and diversity; Analog-Digital converters; Exhaust Gas Temperature (EGT);

Lista de abreviaturas e siglas

ADC	*Analog to Digital Converter*
CHT	*Cylinder Head Temperature*
CCD	*Charge Coupled Devices*
CI	Circuito Integrado
CMOS	*Complementary Metal-Oxide Semicondutor*
DD	*Displacement Damage*
DMA	*Direct Memory Access*
DUT	*Device Under Test*
EHM	*Engine Health Monitoring*
EICAS	*Engine-Indicating and Crew-Alerting System*
EGT	*Exhaust Gas Temperature*
HCI	*Hot Carrier Injection*
IHM	*Interface Homem-Máquina*
IO	*Input-Output*
LRU	*Line-Replaceable Unit*
MBU	*Multiple Bit Upset*
MCU	*Multiple Cell Upset*
MOS	*Metal-Oxide-Semiconductor*
MOSFET	*Metal-Oxide-Semiconductor Field Effect Transistor*
PSoC	*Programmable System-on-Chip*
RCG	Raios Cósmicos Galáticos
SAA	*South Atlantic Anomaly*
SAD	Sistema de Aquisição de Dados
SAR	*Successive Approximation Register*
SBU	*Single Bit Upset*
SEB	*Single Event Burnout*
SEE	*Single Event Effects*
SEFI	*Single Event Functional Interrupt*
SEGR	*Single Event Gate Rupture*
SEL	*Single Event Latch up*
SER	*Soft Error Rate*
SESB	*Single Event Induced Snap-Back*
SET	*Single Event Transient*
SEU	*Single Event Upset*
SHE	*Single Hard Error*
SoC	*System-on-Chip*
SPI	*Serial Peripheral Interface*
SRU	*Shop-Replaceable Unit*
TID	*Total Ionization Dose*
TIT	*Turbine Inlet Temperature*
TMR	*Triple Modular Redundancy*

Lista de Ilustrações

Sumário

1 INTRODUÇÃO

Sistemas e circuitos eletrônicos de aeronaves, sejam elas militares ou comerciais, são compostos de diversos sistemas de medições e sensores destinados a obtenção de melhor performance do avião aliado a proteção dos equipamentos de vôo. Dentro destes sistemas de instrumentação destaca-se o *Engine Health Monitoring* (EHM), fundamental para identificação de problemas no motor e de prover dados para manutenção, que contém entre outros parâmetros, duas medições de temperatura utilizando termopares: A temperatura do gás de exaustão (EGT, *Exhaust Gas Temperature*) e a temperatura da cabeça do cilindro (CHT, *Cylinder Head Temperature*) (ORTIZ, 2009). Este tipo de medição geralmente contém mostradores analógicos conectados diretamente aos sensores, que trabalham como interface homem-máquina (IHM), mas cada vez mais atualmente as aeronaves são modernizadas e contém diversos computadores de missão e displays eletrônicos, que demandam que os sinais de medição de temperatura sejam condicionados, processados e convertidos, antes de serem mostrados ao operador final. Dado a criticidade da operação, existem redundâncias em diversos níveis para a mitigação de problemas.

Entretanto, esses sistemas de aquisição e conversão de dados estão sujeitos a diversas fontes de degradação e distúrbios ambientais que podem produzir efeitos indesejáveis, resultando em erros e falhas (BALEN, 2010). Considerando as características da tecnologia CMOS, utilizada no condicionamento e processamento de sinais, uma quantidade significativa de erros podem surgir durante a vida útil do circuito, que geralmente estão associadas ao envelhecimento do componente e as condições ambientais (CHENET, 2015). Alguns exemplos são a quebra de óxido fino, causada por grandes campos elétricos no isolador (SALIVA, 2014); eletromigração - a deriva de átomos de metal devido a alta densidade de corrente nas linhas de metal (KLUDT, 2014); injeção de transportadora de alta energia (HCI, *Hot Carrier Injection*), no óxido causado por campos elétricos grandes no canal (TAKEDA, 1983); falhas permanente ou transientes produzidas pela radiação, tais como partículas alfa, raios cósmicos e reações secundárias de neutrons (FERLET-CAVROIS, 2013); atividade elétrica anormal causada por interferências eletromagnéticas (WAN, 2011), bem como descargas eletrostáticas (LIOU, 2012) e outros defeitos causados por um conjunto de fatores, como estresse mecânico e corrosão.

Uma das técnicas em nível de sistema amplamente utilizada para mitigar estes efeitos nos circuitos eletrônicos é a Redundância Modular Tripla (TMR, *Triple Modular Redundacy),* que é melhorada com a adição de técnicas de diversidade (BECKER, 2017). Neste contexto, este projeto apresenta uma topologia baseada nestas duas técnicas na implementação de um sistema de aquisição de dados (SAD) analógico-digital. O estudo ainda conta com um SoC (*System-On-Chip*) da Cypress Semiconductor, chamado de PSoC (*Programmable System-On-Chip*) para leitura, votação dos sinais e injeção de falha por simulação.

Esse trabalho é organizado da seguinte forma: a seção 2 fornece dados básicos das turbinas mais comumente utilizadas na aviação; a seção 3 apresenta as informações principais da aplicação foco do trabalho, como a localização e posicionamento dos sensores, noções de sistemas aviônicos existentes que fazem a leitura destes sinais de temperatura e como ele é mostrado ao operador do equipamento; a seção 4 apresenta

as fontes da radiação especial, com noções dos efeitos de TID e DD e discute os principais SEEs, tanto os causadores de soft errors quanto os causadores de hard errors; a seção 5 mostra as principais técnicas de proteção dos circuitos eletrônicos aos soft errors, incluindo as técnicas em nível de dispositivo, circuito e sistema; a seção 6 apresenta o sistema de aquisição de dados proposto com um estudo de injeções de falhas por emulação; a seão 7 apresenta a validação deste sistema; a seção 8 apresenta os resultados e as considerações finais; e finalmente a seção 9 apresenta as conclusões deste projeto de diplomação.

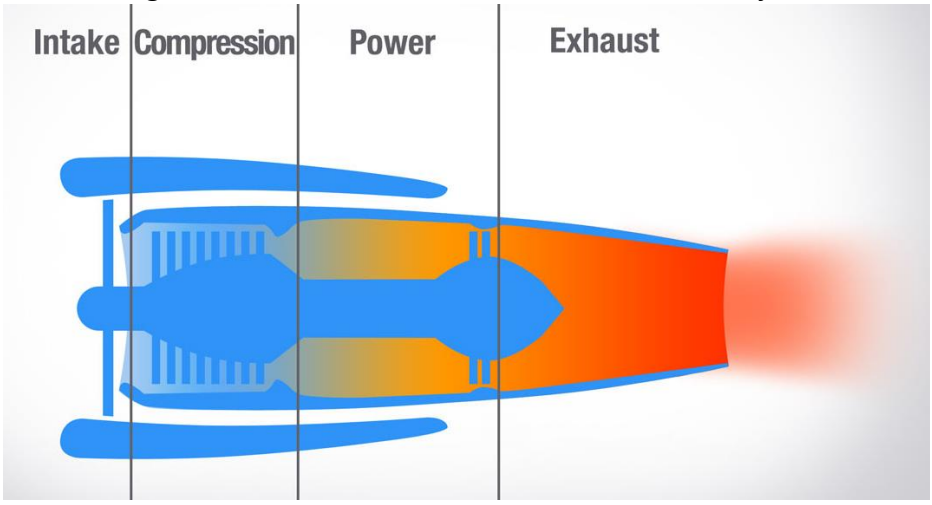
2 MOTORES DE REAÇÃO AERONÁUTICOS

Motores de reação são amplamente usados em aeronaves, gerando o impulso para o deslocamento, além de expulsar os gases de escape em alta velocidade do motor. Os tipos de motores de propulsão de reação mais comuns são turbojatos, *turbofans* e foguetes, e dado que atualmente a maioria das aeronaves comerciais e militares utiliza turbojatos e turbofans, este trabalho de pesquisa irá dar enfoque a aplicações de medições de temperatura nestes dois tipos de motores.

2.1 Turbofan

Motores a jato, também chamados de turbinas a gás, são o tipo motor mais presente em aviãoes de linha atualmente (Glenn Research Center, 2015). Eles funcionam sugando ar para a frente do motor usando um ventilador. A partir daí, o motor comprime o ar, mistura combustível com ele, inflama esta mistura e a joga para a parte de trás do motor, criando empuxo. A Figura 1 mostra um panorama geral deste processo.

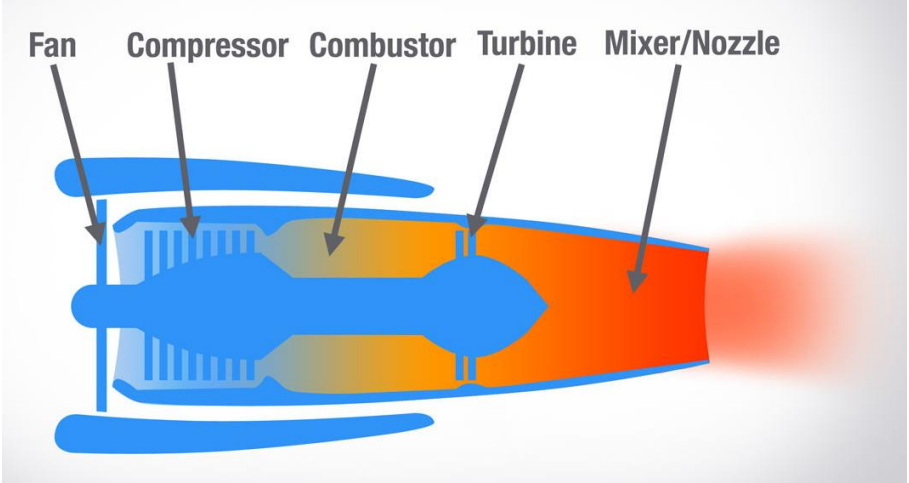
Figura 1 – Funcionamento básico de um motor a jato



Fonte: Adaptado de Glenn Research Center – NASA, 2015.

O motor é basicamente composto por cinco partes: Ventilador, compressor, combustor, turbina e o misturador, que são mostrados na Figura 2.

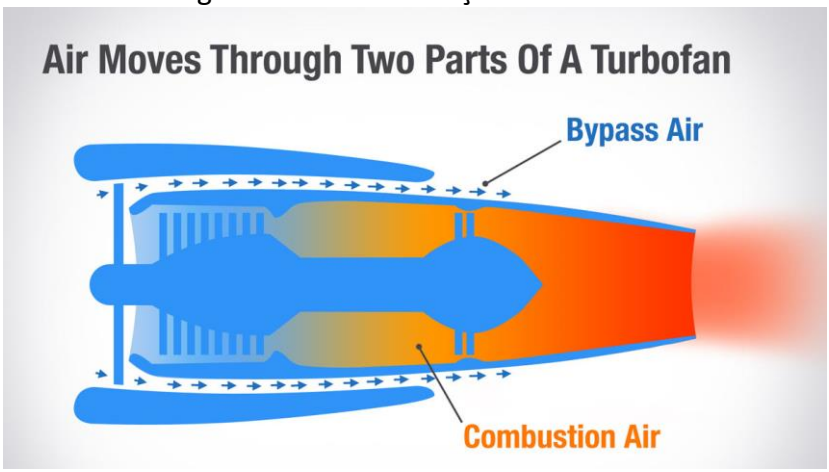
Figura 2 – Partes de um motor a jato.



Fonte: Glenn Research Center – NASA, 2015.

O ventilador, que quase sempre é feito de lâminas de titânio, suga grandes quantidades de ar para o motor, que se move através de duas partes. Parte do ar é direcionado ao núcleo do motor, onde a combustão ocorrerá. O resto do ar, chamado de *bypass air*, é movimentado em torno do exterior do núcleo do motor através de um duto. Esse ar de desvio cria impulso adicional, resfria o motor e torna o motor mais silencioso, cobrindo o ar de escape que sai do motor, conforme ilustrado na Figura 3. Nos turbofans modernos de hoje, o *bypass air* produz a maior parte do empuxo do motor.

Figura 3 – Movimentação do ar na turbina.

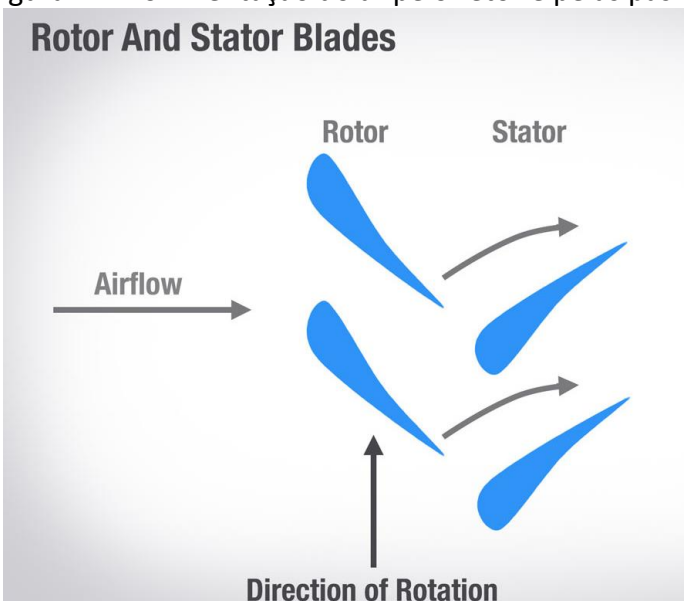


Fonte: Glenn Research Center – NASA, 2015.

O compressor está localizado na primeira parte do núcleo do motor. Ele usa uma série de pás giratórias aerodinâmicas para acelerar e comprimir o ar. É chamado de fluxo axial, porque o ar passa pelo motor em uma direção paralela ao eixo do motor (em oposição ao fluxo centrífugo). Conforme o ar se move através do compressor, cada conjunto de lâminas é um pouco menor, adicionando mais energia e compressão ao ar.

Entre cada conjunto de lâminas de compressor estão lâminas sem movimento de aerofólio chamadas "estatores". Esses estatores (que também são chamados de palhetas) aumentam a pressão do ar convertendo a energia rotacional em pressão estática. Os estatores também preparam o ar para entrar no próximo conjunto de lâminas rotativas. Em outras palavras, eles "endireitam" o fluxo de ar. A Figura 4 mostra o direcionamento deste fluxo de ar pelo compressor.

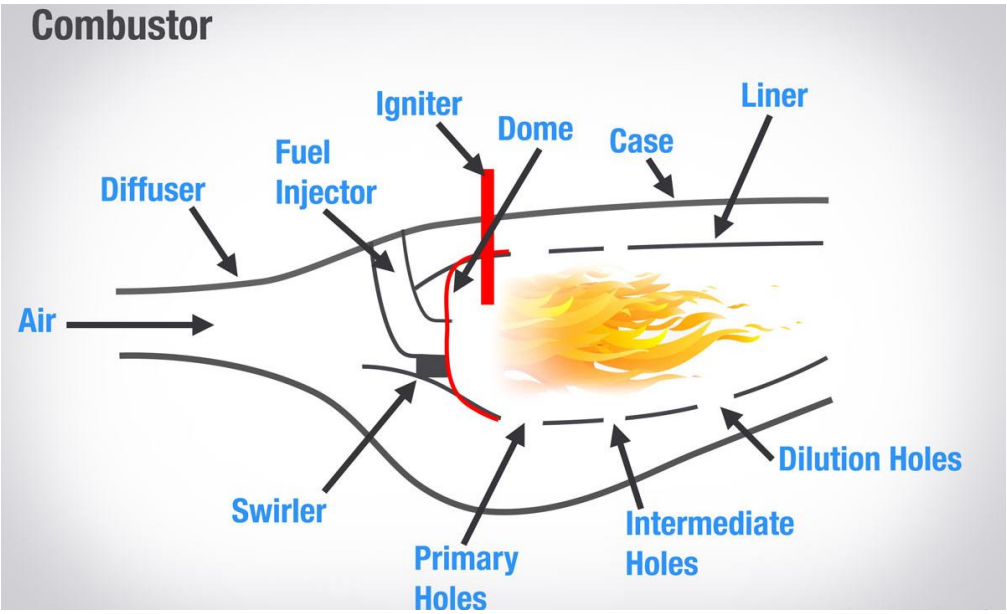
Figura 4 - Movimentação do ar pelo retor e pelas pás.



Fonte: Glenn Research Center – NASA, 2015.

O combustor é o local onde o fogo inicia. Quando o ar sai do compressor e entra no combustor, ele é misturado com combustível e é aceso. Este estágio necessita manter uma combustão estável da mistura combustível-ar,o que pode ser complicado dado que o ar está se movendo através do combustor a uma taxa extremamente rápida. A caixa contém todas as partes do combustor e, dentro dele, o difusor é a primeira parte que funciona. O difusor retarda o ar do compressor, facilitando a ignição. A cúpula e o redemoinho adicionam turbulência ao ar para que ele possa se misturar mais facilmente com o combustível. O injetor de combustível borrifa combustível no ar, criando uma mistura combustível-ar que pode ser acessa. O avião de carreira tem várias entradas, permitindo que o ar entre em vários pontos na zona de combustão. A última parte principal é o ignitor, que é muito semelhante às velas de ignição de um carro convencional. Uma vez que o acendedor acende o fogo, ele é auto-sustentável, e o acendedor é desligado (embora seja freqüentemente usado como reserva em condições ruins de tempo e em caso de gelo). Todas estas etapas são detalhadas na Figura 5.

Figura 5 – Detalhamento do combustor.



Fonte: Glenn Research Center – NASA, 2015.

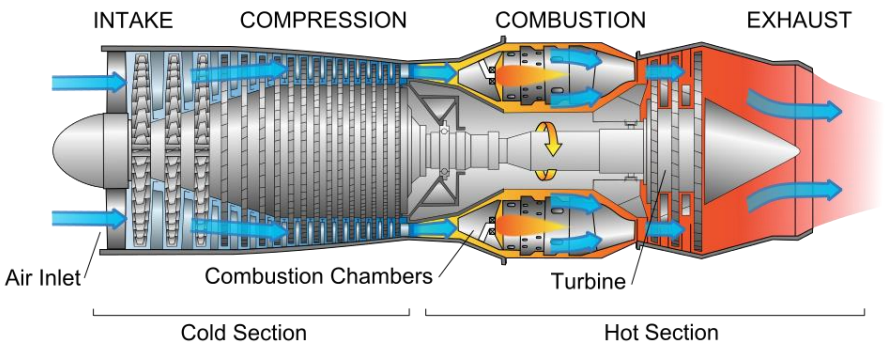
Uma vez que o ar atravessa o combustor, ele flui através da turbina. A turbina é uma série de pás em forma de aerofólio que são muito semelhantes às pás do compressor. À medida que o ar quente e de alta velocidade flui pelas pás da turbina, eles extraem energia do ar, girando a turbina em torno de um círculo e girando o eixo do motor ao qual ela está conectada. Este é o mesmo eixo ao qual o ventilador e o compressor estão conectados, portanto, girando a turbina, o ventilador e o compressor na frente do motor continuam sugando mais ar, que em breve será misturado com combustível e queimado.

A última etapa do processo acontece no bocal. O bocal é essencialmente o duto de exaustão do motor, e é onde o ar de alta velocidade dispara pelas costas. Em alguns motores, há um misturador no bocal de escape também, que mistura parte do ar de desvio que flui ao redor do motor com o ar quente e queimado, tornando o motor mais silencioso.

2.2 Turbojet

Diferente de um turbofan, o turbojet não possui um ventilador para captação do ar, portanto toda entrada de ar é feita diretamente pelo tubo que fica localizado na frente do motor. Além disto, todo ar entra no compressor e segue o processo de explosão, sem conter o *bypass air* característico do turbofan. A Figura 6 mostra um turbojet típico e seus elementos.

Figura 6 – Partes constituintes de um turbojet.



Fonte: CAST SAFETY ORG, 2018.

O principal de funcionamento, entretanto, é o mesmo: O ar é introduzido no compressor giratório através da entrada e comprimido a uma pressão superior antes de entrar na câmara de combustão. O combustível é misturado com o ar comprimido e inflamado por uma faísca. Este processo de combustão aumenta significativamente a temperatura do gás. Os produtos quentes da combustão que saem do combustor expandem-se através da turbina, onde a potência é extraída para dirigir o compressor. Embora este processo de expansão reduza a temperatura e a pressão do gás da saída da turbina, ambos os parâmetros estão geralmente ainda bem acima das condições ambiente. O fluxo de gás saído da turbina expande-se até à pressão ambiental através do bocal de propulsão, produzindo um jato de alta velocidade à saída do motor. Se o momentum do fluxo da saída exceder o momentum do fluxo de entrada, o impulso é positivo, assim, há uma impulsão líquida para avante sobre a fuselagem.

Em comparação ao *turbofan*, o *turbojet* tem um projeto mais simples, ocupam menos espaço e são capazes de atingir maiores velocidades, além de serem mais leves e mais eficientes em altas altitudes. Entretanto, eles consomem mais combustível e fazem mais barulho durante funcionamento.

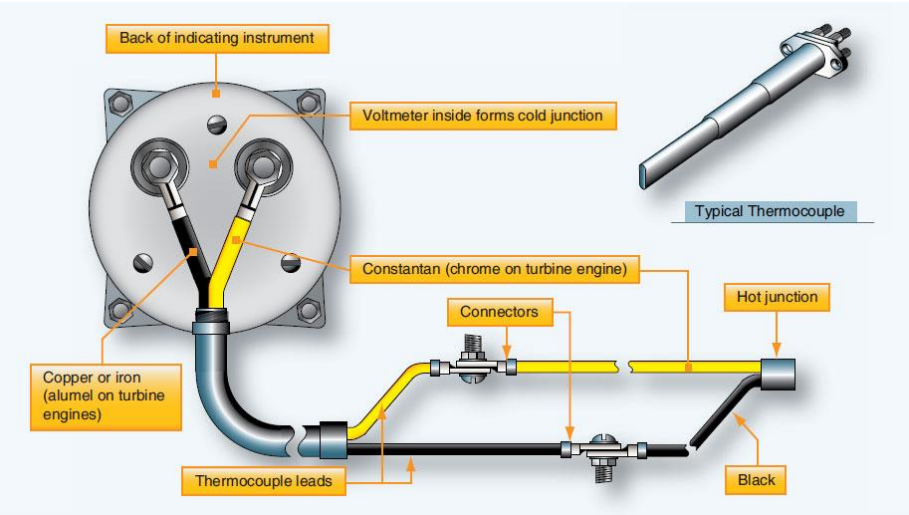
3 MEDIÇÕES DE TEMPERATURA E INSTRUMENTAÇÃO

Termopares geralmente são usados para medir altas temperaturas. Duas aplicações comuns na aviação são a medição da temperatura da cabeça do cilindro (CHT) em motores de pistão e a temperatura do gás de escape (EGT) em motores de turbina. Os eletrodos de termopar são feitos de uma variedade de metais, dependendo da temperatura máxima a que estão expostos. Ferro e constantan, ou cobre e constantan, são comuns para a medição de CHT. O cromel e o alumínio são usados para termopares EGT de turbina (FAA, 2018).

3.1 Medições com termopares em aviônicos

Um termopar é um circuito ou conexão de dois metais diferentes, onde estes metais estão em contato em duas junções separadas. Se uma das junções é aquecida a uma temperatura mais alta que a outra, uma força eletromotriz é produzida no circuito. Essa tensão é diretamente proporcional à temperatura. Então, medindo a quantidade de força eletromotriz, a temperatura pode ser determinada. Quanto mais quente a junção de alta temperatura (junção quente) se torna, maior a força eletromotriz produzida e maior a indicação de temperatura no medidor. A Figura 7 mostra a ligação típica de um termopar a um sistema de medição.

Figura 7 – Típica ligação do termopar com o medidor (voltímetro).



Fonte: Adaptado de FAA, 2018.

A quantidade de diferença de potencial produzida pelos metais dissimilares quando aquecida é medida em milivolts. Portanto, os terminais do termopar são

projetados para fornecer uma quantidade específica de resistência no circuito do termopar, quanto menor possível. Seu material, comprimento ou tamanho da seção transversal não pode ser alterado sem compensação pela alteração na resistência total que resultaria. Cada condutor que faz uma conexão de volta ao voltímetro deve ser feito do mesmo metal que a parte do termopar ao qual ele está conectado. Por exemplo, um fio de cobre é conectado à parte de cobre da junção quente e um fio constante é conectado à parte constante.

A junção quente de um termopar varia em forma dependendo da sua aplicação. Dois tipos comuns são a junta e a baioneta. No tipo de junta, dois anéis dos metais dissimilares são pressionados juntos para formar uma junta que pode ser instalada sob uma vela de ignição ou porca de fixação do cilindro. No tipo baioneta, os metais se juntam dentro de uma bainha protetora perfurada. Os termopares de baioneta encaixam em um orifício ou poço em uma cabeça de cilindro. Nos motores de turbina, eles são encontrados montados na caixa de entrada ou saída da turbina e se estendem através da caixa até a corrente de gás.

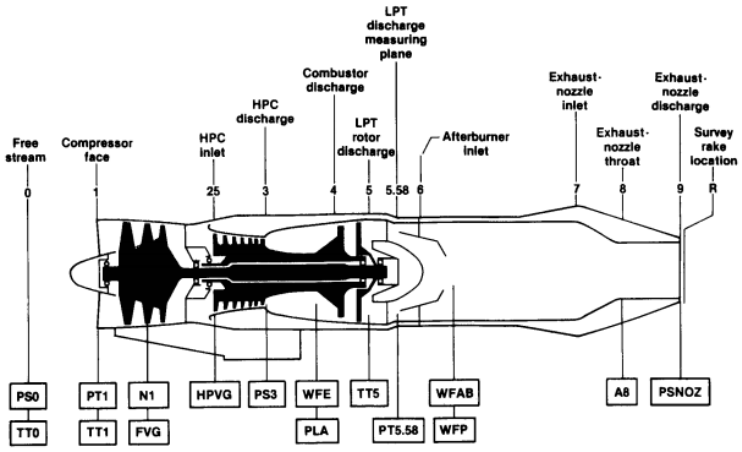
Ambas medições de EGT e CHT tem como característica usual a utilização de termopares tipo K (cromo/alumínio). Este tipo de termopar atende faixas de temperatura de -200°C até 1370°C, com sensibilidades típicas de 41µV/°C.

3.1.1 EGT

O EGT é uma variável crítica da operação do motor de turbina. O sistema de indicação do EGT fornece uma indicação visual de temperatura na cabine dos gases de exaustão quando eles deixam a unidade da turbina. Em certos motores de turbina, a temperatura dos gases de escape é medida na entrada da unidade de turbina. Isto é referido como um sistema indicador de temperatura de entrada da turbina (TIT).

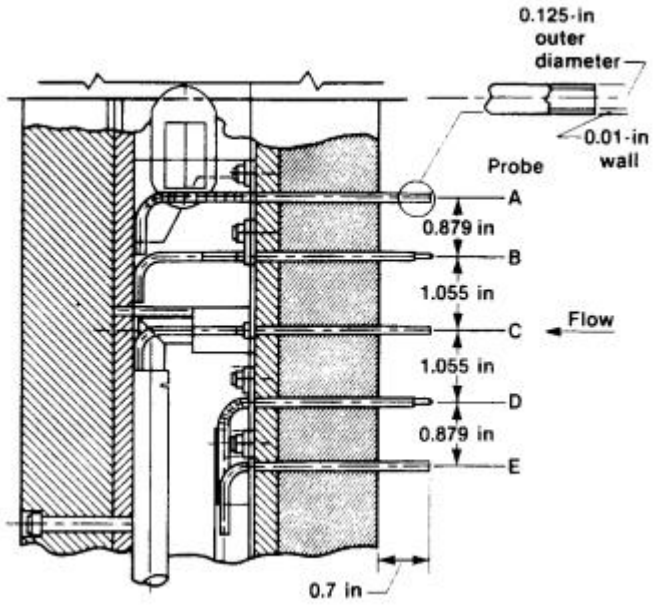
Os sensores de EGT ou TIT podem ser posicionados em diversos pontos da turbina, dependendo das suas características de funcionamento e também da presença ou não de afterburner. Nos motores sem pós-combustão, a temperatura de saída no interior do bocal é de cerca de 600°C com impulso contínuo total. Se a temperatura ambiente for baixa, será necessário menos aquecimento para o mesmo empuxo e as temperaturas de saída serão menores. A temperatura máxima para um determinado motor é, no entanto, independente da altitude e dada pelos materiais dentro do motor. Em vôo, o máximo contínuo pode ser maior do que no solo, quando o resfriamento pelo fluxo externo é menos efetivo. As Figura 8 e Figura 9 mostram posicionamentos típicos dos sensores, neste exemplo específico em um F404-GE-400 turbofan.

Figura 8 – Vista superior da turbina e do posicionamento dos sensores.



Fonte: WATSON, 1986.

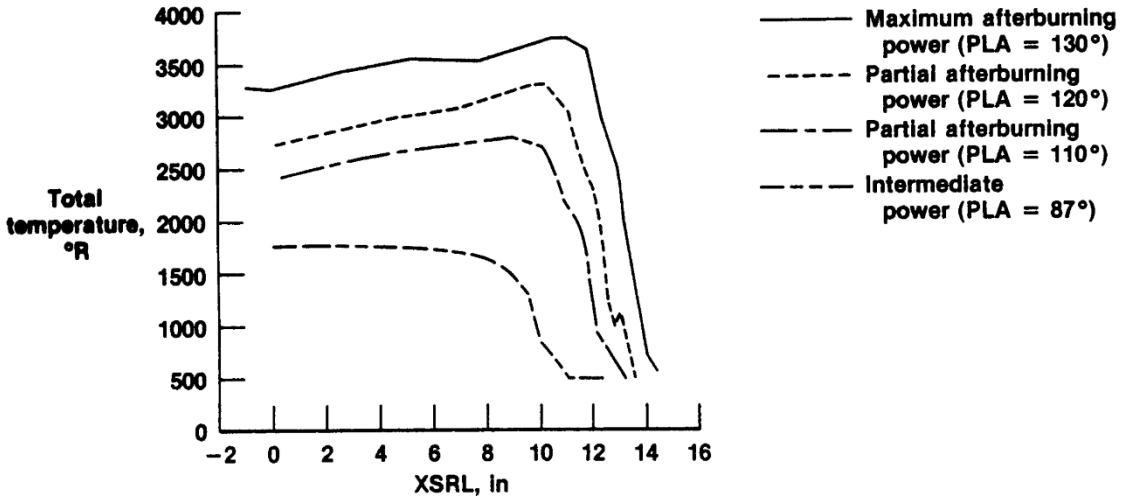
Figura 9 – Corte do posicionamento das sondas de medição de temperatura.



Fonte: WATSON, 1986.

Em um ensaio ensaio realizado por Watson e Burcham Jr., utilizando um afterburner em um motor GE F-404, as temperaturas de bocal chegaram a 1838°C (3800°R), conforme gráfico apresentado na Figura 10, onde a posição de 0 XSRL é o centro do motor. Este valor de temperatura atingido é comum em caças ou aeronaves de grande porte, onde se deseja obter grande empuxo a partir das turbinas.

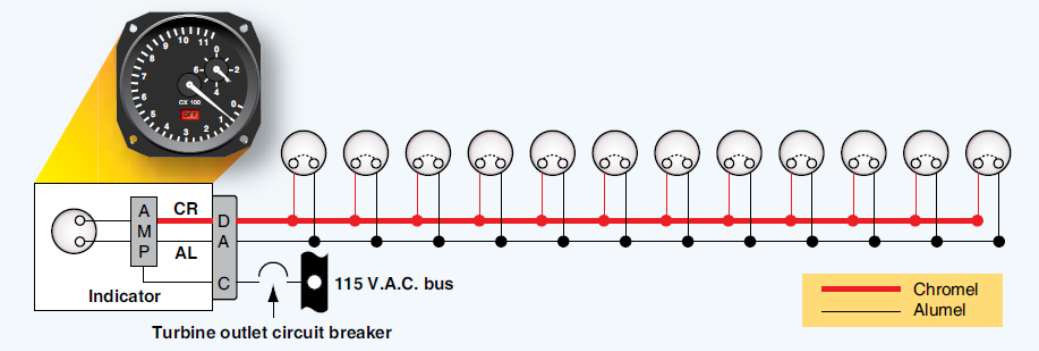
Figura 10 – Medições de temperatura no bocal de um GE-F404.



Fonte: Adaptado de WATSON, 1986.

Vários termopares são associados na medição de EGT ou TIT. Eles estão espaçados em intervalos ao redor do perímetro da carcaça da turbina do motor ou do duto de exaustão, por razões de redundância e confiabilidade. A diferença de potencial do termopar são tipicamente usadas para energizar um servomotor que aciona o indicador, o que é ilustrado na Figura 11, ou é condicionada através de amplificadores de instrumentação e processa em um circuito digital, além de ser enviada e condicionado em circuitos eletrônicos, no caso da presença de displays ou computadores de missão.

Figura 11 – Arranjo de termopares para medições de EGT ou TIT.



Fonte: Adaptado de KEE, 1999.

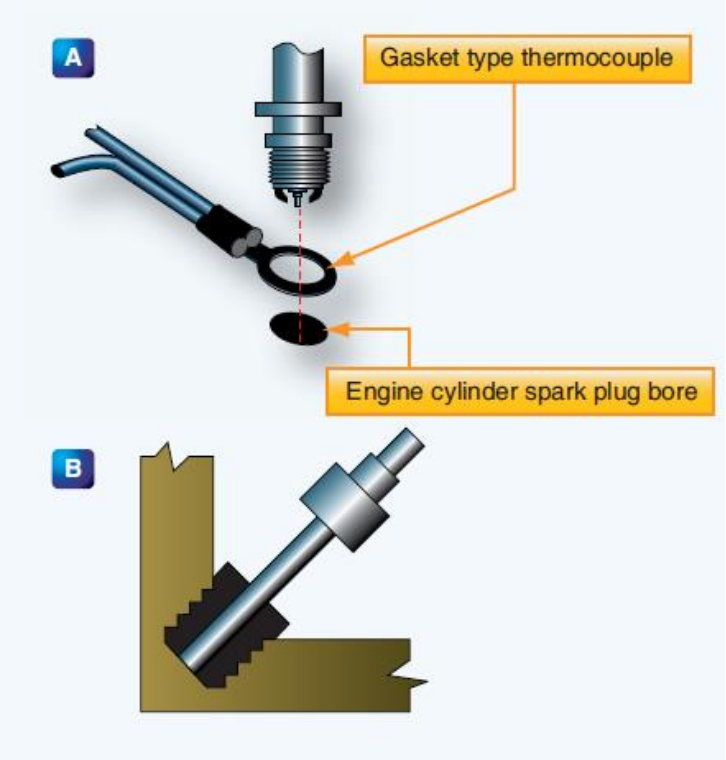
Um sistema de indicação TIT fornece uma indicação visual no painel de instrumentos da temperatura dos gases que entram na turbina. Uma grande variedade de termopares podem ser usados com a tensão média representando o TIT. Dois

termopares existem contendo duas junções eletricamente independentes dentro de uma única sonda. Um conjunto desses termopares é paralelizado para transmitir sinais ao indicador da cabine. O outro conjunto de termopares paralelos fornece sinais de temperatura para os sistemas de monitoramento e controle do motor, onde cada circuito é eletricamente independente, fornecendo confiabilidade de sistema dual.

3.1.2 CHT

Um medidor de temperatura da cabeça do cilindro (CHT) é uma ferramenta essencial usada para monitorar a temperatura e a saúde de funcionamento de um motor ou turbina. Observe que, para a indicação de CHT, o cilindro escolhido para a instalação do termopar é o que está mais quente na maioria das condições de operação, dado que ele tende a ser o cilindro mais crítico. A Figura 12 mostra um termopar tipo junta e outro tipo baioneta utilizado para medições de CHT.

Figura 12 – Tipos de termopares para diferentes aplicações.

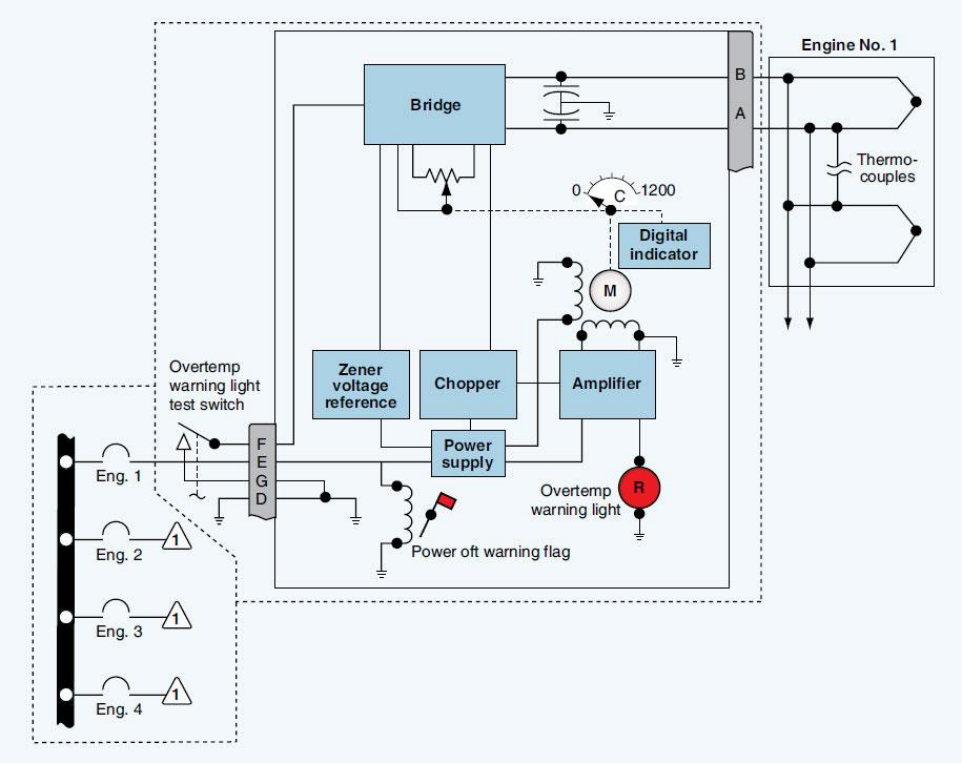


Fonte: FAA, 2018.

3.1.3 Mostradores analógicos e sistemas digitais com EICAS

Um esquema para o sistema de temperatura de entrada da turbina para um motor de uma aeronave de turbina de quatro motores é mostrado na Figura 13. Os circuitos para os outros três motores são idênticos a este sistema. O indicador contém um circuito de ponte, um circuito de limitação, um motor de duas fases para acionar o ponteiro e um potenciômetro de realimentação. Também estão incluídos um circuito de referência de tensão, um amplificador, um sinalizador de desligamento, uma fonte de alimentação e uma luz de advertência sobre temperatura excessiva. A saída do amplificador energiza o campo variável do motor bifásico que posiciona o ponteiro principal do indicador e um indicador digital. O motor também aciona o potenciômetro de realimentação para fornecer um sinal de áudio para parar o motor quando a posição correta do ponteiro, em relação ao sinal de temperatura, for atingida. O circuito de referência de tensão fornece uma tensão de referência estritamente regulada no circuito de ponte para evitar erro da variação da tensão de entrada para a fonte de alimentação do indicador.

Figura 13 – Diagrama analógico típico do indicador de temperatura.



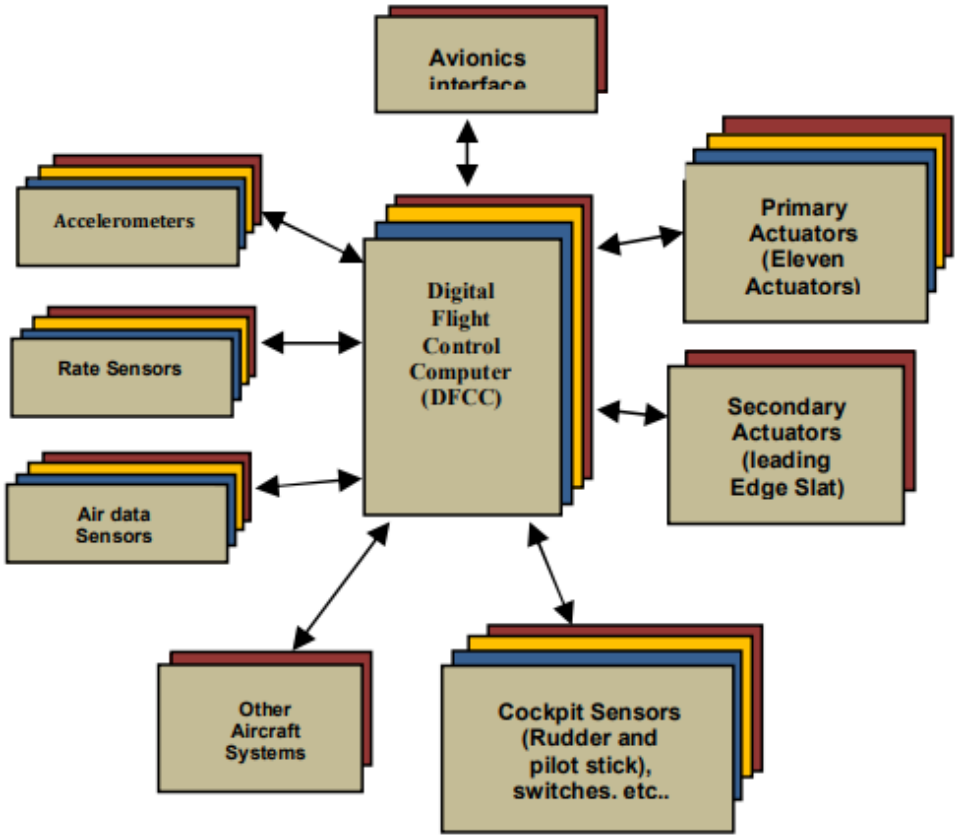
Fonte: FAA, 2018.

A luz de aviso de sobretemperatura no indicador acende quando o TIT atinge um limite predeterminado. Um interruptor de teste externo é normalmente instalado de

modo que as luzes de aviso de temperatura excessiva para todos os motores possam ser testadas ao mesmo tempo. Quando a chave de teste é operada, um sinal de superaquecimento é simulado em cada circuito de ponte de controle de temperatura do indicador.

Em aeronaves modernas, é comum a presença de um *engine-indicating and crew-alerting system* (EICAS), sistema integrado que fornece dados dos motores e outros sistemas de instrumentação a tripulação. O EICAS inclui tipicamente instrumentação de vários parâmetros do motor, incluindo por exemplo rotações por minuto, valores de temperatura, fluxo e quantidade de combustível, pressão do óleo, etc. Outros sistemas de aeronaves monitorados pelo EICAS são, por exemplo, hidráulicos, pneumáticos, elétricos, degelo, ambientais e de controle sistemas. A Figura 14 mostra elementos típicos, assim como o EICAS, ligados a um computador central de processamento.

Figura 14 – Exemplo de elementos conectados a um computador de aeronave.

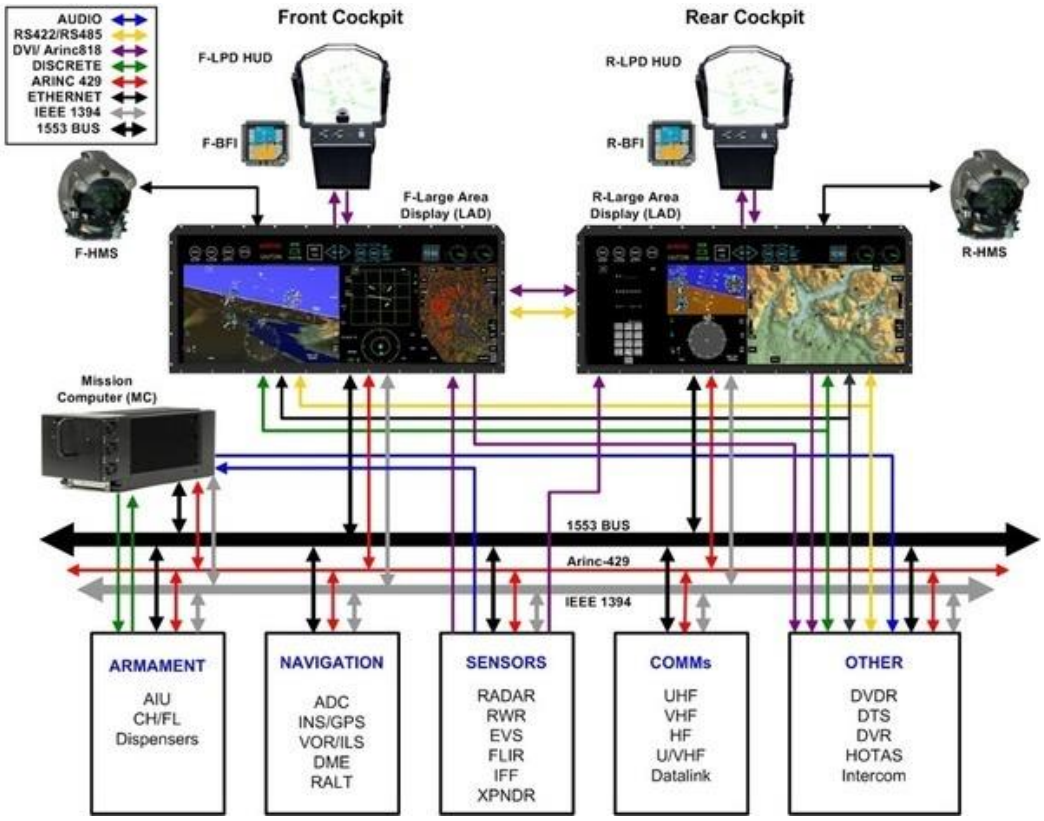


Fonte: Adaptado de RAO, 2016.

O EICAS possui alta conectividade e fornece aquisição e roteamento de dados. Além disso, é uma função essencial de um sistema de cockpit de vidro, que substitui todos os medidores analógicos por monitores eletrônicos acionados por software

(ESTERLINE, 2018). A maior parte da área de exibição é usada para exibições de navegação e orientação, mas uma exibição ou uma seção de uma exibição é reservada especificamente para EICAS. O sistema de alerta de tripulação (CAS) é usado no lugar do painel do anunciador em sistemas mais antigos. A Figura 15 mostra a interligação típica existente entre os computadores e os displays e projeções no cockpit de vidro.

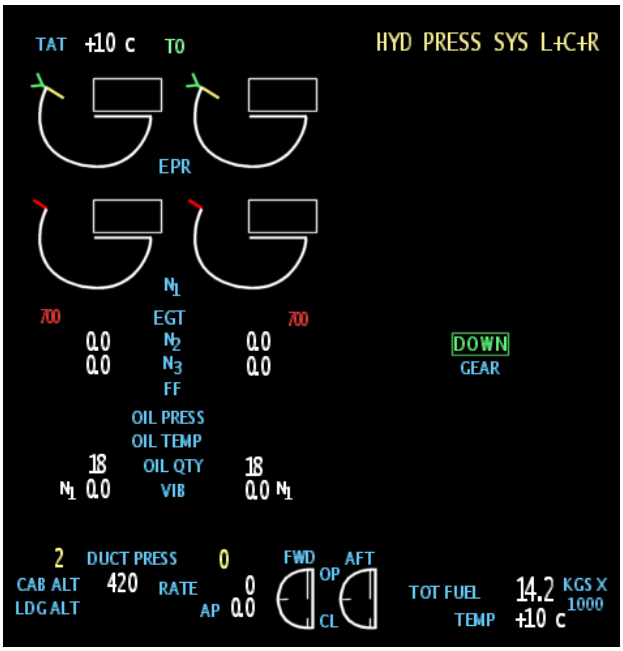
Figura 15 – Interligação entre os sensores, computadores e displays.



Fonte: Esterline, 2018.

Em vez de sinalizar uma falha do sistema ao acender uma luz atrás de um botão translúcido, as falhas são mostradas como uma lista de mensagens em uma pequena janela perto das outras indicações do EICAS (WELLS, 2003). A Figura 16 mostra uma tela típica de EICAS de uma das telas da aeronave contendo informações como a temperatura de EGT.

Figura 16 – Tela típica de EICAS mostrada em uma das telas da aeronave.



Fonte: WELLS, 2003.

4 CIRCUITOS ELETRÔNICOS E SUAS FONTES DE ERRO

Esta seção apresenta as fontes de radição espacial e seus principais efeitos em circuitos eletrônicos. O foco deste trabalho é apresentar uma solução que garanta a confiabilidade do sistema de medição proposta, especialmente para efeitos singulares, portanto é importante o conhecimento das fontes e efeitos destes erros para que sejam mitigados posteriormente.

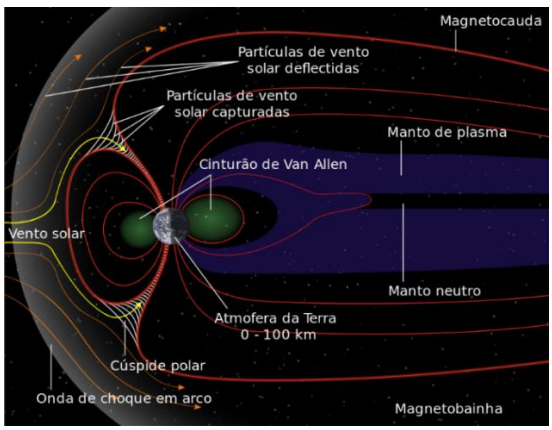
4.1 Radiação Espacial

A radição especial que pode causar efeitos indesejáveis nos circuitos eletrônicos é dita ionizante porque possui energia suficiente para retirar elétrons de átomos, criando íons (BALEN, 2010). As principais partículas que podem causar efeitos indesejados nos circuitos eletrônicos são elétrons, prótons, nêutrons, partículas alfa e íons pesados, além da radiação eletromagnética, tais como raio-x e raios-gama (STASSINOPOULOS & RAYMOND, 1988).

4.1.1 Cinturões de Van Allen

Existem regiões do espaço em torno da terra, que através do campo magnético terrestre, aprisionam prótons e elétrons provenientes do vento solar. A Figura 17 mostra uma ilustração da magnetosfera e os dois cinturões de Van Allen (interno e externo), sendo o cinturão externo o que contém partículas com maior energia. Ions pesados também são aprisionados, entretanto sua pequena abundância associado com sua baixa energia os tornam significativamente pequenos para geração de efeitos em circuitos eletrônicos (STASSINOPOULOS, 2013).

Figura 17 – Cinturões de Van Allen e magnetosfera da terra.

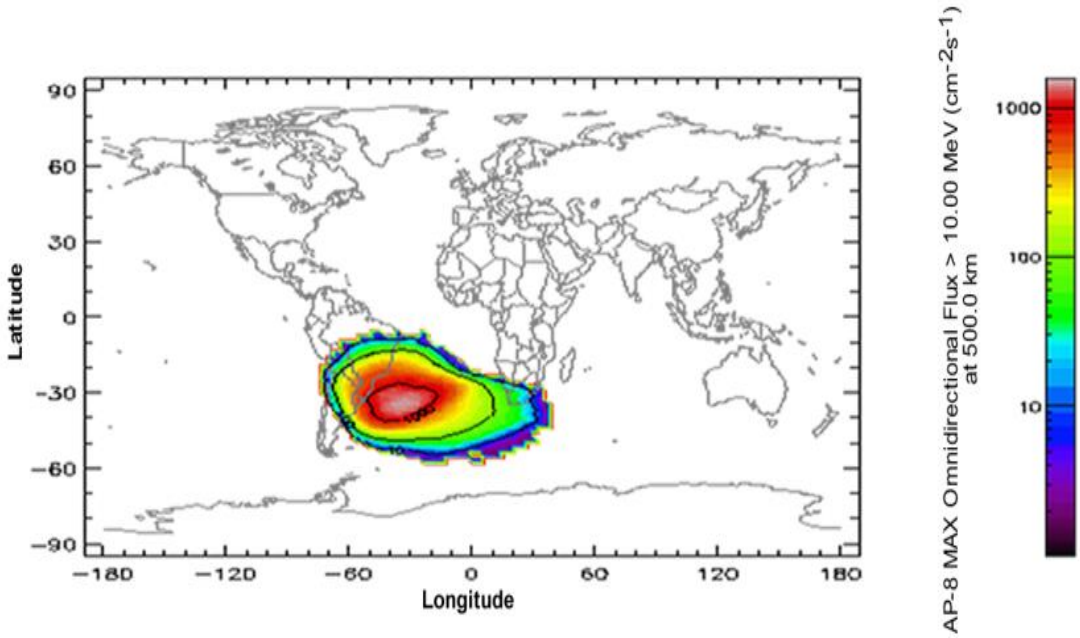


Fonte: Adaptado de UNIVERSITÉ DE LYON, 2018.

O cinturão interno contém elétrons com spectro de energia de até 5MeV e localize-se em altitudes entre 100km e 10.00km. Já o cinturão externo contém elétrons de até 7MeV e situa-se emaltitudes de aproximadamente 20.00km até 60.00km (STASSINOPOULOS & RAYMOND, 1988). Outros cinturões podem surgir temporariamente, em função de tempestades magnéticas, com elétrons de energia de até 30MeV e prótons de até 50MeV (HEYNDERICK, 1996).

A inclinação do eixo de rotação terrestre em relação ao eixo do campo magnético terrestre influencia na distribuição do fluxo de partículas, criando uma espécie de região de depressão, que produz efeitos indesejados nos equipamentos eletrônicos de espaçonaves e satélites (TORRES, 2013). Essa região é conhecida por Anomalia do Atlântico Sul (do inglês, *South Atlantic Anomaly*, SAA). Essa anomalia é conhecida como uma região Vile Vortices, onde ocorrências de falhas em dispositivos eletrônicos são frequentes. A radiação proveniente da SAA é famosa por causar avarias geradas em naves espaciais e, igualmente, a SAA é responsável pela maior parte de radiação incidente nos satélites de baixa órbita (STASSINOPOULOS & RAYMOND, 1988). Já na Figura 18, o fluxo de elétrons é crítico na SAA e nas regiões onde o cinturão mais externo atinge altitudes mais baixas, próximo dos polos terrestres, nessa última região a concentração de partículas pode chegar a 100 vezes maior do que em outras regiões, como por exemplo, sobre o sul do Brasil (SPENVIS, 2013).

Figura 18 – Simulação do fluxo de prótons utilizando o modelo AP-8 para altitude de 500km na região da SAA.



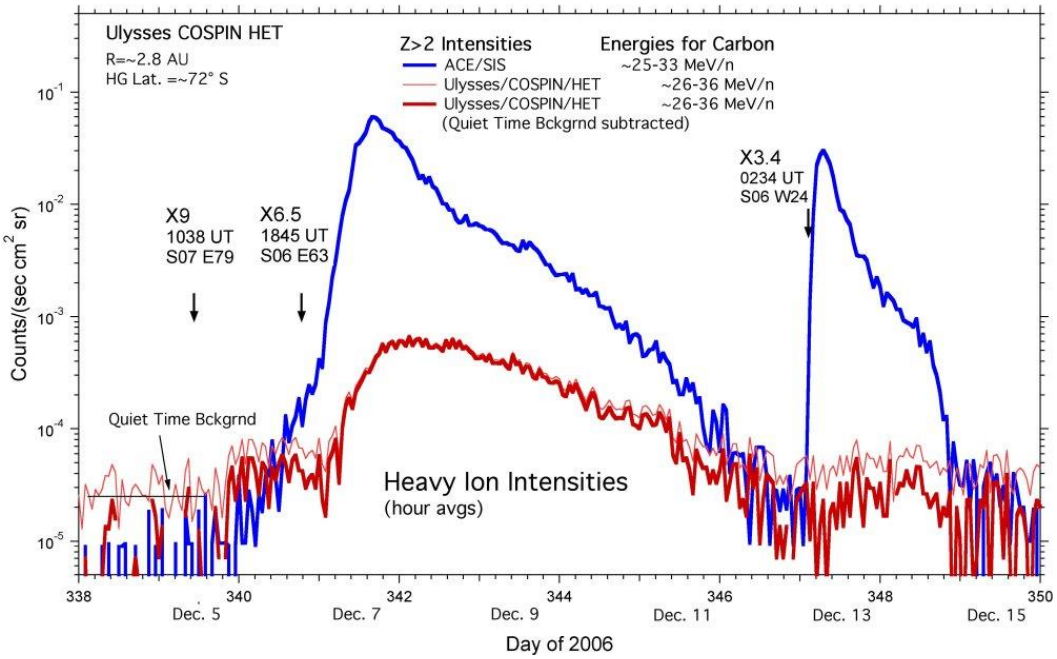
Fonte: Adaptado de SPENVIS, 2013.

4.1.2 Atividade Solar

Um exemplo de fluxos de íons pesados associado as explosões solares é mostrado na Figura 19. Como as explosões solares ocorrem esporadicamente, o mesmo ocorre com o ambiente associado. Como no caso dos raios cósmicos galácticos, são as partículas energéticas carregadas, aceleradas perto do sol durante alguns eventos de erupção solar, que causam fenômenos de evento único e dano de dose total. Nossa compreensão da atividade solar é muito rudimentar para podermos prever com muita antecedência o tempo exato de início de uma erupção solar (KOLASINSKI, 1989).

As ocorrências individuais de explosões parecem ser bastante aleatórias, exceto que sua frequência segue o ciclo de manchas solares de 11 anos, compreendendo 7 anos de alta atividade e 4 anos de baixa atividade (BOUDENOT, 2007). A maioria das erupções solares produz fluxos de prótons que não contribuem significativamente para a taxa de evento único. Geralmente, o fluxo dos íons mais pesados nessas chamas também não é muito significativo. No entanto, como mostra o exemplo da Figura 19, surtos ricos em íons pesados ocorrem algumas vezes, e o fluxo de íons pesados de energia média de tal flare pode exceder o fundo galáctico em mais de uma ordem de grandeza. É de fato afortunado que esses eventos sejam raros e de duração relativamente curta (PETERSEN, 2011).

Figura 19 – Fluxo de partículas devido a explosão solar.



Adaptado de: National Aeronautics and Space Admin, 2007.

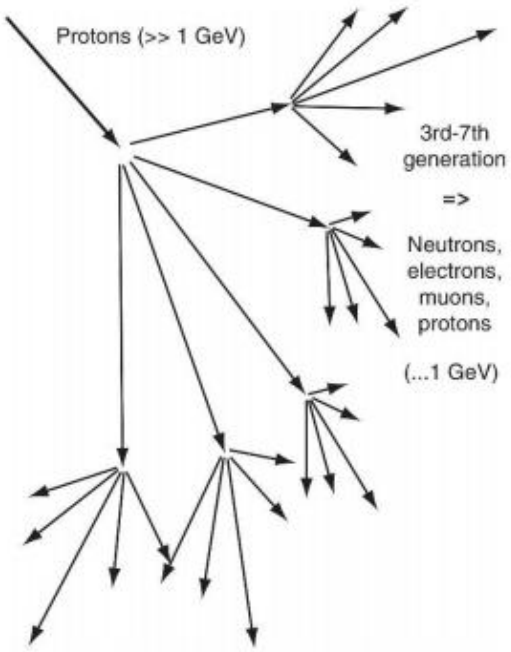
4.1.3 Raios Cósmicos Galáticos

Embora chamados de raios, os RCGs são partículas com alta energia provenientes de fora do sistema solar. A radiação galática consiste de íons diversos elementos da tabela periódica, tendo composição aproximada de 83% de prótons, 13% de partículas alfa, 3% de elétrons e 1% de íons de núcleos pesados (BARTH, 1997). Possuem energias típicas maiores que 10GeV por núcleon e chega às regiões próximas da terra com aproximadamente 1GeV. O fluxo de RCGs fora da magnetosfera, em distâncias equivalentes à distância terra-sol, é de aproximadamente 4 partículas/cm²s (STASSINOPOULOS & RAYMOND, 1988).

Considerando a composição de partículas dos RCGs, para a geração de efeitos danosos nos dispositivos eletrônicos, são significativos apenas os prótons e íons. Quanto aos elétrons, sua densidade é de uma ordem de grandeza menor que a sua densidade no vento solar, e portanto não necessitam ser considerados (BARTH, 1997).

Os RCGs também originam partículas secundárias quando entram na atmosfera terrestre. A interação com átomos de nitrogênio e oxigênio resulta em uma cascata composta por prótons, elétrons, nêutrons, íons pesados, múons e píons. Esse fenômeno é conhecimento como “chuveiro” de partículas, demonstrado na Figura 20. Em termos de efeito da radiação na atmosfera, o mais importante produto desta interação de partículas são os nêutrons. Eles são mensuráveis à 330 km de altitude, e sua densidade aumenta com o decremento da altitude, alcançando a saturação em aproximadamente 20km (BARTH, 2003).

Figura 20 – Estrutura do chuveiro de partículas formado pelo RCGs.

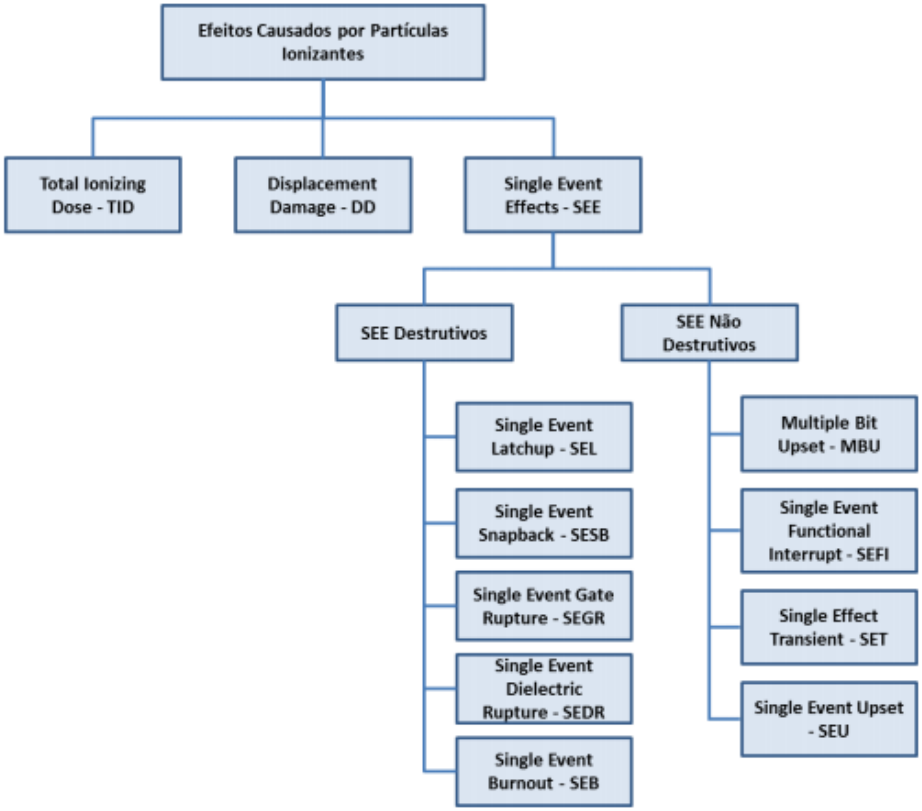


Fonte: Adaptado de LERAY & PETERSEN, 2011.

4.2 Efeitos da Radiação em Dispositivos Eletrônicos

Os efeitos das interações da radiação de partículas em componentes eletrônicos são diversos, e podemos classificá-los de diversas maneiras. Ao se considerar o efeito final, podemos distinguir entre efeitos transitórios e efeitos permanentes. Os efeitos permanentes estão ligados à modificações estruturais no material do componente eletrônico, não sendo reversível tanto ao se retirar a fonte de partículas quanto ao se efetuar uma ciclagem de energia do componente. Ao se considerar os aspectos físicos do componente, os efeitos podem ser classificados em danos por deslocamento e danos por ionização (ARRUDA, 2006). Na sequência, são descritos os conjuntos mais comuns de efeitos, conforme ilustrado na Figura 21.

Figura 21 – Tipos de efeitos causados por partículas ionizantes.



Fonte: Adaptado de LERAY & PETERSEN, 2011.

A Figura 22 mostra um resumo da relação entre a origem da radiação e o efeito causado nos componentes. A radiação do cinturão de Van Allen, que é composto de prótons e elétrons altamente energizados, contribui para todos os tipos de efeitos, igualmente aos prótons energizados provenientes das erupções solares. Os raios cósmicos são compostos de íons altamente energizados capazes de induzir SEE, mas que

são raros e não possuem quantidade suficiente para contribuir para a degradação do componente (FERLINI, 2012).

Figura 22 – Relação entre as fontes de radiação e seus efeitos.

Fonte: Adaptado de FERLINI, 2012.

4.2.1 Total Ionization Dose (TID)

A radiação ionizante, ao atingir componentes eletrônicos com energia suficiente, interage com os materiais que ao receberem energia da partícula, liberam elétrons, gerando pares de elétrons-lacuna (um elétron livre e uma lacuna do átomo original). Durante o funcionamento do circuito, muitas cargas são recombinadas e estes pares são eliminados. No entanto, algumas cargas podem persistir, acumulando-se no componente e modificando seu desempenho, de acordo com esta energia ionizante total absorvida (ARRUDA, 2006). No caso do óxido de silício (SiO_2), comumente usado como material isolante em componentes eletrônicos, as lacunas, por se difundirem lentamente, têm maior probabilidade de se acumularem no volume do óxido, ou na interface do óxido-silício, tornando-se assim uma carga positiva. Como consequência, temos a criação de um campo elétrico parasita que, de acordo com seu efeito acumulado, modifica e prejudica o funcionamento do componente. Assim, a Dose Total de Ionização (TID), que pode ser considerada um dano por ionização, pode ser entendida como uma medida da dose total da energia acumulada no material do componente eletrônico por radiação (ARRUDA, 2006). A dose é usada para quantificar os efeitos da liberação de carga por ionização, é definida como a energia depositada pela unidade de massa do material (que deve ser especificado), e pode ser medida em termos de J/kg ou rad, onde 1 rad = 100 ergs/g (DYER, 2001).

4.2.2 Displacement Damage (DD)

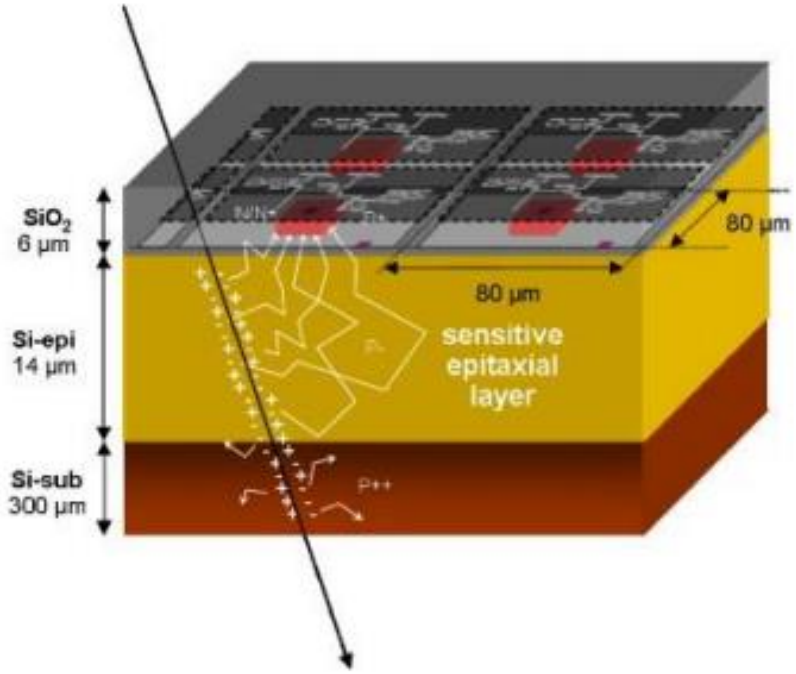
Os átomos dos componentes eletrônicos semicondutores são dispostos em uma rede cristalina, que caracteriza seu funcionamento. Ao serem atingidos por partículas de alta energia, pode haver um deslocamento do núcleo de um ou mais átomos da rede, o que altera as características eletrônicas do componente (MACHADO, 2014). Este fenômeno é também chamado de Dose Total Não Ionizante (*Total Non Ionization Dose*), de forma a distinguir este fenômeno do TID (ECSS E-HB-10-12A, 2010). A energia e o momento linear transferido pela partícula variam principalmente de acordo com a massa e a energia que a partícula incidente detém. Alguns exemplos de efeitos em componentes eletrônicos são: redução no ganho de transistores bipolares, redução na eficiência de células solares e fotodetectoras, ineficiência na transferência de carga em dispositivos CCD (*Charge Coupled Devices)* e degradação na resolução de sensores de estado sólido. Assim, podemos definir os Danos por Deslocamento (DD), como danos na estrutura cristalina de um material causados pela colisão, elástica ou não, de uma partícula de alta energia (ARRUDA, 2006).

4.2.3 Efeitos Singulares

Partículas ionizadas de alta energia provocam em circuitos integrados fabricados com tecnologia CMOS efeitos indesejados no seu funcionamento. Esses efeitos são causados pelo choque dessas partículas com regiões sensíveis do material semicondutor dos dispositivos eletrônicos. Um dos pontos sensíveis do material semicondutor nos circuitos integrados são as vizinhanças das junções dos drenos dos transistores polarizadas reversamente (TORRES, 2013). Se uma partícula de alta energia atravessa a junção pn de um transistor CMOS em estado off, um caminho de baixa impedância momentânea pode ser criado entre o substrato e o terminal de dreno, mostrado na Figura 23, similar ao comportamento de um curto circuito elétrico. Entretanto, se esse este comportamento não ocorre, a quantidade de carga armazenada resultante da penetração dessa partícula no material poderá ser suficiente para produzir um pequeno pulso de corrente transiente, que poderá persistir por um pequeno intervalo de tempo δ até que o efeito dessa carga desapareça, seja por recombinação com outras cargas ou através de um caminho livre para Vdd ou Vss no circuito integrado.

Nesse último caso, a lógica do dispositivo continuará a funcionar de forma normal assim como fora projetado sem causar erros no sistema. Todavia, se a recombinação de cargas não desaparecer dentro de um período de tempo no qual certo dado estiver sendo computado ou armazenado por um elemento de memória, o efeito dessa partícula produzirá um fenômeno conhecido como *Single Event Effects* (SEEs), que se comporta no universo da informação como uma inversão de estado lógico booleano (*bit-flip*) (HIGUERETA, 2008).

Figura 23 – Single Event Effect em um transistor MOS.

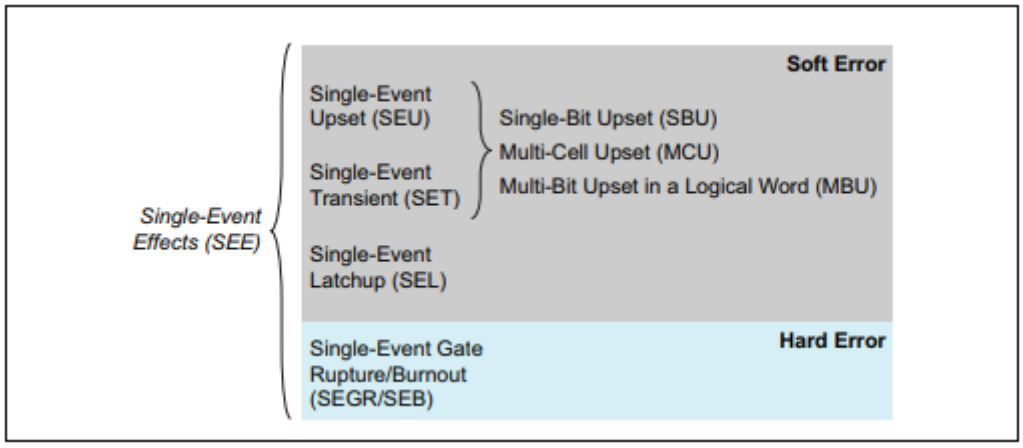


Fonte: Adaptado de HIGUERETA, 2008.

Os efeitos físicos do impacto da radiação em materiais são fenômenos muito complexos e requerem modelos matemáticos e estatísticos específicos. As simulações desses modelos são altamente custosas em tempo devido à complexidade e à quantidade de informação envolvida. Esse quadro agrava-se extremamente quando consideramos circuitos construídos com as tecnologias modernas que integram bilhões de transistores. Contudo, comportamentos comuns entre os circuitos afetados podem ser percebidos. Esses comportamentos estão ligados ao funcionamento do circuito e não aos princípios físicos do fenômeno da radiação. Nesse contexto, são introduzidos os modelos de falhas que estão relacionados ao comportamento do circuito, ou melhor, à mudança de comportamento quando o circuito é afetado por radiação (FERLINI, 2012).

O endurecimento de CIs comerciais, em geral, perante os efeitos duradouros, tem melhorado nos últimos anos (VELAZCO, 2007). O contrário tem acontecido com a sensibilidade dos dispositivos diante aos efeitos radioativos transientes, que tem se agravado devido à miniaturização da tecnologia de dispositivos. Dessa maneira, serão apresentados os modelos de falhas do tipo SEE. Esses efeitos podem ser divididos entre Hard Errors, que correspondem a falhas destrutivas, e Soft Errors, onde o efeito temporário da falha muitas vezes desaparece, principalmente, por meio da simples reinicialização do dispositivo. A Figura 24 mostra os tipos de erros comuns, classificando os efeitos que causam Hard Error ou Soft Error no dispositivo.

Figura 24 – Acrônimos de Single Event Effects e tipos de erros.



Fonte: Adaptado de Altera, 2013.

A seguir são detalhados e especificados os principais SEEs, baseados em (ECSS-E-HB-10-12A, 2010).

4.2.3.1 *Single Event Upset* (SEU)

Single Event Upset (SEU) é uma mudança de estado lógico ou tensão causada por uma partícula energizada que colide com o nodo sensível de um circuito. A norma ECSS-E-ST-10-04C define SEU como uma mudança de estado de um bit em um elemento digital, causado tanto por uma partícula ionizante atravessando um componente, como por emissão de um núcleo que interage com uma partícula. De acordo com (NICOLAIDS, 2011), a literatura por vezes utiliza o termo SEU como sendo o conjunto dos SEEs não destrutivos. Embora seja o tipo mais comum de SEE não destrutivo, para o presente trabalho o SEU deve ser entendido em seu sentido mais restrito. O SEU ocorre em circuitos digitais quando uma partícula com alta energia (partícula ionizante) faz com que um elemento de eletrônica digital mude de estado. Isto pode ocorrer em micro circuitos, como por exemplo, chips de memória, dispositivos de comunicação, circuitos de potência ou microprocessadores. Esta mudança de valor pode levar a comportamentos observáveis em nível de sistema, como o travamento de um subsistema ou um comando inesperado do mesmo. Nos circuitos CMOS, as partes mais sensíveis à SEU são a região do canal de transistores NMOS desligados e a região do dreno de transistores PMOS desligados (WANG; AGRAWAL, 2008).

Os SEUs podem ainda ser classificados em subgrupos em função de sua abrangência. *Single Bit Upset* (SBU) ocorre quando a colisão de uma partícula causa um único bit-flip (HEIJMEN, 2011). *Multiple Cell Upset* (MCU) causa dois ou mais bit-flips e *Multiple Bit Upset* (MBU) causa dois ou mais bit-flips na mesma palavra (JOINT ELECTRON DEVICE ENGINEERING COUNCIL, 2012). Múltiplos erros podem ser criados

quando uma partícula cruza regiões sensíveis de diferentes células ou quando os portadores livres gerados por partículas ionizantes são coletados por diferentes junções de transistores de diversas células de memória (GAILLARD, 2011). Neste primeiro caso citado, a incidência da radiação em um ângulo rasante é determinante para potencializar múltiplos erros. A redução das dimensões dos circuitos integrados nas tecnologias recentes também aumenta a probabilidade de múltiplos erros.

4.2.3.2 *Single Event Functional Interrupt* (SEFI)

Single Event Functional Interrupt (SEFI): mudança de um sinal de controle de um circuito ou em processadores, causado por uma partícula ionizante, tendo como conseqüência a interrupção do funcionamento normal do circuito ou dispositivo.

O termo SEFI foi mencionado pela primeira vez em 1996, pelo Joint Electron Device Engineering Council (JEDEC). É caracterizado pela perda da funcionalidade em circuitos integrados complexos, ocasionada por exemplo, por travamento, ou perturbação em registradores de controle, sinais de clock ou sinais de reset. Além disso, pertence à classe de soft errors pois pode ser recuperado após um reset na alimentação, um reset funcional ou recarga dos registradores de configuração (GAILLARD, 2011; HEIJMEN, 2011; JOINT ELECTRON DEVICE ENGINEERING COUNCIL, 2012). SEFIs são geralmente reportados em dispositivos como memórias DRAM síncronas, SRAM e Flash, FPGAs (Field Programmable Gate Array) baseados em SRAM, microprocessadores e microcontroladores.

Os SEFIs estão associados aos SEUs, porém se diferenciam quando provocam a perda da funcionalidade temporária ou a interrupção da operação normal do circuito afetado. Em alguns casos eles duram por todo o tempo que a alimentação é mantida, enquanto em outros ele duram por um tempo finito.

O artigo de Koga et al., de 1997a, considera que os SEFIs são causados por SEUs em dispositivos em que não se tem um entendimento detalhado de sua arquitetura, ou em pontos sensíveis do circuito os quais não temos acesso direto. Uma vez que não é possível identificar a localização exata do SEU, é possível apenas observar a falha funcional do dispositivo. Nesse contexto, tem-se uma definição para SEFI um pouco diferente da apresentada inicialmente: é um upset para o qual não é possível medir diretamente o bit alterado ou prontamente observar a indicação do upset, e consequentemente é detectada apenas a anomalia funcional (KOGA et al., 1997a). Se o mecanismo exato para um upset é conhecido, ele não pode ser considerado um SEFI.

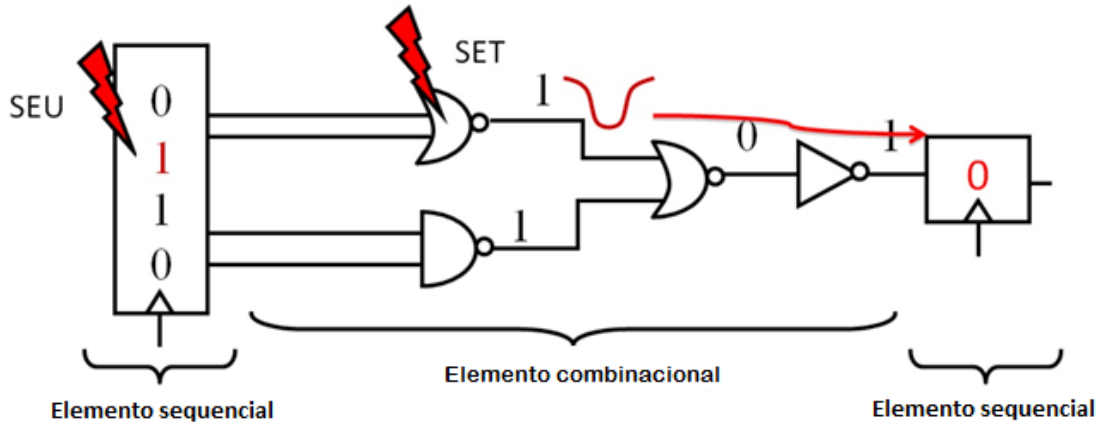
4.2.3.3 *Single Event Transient* (SET)

Single Effect Transient (SET): efeito que gera um sinal espúrio na forma de uma excursão no valor da tensão em um semicondutor (*voltage spike*), que pode ser propagado e capturado como um valor lógico errado.

SET refere-se a um pulso transiente em circuitos integrados causado pela passagem de uma única partícula energética (JOINT ELECTRON DEVICE ENGINEERING COUNCIL, 2012). Pode ou não ser capturado por um elemento de memória (BALEN, 2010), e ocorre tanto em dispositivos digitais (DIEHL et al., 1983) quanto em analógicos (TURFLINGER, 1996). Os erros mais importantes devido à SET ocorrem em circuitos combinacionais e subsistemas analógicos (PETERSEN, 2011).

Nos circuitos combinacionais, mesmo que o pulso transiente não induza uma inversão de bit no nó afetado, ele pode se propagar pelo circuito e quando encontrar um latch ou elemento de memória, ser armazenado como um dado incorreto, como mostrado no circuito da Figura 25.

Figura 25 – SET em um circuito lógico combinacional simples.



Fonte: Adaptado de Aguiar, 2016.

Adicionalmente a esses fenômenos de mascaramento, outros dois fatores são importantes no impacto dos SETs em circuitos combinacionais: a frequência do clock e a largura do pulso transiente (BAUMANN apud MUNTEANU; AUTRAN, 2008). Com o aumento da frequência do clock existem mais bordas de relógio que podem capturar o pulso (em um dado instante de tempo), e assim a taxa de erros aumenta. Um pulso menos estreito aumenta a probabilidade do SET chegar no latch na borda do *clock* de captura. Se o pulso se tornar mais largo que o período do *clock*, então todos os transientes são capturados (GADLAGE et al., 2004).

Mitra et al., em 2005, fizeram estimativas quantitativas das contribuições para a SER total em microprocessadores, processadores e controladores de elevada velocidade. Considerando a tecnologia estado da arte do ano em questão, se obteve que aproximadamente 10% dos *soft errors* tinham origem nos circuitos combinacionais, sendo o restante com origem em elementos de memória. A tendência para as tecnologias avançadas, de dimensões reduzidas, é de que o SET em circuitos combinacionais passe a dominar as taxas de erro (BAUMANN apud MUNTEANU; AUTRAN, 2008; DODD et al., 2010). As principais razões para tal são (MUNTEANU; AUTRAN, 2008): tecnologias de dimensões reduzidas permitem que mais transientes tenham largura de pulso e amplitude suficientes para serem capturados, devido à elevada frequência de operação; o aumento da frequência de operação aumenta a habilidade do transiente se propagar pelo circuito; e com a redução das dimensões, a carga representando o nível lógico alto é reduzida, aumentando o número de SETs.

Nos circuitos digitais, o impacto de um SET é mensurado em última instância pela ocorrência de um bit-flip, que tem efeito de longa duração e frequentemente a habilidade de alterar a funcionalidade de um sistema, como um reset ou modificação de um algoritmo. Já nos circuitos analógicos, o impacto de um SET requer uma dimensão ou atributo adicional para ser adequadamente caracterizado, como por exemplo uma tensão associada, corrente ou dimensão de tempo. Por esse motivo, o estudo do SET em circuitos analógicos é mais recente do que em circuitos digitais, com um crescente de publicações a partir do final dos anos de 1980 (TURFLINGER, 1996).

4.2.3.4 *Single Event Latchup* (SEL)

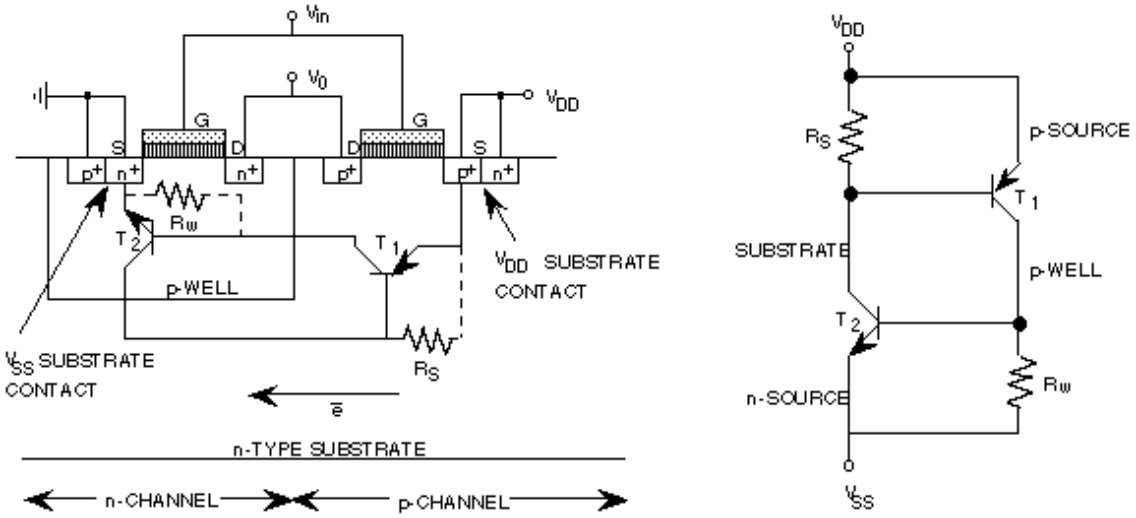
Single Event Latchup (SEL) é uma condição potencialmente destrutiva envolvendo correntes parasitas as quais podem exceder o valor máximo suportado pelo componente, levando à perda do componente quando não há limitação de corrente. É geralmente referido a circuitos CMOS, onde o efeito principal é a criação de tiristores parasitas PNPN.

SEL refere-se à ocorrência de uma corrente anormal e elevada no dispositivo, causada pela passagem de uma única partícula energética em regiões sensíveis de sua estrutura, e que permanece depois que o disparo é removido. Para reestabelecer a operação normal do dispositivo é necessário um reset na alimentação, embora o efeito também possa causar dano permanente (HEIJMEN, 2011; JOINT ELECTRON DEVICE ENGINEERING COUNCIL, 2012; SEXTON, 2003). Neste último caso, a dissipação térmica devido à elevada corrente gera a vaporização das linhas de metal, fusão dos fios de ligação ou derretimento do silício.

Nos dispositivos CMOS o SEL ocorre através de transistores bipolares parasitas, conforme mostrado na Figura 26. O coletor, base e emissor do transistor npn são formados pelo substrato n-, poço p- e difusão n+, respectivamente. Similarmente, coletor, base e emissor do transistor pnp são formados pelo poço p-, substrato n- e

difusão p+, respectivamente. O circuito de polarização desses transistores é formado pela resistência entre a base e o emissor do transistor pnp e a resistência entre a base e o emissor do transistor npn.

Figura 26 - Transistores bipolares parasitas na tecnologia CMOS bulk.



Fonte: Adaptado de SEXTON, 2013.

A Figura 26 mostra um esquema simplificado do circuito parasita. Um latchup é disparado quando um SEE liga um dos transistores parasitas. Caso o evento ocorra em um transistor pnp, por exemplo, uma parcela de sua corrente de coletor é direcionada à base do transistor npn, que a amplifica. A corrente de coletor do npn aumenta, a queda de tensão sobre RS aumenta, a corrente de coletor do transistor pnp aumenta e, consequentemente, a corrente de base do npn aumenta novamente. Esse ciclo se repete indefinidamente, gerando uma corrente elevada que caracteriza o SEL. O latchup se mantém apenas quando o produto do ganho dos transistores parasitas excede 1, o circuito de polarização mantém as junções base emissor polarizadas diretamente e as linhas de alimentação podem garantir corrente suficiente para manter a condição (SEXTON, 2003).

4.2.3.5 *Single Event Burnout* (SEB)

Single Event Burnout (SEB) é o disparo de um dispositivo semicondutor de potência, geralmente operando em altas tensões, gerando altas correntes e perda do componente.

SEB é um evento em que a colisão de uma única partícula energética induz uma corrente elevada que resulta na queima do dispositivo (JOINT ELECTRON DEVICE ENGINEERING COUNCIL, 2012). Afeta transistores bipolares de potência (GAILLARD, 2011) e principalmente transistores MOSFETs (Metal-Oxide-Semiconductor Field Effect Transistor) de potência (BOUDENOT, 2007), como o DMOSFET (Double-Diffused Metal-Oxide-Semiconductor Field Effect Transistor) (GALLOWAY; JOHNSON, 1996; TITUS, 2013). Devido a esses transistores não estarem no foco desse trabalho, o SEB não é abordado aqui em maiores detalhes.

4.2.3.6 *Single Event Gate Rupture* (SEGR)

Single Event Gate Rupture (SEGR) é a formação de um caminho condutor disparado por uma partícula ionizante que atinge uma região de alto campo elétrico no óxido da porta de um transistor.

SEGR ocorre quando a colisão de uma única partícula energética provoca a ruptura do dielétrico de porta do transistor MOSFET, criando um caminho de condução através do óxido de porta (JOINT ELECTRON DEVICE ENGINEERING COUNCIL, 2012). Nos transistores MOSFETs de potência, frequentemente ocorre simultaneamente com o SEB (SEXTON, 2003). Da mesma forma que o SEB, o SEGR também não é abordado em profundidade nesse trabalho, dada a relação desse efeito com os transistores de potência, os quais não estão em foco.

4.2.3.7 *Single Event Induced Snap-Back* (SESB)

Single Event Snapback (SESB) é similar ao SEL, ocorrendo geralmente em transistores N-MOS, onde uma partícula ao atingir a região do dreno, cria um caminho de alta corrente.

SESB refere-se à ocorrência de uma corrente anormal e elevada no dispositivo, devido à passagem de uma única partícula energética. Essa corrente permanece depois que o disparo é removido e pode causar dano permanente ao dispositivo. Diferentemente do SEL, em que é necessária a existência de uma estrutura pnpn, o SES ocorre através de um transistor bipolar parasita. Não acontece em transistores MOS de canal p, e além de ionização, pode ser ocasionado por sobretensão (SEXTON, 2003; KOGA; KOLASINSKI, 1989).

4.2.3.8 *Single Hard Error* (SHE)

SHE é um *hard error* que está associado à inversão de bits de memória devido à dose total ionizante depositada em dielétricos, através da incidência de uma ou duas

partículas energéticas. Pode ser um erro semipermanente, porque em alguns casos pode ser revertido submetendo o dispositivo danificado à radiação ultravioleta e à altas temperaturas (DUFOUR, 1992). Embora tenha semelhança com o tradicional efeito de TID, o SHE é diferente pois é um efeito de dose total localizada, em que a degradação ocorre no local da colisão da partícula ionizante. Por causa da natureza localizada do mecanismo, a dose total entregue por uma única partícula é chamada de microdose (GALLOWAY; JOHNSON, 1996).

5 TÉCNICAS DE PROTEÇÃO E CONFIABILIDADE

Esta seção apresenta técnicas referentes à mitigação de *soft errors* e também de técnicas para redução de ruídos, distorções de pequenos sinais e interferências externas na cadeia de medição. Todos estes elementos são fundamentais para que o sinal amostrado pelo sensor chegue em boas condições e sem apresentar erros no sistema onde ele será lido e processo.

As técnicas de proteção dos circuitos eletrônicos à radiação são aqui divididas em três grupos distintos: as técnicas em nível de dispositivo estão relacionadas ao processo e tecnologia de fabricação (seção 5.1); em nível de circuito se referem ao projeto do circuito para reduzir a sensibilidade aos efeitos da radiação (seção 5.2); e as técnicas em nível de sistema se preocupam com a arquitetura do sistema (seção 5.3). Além disso e não menos importante, métodos de acomplamaneto e blindagem de ligações e cabos são apresentadas na seção 5.4, enquanto as técnicas de filtragem e condicionamento do sinal são mostradas na seção 5.5.

5.1 Técnicas em nível de tecnologia

A presença de algumas substâncias específicas nos materiais usados durante a fabricação e principalmente no encapsulamento dos circuitos integrados pode provocar *soft errors*. Isótopos radioativos de Urânio e Tório ocorrem naturalmente e emitem partículas alfa quando seus núcleos decaem para um estado de menor energia, as quais são capazes de ionizar o semicondutor. Já o Boro pode ser usado como dopante do tipo p e na formação do dielétrico BPSG empregado também nos encapsulamentos. O isótopo ^{10}B é instável quando exposto aos nêutrons, e neste caso emite núcleos de Lítio e partículas alfa, ambos também capazes de ionizar o semicondutor (BAUMANN, 2001). Assim, uma melhoria significativa na SER dos dispositivos microeletrônicos pode ser obtida através da pureza do material de fabricação, eliminando o Urânio, Tório e o ^{10}B.

No caso dos isótopos de Urânio e Tório, a utilização de um encapsulamento de elevada pureza reduz a emissão de partículas alfa de 5 a 10 partículas/cm².h para aproximadamente 0,001 partículas/cm².h (WANG; AGRAWAL, 2008). Efeitos devido ao ^{10}B. no BPSG podem ser mitigados eliminando completamente o BPSG do processo ou utilizando um ^{10}BPSG enriquecido ao invés do convencional.

Uma abordagem para a mitigação dos efeitos singulares é baseada na redução da carga coletada após o efeito de funil causado por uma colisão, que pode ser feita pela criação de barreiras de potencial com a adição de poços nos dispositivos CMOS. Uma técnica que implementa essa ideia é conhecida como poço triplo, e consiste em utilizar um poço p para o transistor NMOS e isolá-lo do substrato p com um poço n profundo. Estes dois poços, juntamente com o poço n do transistor PMOS, formam os três poços que dão nome a técnica.

Outra abordagem de proteção aos efeitos singulares que também se baseia na redução da carga coletada, e que atualmente já é utilizada em larga escala, é a tecnologia de silício sobre isolante (SOI, Silicon on Insulator). Com base no fato de que em um transistor MOS o transporte de carga no canal ocorre apenas numa região muito superficial do silício e que o restante do volume é usado apenas como suporte mecânico, a tecnologia SOI consiste em separar a superfície ativa do dispositivo do substrato através de uma camada fina (< 300 nm) de isolante, chamada de óxido enterrado (BOX, *Buried Oxide*).

Além disto, a adoção de técnicas *Radiation Hardened-by-Design* (RHBD) com *enclosed layout transistor (*ELT) são uma laternativa adequada para aumentar a tolerância à radiação de CIs, independente da tecnologia do nó. Esta técnica de layout é uma maneira eficaz de reduzir a degração elétrica causada por radiação ionizante em transistores MOS. Geometrias fechadas impedem as regiões de óxido espesso nas boardas do canal, impedindo o aumento de indução de radiação nos caminhos de corrente de fuga (CARDOSO, 2017).

5.2 Técnicas em nível de circuito

As técnicas de proteção em nível de circuito têm uma vantagem sobre as técnicas em nível de dispositivo, pois não necessitam de mudanças fundamentais no processo de fabricação (DODD et al., 2010). Uma técnica básica em nível de circuito consiste na adição de um capacitor no nó a ser protegido. Dado que a carga crítica (Q_{crit}) representa a mínima quantidade de carga coletada (Q_{col}) em um nó necessária para causar um *upset*, a ideia é aumentar a Q_{crit}, de maneira que o pulso de corrente gerado pelo SEE seja dissipado e não se propague pelos estágios seguintes do circuito (BAZE et al., 2002).

De acordo com (CHENET, 2015) a Q_{crit} também pode ser definida conforme a equação 6, em que C_{load} é a capacitância do nó e $V_{noise-margin}$ é a tensão de margem de ruído.

$$Q_{crit} = C_{load} * V_{noise-margin} \quad (6)$$

Assumindo que a tensão transiente $V_{dd}/2$ caracteriza um *upset*, em que V_{dd} é a tensão de alimentação do circuito, $V_{noise-margin}$ pode ser definida conforme a equação 7.

$$V_{noise-margin} = \frac{V_{dd}}{2} \quad (7)$$

Assim, a Q_{crit} pode ser expressa conforme a equação 8.

$$Q_{crit} = C_{load} * \frac{V_{dd}}{2} \quad (8)$$

Se um capacitor é adicionado C_{load} aumenta e a Q_{crit} aumenta, tornando o circuito mais tolerante à radiação. Se a carga depositada $\left(Q_{deposited}\right)$ por um SEE é conhecida, a capacitância adicional requerida (C_{add}) pode ser expressa conforme a equação 9.

$$C_{add} = \frac{C_{load}*(Q_{deposited} - Q_{crit})}{Q_{crit}} \quad (9)$$

O acréscimo de um capacitor pode funcionar como um filtro passa baixa em circuitos integrados com elevadas frequências, alterando seu funcionamento. O capacitor adicional também aumenta o atraso de propagação, o consumo de potência e a área em silício dentro do semicondutor. No caso de elevada carga depositada, um capacitor de valor alto pode ser impraticável (SAYIL, 2010).

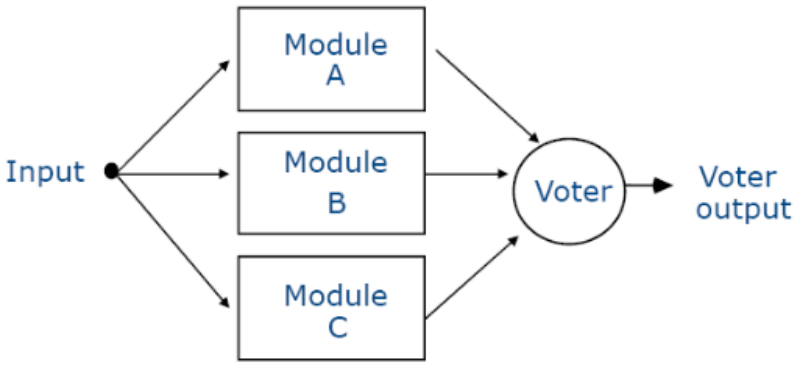
5.3 Técnicas em nível de sistema

A maior parte das técnicas de proteção em nível de dispositivo e em nível de circuito atuam na prevenção de *soft errors*, o que pode não ser suficiente nos circuitos integrados modernos, em virtude de suas características como tensões de alimentação reduzidas, dimensões reduzidas e frequência de clock elevada. Nesse contexto, as técnicas em nível de sistema são a alternativa, pois possuem em grande parte mecanismos de recuperação on-line (WANG; AGRAWAL, 2008). Elas também são adequadas quando não existe o acesso em nível de dispositivo e circuito (DODD et al., 2010). As principais técnicas em nível de sistema podem ser classificadas em primitivas, baseadas em detecção de erro e correção (EDAC, Error Detection and Correction) e baseadas em redundância e votação.

A técnica baseada em redundância e votação amplamente usada é a Redundância Modular Tripla (TMR, *Triple Modular Redundancy*) (VON NEUMANN, 1956), que é uma forma de redundância N-modular tolerante a falhas, na qual três sistemas realizam um processo e esse resultado é processado por um sistema de votação de modo a produzir uma única saída, conforme ilustrado na Figura 27. O sistema

de votação irá seleccionar a saída da maioria dos componentes. Assim, se um componente falhar, o erro não será reflectido na saída do sistema de votação.

Figura 27 – Sistema de redundância modular tripla.



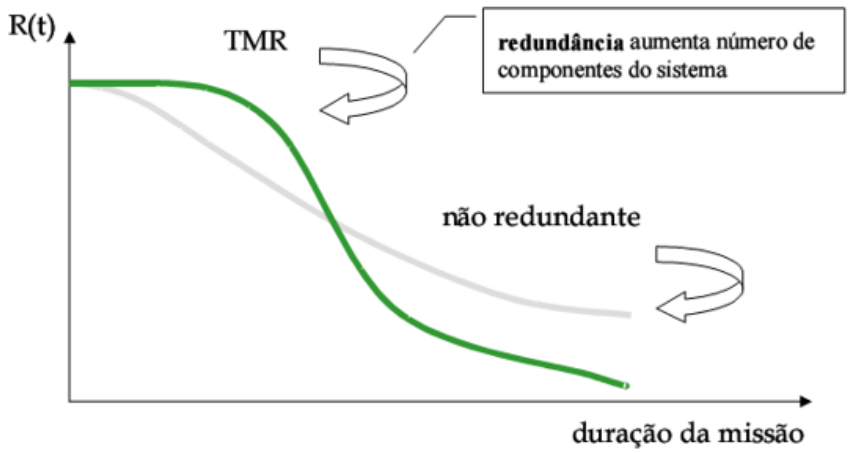
Fonte: Adaptado de Miranda, 2017.

A técnica de diversidade (AVIZIENIS; KELLY, 1984) pode ser usada juntamente com a TMR para aumentar a tolerância à falhas. Nela, os elementos de *hardware* e *software* usados para computação múltipla não são cópias, mas projetos independentes que atendem os critérios do sistema. A diversidade pode ocorrer em três domínios: diversidade de *hardware*, também referida como diversidade espacial; diversidade de *software*, também chamada de diversidade de programa; e a diversidade temporal, que consiste na repetição da computação em instantes de tempo distintos ou com frequência diferente. Essa técnica é especialmente útil para lidar com falhas de modo-comum, devido aos diferentes níveis de resiliência de cada elemento que o projeto contém, sejam eles *hardware*, *software* e/ou tempo. Além disso, outra característica da diversidade que diferentes projetistas e ferramentas empregam em cada cópia, é evitar sistematicamente pontos em comum, o que geralmente faz com que as falhas de projeto não produzam erros similares, aumentando assim sua resiliência. Uma desvantagem dessa técnica é que ela pode adicionar complexidade ao sistema, como por exemplo, a necessidade de um circuito para sincronização dos elementos de computação múltipla.

A TMR no geral é recomendada especialmente para tempos de missões curtos, pois ela ainda é frágil a falhas permanentes, conforme ilustrado na Figura 28. A variável R(t) diz respeito a confiabilidade do sistema e está representado no eixo y, enquanto o tempo de missão está representado no eixo x. Na figura, podemos observar que antes de um certo tempo de missão, que irá depender de diversas características do sistema, a TMR adiciona confiabilidade ao sistema, mas que passado um tempo a sua confiabilidade é menor que um sistema simples pois possui mais elementos suscetíveis a falhas. Portanto, a confiabilidade do sistema só é alta em períodos onde se possa

garantir que irão ser mitigadas apenas falhas transitórias em seus elementos de *hardware*.

Figura 28 – TMR versus tempo de missão.



Fonte: Adaptado de Weber, 2011.

Além disso, um método simples para melhorar a tolerância a erros de conteúdo nos circuitos combinatórios é a redundância temporal, que repete a computação no tempo (WEBER, 2011). Este método evita custo de hardware adicional, aumentando o tempo necessário para realizar uma computação. É utilizado em sistemas onde operações de tempo não são críticas, ou onde o processador trabalha com ociosidade, sendo idealmente aplicada para situações de falhas transitórias. Esse método exige o escalonamento apropriado das tarefas (JACOBS; WULF; GEORGE, 2013), e também consiste no processamento da função mais de uma vez, realizando votações e consolidando o resultado final.

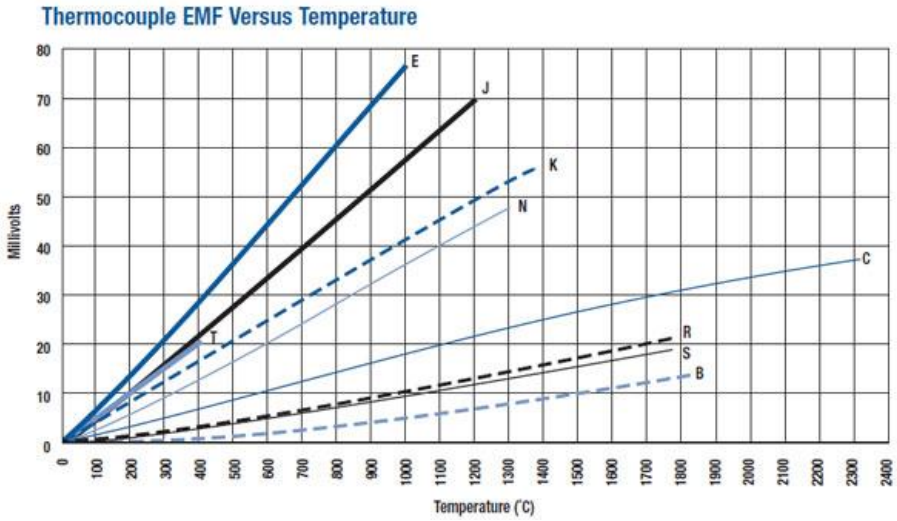
6 CASO DE ESTUDO COM INJEÇÃO DE FALHAS POR EMULAÇÃO

Os estudos desse trabalho adotaram um esquema baseado em TMR e diversidade espacial-temporal para a implementação de um sistema de aquisição de dados (SAD) analógico-digital. O estudo dessa seção utiliza a injeção de falhas por mascaramento de informações para observar o comportamento dos conversores de dados e avaliar a eficácia do esquema adotado baseado em TMR e diversidade. Essa seção está organizada da seguinte forma: em 6.1 é apresentada a cadeia de medição e dados referentes a instrumentação da da leitura do sinal de temperatura; em 6.2 é apresentado o condicionamento do sinal adquirido através de amplificadores operacionais e o filtro analógico aplicado; em 6.3 é detalhado o esquema do SAD adotado; em 6.4 é apresentado o esquemático detalhado completo e a placa de circuito impresso; em 6.5 são apresentadas as características do SoC (System-on-Chip) em que o as falhas são emuladas;

6.1 Cadeia de medição e instrumentação

O presente trabalho utilizou um termopar tipo K, conforme apresentado na seção 3.1, em função de ser o mais utilizado comercialmente para medições de EGT. A temperatura de ensaios será limitada em 370°C para possibilitar a execução dos testes em bancada e em forno, apesar de a metodologia do projeto poder ser aplicada para temperaturas maiores, bastando apenas ajustar o ganho proposto. Conforme Figura 29, podemos observar o comportamento linear do termopar tipo K em função da temperatura, enquanto a **Error! Reference source not found.** mostra que teremos 20,2mV como tensão em fundo de escala em função da temperatura (fenômeno físico).

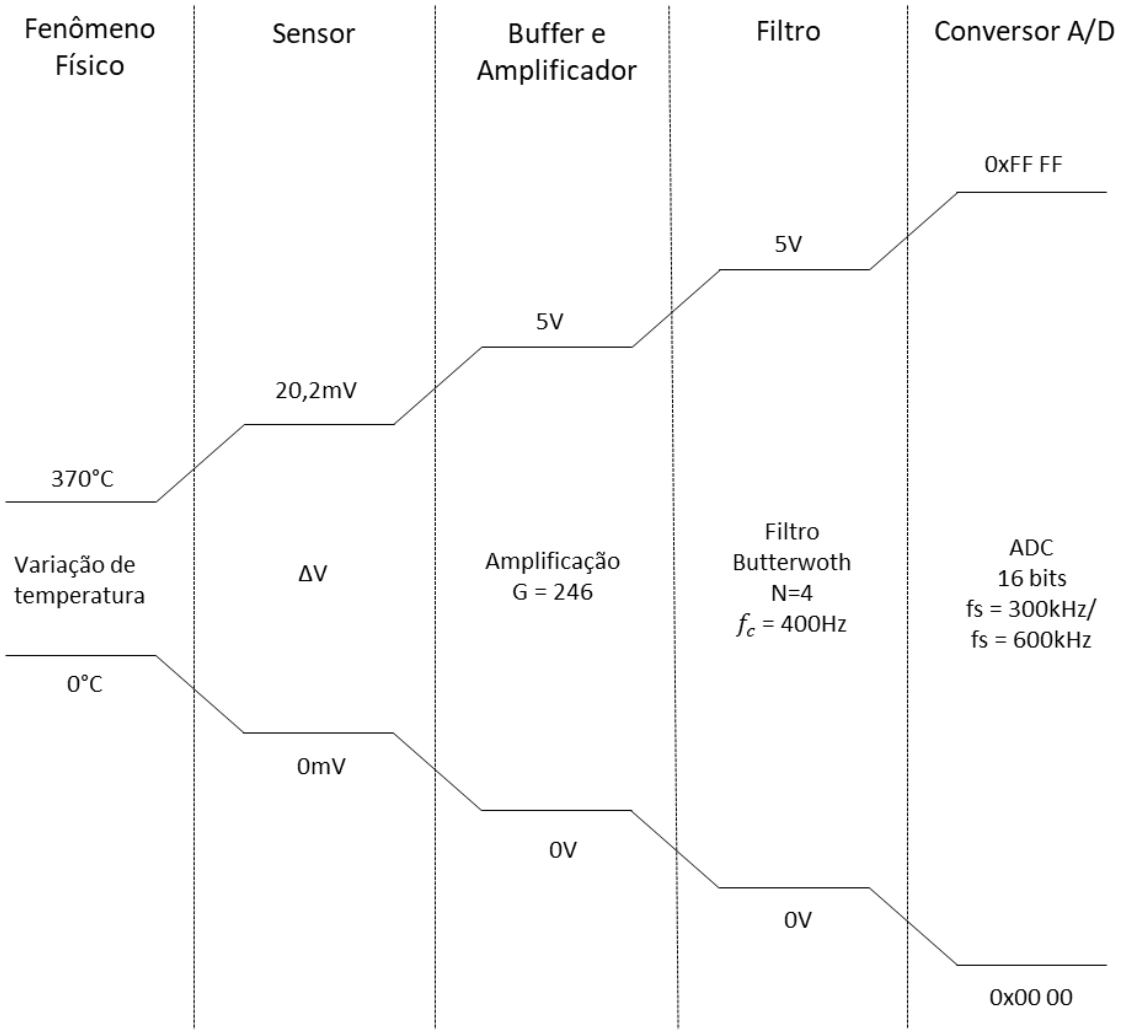
Figura 29 – Temperatura versus tensão dos termopares.



Fonte: Adaptado de KEE, 1999.

A seguir serão demonstrados e discutidos os estágios de processamento típicos, característicos de estarem presentes dentro dos computadores de missão mencionados na seção anterior. A cadeia de medidas mostrada na Figura 30 foi proposta para a elaboração das medições do projeto, apresentado todos os estágios até que a leitura esteja em formato digital para ser avaliada e processada pelos demais dispositivos do sistema.

Figura 30 – Cadeia de medição da variação de temperatura do EGT.

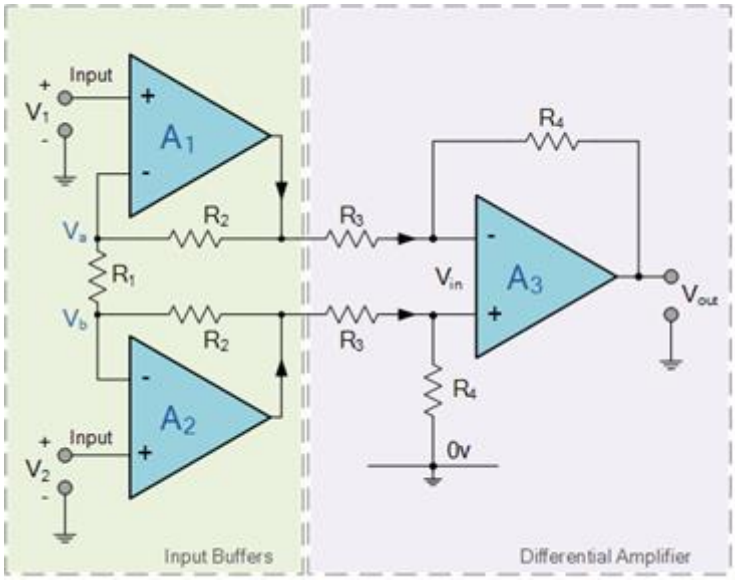


A cadeia de medidas foi dividida em cinco etapas. A primeira etapa diz respeito ao aquecimento e variação de temperatura do EGT, tratando portando do fenômeno físico a ser mensurado. A segunda e terceira etapas estão relacionadas a variação de diferença de potencial do termopar quando sujeitos à variação de temperatura, e engloba todo o sistema de condicionamento de sinais e sua amplificação de instrumentação. A quarta etapa envolve o sistema de filtro ativo de ruídos, e a quinta etapa envolve somente a conversão A/D do sinal adquirido.

6.2 Condicionamento e Filtragem

O sinal do sensor de temperatura, que tem uma saída diferencial, primeiro passará por um amplificador de instrumentação para depois ser amplificado com ganho não inversor de G=246±V/V, cujo esquemático pode ser visto na Figura 31. A equação (1) mostra a saída deste amplificador em função da tensão elétrica de entrada diferencial Vin.

Figura 31 – Esquemático dos estágios do condicionamento de sinal



$$V_{out} = (V_1 - V_2)[1 + \frac{2R2}{R1}] \left(\frac{R4}{R3}\right) \quad (1)$$

Dado que a relação $\left[1 + \frac{2R2}{R1}\right]\left(\frac{R4}{R3}\right)$ resulta num valor número de 223, foram escolhidos resistores R1 = 10kΩ , R2 = 10kΩ , R3 = 10k , R4 = 820k resultando em um ganho de 246. A saída máxima da cadeia de medição, portanto, será de 5V.

Quanto ao design do filtro, foi escolhido um modelo Butterworth, para obtenção de resposta em frequência mais plana na banda passante, se comparado a outros tipos de filtro. Considerando uma frequência de passagem de 0 a 400Hz e com -3dB na frequência de corte, deverá ser utilizado um filtro de ordem quatro no mínimo. Dado que o filtro de quarta ordem atende os requisitos, ele foi o escolhido, onde haverão dois estágios de segunda ordem em sua composição.

O filtro terá como frequência de corte $f_c = 100Hz$, em função da maioria das aeronaves terem energia elétrica proveniente da turbina em115Vac/400Hz. Portanto, dado que o gerador de energia e principal fonte de interferência devido a intensidade

de seu sinal está nesta frequência, ela será o principal elemento a ser filtrado, e o objetivo do filtro é que o sinal já esteja completamente atenuado nesta frequência.

$$H(s) = \frac{A_0}{1+ w_c\,[C_1(R_1+R_2)-(1- A_0)R_1C_2]s+ w^2R_1R_2C_1C_2s^2} \quad (2)$$

Assumindo que o ganho será unitário, $A_0 = 1$, portanto a equação (2) poderá ser simplificada conforme abaixo:

$$H(s) = \frac{1}{1+ w_c\,C_1(R_1+R_2)s+ w^2R_1R_2C_1C_2s^2} \quad (3)$$

Para resolver o polinômio, os valores de C_1 *e* C_2 serão estimados primeiro, para que, então, a relação obtida pode ser manuseada para a obteção dos valores de resistores:

$$R_{1,2} = \frac{a_1\,C_2 \pm \sqrt{{a_1}^2{C_2}^2-4b_1C_1C_2}}{4\Pi f_c\,C_1C_2} \quad (4)$$

Devido a necessidade de se obter valores reais para a raiz acima, C_2 deve obedecer a seguinte relação:

$$C_2 \geq C_1\,\frac{4b_1}{{a_1}^2} \quad (5)$$

Assumindo C_1 =100nF, teremos que $C_2 = 117\text{nF}$ através da solução da equação (5). Dado que o valor não é comercial, o valor escolhido de maneira a viabilizar a montagem do circuito é de $C_2 = 120\text{nF}$.

Resolvendo a equação (4) para a obtenção dos valores de resistores, teremos que $R_1 = 4954\Omega$ e R2 = 3636Ω. Com isto, todos os valores do primeiro estágio do filtro foram selecionados.

Quanto ao segundo estágio do filtro, é necessário apenas mudar os valores dos coeficientes para a_2 *e* b_2. Escolhendo C_1 =1,5nF, teremos que $C_2 = 10{,}3\text{nF}$. Repetindo a metodologia para obtenção dos resistores, teremos $R_1 = 133\text{k}\Omega$ e R2 = 110kΩ. O desenho final do filtro completo com ambos seus estágios pode ser visto na Figura 32, enquanto a Figura 33 apresenta a resposta simulada deste filtro utilizando o WEBENCH da empresa *Texas Instruments*. É importante notar que, baseado que os valores dos resistores encontrados não são comerciais, foram adicionados trimpots para um ajuste de precisão ao filtro.

Figura 32 – Filtro butterworth de quarta ordem com dois estágios.

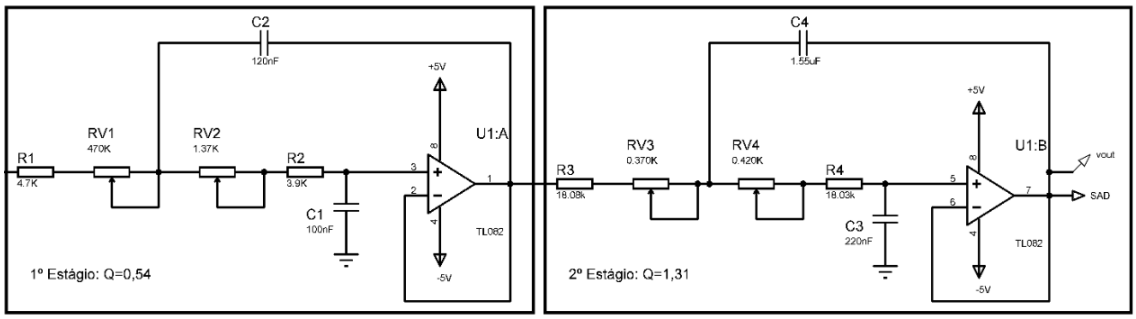
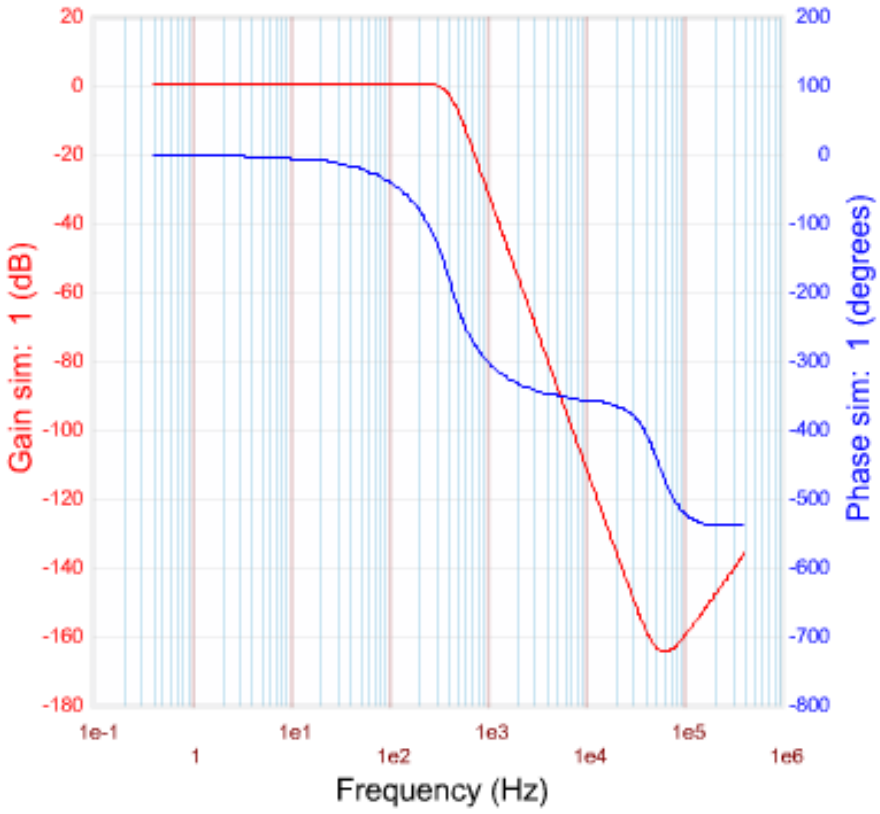


Figura 33 – Resposta simulada do filtro em ganho e fase.

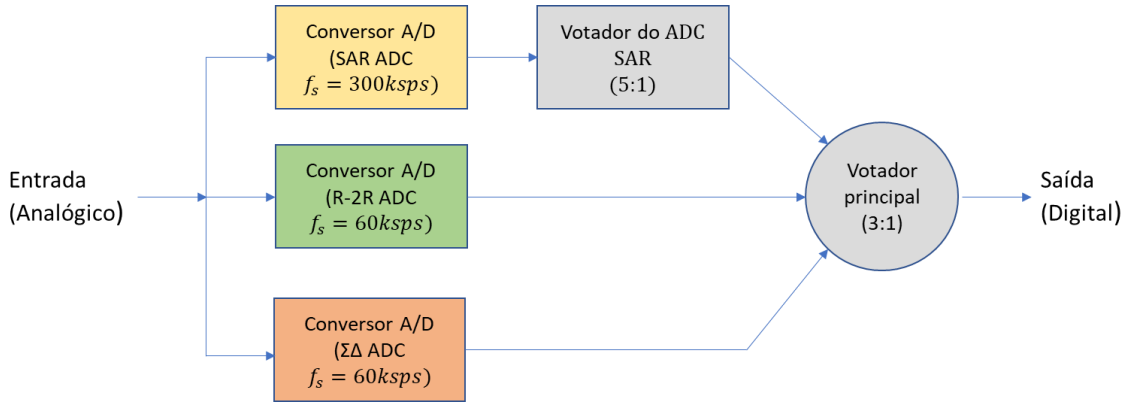


6.3 Sistema de aquisição de dados com TMR e diversidade

O sistema de aquisição de dados (SAD) adotado com base em TMR e diversidade é mostrado na Figura 34. Ele usa três conversores analógico-digitais (ADCs*, Analog to Digital Converters*) em paralelo: 1 com arquitetura de registrador de aproximações sucessivas (SAR, Successive *Approximations Register)* 1 com arquitetura Sigma-Delta e 1 com arquitetura R-2R. O ADC R-2R e o ADC Sigma-Delta operam com taxa de amostragem de 60 ksps e o ADC SAR opera à 300 ksps. O sistema é composto também por dois votadores: o votador principal, que realiza a votação entre os três conversores

e um votador chamado de votador do ADC SAR, que realiza a votação com 5 amostras geradas pelo ADC SAR que opera a 300 ksps. Nesse esquema adotado cada cópia do ADC compõem a abordagem TMR. As três arquiteturas diferentes dos conversores caracterizam a diversidade espacial, e as duas taxas de amostragem diferentes a diversidade temporal.

Figura 34 – Sistema de aquisição de dados com TMR e diversidade.



6.3.1 Conversores A/D

Como mostrado na Figura 34, os conversores A/Ds foram escolhidos de forma a compor um sistema de diversidade. Todos os A/Ds selecionados tem como característica principal uma resolução de saída de 16 bits, afim de não alterar a precisão do valor lido após a votação, e também ter uma saída SPI (*Serial Peripheral Interface*) de comunicação serial, que será utilizada para aquisição dos dados. Portanto todo o sistema de aquisição de dados é feito externamente, em *hardware* dedicado. O controle de sincronismo e o processamento destes dados é mostrado na seção 6.5, onde um microcontrolador foi escolhido para fazer estas funções, comportando-se como um computador de missão em aplicações aviônicas típicas.

6.4 Placa de circuito impresso

Com os A/Ds definidos, a placa de circuito impresso foi confecionada. A ideia do projeto aqui é definir o sistema em dois grandes blocos: O primeiro bloco de aquisição do sinal, contendo o amplificador de instrumentação, condicionamento, filtros e conversão A/D; O segundo bloco de processamento de dados, que é apresentado na seção 6.5. Dado que a maior parte dos equipamentos aviônicos é baseado no conceito de conter diversos *Shop-Replaceable Unit* (SRU) dentro de unidades chamadas de Line-*Replaceable Unit* (LRU), podemos encarar o sistema proposto como a composição de duas SRUs: A SRU de EICAS e a SRU de processador.

A Figura 35 mostra o esquemático desenvolvido, a Figura 36 mostra o *layout* da placa, enquanto a Figura 37 apresenta uma fotografia da confecção da placa.

Figura 35 – Esquemático elétrico do sistema de aquisição de dados.

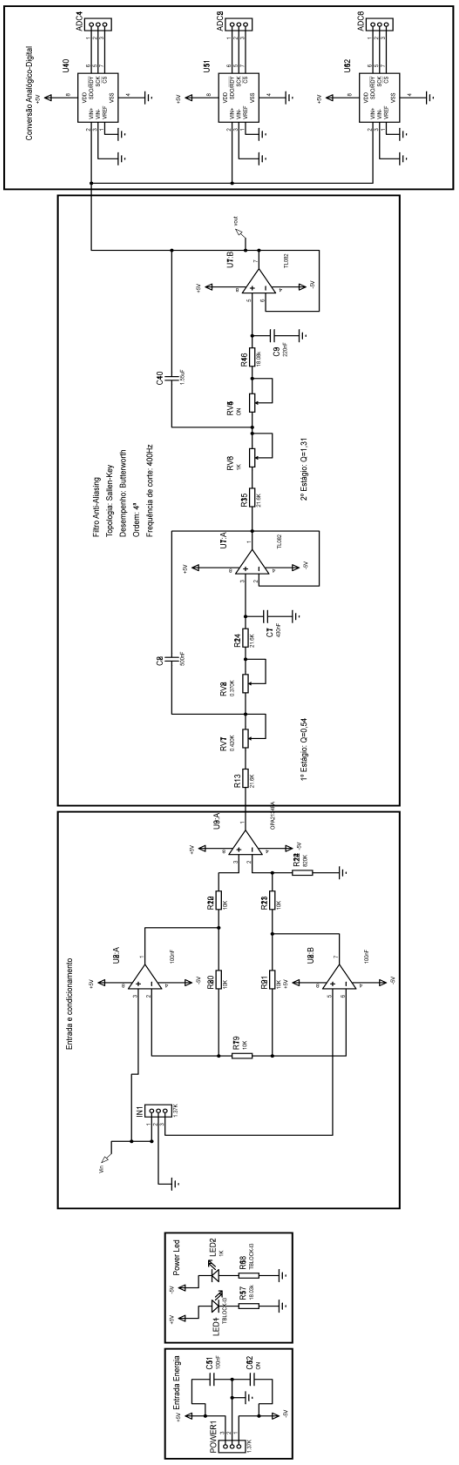
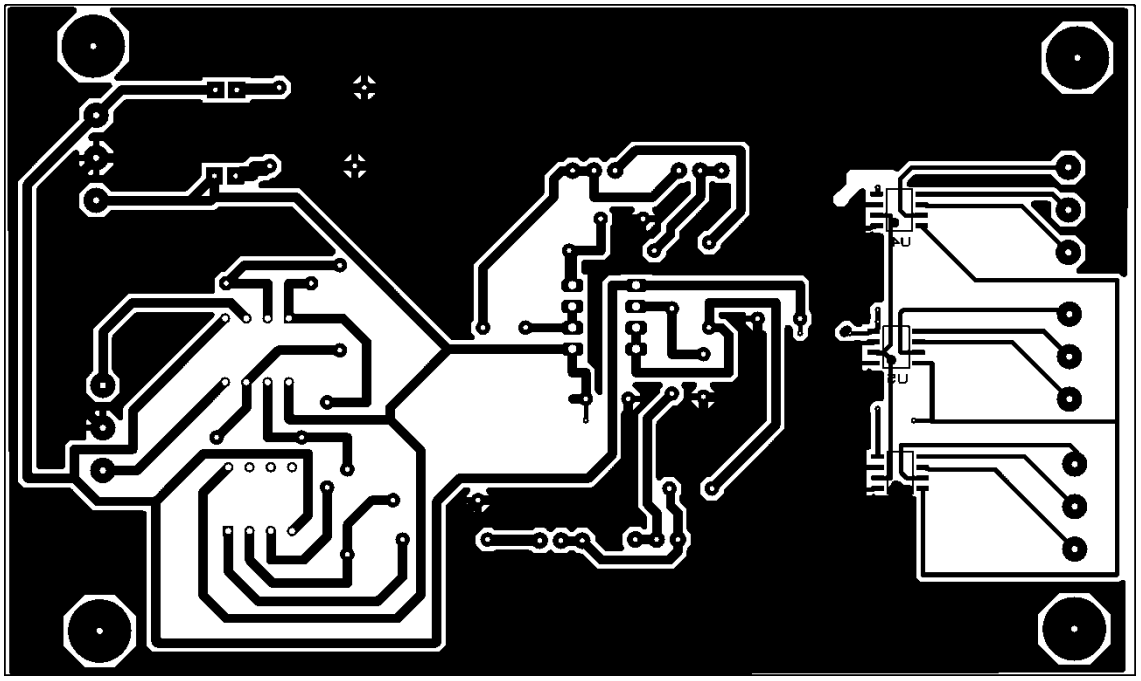


Figura 36 – Placa de circuito impresso.



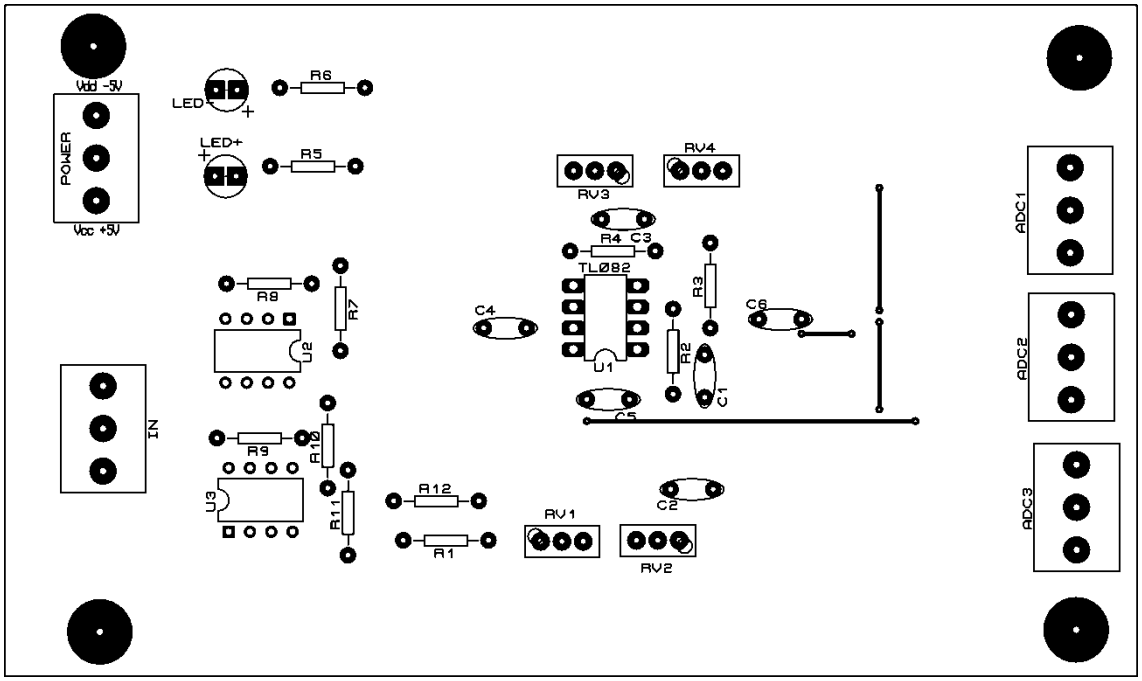
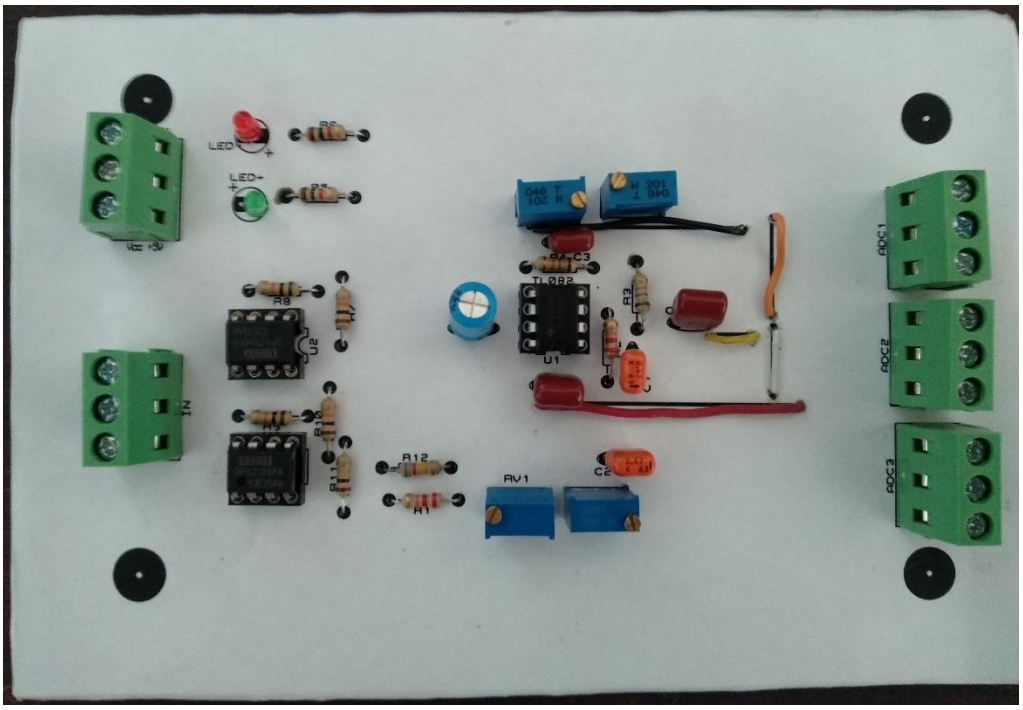
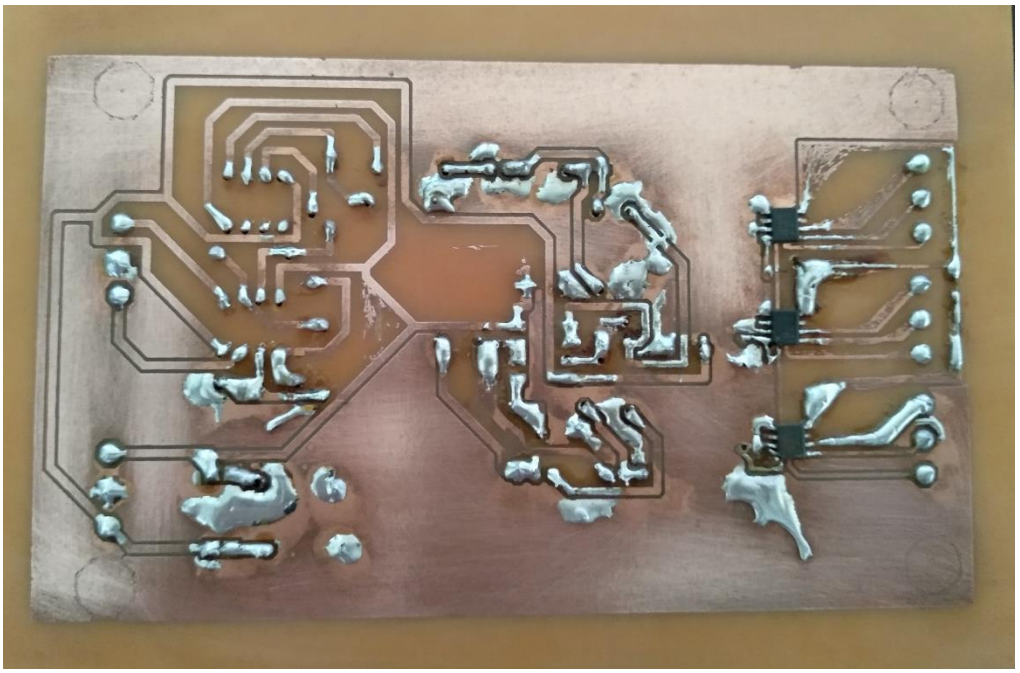


Figura 37 – Fotografia da placa confecionada.





6.5 PSoC 5LP

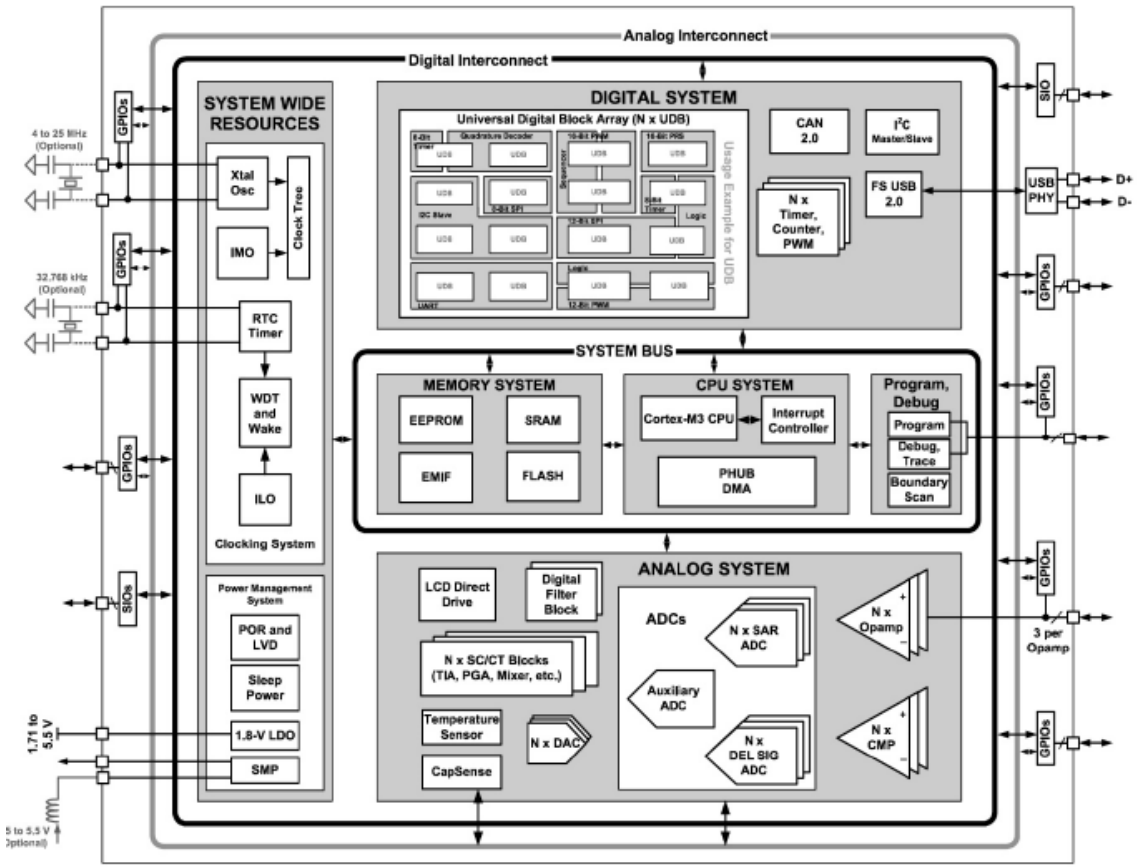
O SAD dos estudos desse trabalho foi totalmente implementado com ADs externos e auxílio de um SoC comercial, programável e de sinais mistos, chamado PSoC 5LP (CYPRESS SEMICONDUCTOR, 2013a). O PSoC será importante para executar a comunicação serial com o computador, a injeção de falhas através de mascaramento,

além de executar o sincronimos da leitura dos três conversores ADs que estão na placa de aquisição de sinais, externas ao PSoC, que serão apresentados na próxima seção deste projeto.

O diagrama de blocos da arquitetura desse circuito integrado é mostrado na Figura 38, onde o part number comercial do PSoC 5LP utilizado é o CY8C5588AXI-060. Esse dispositivo é fabricado em tecnologia CMOS de 130 nm e consiste de uma CPU ARM Cortex-M3 de 32 bits e 80 MHz, memória flash de 256 KB, memória SRAM de 64 KB, memória EEPROM de 2 KB e 24 canais de DMA (*Direct Memory Access*). Além disso, contém periféricos digitais como interfaces de comunicação e PLDs (*Programmable Logic Devices*) baseados em UDBs (*Universal Digital Blocks),* os quais proporcionam a implementação de diversas funções como temporizadores, contadores e outros.

Todos esses periféricos estão disponíveis em uma biblioteca de componentes pré-construídos presente na ferramenta de software PSoC Creator, utilizada em projetos com o PSoC 5LP. A edição de projetos no PSoC Creator é feita por meio de esquemático, usando os componentes pré-construídos, e também por *software* em linguagem C.

Figura 38 – Diagrama de blocos do PSoC 5LP.



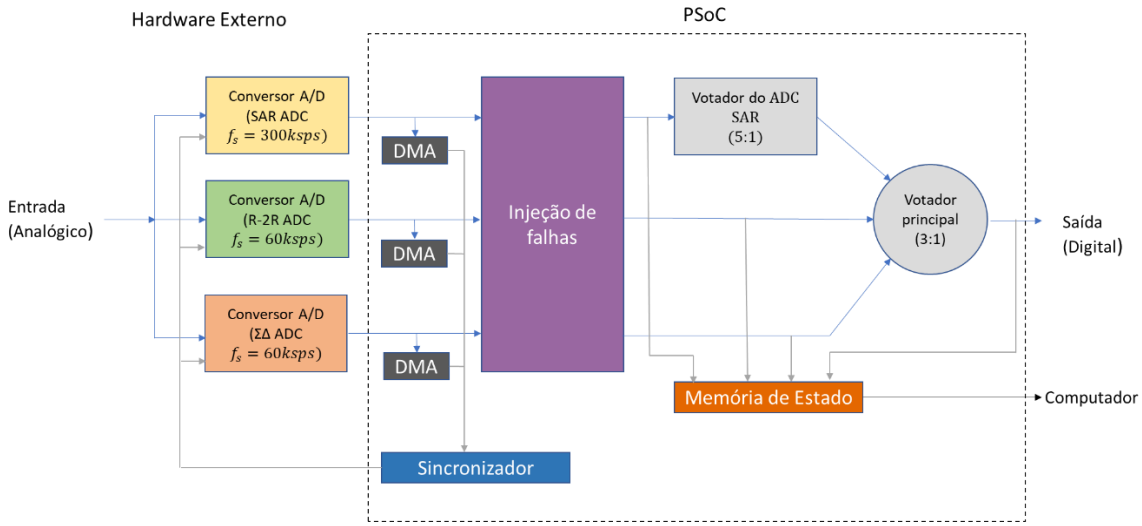
Fonte: Adaptado de Cypress Semiconductor, 2013.

6.5.1 Implementação no PSoC

A Figura 39 mostra o detalhamento da implementação elaborada no PSoC 5LP. Além dos votadores inicialmente previstos, a implementação demanda também circuitos de acesso direto à memória DMA e um sincronizador. O DMA auxilia o sincronizador por sinalizar quando a tarefa foi concluída e não utiliza o processador, reduzindo os atrasados impostos pelo mesmo. Além disso, para atender aos objetivos do estudo, são também necessários um bloco de injeção de falhas e um bloco de registro de estado. O restante do hardware, onde estão os módulos ADs, filtros e aquisição do sinal estão em uma placa externa, conforme apresentado nas seção 6.4.

É importante mencionar que para o escopo do projeto, não foram adicionados mostradores, porque o atraso entre as leituras dos diferentes ADs não é significativo comparado a inércia térmica do motor e seu gás de exaustão. Outras aplicações onde a leitura analógica é mais dinâmica, ou onde o transdutor tem por caracteristica ter um comportamento de tensão alternada, demandariam amostradores para garantir que todas as leituras fossem executadas com base no mesmo parâmetro e no mesmo instante de tempo.

Figura 39 – Detalhes da implementação do SAD com auxílio do PSoC 5LP.

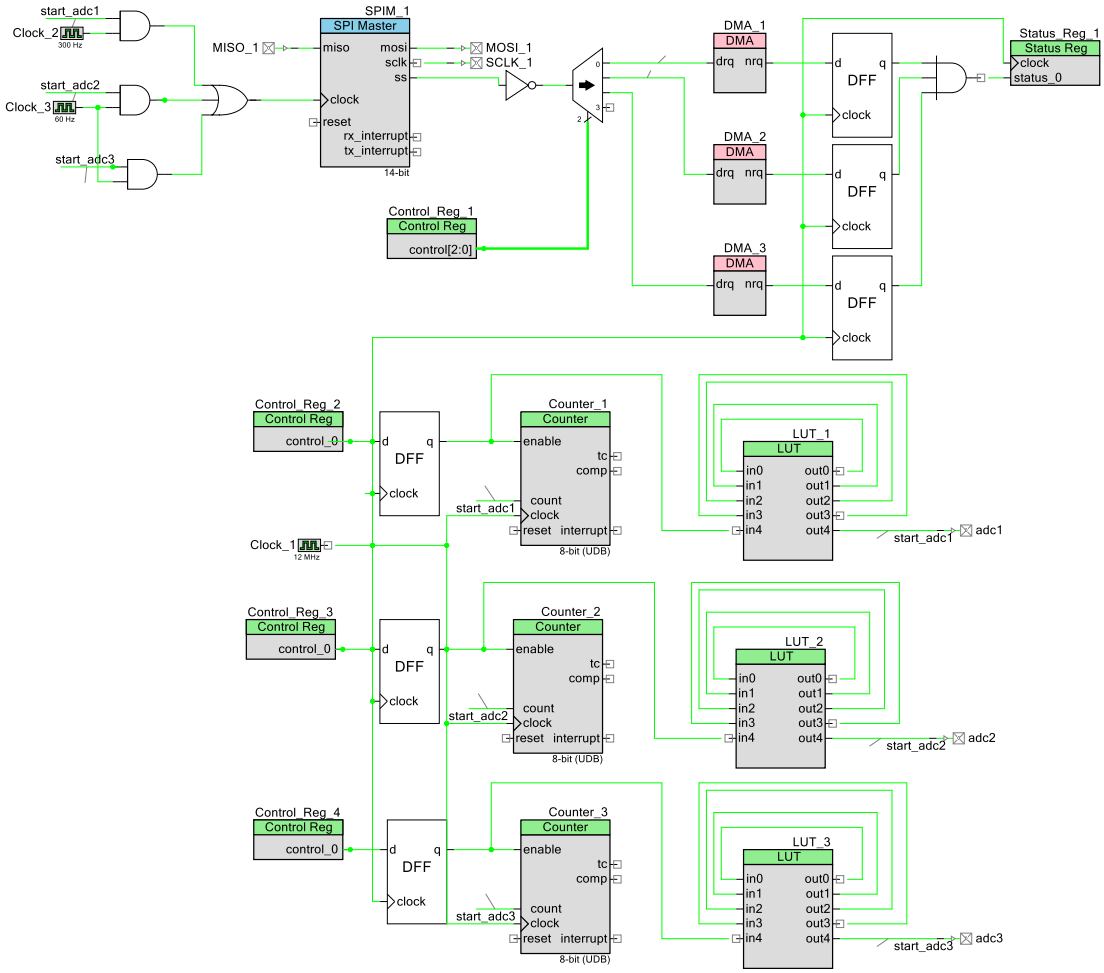


A Figura 40 mostra os principais blocos inseridos no esquemático da ferramenta PSoC Creator para a implementação do SAD. Os votadores não são representados porque são implementados inteiramente em software.

Sempre é válido ressaltar que o PSoC trabalha também com software em linguagem C. Portanto parte do sistema proposta na Figura 39 não é representado nos blocos do PSoC creator porque foi implementado puramente em código C, como é o caso dos votadores, da injeção de falhas e da comunicação entre o PSoC e a porta USB

do computador. De qualquer forma, na seção a seguir cada bloco é abordado em profundidade, onde é descrito a forma de implementação que foi escolhida, o motivo principal, sua funcionalidade e detalhamentos de sua elaboração.

Figura 40 – Blocos utilizados no PSoC Creator para implementação do SAD.



6.5.1.1 Leitura Serial

Para aquisição dos dados dos conversores foi utilizado um padrão de comunicação serial chamado de SPI (*Serial Peripheral Interface*). SPI é uma especificação de interface de comunicação série síncrona usada para comunicação de curta distância, principalmente em sistemas embarcados. A interface foi desenvolvida pela Motorola e tornou-se um padrão de facto.

Os dispositivos SPI comunicam entre si em modo "full duplex" usando uma arquitectura "master-slave" com um único mestre, onde neste caso o PSoC é o mestre principal da rede. O dispositivo mestre inicia a comunicação para a leitura dos

conversores. Múltiplos dispositivos escravos são suportados através de selecção com linhas de selecção de escravos individuais.

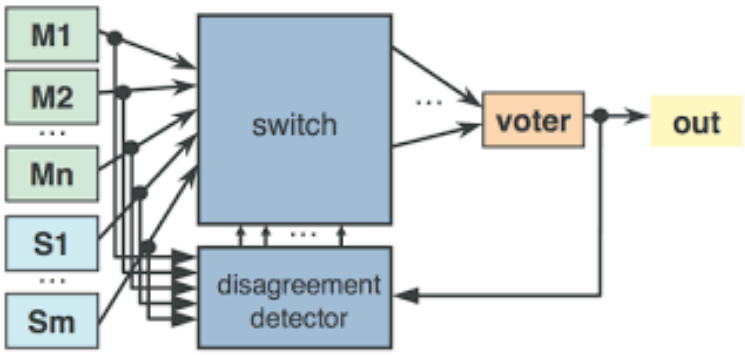
Em modo "escravo", o microcontrolador comporta-se como um componente da rede, recebendo o sinal de Clock. Em modo "mestre", o microcontrolador gera um sinal de relógio e deve ter um pino de entrada/saída para habilitação de cada periférico. Toda configuração do SPI é feita diretamente através do bloco inserido no PSoC Creator.

6.5.1.2 Votadores

O conceito básico de votação adotado na implementação do votador principal e do votador do ADC SAR foi o de votação de maioria. Uma discussão sobre algumas das principais técnicas de votação é apresentada em Lorczak, Caglayan e Eckhardt (1989). A votação de maioria é a técnica usada com mais frequência, embora apresente a desvantagem de implicar em uma perda na precisão, no caso específico deste sistema proposto. Essa perda ocorre devido a necessidade de um limiar para a construção de uma janela de tolerância, que permite determinar a maioria quando o dado a ser votado é aproximado, mas não exatamente o mesmo, como ocorre em grande parte dos sistemas reais.

O votador de palavra, como ilustrado na Figura 41 e com código C apresentado na Figura 42, consiste de três comparadores e um elemento de decisão. Os comparadores realizam a comparação mútua por subtração decimal entre as saídas das três cópias redundantes (compreendendo todas as combinações possíveis), produzindo três sinais de erro. Com base nesses sinais de erro, o elemento de decisão seleciona o valor correto para a saída do votador. No votador de palavra a integridade dos dados é maior no contexto de falhas de modo-comum e múltiplas falhas, porque ele pode detectar condições errôneas que no bit-a-bit produzem saídas incorretas. Nesse caso, o votador de palavra pode gerar uma indicação para que uma ação apropriada seja iniciada.

Figura 41 – Exemplo de votador de palavra com indicação de erro.



Fonte: Adaptado de Troger, 2015.

Figura 42 – Código do votador principal.

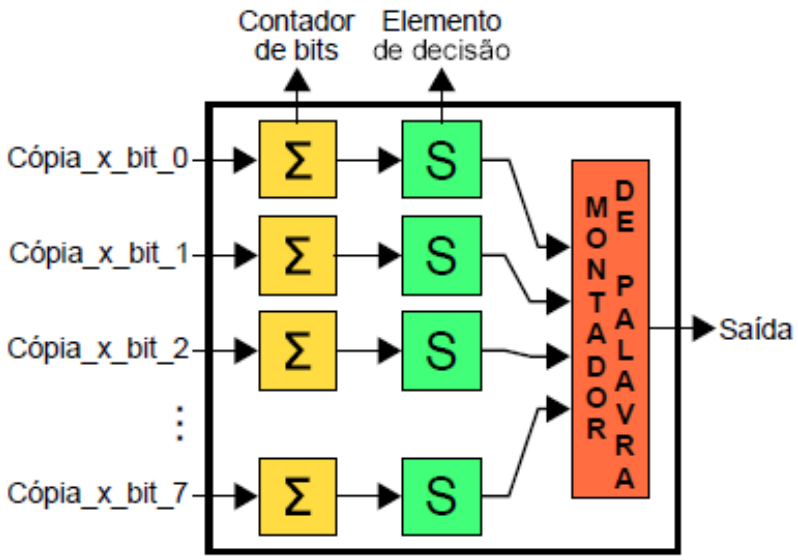
```
void main_voter()
{
        error12 = abs (SAR_voter_data - module2Data[1]);
        error23 = abs (module2Data[1] - module3Data[1]);
        error31 = abs (module3Data[1] - SAR_voter_data);
        tolerance = 8;

        if (error12 <= tolerance)
                voter_output = SAR_voter_data;
        else if (error23 <= tolerance)
                voter_output = module2Data[1];
        else if (error31 <= tolerance)
                voter_output = module3Data[1];
        else
                voter_output = 0;

        if ((error12 > tolerance) || (error21 > tolerance) || (error31 > tolerance))
                voter_error_det = 1;
}
```

A implementação de votadores de maioria no domínio digital pode ser feita seguindo os conceitos de votação bit-a-bit ou votação de palavra (MITRA; MCCLUSKEY, 2000). O votador do ADC SAR foi baseado no conceito de votação bit-a-bit, ilustrado na Figura 43, devido ao fato de que essa escolha é vantajosa devido ao número de entradas (5 entradas). O código em linguagem C que faz essa implementação é mostrado na Figura 44. Em caso de em alguma posição de bit serem detectados estados lógicos diferentes entre as palavras, uma indicação (SAR_voter_error) é gerada.

Figura 43 – Votação bit-a-bit.



Fonte: Adaptado de Chenet, 2015.

Figura 44 – Código em C da votação bit-a-bit.

```
void SAR_voter()
{
        SAR_voter_data = 0;
        bits = 16;
        samples = 5;
        tolerance = 6;
        for (i = 0; i < bits; i++)
                bit_counter[i] = 0;

        for (i = 0; i < bits; i++)
        {
                for (j = 1; j < samples; j++)
                {
                        bits = (module1Data[j] & mask[i]) != 0;
                        if (bits == 1)
                        bit_counter[i] = bit_counter[i] + 1;
                }
        }

        for (i = 0; i < bits; i++)
        {
                if (bit_counter[i] > tolerance)
                SAR_voter_data = SAR_voter_data + mask[i];
        }

        bigger = module1Data[1];
        smaller = module1Data[1];
        for (pointer = 2; pointer < samples; pointer++)
        {
                if (bigger < module1Data[pointer])
                bigger = module1Data[pointer];
                if (smaller > module1Data[pointer])
                smaller = module1Data[pointer];
        }
        error = bigger - smaller;
        if (error > tolerance)
        SAR_voter_error_det = 1;
}
```

6.5.1.3 Circuitos de acesso direto à memória

O DMA é um recurso de *hardware* para armazenar na memória dados gerados ou lidos pelos periféricos sem envolver o processador nessa tarefa. As vantagens de usar o DMA são de liberar o processador para outras tarefas e reduzir o atraso imposto pelo processador, uma vez que o DMA é um *hardware* dedicado e portanto mais rápido. No SAD foram usados três DMAs, um para cada ADC. Com base nisto, foi utilizado um periférico de leitura SPI que irá escrever na DMA os valores obtidos. Além disto, elas geram uma indicação de pronto para que o sincronizador inicie um novo ciclo de votação e aquisição de dados. Sua configuração foi feita através de código no PSoC, e encontra-se disponível no apêndice A.

6.5.1.4 Sincronizador

O conceito de sincronização usa o termo ciclo de votação, que se refere ao intervalo de tempo entre a geração pelas cópias de sucessivas saídas que estão em votação. Para formalizar a noção de sincronização, considera-se um conjunto de duas ou mais cópias, cada uma tendo n ciclos de votação em um intervalo de tempo. Sequencialmente numeram-se os ciclos de votação de cada cópia, começando com 1. Um conjunto de cópias é dito sincronizado se para cada i, com $1 \leq i \leq n$, existe um

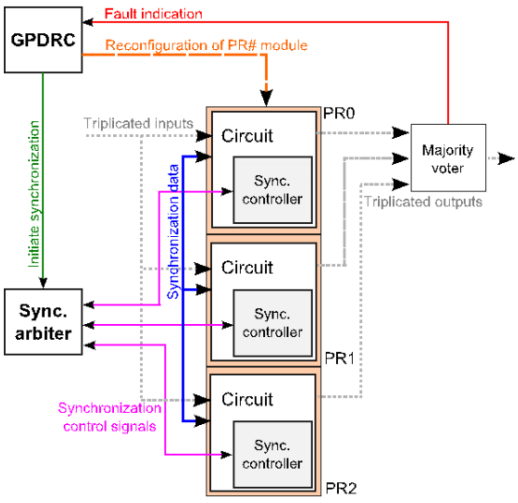
intervalo de tempo não nulo em que cada uma das cópias está no ciclo de votação i. Assim, essa noção caracteriza o comportamento temporal das cópias TMR (DAVIES; WAKERLY, 1978). A sincronização é um ponto chave na técnica TMR, principalmente quando utiliza-se diversidade, pois ela afeta a controlabilidade e complexidade do sistema.

Davies e Wakerly (1978) classificam a sincronização em três tipos: bases de tempo independentes e precisas, fundamentado na precisão das bases de tempo individuais; referência externa comum; e realimentação mútua, em que as cópias monitoram umas às outras e corrigem os desvios por elas próprias. A técnica de votação sincronizada é proveniente do tipo realimentação mútua.

O modelo de sistema baseado na técnica de realimentação mútua é ilustrado na Figura 45. Cada cópia TMR tem uma comunicação com o sincronizador contendo um sinal de entrada e um sinal de saída discretos. A entrada informa à cópia que o ciclo de votação iniciou e a saída indica que a cópia concluiu o ciclo e está pronta para iniciar o próximo. A saída pode ser usada por um sincronizador para iniciar a votação e portanto controlar o ciclo de votação. Desta forma, existe controle total de quando uma leitura foi iniciada, quando foi concluída, e quando a votação pode ser iniciada dado que todos os dados foram adquiridos e o ciclo pode ser encerrado.

Devido à flexibilidade de implementar a técnica de votação sincronizada na aplicação e os recursos disponíveis no PSoC 5LP que facilitam a implementação, essa foi a técnica adotada. Como os conversores externos realizam uma conversão por disparo de clock para enviar o bit via SPI, o sincronizador dispara o enable dos A/Ds bem como seu sinal de clock, com base na indicação de pronto fornecida pelos DMAs e pelo próprio enable do A/D. A implementação do sincronizador foi feita parte em software e parte em hardware, onde a parte em software encontra-se distribuída ao longo do código em C no apêndice A e a parte em hardware pode ser visualizada no esquemático presente na Figura 40.

Figura 45 – Exemplo de sistema de sincronização.



Fonte: Adaptado de Szurman, 2017.

6.5.1.5 Emulação de Falhas

Um bloco de emulação de falhas foi implementado para reproduzir a ocorrência de *soft-errors* no sistema de aquisição de dados em questão. Ele utiliza dois registradores do PSoC 5LP para gerar interrupções e inverter bits do dado gerado na saída dos A/D. Um dos registros escolhe o bit a ser mascarado e o segundo registro escolhe qual das leituras obtidas, agora disponíveis na DMA, que terá o bit invertido. O mascaramento basea-se em realizar uma uma operação lógica XOR (Ou exclusivo) entre o registro e a variável auxiliar que contém o bit-flip que deseja-se aplicar. Além disto, foi definido um gerador aleatório, que irá escolher em quais ADs ocorrerá o bit-flip, caso ocorra, e um segundo gerador aleatório onde se escolhe qual o mascaramento que será aplicado.

O primeiro gerador aleatório gera um valor randômico de 0 a 5. Quando valores de 0 a 2 são geradores, nenhum bit-flip ocorrerá. Desta forma, teremos uma probabilidade de 50% de que nenhum bit-flip aconteça, e os mesmos 50% de que ocorrerá uma inversão de bit para emular um SEE. De 3 a 5, um gerador auxiliar irá determinar se o mascaramento será simples, duplo, ou triplo, onde neste projeto se determinou uma probabilidade de 16% da falha ser duplo (aproximadamente um terço de 50%) e 5% da falha ser tripla (aproximadamente um terço da probabilidade de falha dupla). Este gerador também é importante porque em caso de falhas simples, existe uma legenda onde 3 = A/D SAR, 4 = Delta-Sigma e 5 = R-2R, e em caso de falhas duplas 3 = A/D SAR e Delta-Sigma, 4 = Delta-Sigma e R-2R e 5 = A/D SAR e R-2R, portanto seu valor é mantido até o final da execução do código.

O segundo gerador aleatório gera um valor randômico de 0 a 15, correspondendo ao bit que será mascarado. Desta forma, o registrador mascaramento conterá uma palavra de valor 0b000000000000 e o valor lógico “1” na posição determinada por este gerador. Uma função auxiliar foi criada para fazer a conversão do valor decimal para a máscara desejada. Ao final do processamento desta rotina, são executadas operações lógicos ou exclusivo entre o mascaramento e os registros de memória que contém os valores lidos pelo A/Ds com a finalidade de obter-se o valor final com a emulação da falha. Além disto, para o A/D SAR, que contém 5 amostras, a falha foi emulada em todas as suas leituras, com o intuito de levar em consideração que os efeitos singulares causariam um transtorno maior que apenas um período de clock do sistema. A Figura 46 mostra o código em C que executa a rotina descrita nesta seção.

Figura 46 – Código em C para emulação de SEEs.

```
void masker()
{
	long mask1 = 0x0000;                                                    //0b0000000000000000
	long mask2 = 0x0000;                                                    //0b0000000000000000
	long mask3 = 0x0000;                                                    //0b0000000000000000
	int lower = 0;                                                          // 0 a 2 sem bit-flip - 50%
	int upper = 5;                                                          //3 a 5 com bit-flip - 50%
	int simples = 0;
	int duplo = 0;
	int triplo = 0;
	int maxbits = 15;                                                       //16 bits - 1, contagem inicia em 0
	int aux = 0;

	int num = (rand( ) % upper + 1 - lower)) + lower;

	if (num >= 2)                                           //ocorrera mascaramento
		{

		duplo = (rand( ) % upper + 1 - lower)) + lower;                                 //16%
		if (duplo == 5)                                 //minimo duplo
			{

			upper = 20;
			triplo = (rand( ) % upper + 1 - lower)) + lower;            //5%
			if (triplo == 30)
				{
				aux = (rand( ) % maxbits + 1 - lower)) + lower;
				long masking (aux, mask);
				mask1 = mask;
				aux = (rand( ) % maxbits + 1 - lower)) + lower;
				long masking (aux, mask);
				mask2 = mask;
				aux = (rand( ) % maxbits + 1 - lower)) + lower;
				long masking (aux, mask);
				mask3 = mask;
				}
			else
			if (num == 3)
				{
				aux = (rand( ) % maxbits + 1 - lower)) + lower;
				long masking (aux, mask);
				mask1 = mask;
				aux = (rand( ) % maxbits + 1 - lower)) + lower;
				long masking (aux, mask);
				mask2 = mask;
				}
			if (num == 4)
				{
				aux = (rand( ) % maxbits + 1 - lower)) + lower;
				long masking (aux, mask);
				mask2 = mask;
				aux = (rand( ) % maxbits + 1 - lower)) + lower;
				long masking (aux, mask);
				mask3 = mask;
				}
			if (num == 5)
				{
				aux = (rand( ) % maxbits + 1 - lower)) + lower;
				long masking (aux, mask);
				mask1 = mask;
				aux = (rand( ) % maxbits + 1 - lower)) + lower;
				long masking (aux, mask);
				mask3 = mask;
				}
			}
		else
		if (num == 3)
			{
			aux = (rand( ) % maxbits + 1 - lower)) + lower;
			long masking (aux, mask);
			mask1 = mask;
			}

		if (num == 4)
			{
			aux = (rand( ) % maxbits + 1 - lower)) + lower;
			long masking (aux, mask);
			mask2 = mask;
			}

		if (num == 5)
			{
			aux = (rand( ) % maxbits + 1 - lower)) + lower;
			long masking (aux, mask);
			mask3 = mask;
			}

	SAR_voter_data ^= mask1;
	module2Data[1] ^= mask2;
	module3Data[1] ^= mask3;
}
```

Nesse modelo uma falha é inserida durante aproximadamente 500 milissegundos, ficando portanto ativa por um período consideravelmente maior à duração de um pulso transiente de um soft error, o qual é da ordem de pico a nanossegundos. A escolha desse modelo leva em consideração a possibilidade da duração de um SET ser maior do que a duração do seu pulso transiente, devido a este pulso ser capturado por um latch ou afetar algumas aproximações sucessivas do A/D tipo SAR. Esta duração também é importante para efeitos práticos da aplicação do projeto, onde os computadores de missão teriam tempo útil de compartilhar informações e possivelmente identificarem que suas leituras estão inconsistentes se comparadas entre si. Além disto, falhas de curta duração, como por exemplo por um período de clock, seriam facilmente toleradas pela redundância temporal do A/D SAR, portanto foi escolhida uma duração superior a redundância temporal para avaliar a resiliência do conjunto como um todo.

6.5.1.6 Registros e monitoração

Um bloco contendo diversas memórias de registrado de estados foi implementado para fins de monitoração e relatório, para que os dados sejam analisados durante e posteriormente aos testes. Este bloco é composto por um buffer, um controle de escrita e uma interface serial RS-232/UART, que será utilizada para leitura de dados através de um computador de monitoração. O buffer armazena os dados dos 100 ciclos de votação mais recentes, guardando as seguintes informações: valores de saída dos três conversores, valor de saída do votador principal e do votador do ADC SAR, valor de um contador sequencial de ciclos e o valor do indicador de erro gerado pelos conversores.

O bloco de controle de escrita funciona da seguinte forma: se nenhum erro for detectado, aproximadamente a cada 30 segundos as informações de quantidade total de erros e tempo transcorrido de teste são enviadas; caso algum erro é detectado, o conteúdo do buffer é enviado. O envio de informações é feito através da interface serial RS-232/UART, disponível na biblioteca de componentes pré-construídos do PSoC 5 LP. A interface foi configurada para operar com taxa de 115,2 kbps, dados de 8 bits, 1 bit de parada e sem paridade e controle de fluxo. Um exemplo de parte do relatório criado é mostrado na Figura 47, onde as informações são lidas no *software* Hyperterminal da Microsoft. O código em linguagem C que faz essa implementação encontra-se disponível no apêndice A.

Figura 47 – Relatório criado com o registrado de estado.

```
*******State memory******
Total Faults: 0
************************
ADC_1_[01]: 1001001
ADC_1_[02]: 1001001
ADC_1_[03]: 1001001
ADC_1_[04]: 1001001
ADC_1_[05]: 1001001
ADC_2: 1001001
ADC_3: 1001010
SAR_ADC_voter: 1001001
Main_voter: 1001001
Cycle_counter: 231
************************
```

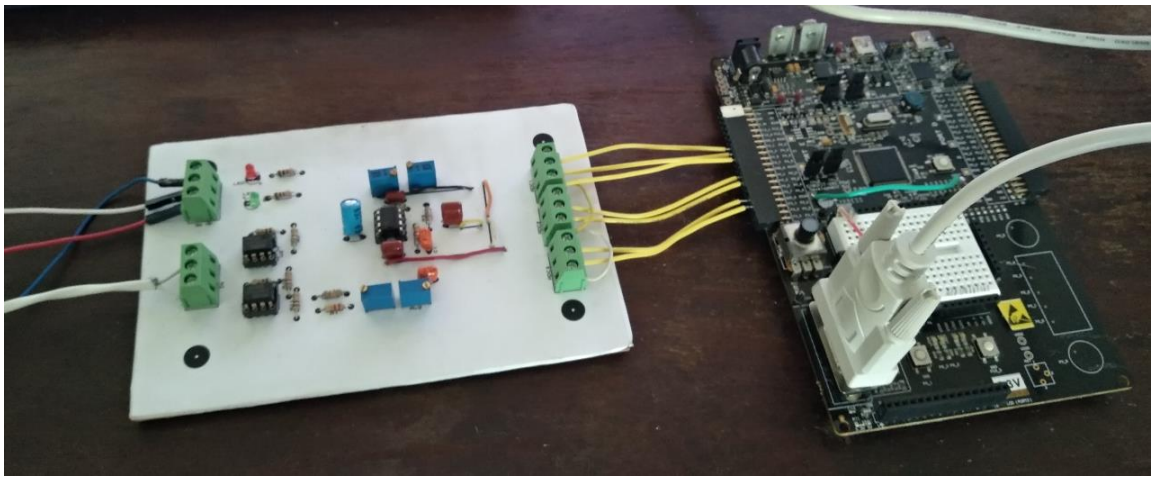
7 PROCEDIMENTO DE VALIDAÇÃO E TESTES

Concluídas as implementações do DUT, o setup de teste foi montado conforme descrito no início da seção 6. A placa de circuito impresso, com esquemático completo do projeto foi apresentada na seção 6.4, enquanto a diagramação do PSoC foi apresentada na Figura 39, e seu código C está disponível no apêndice A. O sinal analógico usado para estimular a entrada do SAD é de um termopar tipo K, especial para a finalidade de medições de EGT, onde seu encapsulamento pode suportar temperaturas superiores a 1000°C.

A temperatura do EGT foi controlada através da utilização de um forno industrial. O forno utilizado é do modelo *Universal Oven U*, da fabricante Memmerts, com capacidade de 110 litros, e tem por característica operar com temperaturas controladas de 30° a 300°C (Memmerts, 2018), que determinou a faixa da execução dos testes. O forno contém precisão de 0,5°C para temperaturas acima de 100°C, portanto o seu controlador foi considerado como parâmetro para validar o setup e as medições do SAD. A Figura 48 apresenta o setup utilizado para criar as condições de temperatura dos ensaios, enquanto a Figura 49 mostra a placa de circuito impressa confecionada, com todas as suas conexões a alimentação, ao PSoC e ao sensor, mostrando todas as condições onde os ensaios e testes foram executados.

Figura 48 – Forno com sensor de EGT posicionado dentro.

Figura 49 – Placa de circuito impresso com conexões ao sensor e ao PSoC.



A validação foi realizada através da inibição da injeção de falhas por software, para verificação da qualidade do sinal obtido, observados os seguintes critérios para considerar o setup como validado:

a) Os valores gerados pelos A/Ds ficam dentro das janelas de tolerâncias definidas nos votadores, e portanto o sistema não reporta erros;

b) O valor binário convertido do sistema de aquisição de dados, após votador, tem a mesma grandeza da temperatura ajustada do forno industrial;

A continuação da validação foi realizada agora habilitando a injeção de falhas por software, observados os seguintes critérios para considerar o setup como validado:

a) Os votadores detectam a ocorrência de erro e o conteúdo do buffer com os valores dos 100 ciclos de conversão mais recentes é reportado;

b) O valor binário convertido do sistema de aquisição de dados, após votador, tem a mesma grandeza da temperatura ajustada do forno industrial em casos onde ainda existem duas conversões válidas no SAD, e não existe leitura binária na saída do votador em evento dos três A/Ds terem sofrido mascaramento através da injeção de falhas;

Para a validação, os testes foram executados para seis temperaturas diferentes (50°C a 300°C com graduação de 50°C), durante um período de 30 minutos para cada temperatura selecionada, para garantir a funcionalidade do sistema em período de constante operação e condizentes com tempos reais de vôo de aeronaves militares. Os

primeiros dez minutos de cada teste foram descartados com o objetivo de deixar o sistema obter o equílibrio térmico e as medidas conterem valores válidos e confiáveis de medição.

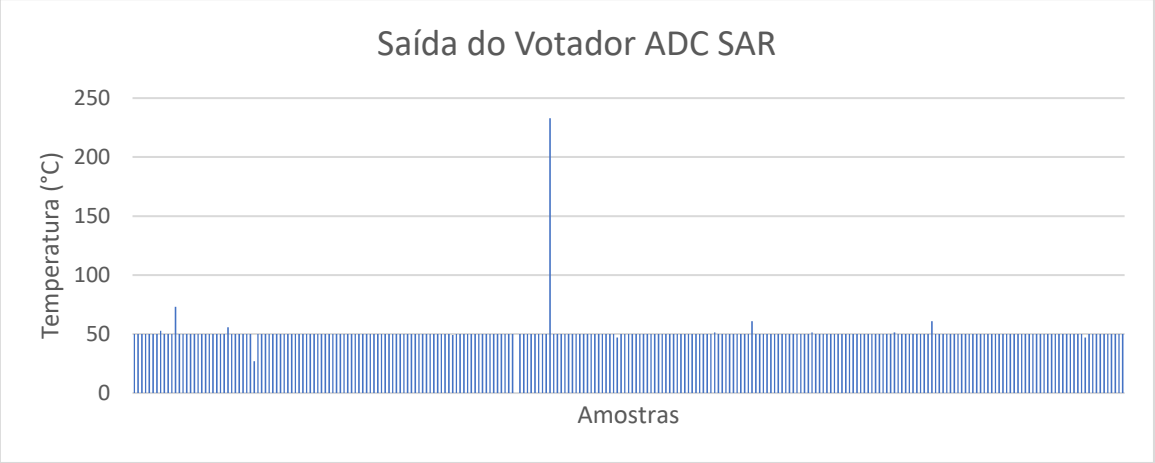
Além disto, foi definida uma tolerância de 8°C como aceitável para o votador, tendo como base 2% do valor de fundo de escala. Com base nisto, variações nos onze primeiros bits, caso ocorram simultaneamente em dois ou mais A/Ds, foram consideráveis aceitáveis por estarem provocando uma alteração máxima de 5°C na saída do sistema, que não afetaria a identificação da situação atual por parte do operador. A partir do décimo segundo bit em diante, o efeito da injeção de falhas é superior a 11,5°C, portanto houve a indicação de erros no sistema de aquisição de dados e votação, salvo em caso de falhas duplas ou múltiplas, que podem afetar a o votador principal e a falha ser reproduzida na saída por ter sido tolerada pelo sistema.

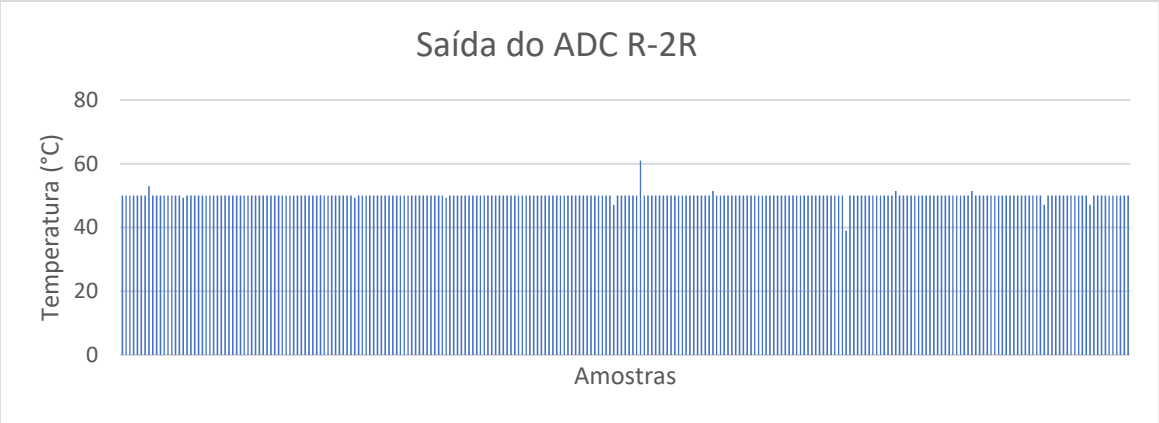
8 RESULTADOS E CONSIDERAÇÕES

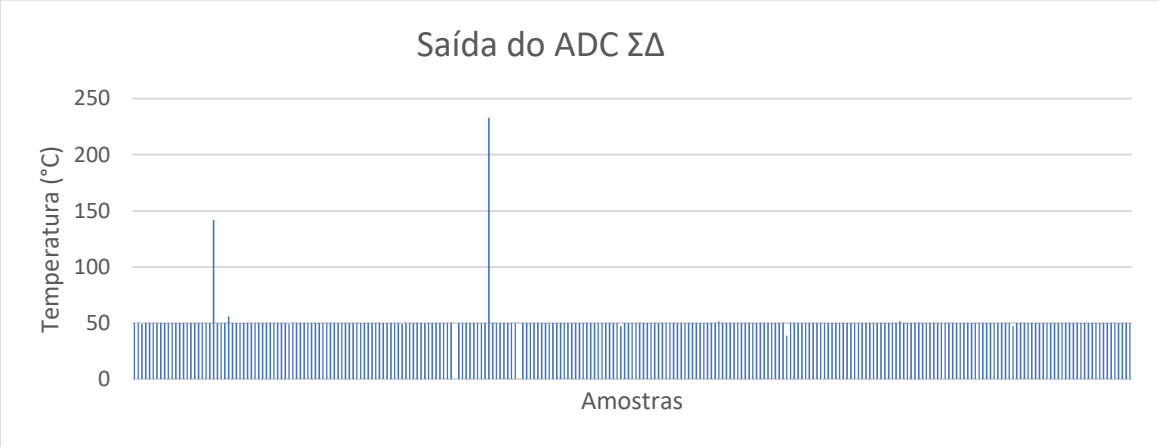
Os ensaios descritos nesta seção foram realizados por períodos de 10 minutos totais, para estabilização de temperatura, onde os últimos 3 minutos foram escolhidos para representar graficante o sistema e seu comportamento a emulação de falhas. Todas os gráficos aqui contidos representam 360 pontos de amostras agrupados, onde todos os valores lidos, sem apresentar alteração a cada 500mS, foram concatenados em uma ponto visual. Portanto, cada valor do eixo x contém pouco mais de 5000 medições que foram agrupadas em um único resultado para efeitos de visualização. O período de 500mS foi adotado por ele ter a mesma duração que uma falha permanece emulada, portanto não há alteração do valor lido do A/D neste intervalo de tempo e consequentemente o dado agrupado reflete todo o conjunto.

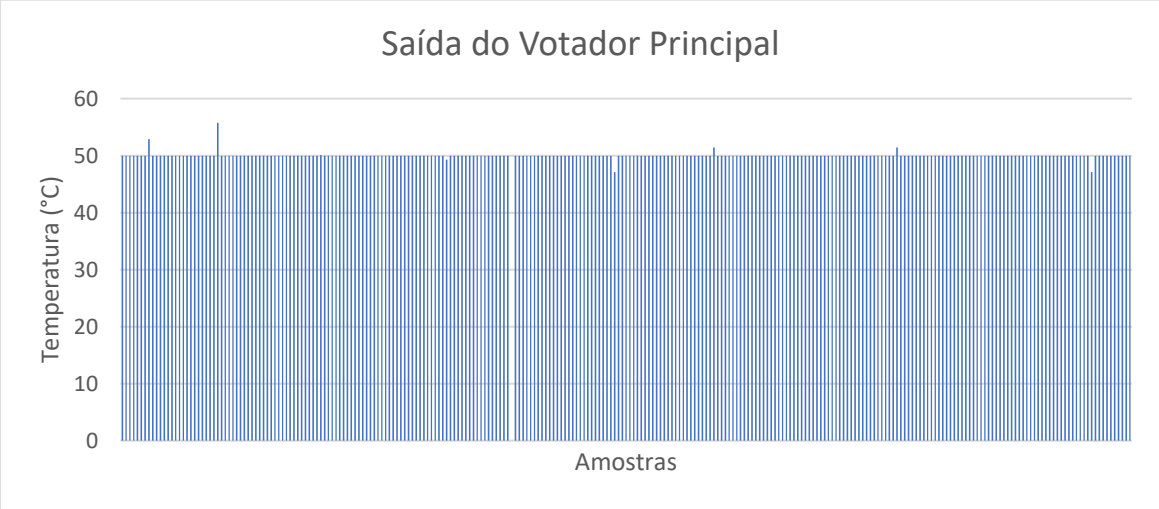
O primeiro ensaio foi realizado com uma temperatura ajustada em 50°C, por um período de aquisição de dados de 3 minutos. Cada A/D, de forma aleatória, sofreu em torno de 30 falhas não significativas, que afetaram apenas bits não sinigicativos na medição, e outras 30 falhas signicativas, totalizando 90 falhas significativas. Dentro destas 90 falhas, 8 delas foram duplas, enquanto que ocorreu apenas um evento onde os três A/Ds tiveram mascaramento aplicado ao seu sinal digital em bits de maior representatividade no valor de temperatura, que ocasionou em uma saída nula de temperatura. Na Figura 50 é possível observar que a saída do votador do A/D SAR, por ser a primeira utilizada na votação do votador principal, é a mais similar a saída, pois basta seu valor estar similar entre uma duas outras duas amostras para ser o prevalencente. Além disto, em casos de falha simples (singular), o SAD conseguiu tolerar esta falha e apresentou valores condizentes lidos através dos dois ADCs que continham valores válidos. Houve casos de falhas múltiplas (duplas e triplas), que levaram a votação principal a levar o resultado errôneo na saída, mas com frequência consideravelmente menor do que a quantidade de falhas injetadas ao longo do tempo.

Figura 50 – Saída de todos os pontos de medição do SAD.





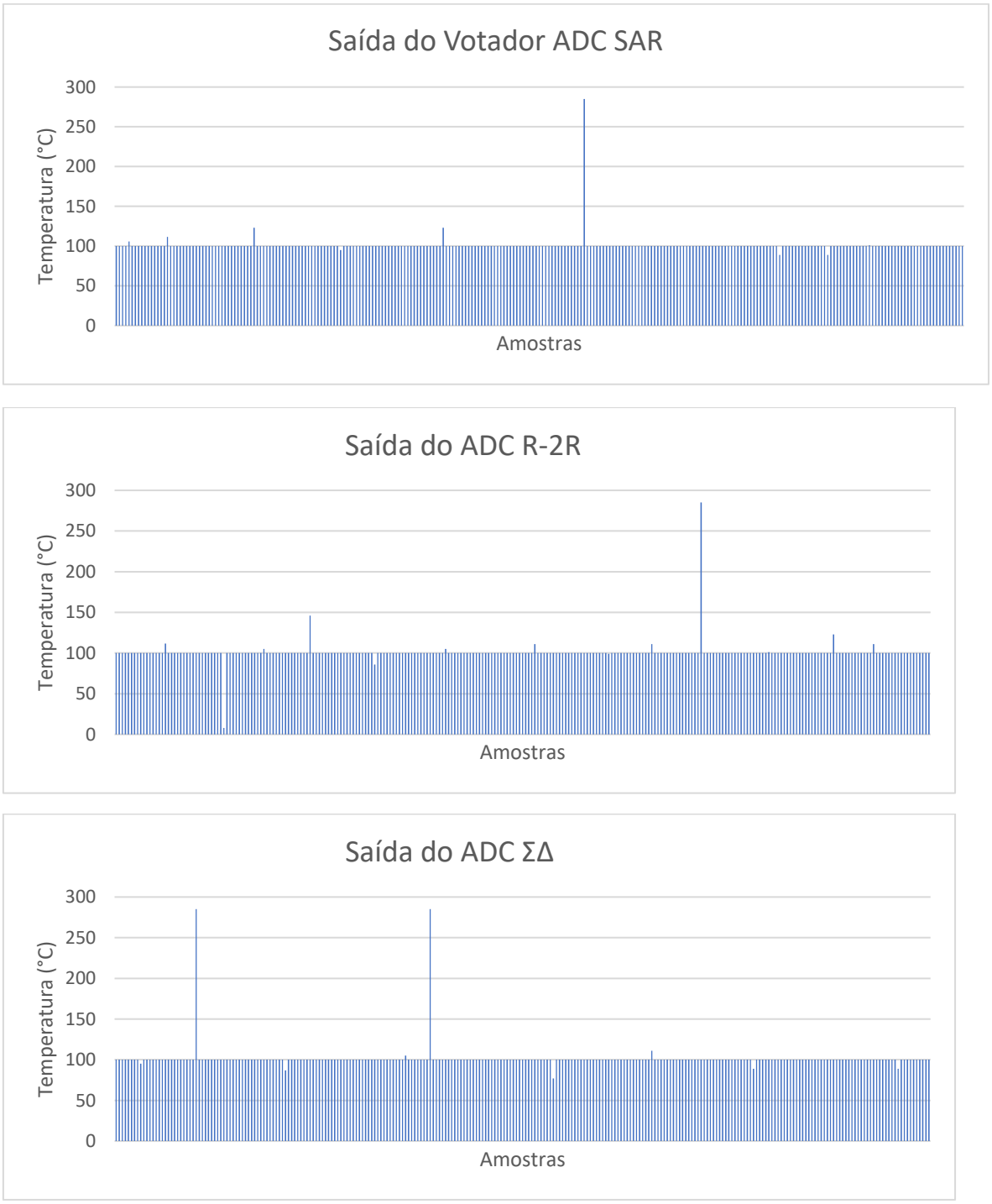


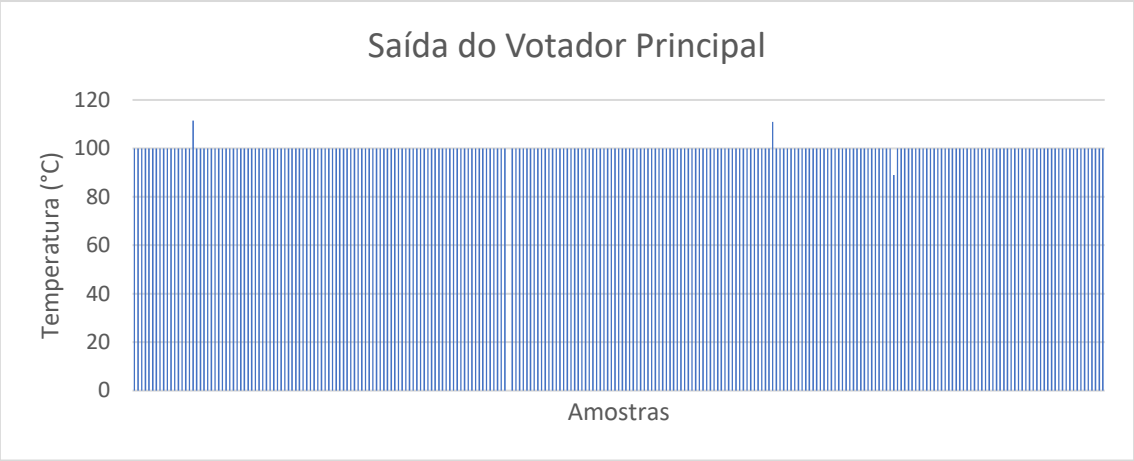


O segundo ensaio foi realizado com uma temperatura ajustada em 100°C, Na Figura 51 são possíveis visualizar todas as saídas do sistema, com o mesmo comportamento observado no ensaio anterior.Cada ADC, sofreu em torno de 32 mascaramentos de bit significativos aleatoriamente gerados pelo PSoC, totalizando 96 erros. Dentro destas 96 falhas, 9 delas foram duplas, enquanto que ocorreu apenas um evento onde os três ADCs sofreram mascaramento de bit significativo. As falhas duplas causaram algumas divergências no valor da saída, com dois pontos onde a temperatura

medida foi de 108°C, e também houve uma saída nula em função da ocorrência de emulação tripla.

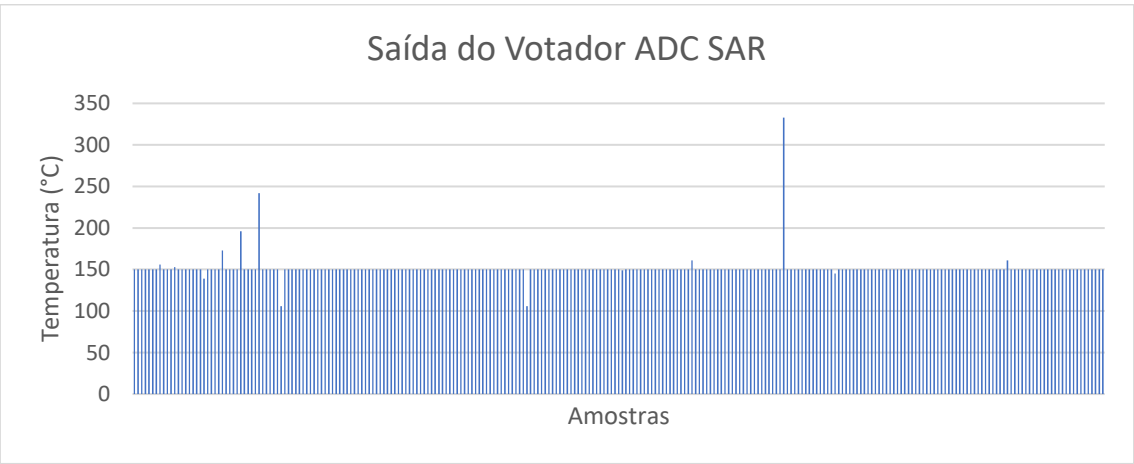
Figura 51 – Saída do SAD com temperatura de medição a 100°C.

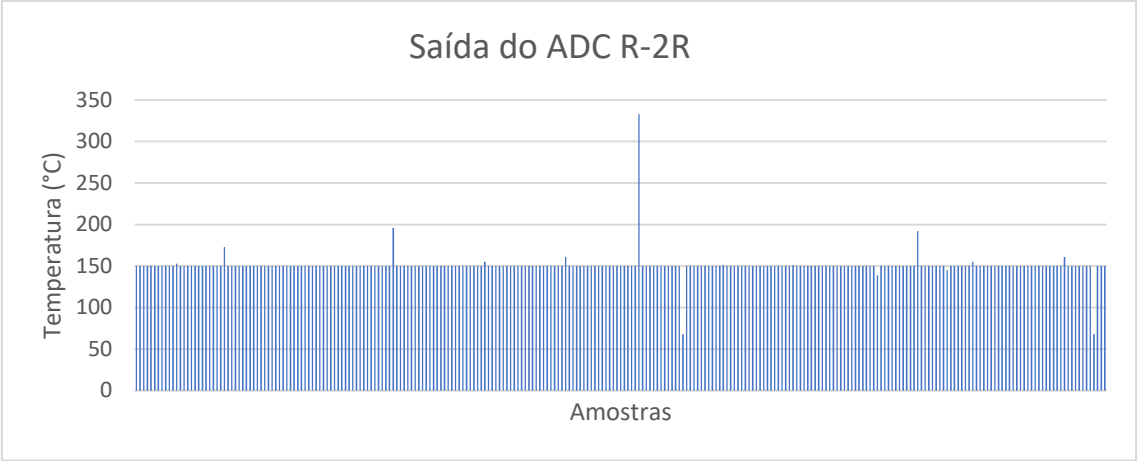


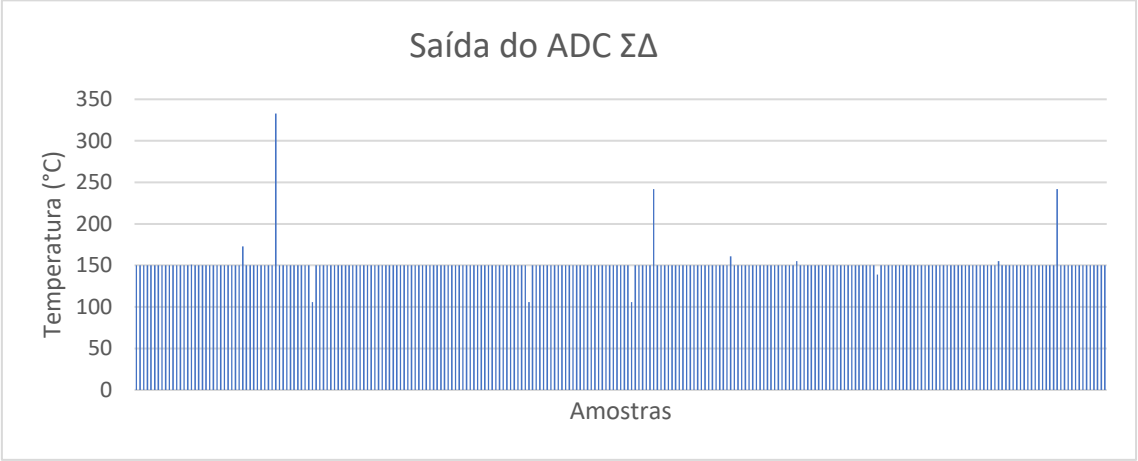


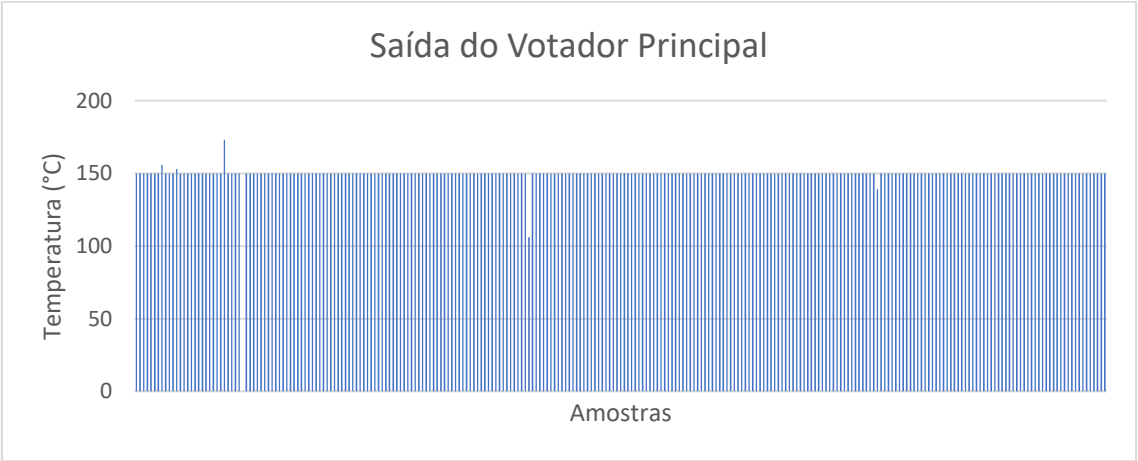
Mais um ensaio foi realizado com uma temperatura ajustada em 150°C, Na Figura 52 são possíveis visualizar todas as saídas do sistema, com o mesmo comportamento observado no ensaio anterior. Cada ADC, sofreu em torno de 35 falhas aleatoriamente geradas pelo PSoC, totalizando 105 falhas significativas. Dentro destas 105 falhas, 8 delas foram duplas, enquanto que em 2 ocasiões tivemos múltiplos mascaramento de bit significativos nos A/Ds. Neste caso, também houve um evento de saída nula e um evento onde a temperatura medida foi de 107°C, abaixo do esperado.

Figura 52 – Saída do SAD com temperatura de medição a 150°C.





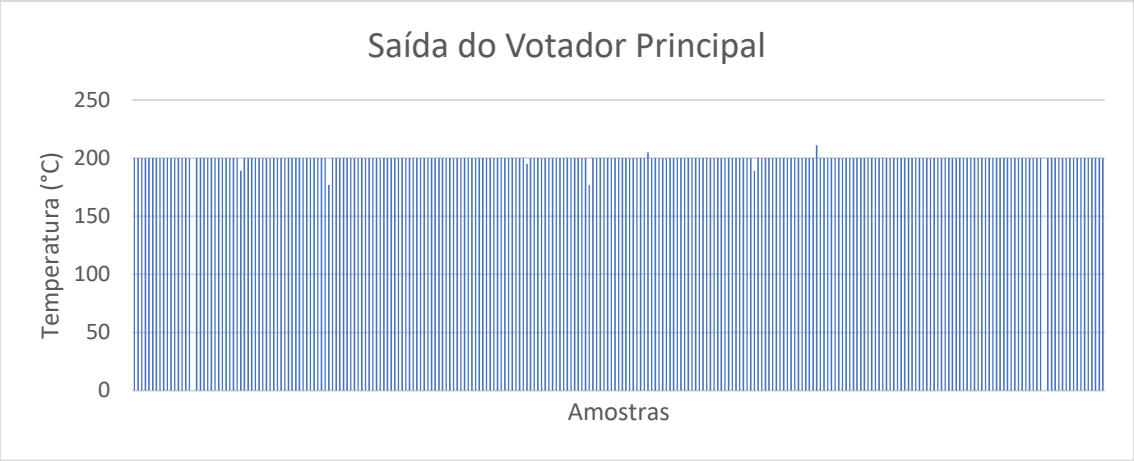




O quarto ensaio foi realizado com uma temperatura ajustada em 200°C, Na Figura 53 são possíveis visualizar todas as saídas do sistema. Cada A/D, aleatóriamente, sofreu em torno de 34 falhas significativas geradas pelo PSoC, totalizando 102 falhas. Dentro destas 102 falhas, 12 delas foram duplas, enquanto que em 3 ocasiões tivemos múltiplos mascaramento de bit signicativos nos A/Ds, causando erros de leitura consideráveis na saída do sistema mesma após as votações. É interessante notar também que o acréscimo de temperatura faz com que agora erros de menores

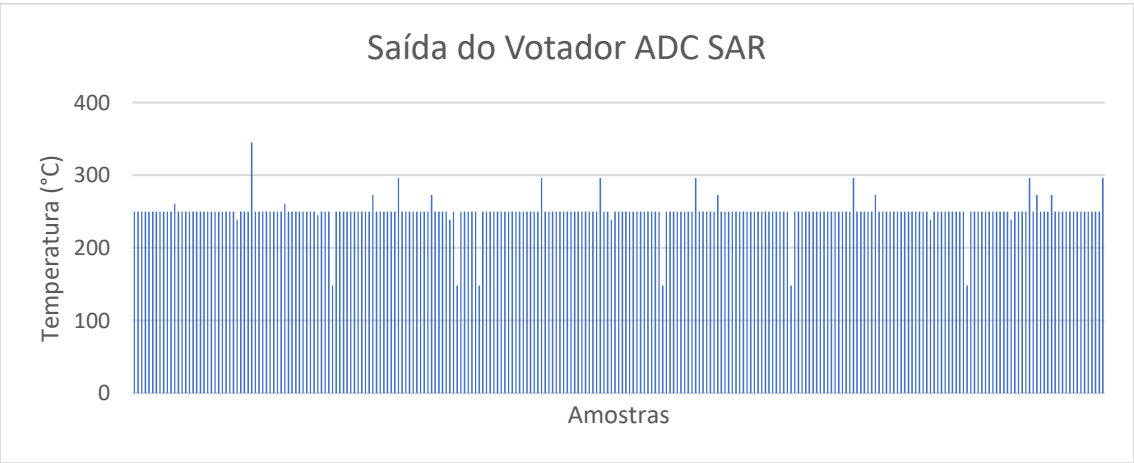
proporções se tornem ainda menos significativos no sistema, mesmo em casos onde exista uma emulação de falha dupla ou tripla.

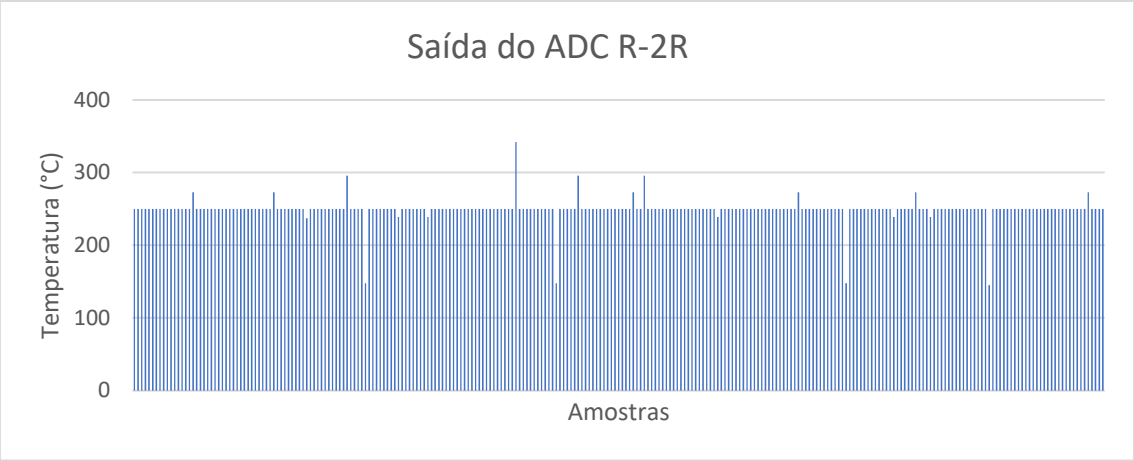
Figura 53 – Saída do SAD com temperatura de medição a 150°C.

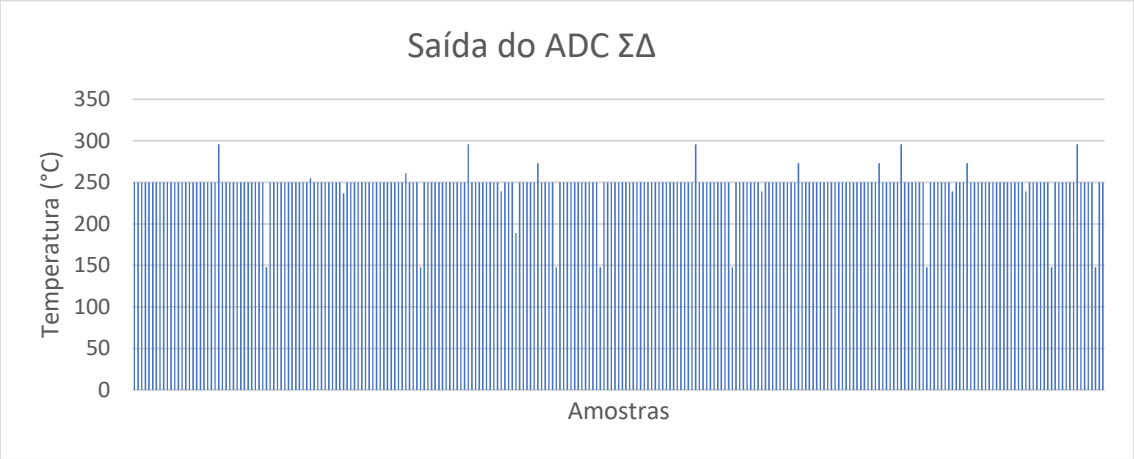


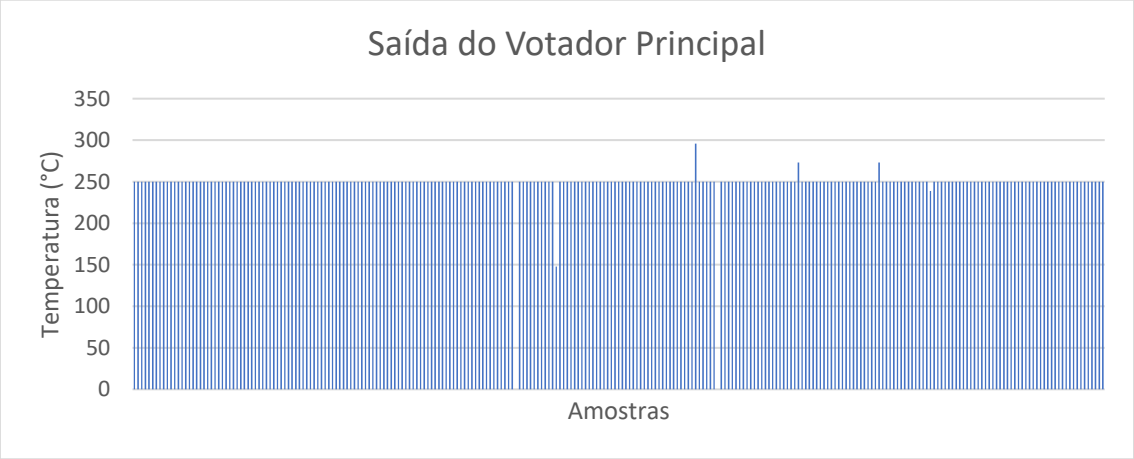
O quinto ensaio foi realizado com uma temperatura ajustada em 250°C, Na Figura 54 são possíveis visualizar todas as saídas do sistema, com o mesmo comportamento observado no ensaio anterior.Cada ADC, aleatóriamente, sofreu em torno de 36 falhas significativas geradas pelo PSoC, totalizando 118 falhas. Dentro destas 118 falhas, 19 delas foram duplas, enquanto que em 2 ocasiões tivemos múltiplos mascaramento de bit nos ADCs. Neste ensaio, não houve nenhuma divergência grande entre os três A/Ds que causasse o votador principal a indicar uma saída nula de temperatura.

Figura 54 – Saída do SAD com temperatura de medição a 250°C.



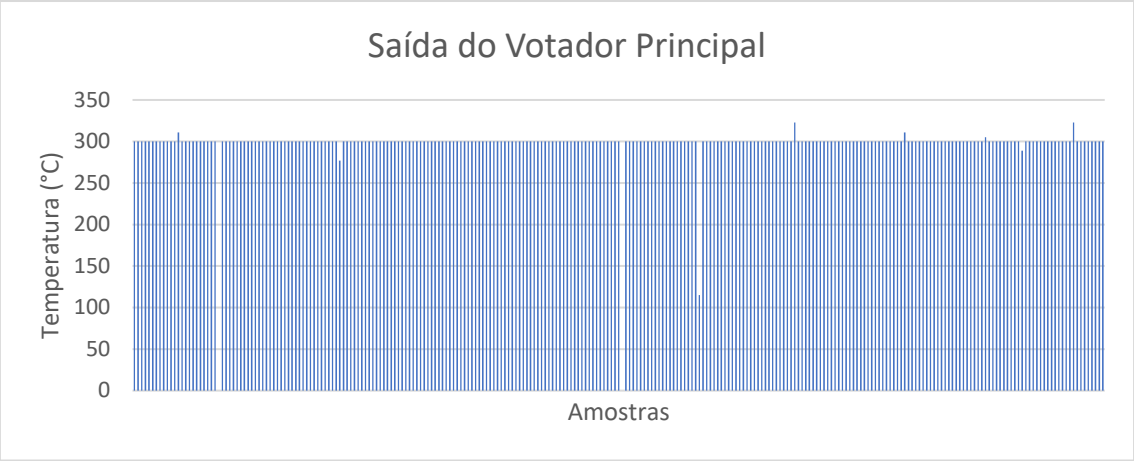






O último ensaio foi realizado com uma temperatura ajustada em 300°C, Na Figura 55 são possíveis visualizar todas as saídas do sistema, onde é possível observar que cada vez menos alterações de baixo valor tem impacto no sistema final. Cada A/D sofreu em torno de 34 falhas significativas aleatoriamente geradas pelo PSoC, totalizando 102 falhas. Dentro destas 102 falhas, 12 delas foram duplas, enquanto que em 2 ocasiões tivemos múltiplos mascaramento de bit significativos nos A/Ds.

Figura 55 – Saída do SAD com temperatura de medição a 300°C.



Com os seis ensaios concluídos, foi feito um resumo da quantidade de emulação de erros e a resiliência apresentada pelo sistema, que é apresentado na Figura 56. Foram emuladas ao todo 1111 falhas, onde 689 delas causaram algum tipo de transtorno ao sistema: 613 delas foram do tipo simples, 68 duplas e 8 triplas. Foi possível observar que, apesar da quantidade de emulação de falhas, todos os mascaramentos simples que houveram foram mitigados em função do votador principal do sistema. Este tipo de ocorrência representou 55% dos erros significativos emulados. Portanto, menos de 7% dos erros emulados não foram tolerados, onde causaram dois estados diferentes ao sistema: uma indicação de temperatura incorreta, em função do mascaramento ter ocorrido de forma similar em dois A/Ds; uma indicação nula de temperatura no sistema, em função de as três medidas estarem incoerentes entre si.

Figura 56 – Quadro geral da emulação de falhas e resiliência do sistema.

		Significativos		
	Erros	Simples	Duplos	Triplos
50°C	179	90	8	1
100°C	185	96	9	1
150°C	191	105	8	2
200°C	183	102	12	0
250°C	195	118	19	2
300°C	178	102	12	2
		55%	**6%**	**1%**

A quantidade de erros duplos ou triplos de grande escala ficou abaixo do projetado estatisticamente no *software*, muito em função da quantidade de bits do sistema, que dificultou a probabilidade de ocorrências significativas simultâneas e acabou por tolerar mascaramentos que afetaram os bits de menor impacto na medição de temperatura.

No caso de falhas simples, o SAD é capaz de tolerar 100% das falhas, como já é de conhecimento comum para sistemas baseados em TMR. No caso de falhas duplas, o SAD em teoria é capaz de tolerar aproximadamente 60% das falhas, o que configura um ganho de confiabilidade em relação ao TMR clássico e, como visto no resultado final, houve uma parcela considerável das 68 ocorrências que não causaram impactos no sistema. No caso de falhas múltiplas, o SAD é capaz de tolerar aproximadamente 15% das falhas. Considerando a baixa probabilidade de ocorrerem múltiplas falhas em distintos domínios espacial e temporal, a tolerância do SAD no caso de falhas múltiplas pode não ser considerada crítica. Portanto, este estudo demostra, que em comparação com circuitos simples a técnica TMR clássica, com a adição de diversidade temporal, gera um ganho significativo na tolerância de falhas independente de sua ordem.

9 CONCLUSÕES

O estudo dos efeitos da radiação espacial nos circuitos eletrônicos teve início na década de 1960 e desde então estes efeitos vêm apresentando comportamentos diferentes à medida que a eletrônica evolui. Atualmente os efeitos singulares, em especial os soft errors, são a maior preocupação para a garantia da confiabilidade e disponibilidade das tecnologias CMOS, dado a redução das dimensões dos transistores, o aumento da velocidade de operação dos sistemas e a redução da tensão de alimentação dos circuitos integrados. TMR e diversidade são técnicas em nível de sistema de proteção contra os soft errors, usadas geralmente em aplicações críticas. Nesse contexto, esse trabalho adotou um esquema baseado nessas duas técnicas para a implementação de um sistema de aquisição de dados (SAD) analógico-digital, com o objetivo de avaliar o comportamento dos conversores de dados frente a emulação de SEEs, e avaliar a eficácia de um sistema baseado em TMR e diversidade espacial-temporal contra esses efeitos da radiação. A aplicação escolhida foi a de medição de EGTs, por ser uma aplicação crítica e também que, dado que foi abordado na seção 4 que em maiores altitudes os campos elétricos e magnéticos são maiores, e que portanto os equipamentos eletrônicos estão mais suscetível a efeitos indesejáveis. A implementação foi feita através do desenvolvimento de uma placa de circuito impresso contendo o amplificador de instrumentação, filtro ativo de quarta ordem para eliminação de ruídos, três conversores A/Ds e saídas de conexão para o SoC PSoC 5LP, da Cypress Semiconductor, que fez o monitoramento dos dados e emulação de falhas.

A utilização de TMR e diversidade ao longo desse trabalho propiciou esclarecer alguns aspectos importantes dessas técnicas, os quais não são inéditos, mas que são ressaltados com pouca frequência na literatura do assunto, especialmente focado em algum tipo de aplicação industrial. São eles:

a) a técnica TMR clássica é eficaz para tolerar falhas aleatórias que presumidamente ocorrem de maneira independente nas cópias redundantes;

b) a técnica de diversidade é especialmente útil para lidar com falhas de modo comum, que é quando múltiplas cópias de um sistema redundante sofrem falhas simultaneamente, geralmente devido a uma única causa. Essa característica da diversidade é devido aos diferentes níveis de resiliência de cada *hardware*, *software* e tempo, que fazem com que erros devido a pontos em comum sejam sistematicamente evitados;

c) a proteção com TMR e diversidade implica em um aumento da complexidade de projeto do sistema. Exemplos de fatores que contribuem para isso são: a necessidade de um sincronizador para controlar as cópias e os votadores, e a necessidade de atenção com os parâmetros analógicos quando se replica e conecta circuitos analógicos. O sincronizador e os votadores são elementos-chave em TMR e diversidade;

d) devido ao seu custo elevado, TMR e diversidade são geralmente mais apropriadas para aplicações críticas, que requerem elevada confiabilidade e

disponibilidade. Como por exemplo em naves espaciais, aviões comerciais e instalações nucleares, e por consequência este trabalho de pesquisa utilizou uma aplicação comum em naves especiais para também enaltecer este viés e estar condizente a realidade.

No presente trabalho, a implementação do sistema baseado em TMR e diversidade foi desenvolvida em uma placa dedicada, apresentada na seção 6. Ela contém o amplificador de instrumentação, filtro ativo e três A/Ds distintos: SAR, R-2R e Delta-Sigma. Além disto, foi utilizado um SoC programável para realizar o sincronismo, emulação de falhas e votação dos sinais adquiridos. Esta divisão foi baseada no conceito de SRUs em sistemas aviônicos, onde a placa dedicada pode ser interpretada com o módulo EICAS, enquanto o SoC programável como uma SRU de processamento. Aqui, podemos observar que a confiabilidade do sistema foi incrementada às custas do acréscismo de complexidade do sistema, onde foi necessário desenvolver diversos blocos de processamento e controle no SoC.

A emulação de falhas foi abordada na seção 8. Foram emuladas separadamente falhas nos registradores de memória que continham a última leitura válida de cada A/D. Os resultados mostraram que apenas um baixo percentual das falhas injetadas é detectado na forma de erros, devido a resiliência do sistema e sua capacidade em tolerar falhas, especialmente nos casos de mascaramento simples. Esses resultados confirmam o embasamento teórico apontado na seção 5, onde foram apresentadas técnicas de mitigação de falhas a nível de tecnologia, dispositivo, e sistema, e que basearam o projeto na decisão da utilização de uma TMR com diversidade e redundância temporal na amostragem do A/D SAR.

Como sugestões para trabalhos futuros são mencionados:

a) testar o SAD implementado nesse trabalho em instalações de teste para radiação de fluxo de partículas;

b) testar o SAD implementado novamente com a injeção de falhas por software nos registradores de controle dos periféricos do SoC. Desta forma, existiria uma emulação do comportamento da falha nos dois módulos: placa confeccionada e no SoC programável;

c) adicionar outras fontes de erro no sistema, associadas com a emulação ou com a radiação de partículas. Ruídos, interferências eletro-magnéticas de alta frequência e outros sinais, que estão presentes no ambiente de uma aeronave, e podem causar outros tipos de comportamento no circuito, como por exemplo afetar a tensão do termopar lida no SAD.

REFERÊNCIAS

AGUIAR, Y.; ZIMPECK, A.; MEINHARDT, C.; **NFAS-tool: Avaliação da confiabilidade de células combinacionais sob falhas de radiação do tipo SET**. IBERCHIP Workshop, Florianópolis, 2016.

ARRUDA, T. M. **A Influência da radiação em circuitos eletrônicos**. 212f. 2006. Dissertação (Mestrado em Engenharia Aeronáutica e Mecânica e Área de Sistemas Aeroespaciais e Mecatrônica) – Instituto Tecnológico da Aeronáutica, São José dos Campos – SP, 2006.

BALEN, T. R. **Efeitos da radiação em dispositivos analógicos programáveis (FPAAs) e técnicas de proteção**. 2010. 205 f. Tese (Doutorado em Engenharia Elétrica) – Programa de Pós-Graduação em Engenharia Elétrica, Universidade Federal do Rio Grande do Sul, Porto Alegre, 2010.

BALBINOT, A.; BRUSAMARELLO, V. J. **Instrumentação e Fundamentos de Medidas**. v. 1, e. 2. Rio de Janeiro: LTC, 2011.

BARTH, J. Modeling space radiation environments. **UEEE NSREC Short Course Notes**, Snowmass Village, p. I 1-83, July 1997.

BAUMANN, R. C. Soft errors in advanced semiconductor devices-part I: the three radiation sources. **IEEE Transactions on Device and Materials Reliability**, [S. l.], v. 1, n. 1, p. 17-22, Mar. 2001.

BECKER, T. E.; LANOT, A. J. C.; CARDOSO, G. S.; BALEN, T. R; Single Event Transient Effects on Charge Redistribution SAR ADCs. **Microelectronics Reliability**, v. 73, p. 22-35, June 2017.

BOUDENOT, J. C., Radiation Space Environment. In: Velazco, Raoul; Fouillat, Pascal; Reis, Ricardo. (Org). **Radiation Effects on Embedded Systems**, Springer, v. 1, p. 1-9, 2007.

CARDOSO, G. S.; Performance and simulation accuracy evaluation of analog circuits with enclosed layout transistors. **Analog Integr Circ Sig Process**, 93:455-466, 2017.

CAST SAFETY ORG, **Engine Fundamentals**. Disponível em: <http://www.cast-safety.org/pdf/3_engine_fundamentals.pdf>. Acesso em 28 mai. 2018.

CHENET, C. P.; TAMBARA, L. A.; BORGES, G. M.; KASTENSMIDT, F.; LUBASZEWSKI, M. S.; BALEN, T. R. Exploring design diversity redundancy to improve resilience in mixed-signal systems. **Microelectronics Reliability**, v. 55, i. 12, p. 2833-2844, Dezembro 2015.

DODD, P. E., Device Simulation of Charge Collection and Single-Event Upset. **IEE Transactions on Nuclear Science**. [S. 1.], v. 43, n. 2. p. 561-575. Apr, 1996.

DODD, P. E., Current and Future Challenges in Radiation Effects on CMOS Electronics. **IEE Transactions on Nuclear Science**. [S. 1.], v. 57, n. 4. p. 1741-1763. Aug, 2010.

DUFOUR, C. et al. Heavy ion induced single hard errors on submicronic memories [for space application]. **IEEE Transactions on Nuclear Science**, [S. l.], v. 39, n. 6, p. 1693-1697, Dec. 1992.

DYER, C. **Radiation effects on spacecraft & aircraft**. SOLAR CYCLE AND SPACE WEATHER EUROCONFERENCE, 2., 2001, 24 – 29, Vico Equense, Italy. Vico Equence, 2001.

ESTERLINE, **Cockpit 400 NexGen**. Disponível em: <https://www.esterline.com/Portals/17/Documents/en-us/CK-Cockpit-4000-NG-4pager_18-002.pdf>. Acesso em: 06 Mai. 2018.

EUROPEAN COOPERATION FOR SPACE STANDARDIZATION (ECSS). **ECSS-E-HB-10-12A space engineering - calculation of radiation and its effects and margin policy handbook**. Noordwijk, The Netherlands, 2010.

FEDERAL AVIATION ADMINISTRATION, **Aircraft Instrument Systems**. Disponível em:<https://www.faa.gov/regulations_policies/handbooks_manuals/aircraft/amt_airframe_handbook/media/ama_Ch10.pdf>. Acesso em 05 Jun. 2018.

FERLET-CAVROIS, V.; MASSENGILL, L. W.; GOUKER, P., Single event transients in digital CMOS - a review, **IEEE Trans. Nucl. Sci. 60 (3**), p. 1767–1790, June 2013.

FERLINI, F.; **PLAESER – Plataforma de Emulação de Soft Erros Visando a Análise Experimental de Técnicas de Tolerância a Falhas: Uma Prototipação Raída Utilizando FPGAs**. 2012. 159 f. Tese (Mestre em Engenharia Elétrica) – Programa de Pós-Graduação em Engenharia Elétrica, Universidade Federal de Santa Catarina, Florianópolis, 2012.

GALLOWAY, K. F.; JOHNSON, G. H. Catastrophic single-event effects in the natural space environment. **IEEE NSREC Short Course Notes**, Monterey, p. IV 1-72, July 2003.

GLENN RESEARCH CENTER – NASA. **Turbofan Engine**. Disponível em: <https://www.grc.nasa.gov/www/K-12/airplane/aturbf.html>. Acesso em 24 mai. 2018.

HIGUERETA, S.; HUSSON, D.; TROCMÉ, M.; NOURREDINNE, A.; LÊ, T. D.; MICHIELSEN, N., **Measurement of 222Rn at the Bq m−3 level with the AlphaRad chip**, Science Direct, Radiation Measurements, v. 43, i. 2-6, p. 1059-1062, Feb-June 2008.

JOINT ELECTRON DEVICE ENGINEERING COUNCIL. **JESD57**: test procedures for the measurement of single-event effects in semiconductor devices from heavy ion irradiation. Arlington, 1996.

JOINT ELECTRON DEVICE ENGINEERING COUNCIL. **JESD89A**: measurement and reporting of alpha particle and terrestrial cosmic ray-induced soft errors in semiconductor devices. Arlington, 2006.

KEE, R. J.; O'REILLY, P.; FLECK, R.; McENTEE, P. Measurement of Exhaust Gas Temperatures in a High Performance Two-Stroke Engine. **SAE Transactions - Journal of Engines**. 107-3. 2413 –2423. 10.4271/983072, January 1999.

KLUDT, J.; WEIDE-ZAAGE, K.; ACKERMANN, M.; KOVACS, C.; HEIN, V. Reliability Performance of Different Layouts of Wide Metal Tracks, **Reliability Physics Symposium, 2014 IEEE International**, pp. IT.4.1–IT.4.4, June 2014.

KOLASINSKI, W. A., **Effects of Space Radiation on Electronics Microcircuits**. Effects of Space Radiation on Electronic Microcircuits, NASA/SDIO Space Environmental Effects on Materials Workshop, NASA CP 3035, Part 2, p. 383-389. 1989.

LIOU, J. J.; JIANG, C.; CHIA, F. Electrostatic discharge (ESD) protection of RF integrated circuits, Circuits and Systems (APCCAS), **2012 IEEE Asia Pacific Conference**, p. 460–462, December 2012.

MACHADO, S. R. F.; **Estudo de um Processo de Garantia de Confiabilidade de Sistemas Eletrônicos Embarcados a Single Event Upsets Causado Por Partículas Ionizantes**. 2014. 192 f. Tese (Mestrado em Engenharia e Tecnologias Espaciais) – Instituto Nacional de Pesquisas Espaciais. São José Dos Campos, 2014.

MEMMERTS; **Heating and dryings ovens.** Disponível em: <https://www.memmert.com/fileadmin/products/documents/categories/BR-Heating-Ovens-english-D13646.pdf>. Acesso em 20 jul. 2018.

MUNTEANU, D.; AUTRAN, J. L. Modeling and simulation of single-event effects in digital devices and ICs. **IEEE Transactions on Nuclear Science**, [S. l.], v. 55, n. 4, p. 1854-1878, Aug. 2008.

UNIVERSITÉ DE LYON. **Accélération des particules: le rôle de la magnétosphère terrestre**. Disponível em: <http://planet-terre.ens-lyon.fr/article/aurores-polaires-Bernard.xml>. Acesso em: 08 jul. 2018.

NATIONAL AERONAUTICS AND SPACE ADMINISTRATION. **No Safe Place**. Disponível em: < https://science.nasa.gov/science-news/science-at-nasa/2007/22feb_nosafeplace>. Acesso em 20 Jun. 2018.

NICOLAIDIS, M. **Soft errors in modern electronic systems**. New York: Springer Science and Business Media, LLC 2011. ISBN 978-1-4419-6992-7.

ORTIZ, E.; BABBAR, A.; SYRMOS, V.; CLARK, G.; VIAN, J.; ARITA, M. Advanced diagnostics and prognostics for engine health monitoring, **Proc. IEEE Aerospace conference,** 2009. p. 6-10.

PETERSEN, E. **Single Event Effects in Aerospace**. Hoboken: John Wiley & Sons, p. 520, 2011.

TAKEDA, E.; NAKAGOME, Y.; HUME, H.; ASAI, S. New hot-carrier injection and device degradation in submicron MOSFETs, **Solid-State Electron Devices IEE Proc. I 130,** p. 144–150, June 1983.

RAO, V. M; PATEL, V. V.; PRASAD, N**. Advanced Distributed Flight Control System Architecture with Prognosis for High Performance Fighter Aircraft**. International Journal of Engineering Trends and Technology, v. 42, December 2016.

SALIVA, M.; CACHO, F.; HUARD, V.; ANGOT, D.; FEDERSPIEL, X.; DURAND, M.; PARRA, M.; BRAVAIX, A.; ANGHEL, L. New Insights about Oxide Breakdown Occurrence at Circuit Level, **Reliability Physics Symposium, 2014 IEEE International**, pp. 2D.5.1,2D.5.6, June 2014.

SZURMAN, K; State Synchronization of Faulty Soft Core Processors in Reconfigurable TMR Architecture. **Počítačové architektúry & diagnostika PAD**, 2017.

SEXTON, F. W. Destructive single-event effects in semiconductor devices and ICs. **IEEE Transactions on Nuclear Science**, [S. l.], v. 50, n. 3, p. 603-621, June 2003.

STASSNIPOULOS, E.; RAYMOND, J. The Space Radiation Environment for Electronics. **Proceedings of the IEE**, [S. 1.], v. 76, p. 1423-1422, Nov. 1988.

SPENVIS**. European Space Agency: Space environment information system**. Disponível em: <https://www.spenvis.oma.be/help/background/traprad/traprad.html>. Acesso em: 08 Jul. 2018.

TORRES, F. E. **Desenvolvimento de um Sistema de Emulação de Single Event Upsets em Dispositivos COTS Baseado na Metodologia Code Emulating Upsets**. 2013. 12 f. Tese (Mestrado em Engenharia Elétrica) – Programa de Pós-Graduação em Engenharia Elétrica, Universidade Federal de Minas Gerais, 2013.

TURFLINGER, T. L. Single-event effects in analog and mixed-signal integrated circuits. **IEEE Transactions on Nuclear Science**, [S. l.], v. 43, n. 2, p. 594-602, Apr. 1996.

VELAZCO, R.; FOUILLAT, P. e REIS, R. **Radiation Effects on Embedded Systems**. Dordrecht: Springer Netherlands, 2007.

WAN, F.; DUVAL, F.; SAVATIER, X.; LOUIS, A.; MAZARI, B. Effects of conducted electromagnetic interference on analogue-to-digital converter, **Electron. Lett. 47 (1)** p. 23–25, January 2011.

WATSON, J. T.; BURCHAM JR., F. W. **Exhaust-Gas Pressure and Temperature Survey of F404-GE-400 Turbofan Engine**. NASA Technical Memorandum 88273, December 1986.

WEBER, T.S; **Tolerância a Falhas: Conceitos e Exemplos**. Disponível em: <http://www.inf.ufrgs.br/~taisy/disciplinas/textos/ConceitosDependabilidade.pdf.> Acesso em Outubro de 2018.

WELLS, A. T.; RODRIGUES, C. C., **Commercial Aviation Safety**. McGraw-Hill Professional. p. 245, December 2003.

APÊNDICE A – Rotina de Software Desenvolvida no PSoC

```
#include <project.h>
#include <stdlib.h>
#include <stdio.h>

/* main function's variables*/
uint16 DMA_module1Data[5];
uint16 DMA_module2Data[2];
uint16 DMA_module3Data[2];
uint16 module1Data[5] = {0,0,0,0,0};
uint16 module2Data[2] = {0,0};
uint16 module3Data[2] = {0,0};
uint16 DMA_1_Chan;
uint16 DMA_1_TD[1];
uint16 DMA_2_Chan;
uint16 DMA_2_TD[1];
uint16 DMA_3_Chan;
uint16 DMA_3_TD[1];
uint16 cycle_counter = 0;
uint8 tmpStat;
uint8 running_bit = 0;

/* SAR_ADC_voter function's variables*/
uint8 i, j;
uint8 bits;
uint8 mask[8] = {128,64,32,16,8,4,2,1};
uint8 bit_counter[8] = {0,0,0,0,0,0,0,0};
uint8 SAR_ADC_voter_data = 0;
uint8 SAR_ADC_voter_error_det = 0;
uint8 bigger;
uint8 smaller;
uint8 pointer;
uint8 error;

/* main_voter function's variables */
uint8 error1, error2, error3;
uint8 system_output = 0;
uint8 data_invalid_flag = 0;
uint8 main_voter_error_det = 0;

/* buffer's variables */
uint16 buf_1[500];
uint16 buf_2[100];
uint16 buf_3[100];
uint16 buf_4[100];
uint16 buf_5[100];
uint16 buf_6[100];
uint16 buf_7[100];
uint16 buf_pointer_1 = 0;
uint8 buf_pointer_2 = 0;

/* uart_print_values function's variables */
char16 *module1Data_1[3];
char16 *module1Data_2[3];
char16 *module1Data_3[3];
char16 *module1Data_4[3];
char16 *module1Data_5[3];
char16 *module2Data_1[3];
char16 *module3Data_1[3];
char16 *SAR_ADC_voter_data_uart[3];
char16 *system_output_uart[3];
char8 *cycle_counter_uart[5];
char8 *cycle_error_uart[1];
uint16 uart_pointer_1 = 0;
uint8 uart_pointer_2 = 0;

/* uart_control_write function's variables */
uint32 count_alive = 0;
uint8 error_capture = 0;
uint8 buffer_counter = 0;
uint32 fault_counter = 0;
char8 *fault_counter_uart[10];
```

```
/* real time clock */
uint8 second = 0u;
uint8 minute = 0u;
uint8 hour = 0u;
uint8 day_month = 0u;
uint8 month = 0u;
char8 *second_uart[2];
char8 *minute_uart[2];
char8 *hour_uart[2];
char8 *day_month_uart[2];
char8 *month_uart[2];

void DMA_1_Config(void)
{
	#define DMA_1_BYTES_PER_BURST 1
	#define DMA_1_REQUEST_PER_BURST 1
	#define DMA_1_SRC_BASE (CYDEV_PERIPH_BASE)
	#define DMA_1_DST_BASE (CYDEV_SRAM_BASE)

	DMA_1_Chan = DMA_1_DmaInitialize(DMA_1_BYTES_PER_BURST, DMA_1_REQUEST_PER_BURST,
	HI16(DMA_1_SRC_BASE),HI16(DMA_1_DST_BASE));
	DMA_1_TD[0] = CyDmaTdAllocate();
	CyDmaTdSetConfiguration(DMA_1_TD[0], 10, DMA_1_TD[0], DMA_1__TD_TERMOUT_EN | TD_INC_DST_ADR);
	CyDmaTdSetAddress(DMA_1_TD[0], LO16((uint32)Module_1_SAR_WRK0_PTR), LO16((uint32)DMA_module1Data));
	CyDmaChSetInitialTd(DMA_1_Chan, DMA_1_TD[0]);
	CyDmaChEnable(DMA_1_Chan, 1);
}

void DMA_2_Config(void)
{
	#define DMA_2_BYTES_PER_BURST 1
	#define DMA_2_REQUEST_PER_BURST 1
	#define DMA_2_SRC_BASE (CYDEV_PERIPH_BASE)
	#define DMA_2_DST_BASE (CYDEV_SRAM_BASE)

	DMA_2_Chan = DMA_2_DmaInitialize(DMA_2_BYTES_PER_BURST, DMA_2_REQUEST_PER_BURST,
	HI16(DMA_2_SRC_BASE),HI16(DMA_2_DST_BASE));
	DMA_2_TD[0] = CyDmaTdAllocate();
	CyDmaTdSetConfiguration(DMA_2_TD[0], 2, DMA_2_TD[0], DMA_1__TD_TERMOUT_EN | TD_INC_DST_ADR);
	CyDmaTdSetAddress(DMA_2_TD[0], LO16((uint32)Module_2_SAR_WRK0_PTR), LO16((uint32)DMA_module2Data));
	CyDmaChSetInitialTd(DMA_2_Chan, DMA_2_TD[0]);
	CyDmaChEnable(DMA_2_Chan, 1);
}

void DMA_3_Config(void)
{
	#define DMA_3_BYTES_PER_BURST 1
	#define DMA_3_REQUEST_PER_BURST 1
	#define DMA_3_SRC_BASE (CYDEV_PERIPH_BASE)
	#define DMA_3_DST_BASE (CYDEV_SRAM_BASE)

	DMA_3_Chan = DMA_3_DmaInitialize(DMA_3_BYTES_PER_BURST, DMA_3_REQUEST_PER_BURST,
	HI16(DMA_3_SRC_BASE),HI16(DMA_3_DST_BASE));
	DMA_3_TD[0] = CyDmaTdAllocate();
	CyDmaTdSetConfiguration(DMA_3_TD[0], 2, DMA_3_TD[0], DMA_1__TD_TERMOUT_EN | TD_INC_DST_ADR);
	CyDmaTdSetAddress(DMA_3_TD[0], LO16((uint32)Module_3_DEC_SAMP_PTR), LO16((uint32)DMA_module3Data));
	CyDmaChSetInitialTd(DMA_3_Chan, DMA_3_TD[0]);
	CyDmaChEnable(DMA_3_Chan, 1);
}

void masker()
{

	long mask1 = 0x0000;                                    //0b0000000000000000
	long mask2 = 0x0000;                                    //0b0000000000000000
	long mask3 = 0x0000;                                    //0b0000000000000000
	int lower = 0;                                          // 0 a 2 sem bit-flip - 50%
	int upper = 5;                                          //3 a 5 com bit-flip - 50%
	int simples = 0;
	int duplo = 0;
	int triplo = 0;
```

```
int maxbits = 15;                                   //16 bits - 1, contagem inicia em 0
int aux = 0;

int num = (rand( ) % upper + 1 - lower)) + lower;

if (num >= 2)                                       //ocorrera mascaramento
        {

        duplo = (rand( ) % upper + 1 - lower)) + lower;                         //16%
        if (duplo == 5)                             //minimo duplo
                {

                upper = 20;
                triplo = (rand( ) % upper + 1 - lower)) + lower;        //5%
                if (triplo == 30)
                        {
                        aux = (rand( ) % maxbits + 1 - lower)) + lower;
                        long masking (aux, mask);
                        mask1 = mask;
                        aux = (rand( ) % maxbits + 1 - lower)) + lower;
                        long masking (aux, mask);
                        mask2 = mask;
                        aux = (rand( ) % maxbits + 1 - lower)) + lower;
                        long masking (aux, mask);
                        mask3 = mask;
                        }
                else
                if (num == 3)
                        {
                        aux = (rand( ) % maxbits + 1 - lower)) + lower;
                        long masking (aux, mask);
                        mask1 = mask;
                        aux = (rand( ) % maxbits + 1 - lower)) + lower;
                        long masking (aux, mask);
                        mask2 = mask;
                        }
                if (num == 4)
                        {
                        aux = (rand( ) % maxbits + 1 - lower)) + lower;
                        long masking (aux, mask);
                        mask2 = mask;
                        aux = (rand( ) % maxbits + 1 - lower)) + lower;
                        long masking (aux, mask);
                        mask3 = mask;
                        }
                if (num == 5)
                        {
                        aux = (rand( ) % maxbits + 1 - lower)) + lower;
                        long masking (aux, mask);
                        mask1 = mask;
                        aux = (rand( ) % maxbits + 1 - lower)) + lower;
                        long masking (aux, mask);
                        mask3 = mask;
                        }
                }
        else
        if (num == 3)
                {
                aux = (rand( ) % maxbits + 1 - lower)) + lower;
                long masking (aux, mask);
                mask1 = mask;
                }

        if (num == 4)
                {
                aux = (rand( ) % maxbits + 1 - lower)) + lower;
                long masking (aux, mask);
                mask2 = mask;
                }

        if (num == 5)
```

```
                                {
                                aux = (rand( ) % maxbits + 1 - lower)) + lower;
                                long masking (aux, mask);
                                mask3 = mask;
                                }

        SAR_voter_data ^= mask1;
        module2Data[1] ^= mask2;
        module3Data[1] ^= mask3;
}

long masking (aux, mask)
{
        case 0:
        mask = 0b0000000000000001;
        case 1:
        mask = 0b0000000000000010;
        case 2:
        mask = 0b0000000000000100;
        case 3:
        mask = 0b0000000000001000;
        case 4:
        mask = 0b0000000000010000;
        case 5:
        mask = 0b0000000000100000;
        case 6:
        mask = 0b0000000001000000;
        case 7:
        mask = 0b0000000010000000;
        case 8:
        mask = 0b0000000100000000;
        case 9:
        mask = 0b0000001000000000;
        case 10:
        mask = 0b0000010000000000;
        case 11:
        mask = 0b0000100000000000;
        case 12:
        mask = 0b0001000000000000;
        case 13:
        mask = 0b0010000000000000;
        case 14:
        mask = 0b0100000000000000;
        case 15:
        mask = 0b1000000000000000;

        return mask;
}

void SAR_ADC_voter()
SAR_voter_data = 0;
        bits = 16;
        samples = 5;
        tolerance = 6;
        for (i = 0; i < bits; i++)
                bit_counter[i] = 0;

        for (i = 0; i < bits; i++)
        {
                for (j = 1; j < samples; j++)
                {
                        bits = (module1Data[j] & mask[i]) != 0;
                        if (bits == 1)
                        bit_counter[i] = bit_counter[i] + 1;
                }
        }

        for (i = 0; i < bits; i++)
        {
                if (bit_counter[i] > tolerance)
                SAR_voter_data = SAR_voter_data + mask[i];
```

```
	}

		bigger = module1Data[1];
		smaller = module1Data[1];
		for (pointer = 2; pointer < samples; pointer++)
	{
		if (bigger < module1Data[pointer])
		bigger = module1Data[pointer];
		if (smaller > module1Data[pointer])
		smaller = module1Data[pointer];
	}
	error = bigger - smaller;
	if (error > tolerance)
	SAR_voter_error_det = 1;
}

void main_voter()
{
	error1 = abs (SAR_voter_data - module2Data[1]);
	error2 = abs (module2Data[1] - module3Data[1]);
	error3 = abs (module3Data[1] - SAR_voter_data);
	tolerance = 8;

	if (error1 <= tolerance)
		{
		system_output = SAR_voter_data;
		data_invalid_flag = 0;
		}
	else if (error2 <= tolerance)
		{
		system_output = module2Data[1];
		data_invalid_flag = 0;
		}
	else if (error3 <= tolerance)
		{
		system_output = module3Data[1];
		data_invalid_flag = 0;
		}
	else
		{
		system_output = 0;
		data_invalid_flag = 1;
		}

	if ((error1 > tolerance) || (error2 > tolerance) || (error3 > tolerance))
	main_voter_error_det = 1;
}

void buffer()
{
	if (buf_pointer_2 > 100)
		{
		buf_pointer_1 = 0;
		buf_pointer_2 = 0;
		}

	buf_1[buf_pointer_1] = module1Data[1];
	buf_pointer_1++;
	buf_1[buf_pointer_1] = module1Data[2];
	buf_pointer_1++;
	buf_1[buf_pointer_1] = module1Data[3];
	buf_pointer_1++;
	buf_1[buf_pointer_1] = module1Data[4];
	buf_pointer_1++;
	buf_1[buf_pointer_1] = module1Data[5];
	buf_pointer_1++;

	buf_2[buf_pointer_2] = module2Data[1];
	buf_3[buf_pointer_2] = module3Data[1];
	buf_4[buf_pointer_2] = SAR_ADC_voter_data;
	buf_5[buf_pointer_2] = system_output;
```

```
		buf_6[buf_pointer_2] = cycle_counter;
		buf_7[buf_pointer_2] = SAR _voter_error_det || main_voter_error_det;
		buf_pointer_2++;
}

void uart_print_time()
{
		second = RTC_ReadSecond();
		minute = RTC_ReadMinute();
		hour = RTC_ReadHour();
		day_month = RTC_ReadDayOfMonth();
		month = RTC_ReadMonth();

		sprintf(second_uart, "%d", second);
		sprintf(minute_uart, "%d", minute);
		sprintf(hour_uart, "%d", hour);
		sprintf(day_month_uart, "%d", day_month);
		sprintf(month_uart, "%d", month);

		UART_1_PutString(&hour_uart);
		do {tmpStat = UART_1_ReadTxStatus();} while(~tmpStat & UART_1_TX_STS_COMPLETE);
		UART_1_PutString(":");
		do {tmpStat = UART_1_ReadTxStatus();} while(~tmpStat & UART_1_TX_STS_COMPLETE);
		UART_1_PutString(&minute_uart);
		do {tmpStat = UART_1_ReadTxStatus();} while(~tmpStat & UART_1_TX_STS_COMPLETE);
		UART_1_PutString(":");
		do {tmpStat = UART_1_ReadTxStatus();} while(~tmpStat & UART_1_TX_STS_COMPLETE);
		UART_1_PutString(&second_uart);
		do {tmpStat = UART_1_ReadTxStatus();} while(~tmpStat & UART_1_TX_STS_COMPLETE);
		UART_1_PutString(" ");
		do {tmpStat = UART_1_ReadTxStatus();} while(~tmpStat & UART_1_TX_STS_COMPLETE);
		UART_1_PutString(&month_uart);
		do {tmpStat = UART_1_ReadTxStatus();} while(~tmpStat & UART_1_TX_STS_COMPLETE);
		UART_1_PutString("/");
		do {tmpStat = UART_1_ReadTxStatus();} while(~tmpStat & UART_1_TX_STS_COMPLETE);
		UART_1_PutString(&day_month_uart);
		do {tmpStat = UART_1_ReadTxStatus();} while(~tmpStat & UART_1_TX_STS_COMPLETE);
}

void uart_print_values()
{

		UART_1_PutString("**************************");
		do {tmpStat = UART_1_ReadTxStatus();} while(~tmpStat & UART_1_TX_STS_COMPLETE);
		UART_1_PutString("\n\r");
		do {tmpStat = UART_1_ReadTxStatus();} while(~tmpStat & UART_1_TX_STS_COMPLETE);
		sprintf(prev_faults_uart, "%d", prev_faults);
		UART_1_PutString("Voter Prev. faults: ");
		do {tmpStat = UART_1_ReadTxStatus();} while(~tmpStat & UART_1_TX_STS_COMPLETE);

		UART_1_PutString(&prev_faults_uart);
		do {tmpStat = UART_1_ReadTxStatus();} while(~tmpStat & UART_1_TX_STS_COMPLETE);

		UART_1_PutString("\n\r");
		do {tmpStat = UART_1_ReadTxStatus();} while(~tmpStat & UART_1_TX_STS_COMPLETE);

		UART_1_PutString("----------------------");
		do {tmpStat = UART_1_ReadTxStatus();} while(~tmpStat & UART_1_TX_STS_COMPLETE);

		UART_1_PutString("\n\r");
		do {tmpStat = UART_1_ReadTxStatus();} while(~tmpStat & UART_1_TX_STS_COMPLETE);

		UART_1_PutString("Faults in ");
		do {tmpStat = UART_1_ReadTxStatus();} while(~tmpStat & UART_1_TX_STS_COMPLETE);
		uart_print_time();
		UART_1_PutString(" :");
		do {tmpStat = UART_1_ReadTxStatus();} while(~tmpStat & UART_1_TX_STS_COMPLETE);
		UART_1_PutString("\n\r");
		do {tmpStat = UART_1_ReadTxStatus();} while(~tmpStat & UART_1_TX_STS_COMPLETE);

		UART_1_PutString("----------------------");
```

```
            do {tmpStat = UART_1_ReadTxStatus();} while(~tmpStat & UART_1_TX_STS_COMPLETE);
            UART_1_PutString("\n\r");
            do {tmpStat = UART_1_ReadTxStatus();} while(~tmpStat & UART_1_TX_STS_COMPLETE);
            uart_pointer_1 = 0;

for (uart_pointer_2=0; uart_pointer_2 < 101; uart_pointer_2++)
{
            sprintf(module1Data_1, "%d", buf_1[uart_pointer_1]);
            uart_pointer_1++;
            sprintf(module1Data_2, "%d", buf_1[uart_pointer_1]);
            uart_pointer_1++;
            sprintf(module1Data_3, "%d", buf_1[uart_pointer_1]);
            uart_pointer_1++;
            sprintf(module1Data_4, "%d", buf_1[uart_pointer_1]);
            uart_pointer_1++;
            sprintf(module1Data_5, "%d", buf_1[uart_pointer_1]);
            uart_pointer_1++;

            sprintf(module2Data_1, "%d", buf_2[uart_pointer_2]);
            sprintf(module3Data_1, "%d", buf_3[uart_pointer_2]);
            sprintf(SAR_voter_data_uart, "%d", buf_4[uart_pointer_2]);
            sprintf(system_output_uart, "%d", buf_5[uart_pointer_2]);
            sprintf(cycle_counter_uart, "%d", buf_6[uart_pointer_2]);
            sprintf(cycle_error_uart, "%d", buf_7[uart_pointer_2]);

            UART_1_PutString("Mod_1_[01]: ");
            do {tmpStat = UART_1_ReadTxStatus();} while(~tmpStat & UART_1_TX_STS_COMPLETE);
            UART_1_PutString(&module1Data_1);
            do {tmpStat = UART_1_ReadTxStatus();} while(~tmpStat & UART_1_TX_STS_COMPLETE);
            UART_1_PutString("\n\r");
            do {tmpStat = UART_1_ReadTxStatus();} while(~tmpStat & UART_1_TX_STS_COMPLETE);

            UART_1_PutString("Mod_1_[02]: ");
            do {tmpStat = UART_1_ReadTxStatus();} while(~tmpStat & UART_1_TX_STS_COMPLETE);
            UART_1_PutString(&module1Data_2);
            do {tmpStat = UART_1_ReadTxStatus();} while(~tmpStat & UART_1_TX_STS_COMPLETE);
            UART_1_PutString("\n\r");
            do {tmpStat = UART_1_ReadTxStatus();} while(~tmpStat & UART_1_TX_STS_COMPLETE);

            UART_1_PutString("Mod_1_[03]: ");
            do {tmpStat = UART_1_ReadTxStatus();} while(~tmpStat & UART_1_TX_STS_COMPLETE);
            UART_1_PutString(&module1Data_3);
            do {tmpStat = UART_1_ReadTxStatus();} while(~tmpStat & UART_1_TX_STS_COMPLETE);
            UART_1_PutString("\n\r");
            do {tmpStat = UART_1_ReadTxStatus();} while(~tmpStat & UART_1_TX_STS_COMPLETE);

            UART_1_PutString("Mod_1_[04]: ");
            do {tmpStat = UART_1_ReadTxStatus();} while(~tmpStat & UART_1_TX_STS_COMPLETE);
            UART_1_PutString(&module1Data_4);
            do {tmpStat = UART_1_ReadTxStatus();} while(~tmpStat & UART_1_TX_STS_COMPLETE);
            UART_1_PutString("\n\r");
            do {tmpStat = UART_1_ReadTxStatus();} while(~tmpStat & UART_1_TX_STS_COMPLETE);

            UART_1_PutString("Mod_1_[05]: ");
            do {tmpStat = UART_1_ReadTxStatus();} while(~tmpStat & UART_1_TX_STS_COMPLETE);
            UART_1_PutString(&module1Data_5);
            do {tmpStat = UART_1_ReadTxStatus();} while(~tmpStat & UART_1_TX_STS_COMPLETE);
            UART_1_PutString("\n\r");
            do {tmpStat = UART_1_ReadTxStatus();} while(~tmpStat & UART_1_TX_STS_COMPLETE);

            UART_1_PutString("Mod_2: ");
            do {tmpStat = UART_1_ReadTxStatus();} while(~tmpStat & UART_1_TX_STS_COMPLETE);
            UART_1_PutString(&module2Data_1);
            do {tmpStat = UART_1_ReadTxStatus();} while(~tmpStat & UART_1_TX_STS_COMPLETE);
            UART_1_PutString("\n\r");
            do {tmpStat = UART_1_ReadTxStatus();} while(~tmpStat & UART_1_TX_STS_COMPLETE);

            UART_1_PutString("Mod_3: ");
            do {tmpStat = UART_1_ReadTxStatus();} while(~tmpStat & UART_1_TX_STS_COMPLETE);
            UART_1_PutString(&module3Data_1);
            do {tmpStat = UART_1_ReadTxStatus();} while(~tmpStat & UART_1_TX_STS_COMPLETE);
```

```
		UART_1_PutString("\n\r");
		do {tmpStat = UART_1_ReadTxStatus();} while(~tmpStat & UART_1_TX_STS_COMPLETE);

		UART_1_PutString("SAR_ADC_voter: ");
		do {tmpStat = UART_1_ReadTxStatus();} while(~tmpStat & UART_1_TX_STS_COMPLETE);
		UART_1_PutString(&SAR_voter_data_uart);
		do {tmpStat = UART_1_ReadTxStatus();} while(~tmpStat & UART_1_TX_STS_COMPLETE);
		UART_1_PutString("\n\r");
		do {tmpStat = UART_1_ReadTxStatus();} while(~tmpStat & UART_1_TX_STS_COMPLETE);

		UART_1_PutString("System_output: ");
		do {tmpStat = UART_1_ReadTxStatus();} while(~tmpStat & UART_1_TX_STS_COMPLETE);
		UART_1_PutString(&system_output_uart);
		do {tmpStat = UART_1_ReadTxStatus();} while(~tmpStat & UART_1_TX_STS_COMPLETE);
		UART_1_PutString("\n\r");
		do {tmpStat = UART_1_ReadTxStatus();} while(~tmpStat & UART_1_TX_STS_COMPLETE);

		UART_1_PutString("Cycle_counter: ");
		do {tmpStat = UART_1_ReadTxStatus();} while(~tmpStat & UART_1_TX_STS_COMPLETE);
		UART_1_PutString(&cycle_counter_uart);
		do {tmpStat = UART_1_ReadTxStatus();} while(~tmpStat & UART_1_TX_STS_COMPLETE);
		UART_1_PutString("\n\r");
		do {tmpStat = UART_1_ReadTxStatus();} while(~tmpStat & UART_1_TX_STS_COMPLETE);

		UART_1_PutString("Cycle_error: ");
		do {tmpStat = UART_1_ReadTxStatus();} while(~tmpStat & UART_1_TX_STS_COMPLETE);
		UART_1_PutString(&cycle_error_uart);
		do {tmpStat = UART_1_ReadTxStatus();} while(~tmpStat & UART_1_TX_STS_COMPLETE);
		UART_1_PutString("\n\r");
		do {tmpStat = UART_1_ReadTxStatus();} while(~tmpStat & UART_1_TX_STS_COMPLETE);

		UART_1_PutString("----------------------");
		do {tmpStat = UART_1_ReadTxStatus();} while(~tmpStat & UART_1_TX_STS_COMPLETE);
		UART_1_PutString("\n\r");
		do {tmpStat = UART_1_ReadTxStatus();} while(~tmpStat & UART_1_TX_STS_COMPLETE);
}


void uart_control_write()
{

if ((SAR_voter_error_det == 1) || (main_voter_error_det == 1))
		{
		uart_print_status();
		SAR_voter_error_det = 0;
		main_voter_error_det = 0;
		count_alive = 0;;
		}
else
		{
		count_alive++;
		if (count_alive == 245000)
			{
			uart_print_status();
			count_alive = 0;
			}
		}
}


int main()
{
/* Enable led of reset */
Control_Reg_4_Write(1);


/* Initialize UART */
UART_1_Start();

/* Initialize DMA */
DMA_1_Config();
```

```
DMA_2_Config();
DMA_3_Config();


/* Initialize modules */
Module_1_Start();
Module_2_Start();
Module_3_Start();

/* Initialize counters */
Counter_1_Start();
Counter_2_Start();

/* Initialize clock_1 */
Clock_1_Start();

/* Delay to stabilize operation */
CyDelay(5000);

/* First write in UART */

{
UART_1_PutString("***** System started *****");
do {tmpStat = UART_1_ReadTxStatus();} while(~tmpStat & UART_1_TX_STS_COMPLETE);

UART_1_PutString("\n\r");
{tmpStat = UART_1_ReadTxStatus();} while(~tmpStat & UART_1_TX_STS_COMPLETE);

UART_1_PutString("\n\r");
do {tmpStat = UART_1_ReadTxStatus();} while(~tmpStat & UART_1_TX_STS_COMPLETE);


for (;;){}
}


/* Initialize Real time Clock */
RTC_TIME_DATE Start;

Start.Sec = 0u;
Start.Min = 0u;
Start.Hour = 0u;
Start.DayOfMonth = 1u;
Start.Month = 1u;
Start.Year = 2014u;

RTC_WriteTime(&Start);
RTC_Start();

for(;;)
{
/* Reset of flip-flops */
Control_Reg_2_Write(1);
CyDelayCycles(5);
Control_Reg_2_Write(0);

/* Wait for sampling control */
while (Status_Reg_1_Read() == 0)
{
}

/* Wait for the end of the sampling */
CyDelayCycles(660);

/* Start conversions */
Control_Reg_1_Write(1);
CyDelayCycles(5);
Control_Reg_1_Write(0);

/* Processing */
SAR_ADC_voter();
```

```
main_voter();
buffer();
uart_control_write();

/* Wait for the end of the conversions */
while (Status_Reg_2_Read() == 0)
{
}

/* Move DMA to temporary buffer */
module1Data[1] = DMA_module1Data[1];
module1Data[2] = DMA_module1Data[2];
module1Data[3] = DMA_module1Data[3];
module1Data[4] = DMA_module1Data[4];
module1Data[5] = DMA_module1Data[5];
module2Data[1] = DMA_module2Data[1];
module3Data[1] = DMA_module3Data[1];

/* Increments cycle_counter */
cycle_counter++;

/* Sends the running bit */
running_bit = !running_bit;
Control_Reg_3_Write(running_bit);
}
}
```