UNIVERSIDADE D
COIMBRA

João Ricardo Martelo de Castanheira

Desenvolvimento de eletrónica digital para monitorização de processos aerotermodinâmicos a bordo de sistemas de proteção térmica

Dissertação no âmbito do Mestrado em Engenharia Física orientada pelo Doutor Fernando António dos Santos Simões e pelo Professor Doutor Rui Miguel Curado da Silva e apresentada ao Departamento de Física da Faculdade de Ciências e Tecnologia.

Setembro de 2022

Agradecimentos

Agradeço aos meus orientadores Fernando Simões, Rui Silva e Rui Ventura pela ajuda, orientação e ensinamentos.

Agradeço à D. Alice, ao Sr. Fernando e ao Pedro Francisco e à restante família pela ajuda, apoio e motivação.

Agradeço ao Tiago, ao Carlos, ao Henrique e ao Vando e restantes amigos pela ajuda, disponibilidade e estarem presentes.

Agradeço a todos que de uma maneira ou de outra me apoiaram e incentivaram.

Agradeço à Jéssica.

Obrigado a todos que contribuíram para a concretização desta etapa.

Resumo

A ablação ocorrida durante a entrada na atmosfera de vários corpos celestes provoca recessão no escudo térmico de proteção. Medições sobre a recessão ocorrida ajudariam a melhorar e a otimizar os sistemas de proteção térmica, os modelos computacionais de simulação, a compreensão dos fenómenos aerotermodinâmicos, a melhorar a estimativa sobre o rácio da massa total perdida e a perceber melhor a interação entre o material e o plasma circundante. De modo a obter os dados sobre a recessão ocorrida em escudos térmicos, a Active Space Technologies começou a desenvolver o TPIP (*Thermal Protection System sensor based on Impedance Probe*), um sensor não intrusivo, colocado no interior do escudo térmico, que utiliza o princípio da impedância mútua para medir as propriedades dielétricas do material e, assim, calcular a recessão do escudo térmico. O funcionamento do TPIP consiste em dois elétrodos transmissores, cada um emitindo um sinal sinusoidal, e dois elétrodos recetores, que recebem os sinais emitidos pelos elétrodos emissores. Os quatro sinais analógicos são convertidos em digitais pelo ADC de um microcontrolador que, após a aquisição dos ditos sinais, faz o tratamento dos dados, isto é, faz a FFT (*Fast Fourier Transform*) dos quatro sinais com vista a calcular o rácio de amplitude entre emissores e recetores e diferença de fase destes. O TPIP está dividido em três módulos, o módulo analógico que engloba o circuito dos elétrodos, o módulo digital que consiste no circuito do microcontrolador e o módulo de *software* que consiste no tratamento dos dados. Visto que a Active Space Technologies já desenvolveu o módulo analógico, foi agora desenvolvido o módulo digital e parte do módulo de software. Foi escolhido um microcontrolador *rad hard* e um microcontrolador não *rad hard* equivalente. Como este último não possuía canais analógicos suficientes, foi adicionado ao circuito um ADC externo. Escolhido o microcontrolador e o ADC, desenvolveu-se um esquemático do circuito com todas as ligações necessárias e, seguidamente, converteu-se o esquemático para um PCB. Com o PCB montado foi desenvolvido o software. Ao validar o PCB constatou-se que havia problemas no ADC, não se tendo encontrado a causa dos problemas, mas que provavelmente se deveria à soldadura dos componentes. O software desenvolvido para o ADC externo não pôde ser testado. No entanto, foi desenvolvido um novo software que foi testado e com o qual foi possível observar que ambos os canais do ADC interno do microcontrolador fazem a aquisição dos dados simultaneamente, sendo que o ADC tem uma frequência de amostragem máxima de 500 kHz.

Palavras-chave: sistema de proteção térmica, impedância mútua, processamento de sinal, microcontrolador, instrumentação espacial

Abstract

Development of digital electronics for monitoring aerothermodynamic processes on board thermal protection systems

Ablation that occurs during the entry into the atmosphere of celestial bodies causes recession of the thermal protection system. Recession measurements would contribute to improve thermal protection systems, validate numerical models of aerothermodynamic phenomena as well as to assess mass ratio and better understand the interaction with the surrounding plasma. To obtain recession measurements from heat shields, Active Space Technologies is developing the TPIP (Thermal Protection System sensor based on Impedance Probe) sensor, a non-invasive sensor, placed inside the heat shield. The sensor uses the principle of mutual impedance to measure the dielectric properties of the material and subsequently estimate the heat shield recession. The TPIP architecture consists of two transmitting electrodes and two receiving electrodes. The four analog signals are digitized by an ADC (Analogue-Digital Converter) embedded in a microcontroller, which, after acquiring the signals, processes the data, e.g., performs the FFT (Fast Fourier Transform) of the four signals to calculate the receiver to transmitter amplitude as well as the phase difference. The TPIP sensor is divided into three modules: the analog module that also includes the electrodes, the digital module that comprises the microcontroller, and the software module for data processing and communications. Since Active Space Technologies has already developed the analog module, the digital module and part of the software module have been developed in this dissertation. A rad hard microcontroller and its non-rad hard equivalent have been chosen. The project comprises the design, development, manufacturing, assembly, and testing of the digital module. Although the digital module has not been completely validated, an important contribution has been made to the TPIP sensor.

Keywords: thermal protection system, mutual impedance, signal processing, microcontroller, space instrumentation

Acrónimos

ADC - Analogue-Digital Converter

ARAD - Analog Resistance Ablation Detector

CMSIS - Common Microcontroller Software Interface Standard

CPU - Central Processing Unit

EUA - Estados Unidos da América

FFT - Fast Fourier Transform

GPIO - General Purpose Input/Output

HEAT - Hollow aErothermal Ablation and Temperature

I2C - Inter-Integrated Circuit

IDE - Integrated Development Environment

JTAG - Joint Test Action Group

KSPS - KiloSamples Per Second

MIP – Mutual Impedance Probe

NASA - National Aeronautics and Space Administration

PCB - Printed Circuit Board

PICA - Phenolic Impregnated Carbon Ablator

PIT - Periodic Interrupt Timer

SPI - Serial Peripheral Interface

SWD - Serial Wire Debug

TPIP - Thermal Protection System sensor based on Impedance Probe

TPS – Thermal Protection System

UART - Universal Asynchronous Receiver/Transmitter

Índice de figuras

Índice de Tabelas

Índice

1. Introdução

Desde o início da exploração espacial foi necessário ter em atenção as temperaturas elevadas que são atingidas durante o processo de entrada ou reentrada em qualquer atmosfera, de modo a manter tripulantes e instrumentos intactos. Deste modo, foram criados os sistemas de proteção térmica (TPS) que previnem acontecimentos inesperados devido à ablação, fenómeno de erosão do material devido ao calor, que ocorre por consequência do aquecimento aerodinâmico. A ablação provoca uma recessão no escudo de proteção térmica em que uma espessura insuficiente originará estragos à nave e à sua carga. Como apenas se consegue avaliar a recessão das naves que retornam à Terra, os dados da recessão que ocorrem num escudo térmico em outros ambientes planetários ainda são escassos, levando a que a massa do sistema de proteção térmica seja maior do que realmente seria necessário. Como os modelos computacionais não têm dados suficientemente realistas sobre os fenómenos ocorridos durante a ablação, a única maneira de prevenir danos é colocar um sistema de proteção térmica com uma massa e espessura maior à necessária. Porém, uma massa maior provoca um aumento no peso, um parâmetro que deverá ser sempre minimizado. Dados sobre a recessão não ajudariam apenas a melhorar e otimizar o sistema de proteção térmica, como ajudariam também a melhorar a estimativa sobre o rácio da massa total perdida, os modelos computacionais, a compreensão sobre os fenómenos aerotermodinâmicos e a interação entre o material e o plasma formado na presença de temperaturas elevadas. A Active Space Technologies está a desenvolver o TPIP (*Thermal Protection System sensor based on Impedance Probe*), um sensor não intrusivo colocado na parte interior do escudo térmico, baseando-se na técnica de impedância mútua, a qual é capaz de medir as propriedades dielétricas do material, sendo assim possível calcular a recessão. Este projeto propõe-se a melhorar o nível de prontidão tecnológica de um instrumento desenvolvido para monitorizar os processos de ablação em escudos térmicos ocorridos durante a entrada na atmosfera, permitindo recolher mais dados, melhorar os modelos computacionais, refinar a compreensão sobre os fenómenos aerotermodinâmicos e estimar melhor a massa necessária do sistema de proteção térmica.

O projeto desenvolvido pela Active Space Technologies para a monitorização da recessão nos escudos térmicos consiste em quatro etapas, nomeadamente o desenvolvimento da componente analógica, digital, *software* e testes/validação.

A primeira etapa, já feita pela empresa, consistiu no desenvolvimento de um circuito eletrónico analógico que integra os elétrodos e a emissão e receção dos sinais.

A segunda etapa do projeto passará por escolher um microcontrolador capaz de converter os sinais analógicos dos elétrodos em sinais digitais através de um ADC (*Analogue-Digital Converter*), de processar os sinais de modo a calcular o rácio de amplitudes e o desvio de fase necessários para determinar as propriedades dielétricas

e de comunicar com o computador de bordo. Será necessário desenhar o circuito digital com os componentes escolhidos e depois construir um PCB (*Printed Circuit Board*).

A terceira etapa passará por desenvolver o código do *software* necessário para a aquisição dos sinais e do tratamento dos dados obtidos pelo ADC. O processamento será feito com recurso a FFT. Nesta etapa também será implementado um protocolo de comunicação, como o *CANbus*.

A quarta etapa do projeto consistirá em testes funcionais em laboratório, isto é, serão feitos testes onde se avaliará a variação entre o sinal transmitido e recebido e a sua estabilidade. Serão efetuados mais testes em laboratório, sendo simulados os processos de ablação. O objetivo será caracterizar estes processos no material que simula o material dos sistemas de proteção térmicos. Após a conclusão dos testes, começará a análise de dados e a validação do software. Será avaliado o desempenho do sensor e em que circunstâncias é adequado utilizá-lo para a monitorização do sistema de proteção térmica.

A contribuição deste trabalho para o projeto geral consistirá no desenvolvimento do módulo digital, que passará por escolher um microcontrolador *rad hard*, uma alternativa não *rad hard* e restantes componentes. Após a seleção dos componentes será feito um esquemático no Altium Designer onde estarão representados todos os componentes e ligações necessários. A seguir à validação do esquemático por parte do Altium Designer será desenvolvido o PCB. Será também desenvolvido *software* para testar o funcionamento e validar o PCB, especialmente o microcontrolador.

2. Estado da Arte

2.1. Sensores

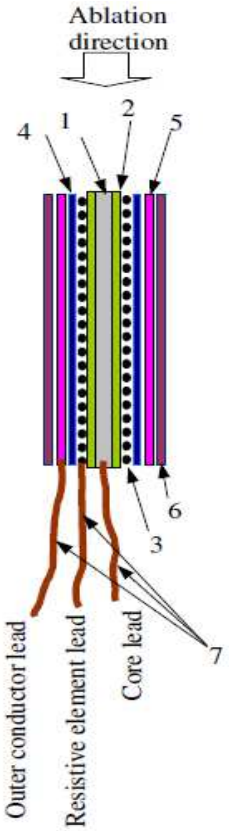
Ao longo dos anos, foram desenvolvidos sensores para dar informação sobre a recessão ablativa. A NASA desenvolveu dois sensores, o ARAD (*Analog Resistance Ablation Detector*) e o HEAT (*Hollow aErothermal Ablation and Temperature*).

2.1.1. ARAD

As versões desenvolvidas pela NASA baseiam-se no sensor ARAD à base de quartzo, desenvolvido pela *General Electric* para o governo dos EUA. O ARAD é um sensor intrusivo, utilizado na missão *Galileo* e no centro de investigação *Ames* da NASA, cujos modelos variam apenas em função dos materiais de construção utilizados, mantendo o mesmo princípio de funcionamento que consiste numa barreira eletricamente isolante que ao queimar devido às altas temperaturas, se torna condutora, completando assim o circuito entre os fios resistivos e condutores. À medida que os fios resistivos e condutores vão queimando, o seu tamanho encurta e, por consequência, a resistência diminui, originando uma diminuição da tensão [1]. O ARAD é constituído por seis camadas cilíndricas concêntricas. O sensor utilizado na missão *Galileo* tem no centro um núcleo de carbono fenólico seguido de camadas de fita de *Kapton*[1], de fio de platina-tungsténio, outra fita de *Kapton*, uma fita de níquel e por uma camada de fita de *Kapton* [2]. O ARAD desenvolvido no centro de investigação *Ames* é composto por um núcleo de grafite envolto por um tubo de *Kapton*, que por sua vez está envolto por um fio de platina-tungsténio e sucessivamente por outro tubo de *Kapton*, uma fita de níquel e por fim um terceiro tubo de *Kapton*. O tubo de *Kapton* atua como isolante entre as diversas camadas resistivas e condutoras, mas ao ser carbonizado devido às altas temperaturas torna-se condutor elétrico, fechando o circuito, permitindo a passagem de corrente. A principal diferença entre o ARAD da missão Galileo e o do ARAD do centro de investigação Ames está na composição do núcleo e na forma do isolante, carbono fenólico e fitas de *Kapton* para a missão *Galileo* e grafite e tubo de *Kapton* do *Ames*, mantendo-se o princípio de funcionamento. A mudança no núcleo deve-se à dificuldade de obtenção de carbono fenólico com um diâmetro muito pequeno [3]. Ao núcleo, ao fio de platina-tungsténio e à fita de níquel estão ligados fios de chumbo para condução

[1] *Kapton* – é uma poliimida, polímero com alta resistência mecânica e elétrica, sendo particularmente apropriado para aplicações espaciais. A sua principal característica é ser um excelente isolante elétrico.

da corrente (Figura 1). Sendo este um sensor intrusivo, é necessário perfurar o escudo térmico para a sua colocação, utilizando-se epóxi[2] para a sua fixação. Os resultados dos testes efetuados indicam que o ARAD por vezes perdia a sua conexão elétrica interna. Mesmo que esta ligação se mantivesse intacta, a incerteza nas medições era grande [1]. Os testes realizados com o ARAD mostram que o sensor tem uma resolução de recessão de ±0,09 cm a uma taxa de recessão de 0,1 cm s^{-1} [2]. Um dos resultados medidos na missão *Galileo*, mostrou uma recessão de 3,63 ± 0,15 cm na parte do escudo que mais sofreu com a ablação [4].



		Quartz	Galileo	Ames
1	Conductive core	Platinum wire and conductive epoxy	Carbon phenol	graphite
2	Inner Insulation barrier	Quartz tube	Polyimide ribbon wound	Polyimide tube
3	Resistive wire wound	Platinum tungsten	Platinum tungsten	Platinum tungsten
4	Middle insulation barrier	Quartz fiber	Polyimide ribbon wound	Polyimide tube
5	Outer conductor	Platinum wire and conductive epoxy	Nickel ribbon wound	Nickel ribbon wound
6	Outer insulation	None	Polyimide ribbon wound	Polyimide tube
7	Lead wires			

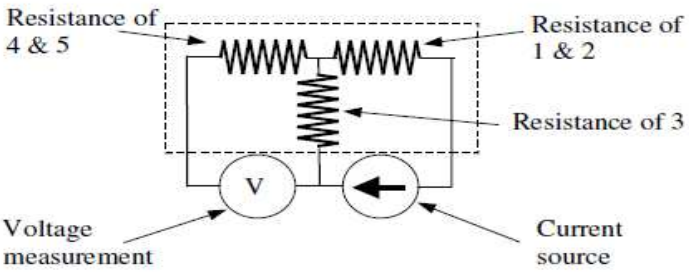


Figura 1 - Representação da estrutura do ARAD e o seu circuito elétrico correspondente [1].

[2] Epóxi – ou poliepóxido, é uma resina.

2.1.2. HEAT

O sensor HEAT é uma evolução do sensor ARAD. É um sensor também intrusivo. Consiste num núcleo feito do mesmo material que o sistema de proteção térmica, envolvido num tubo de *Kapton*, em cuja estrutura se encontram enrolados dois fios de platina-tungsténio (Figura 2). Quando sujeito a altas temperaturas, o *Kapton* vai derreter e tornar-se condutor devido ao carbono formado, fechando o circuito e permitindo a passagem de corrente. À medida que o comprimento do sensor vai sendo reduzido, diminui a resistência e por sua vez a tensão medida, mantendo-se o princípio de funcionamento do ARAD [5]. Os fios de platina-tungsténio estão soldados na ponta do sensor de modo a permitir uma conexão elétrica inicial [6]. Há oito possíveis configurações para o HEAT, variando as dimensões do tubo de *Kapton*, presença ou ausência de revestimento nos fios resistivos e o material com que é fixado o sensor ao escudo térmico. A resistência é de 2 kΩ por 2,54 cm de comprimento. A estabilidade da fonte de corrente afeta a precisão com que se mede o valor da resistência. Caso a fonte de corrente seja instável, podem-se corrigir as incertezas através de uma resistência de referência para monitorizar as flutuações na corrente ao longo do tempo [1]. Na missão do *Mars Science Laboratory* foi utilizado o HEAT, sendo detetada uma recessão que não ultrapassou os 2,54 mm, valor menor do que os 4,1 mm previstos pelas simulações [7].

O sensor HEAT garante várias melhorias nas medições e na sua capacidade de fabrico em relação ao sensor ARAD. O HEAT não sofre das perdas de conexão elétrica, pode ser utilizado com qualquer material do sistema de proteção térmica, tem resultados mais confiáveis, os custos de fabricação são baixos e ainda permite colocar um termopar no núcleo oco para medir a temperatura.

Há que ter em atenção que não se pode desprezar o coeficiente resistivo de temperatura, isto é, as temperaturas elevadas alteram o valor das resistências.

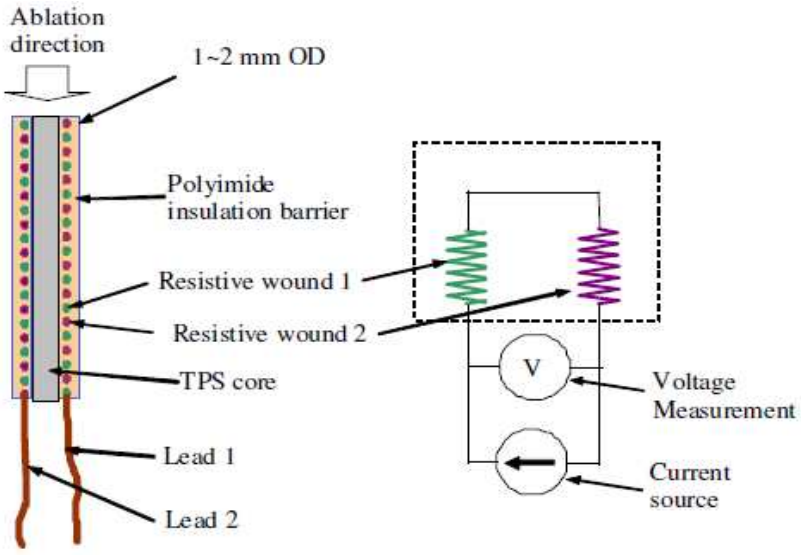


Figura 2 – Representação da estrutura do HEAT e o seu circuito elétrico correspondente [1].

2.2. TPS - Thermal Protection Systems

Durante as entradas atmosféricas os veículos espaciais atingem velocidades hipersónicas que provocam ablação no sistema de proteção térmica. Para proteger a carga destes veículos foram desenvolvidos ao longo dos anos sistemas de proteção térmica (TPS – *Thermal Protection Systems*), tendo vários materiais, geometrias e configurações sido testadas de maneira a melhorar o desempenho. Três métodos podem ser utilizados nos sistemas de proteção térmica: métodos passivos, métodos semi-passivos e métodos ativos.

Um sistema de proteção térmica passivo pode ser de três tipos: dissipador de calor, estrutura quente e estrutura isolada. O dissipador de calor é uma estrutura metálica com grande área de superfície que permite uma maior libertação do calor, devendo ser apenas utilizado para intervalos de tempo curtos quando em contacto com calor. As estruturas quentes possuem alta emissividade, o que permite a emissão por radiação da energia que recebe do calor, sendo a temperatura que esta classe pode suportar um fator limitante. As estruturas isoladas irradiam para o exterior a maior parte do calor que recebem e no interior têm uma camada de um material isolante que atrasa o processo de transferência de calor.

Os dois métodos semi-passivos são: tubos de calor e superfícies ablativas. Os tubos de calor são dispositivos utilizados para dissipar calor de forma eficiente. Todo a energia calorífica é transferida para uma zona onde não se fazem sentir efeitos de calor tão intensos. Tendo sido provado experimentalmente que podem falhar na proteção, foram considerados inadequados. As superfícies ablativas dissipam calor, sacrificando a superfície do material, isto é, à medida que o material vai carbonizando, a sua espessura vai diminuindo. Este método é o mais promissor e o mais utilizado. O material constituinte deste sistema de proteção térmica é o PICA (*Phenolic Impregnated Carbon Ablator*) ou derivados.

Foram testados três métodos ativos que são: refrigeração convectiva, refrigeração de filme e refrigeração por transpiração. A refrigeração convectiva passa por bombear um fluido refrigerante por baixo do escudo térmico para fornecer arrefecimento. A refrigeração de filme é usada em veículos hipersónicos, injetando um fluido refrigerante num local específico da superfície e deste modo o fluido formar uma fina camada fria isolante. A refrigeração por transpiração consiste em fazer passar um fluido refrigerante pelo meio da parede porosa do escudo térmico em direção à superfície, indo o fluido arrefecer a parede e formar uma fina camada fria isolante no exterior [8].

2.3. TPIP - Thermal Protection systems based on Impedance Probe

O sensor que a Active Space Technologies está a desenvolver chama-se TPIP (*Thermal Protection systems based on Impedance Probe*). Este sensor não é intrusivo e o principal requisito é que a resolução seja 0,5 mm para uma espessura do escudo de 20 mm. Nos testes já feitos com P50 e *Calcarb* foi obtida uma resolução melhor que a pretendida para uma espessura de 50 mm. P50 é um composto de cortiça usado nos escudos térmicos sendo um material poroso e um isolante eficaz. *Calcarb* é formado por fibras de carbono provenientes de fibras de *Rayon*. O TPIP pretende medir as propriedades dielétricas do material porque estas variam com a frequência, composição e temperatura geradas pela polarização do meio. A arquitetura do sensor é composta por três módulos: analógico, digital e software. A Active Space Technologies desenvolveu o módulo analógico onde são gerados os sinais dos elétrodos transmissores e onde os elétrodos recetores recebem o sinal. O módulo digital e de software foram para já feitos com recurso a um sistema de aquisição da *National Instruments* e a uma unidade de processamento de sinais do *LabVIEW*. O TPIP utiliza a técnica de impedância mútua, ou seja, injeta uma corrente sinusoidal de amplitude constante entre os dois elétrodos transmissores (Tx1 e Tx2) e induz uma tensão entre os dois elétrodos recetores (Rx1 e Rx2). Depois de efetuados os testes, concluiu-se que a melhor configuração para os quatro elétrodos seria Rx-Tx-Tx-Rx. O TPIP pretende monitorizar os processos de ablação fornecendo dados mais exatos e ajudar a investigar a interação entre o material e o plasma. A melhoria nos dados dever-se-á a uma melhor resolução espacial e temporal da medição.

Na tabela 1 podemos ver os requisitos funcionais do TPIP.

Tabela 1 - Requisitos do TPIP.

Número mínimo de sinais a adquirir	4 (2 emissores e 2 recetores)
Frequência dos sinais de estímulo	1 – 10 kHz
Taxa de amostragem	Pelo menos 10 vezes a frequência do sinal
Resolução dos sinais	12 bits ou superior
Tempo de aquisição	Pelo menos 10 ondas completas
Sincronismo entre sinais	Melhor que 1 grau
Componentes críticos	*Rad hard*
Repetição do ciclo de aquisição	10 s no máximo

2.4. Impedância mútua

Dadas duas antenas, uma transmissora e uma recetora, quando uma corrente percorre a antena transmissora, é induzida uma tensão na antena recetora, onde a impedância representa tanto o meio entre as duas antenas como o rácio entre a tensão induzida e a corrente do circuito. Neste caso não queremos calcular a impedância, mas sim as propriedades dielétricas, fazendo uso do princípio da impedância mútua. Uma sonda de impedância mútua (MIP – *Mutual Impedance Probe*) consiste em quatro elétrodos. Entre os dois elétrodos transmissores é injetado um sinal sinusoidal com amplitude constante e frequência (ω) conhecida, induzindo uma tensão entre os dois elétrodos recetores. A medição da amplitude e da fase nos elétrodos recetores permite calcular as propriedades dielétricas do meio (condutividade e permitividade).

Se (A_0,φ_0) forem a amplitude e a fase no vácuo ou num meio de referência e (A, φ) forem a amplitude e a fase num dado meio, pode-se obter a condutividade, σ, e a permitividade, ε, do meio, através das seguintes equações:

$$\sigma = \frac{A_0}{A}\ \omega\ \varepsilon_0\ \sin(\varphi - \varphi_0) \quad (1)$$

$$\varepsilon = \frac{A_0}{A}\ \cos(\varphi - \varphi_0) \quad (2)$$

onde ε_0 é a permitividade do vácuo.

3. Circuito

3.1. Escolha do microcontrolador *rad hard*

Devido às condições espaciais adversas, é necessário utilizar um *radiation hardened microcontroller*, ou seja, um microcontrolador capaz de resistir às condições espaciais adversas, tais como a exposição a radiação e a temperaturas elevadas. Toda a preparação especial dos materiais *rad hard* serve para prevenir danos físicos (o material quebrar ou derreter) e danos lógicos (perda de comunicações de dados e erros de processamento).

Atualmente, encontra-se disponível uma grande variedade de microcontroladores, mas quando se fala em microcontroladores *rad hard* as possibilidades de escolha diminuem significativamente. Os principais produtores de microcontroladores *rad hard* são a Microchip e a Vorago Technologies. Na Tabela 2, na Tabela 3 e na Tabela 4 podemos ver todos os microcontroladores encontrados e as suas respetivas especificações.

Os principais requisitos do projeto são que o microcontrolador tenha ADC interno, 4 canais de ADC com 12 bits de resolução e valores de memória e *clock* o maior possível.

É preferível um microcontrolador com ADC interno visto que adicionar um ADC externo tornará o circuito mais complexo e será mais uma possível fonte de erros e falhas. São necessários quatro canais de ADC para poder converter em digitais os quatro sinais analógicos dos elétrodos. Para se obter bons resultados com boa precisão precisamos de ter no mínimo um ADC com 12 bits de resolução.

De preferência, o microcontrolador escolhido deverá ter uma versão não *rad hard* para efetuar testes em laboratório devido aos preços elevados dos microcontroladores *rad hard*, mas não será um fator eliminatório na escolha final.

Os microcontroladores da Tabela 2 não vão contar para a escolha final porque são *radiation tolerant,* o que significa que a *total ionizing dose* é baixa comparativamente com os *radiation hardened,* o que pode levar a que sejam danificados mais facilmente e mais depressa, visto apenas serem resistentes a pequenas doses de radiação. Apesar do microcontrolador SAMV71Q21RT ter boas especificações, principalmente memória e *clock*, e cumprir os requisitos do projeto, como é *rad tolerant* foi excluído, embora tivesse sido uma boa opção.

Na Tabela 3 podemos ver vários microcontroladores *rad hard*, ou seja, mais bem preparados com uma maior proteção contra a radiação. Estes microcontroladores têm boas especificações principalmente em valores de *total ionizing dose* e de *clock*, mas não cumprem o requisito dos 12 bits de resolução do ADC nem o requisito dos quatro canais de ADC visto que estes microcontroladores não têm ADC interno, o que é um

problema devido às possíveis falhas ao adicionar um ADC externo, contudo qualquer um dos microcontroladores na Tabela 3 seria uma boa opção.

Das quatro possibilidades restantes que podemos ver na Tabela 4, começamos por excluir o microcontrolador MSP430FR5969-SP devido a ter uma *total ionizing dose* baixa, tal como os valores de memória e *clock*. Apesar do microcontrolador SAMRH707 ter uma arquitetura mais rápida e uma maior taxa de amostragem, estas características não são as mais importante pelo que é preferível o VA41620 ou o VA41630, pelo facto de terem uma *total ionizing dose* maior e um valor de *clock* muito maior em relação ao SAMRH707 e ao MSP430FR5969-SP.

A escolha final será entre o VA41620 e o VA41630. A diferença está na *total ionizing dose* e na existência de uma *non volatile memory* interna. O microcontrolador escolhido foi o VA41630 devido a ter uma NVM interna que poderá auxiliar no armazenamento de dados.

Apesar do VA41630 não ter uma versão não *rad hard* para testes, continua a ser a melhor escolha. Então será necessário construir um circuito equivalente a este microcontrolador para poder realizar testes em laboratório.

Tabela 2 - Microcontroladores rad hard e suas especificações.

Microcontrolador	SAM3X8ERT	ATmegaS128	ATmegaS64M1	SAMV71Q21RT
Marca	Microchip	Microchip	Microchip	Microchip
Tipo	*Rad tolerant*	*Rad tolerant*	*Rad tolerant*	*Rad tolerant*
Total ionizing dose	30 krad	30 krad	30 krad	30 krad
Arquitetura	ARM cortex M3 RISC	AVR 8-bit RISC	AVR 8-bit RISC	ARM cortex M7
Temperatura	-40°C – 105°C	-55°C – 125°C	-55°C – 125°C	-55°C – 125°C
Resolução do ADC	12 bits	10 bits	10 bits	12 bits
Taxa de amostragem do ADC	1 Msps		120 ksps	
Número de canais do ADC	16	8	11	24
CANBus	Sim	Não	Sim	Sim
Tensão de alimentação	3,3 V	3,3 V	3,3 V	3,3 V
UART, I^2C, SPI	Sim	Sim	Sim	Sim
Memória do código	512 kbytes	128 kbytes	64 kbytes	2048 kbytes
Memória dos dados	100 kbytes	4 kbytes	4 kbytes	384 kbytes
NVM				
Clock	84 MHz	8 MHz	8 MHz	300 MHz

Tabela 3 - Microcontroladores rad hard e suas especificações (continuação).

Microcontrolador	SAMRH71	VA10805	VA41628	VA41629
Marca	Microchip	Vorago technologies	Vorago technologies	Vorago technologies
Tipo	*Rad hard*	*Rad hard*	*Rad hard*	*Rad hard*
Total ionizing dose	100 krad	300 krad	300 krad	200 krad
Arquitetura	ARM cortex M7	ARM cortex M0	ARM cortex M4 (32 bits)	ARM cortex M4 (32 bits)
Temperatura	-55°C – 125°C	-55°C – 125°C	-55°C – 125°C	-55°C – 125°C
Resolução do ADC				
Taxa de amostragem do ADC				
Número de canais do ADC	0	0	0	0
CANBus	Sim	Não	Não	Não
Tensão de alimentação	1,65 V – 1,95 V núcleo, 3 V – 3,6 V periféricos	1,5 V núcleo, 3,3 V periféricos	3,3 V	3,3 V
UART, I2C, SPI	Sim	Sim	Sim	Sim
Memória do código	128 kbytes	128 kbytes	256 kbytes	256 kbytes
Memória dos dados	384 kbytes	32 kbytes	64 kbytes	64 kbytes
NVM				
Clock	100 MHz	50 MHz	50 MHz	100 MHz

Tabela 4 - Microcontroladores rad hard e suas especificações (continuação).

Microcontrolador	SAMRH707	MSP430FR5969-SP	VA41620	VA41630
Marca	Microchip	Texas instruments	Vorago technologies	Vorago technologies
Tipo	*Rad hard*	*Rad hard*	*Rad hard*	*Rad hard*
Total ionizing dose	100 krad	50 krad	300 krad	200 krad
Arquitetura	ARM cortex M7 (32 bits)	RISC (16 bits)	ARM cortex M4 (32 bits)	ARM cortex M4 (32 bits)
Temperatura	-55°C – 125°C	-55°C – 105°C	-55°C – 125°C	-55°C – 125°C
Resolução do ADC	12 bits	12 bits	12 bits	12 bits
Taxa de amostragem do ADC	1 Msps por canal	1 Msps	600 ksps	600 ksps
Número de canais do ADC	16	16	8	8
CANBus	Sim	Não	Sim	Sim
Tensão de alimentação	1,65 V-1,95 V núcleo, 3 V-3,6 V periféricos	1,8 V – 3,6 V	3,3 V	3,3 V
UART, I2C, SPI	Sim	Sim	Sim	Sim
Memória do código	128 kbytes		256 kbytes	256 kbytes
Memória dos dados	192 kbytes	2 kbytes	64 kbytes	64 kbytes
NVM	128 kbytes	64 kbytes		256 kbytes
Clock	50 MHz	16 MHz	100 MHz	100 MHz

3.2. Escolha de memória externa *rad hard*

Em caso de necessidade, para auxiliar no processamento dos dados é possível adicionar uma memória SRAM externa ao microcontrolador. Para calcular a quantidade de memória necessária para armazenar os sinais utiliza-se a seguinte fórmula:

$$Memória = (número\ de\ sinais) \times (resolução\ do\ ADC) \times (frequência\ de\ amostragem) \times (tempo\ de\ aquisição)$$

Sendo o número de sinais 4 e a resolução do ADC 12 bits.

Na segunda linha da Tabela 5 podemos ver os vários tempos de aquisição em milissegundos e na primeira coluna as várias frequências de amostragem em *kilohertz*.

Tabela 5 - Memória estimada em bytes para 4 sinais.

	12 bits			
	10 ms	100 ms	500 ms	1000 ms
2 kHz	120	1200	6000	12000
20 kHz	1200	12000	**60000**	**120000**
50 kHz	3000	30000	**150000**	**300000**
100 kHz	6000	**60000**	**300000**	**600000**

Visto que o microcontrolador VA41630 só tem disponíveis 64 kbytes de memória para guardar os dados, podemos ver a negrito os valores de tempo de aquisição e frequência de amostragem impossíveis de utilizar. Em caso de necessidade, pode-se acrescentar uma memória SRAM *rad hard* de 16 Mbit o que corresponde a 2000 kbytes.

No entanto, foi decidido não utilizar no circuito uma memória externa para não aumentar a complexidade.

Se posteriormente se se justificar e for necessário a utilização da memória externa, pode-se utilizar a memória rad hard SMV512K32.

3.3. Escolha de uma alternativa não *rad hard*

Tendo em conta o elevado preço do microcontrolador *rad hard* VA41630, houve a necessidade de escolher uma alternativa mais barata de modo a testar o sistema. As características mais importantes que foram consideradas na pesquisa para um

microcontrolador não *rad hard* foram os 64 kbytes de memória dos dados, 256 kbytes de memória do código, 256 kbytes de NVM e 100 MHz de *clock*.

O microcontrolador encontrado com as características necessárias foi o MK40DX256VLQ10, tendo também disponível o protocolo de comunicação CANBus. No entanto, visto que este não se encontrava em stock, foi utilizado o microcontrolador MK50DX256CLK10 da empresa *NXP Semiconductors,* sendo a única diferença o facto de este não ter o protocolo de comunicação CANBus.

3.4. Escolha do ADC para a alternativa não rad hard

Visto que a melhor opção encontrada, o MK50DX256CLK10, só tem dois canais de ADC quando são precisos quatro para converter os quatro sinais analógicos em digitais, é necessário acrescentar um ADC externo que tenha 12 bits de resolução e pelo menos quatro canais e que seja capaz de comunicar através de um protocolo de comunicação série. Além das condições anteriores, o ADC tem de ter uma taxa de amostragem de pelo menos 150 ksps, que é a taxa de amostragem do ADC do microcontrolador rad hard VA41630. Face a estas condições, foi escolhido o ADC ADS8528SRGCT, que tem uma resolução de 12 bits, oito canais e uma taxa de amostragem de 480 ksps em *serial communication*, preenchendo todos os requisitos.

3.5. Escolha dos restantes componentes

A maior disponibilidade e facilidade de aprovisionamento de uma fonte de alimentação de 12 V fez com que fosse necessário adicionar ao circuito um regulador de tensão para converter os 12 V em 5 V e outro regulador de tensão para converter os 5 V em 3,3 V, pois o microcontrolador é alimentado por uma tensão de 3,3 V e o ADC por tensões de 3,3 V, 5 V, -12 V e 12 V. Para converter os 12 V em 5 V utilizou-se o regulador de tensão ADP3335ARMZ-5-R7 e para converter de 5 V em 3,3 V utilizou-se o BU33SA5WGWZ-E2. Ambos os reguladores de tensão têm uma corrente de saída de 0,5 A.

Foi escolhido o conector DF51A-12DP-2DSA para ligar a alimentação e os sinais ao PCB e para ligar alguns pinos não utilizados do microcontrolador em caso de necessidade de fazer algum teste. Para o JTAG foi escolhido o conector FTSH-110-01-L-DV-K.

As resistências e os condensadores escolhidos são do tamanho 0805. O tamanho vem no formato comprimento x largura e as unidades de medida em polegadas, correspondendo a 2 mm por 1,2 mm.

Todos os componentes do circuito (microcontrolador, ADC, reguladores de tensão, conectores, condensadores e resistências) foram escolhidos conforme a disponibilidade em *stock* na altura da escolha.

3.6. Diagrama de blocos e arquitetura

Microcontrolador *rad hard* (VA41630):

Os sinais analógicos dos quatro elétrodos serão recebidos e digitalizados através do ADC interno do microcontrolador e serão armazenados e processados na *data* SRAM enquanto o microprocessador faz o tratamento dos dados. Este tratamento consistirá em transformar o domínio de tempo em domínio de frequência através de FFT e de seguida em calcular as médias dos sinais transmitidos e recebidos de modo a obter as amplitudes e fases para depois calcular o rácio das amplitudes e a diferença de fase, que serão posteriormente enviados pelo *transceiver* para quem estiver a monitorizar a missão (Figura 3).

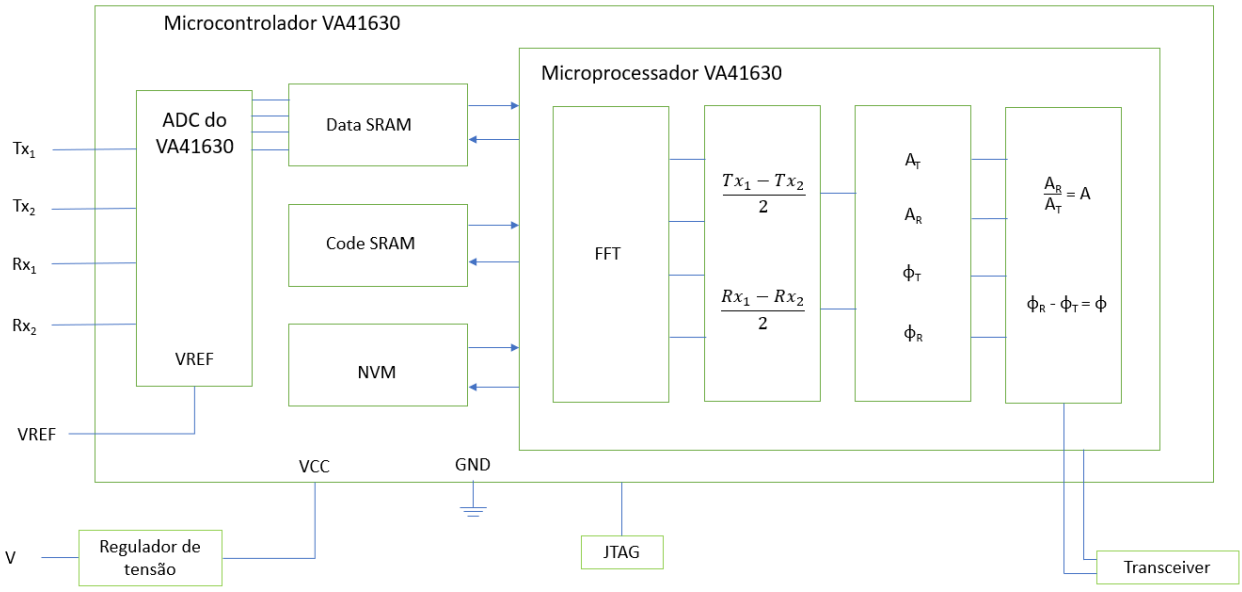


Figura 3 – Diagrama de blocos para o microcontrolador rad hard.

Microcontrolador não *rad hard* (MK50DX256CLK10):

Como o microcontrolador só tem dois canais de ADC e são precisos quatro, os quatro sinais analógicos dos elétrodos serão convertidos em digitais num ADC externo e depois da conversão serão enviados para a data SRAM do microcontrolador, onde este irá fazer o tratamento dos dados (Figura 4). Este tratamento consistirá em transformar o domínio de tempo em domínio de frequência através de FFT e de seguida em calcular as médias

dos sinais transmitidos e recebidos de modo a obter as amplitudes e fases com o objetivo de, posteriormente, calcular o rácio das amplitudes e a diferença de fase.

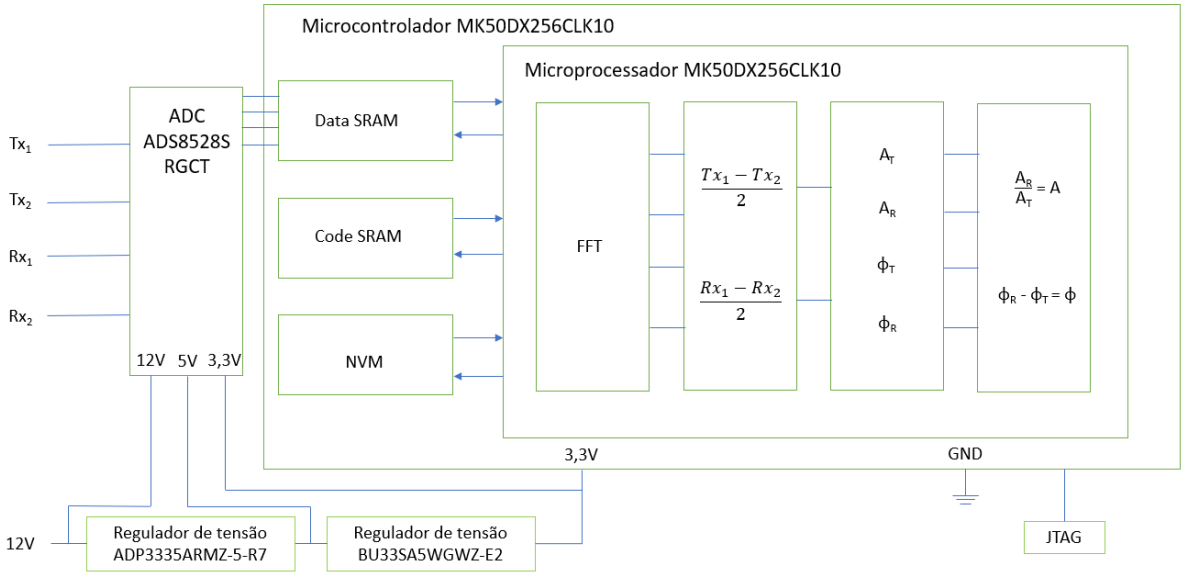


Figura 4 – Diagrama de blocos para o microcontrolador não rad hard.

3.7. Desenho do circuito

Na Figura 5 podemos observar o esquemático do circuito desenvolvido onde podemos ver os diversos componentes e as diversas ligações entre estes. Pode-se observar na imagem quais os pinos de SPI e GPIO utilizados para ligar o microcontrolador ao ADC e em que pinos do microcontrolador foram ligadas as ligações do JTAG. No apêndice A.1. representa-se a figura 5 noutra escala, em maior detalhe.

O esquemático foi realizado utilizando o programa Altium Designer. Altium Designer é um *software* que permite desenhar o esquemático de um circuito e transformá-lo num PCB, tendo várias ferramentas úteis disponíveis e permitindo também observar o PCB em três dimensões.

PCB significa *Printed Circuit Board*. É uma estrutura de camadas condutoras e isolantes onde estão soldados os componentes necessários e onde se encontram as pistas e vias para a passagem de corrente/sinal [9].

Os valores dos condensadores e resistências que são utilizadas no microcontrolador e no ADC foram tirados das *datasheets* do microcontrolador e do ADC, respetivamente. As ligações da alimentação do ADC foram feitas seguindo o exemplo dado na *datasheet* do ADC.

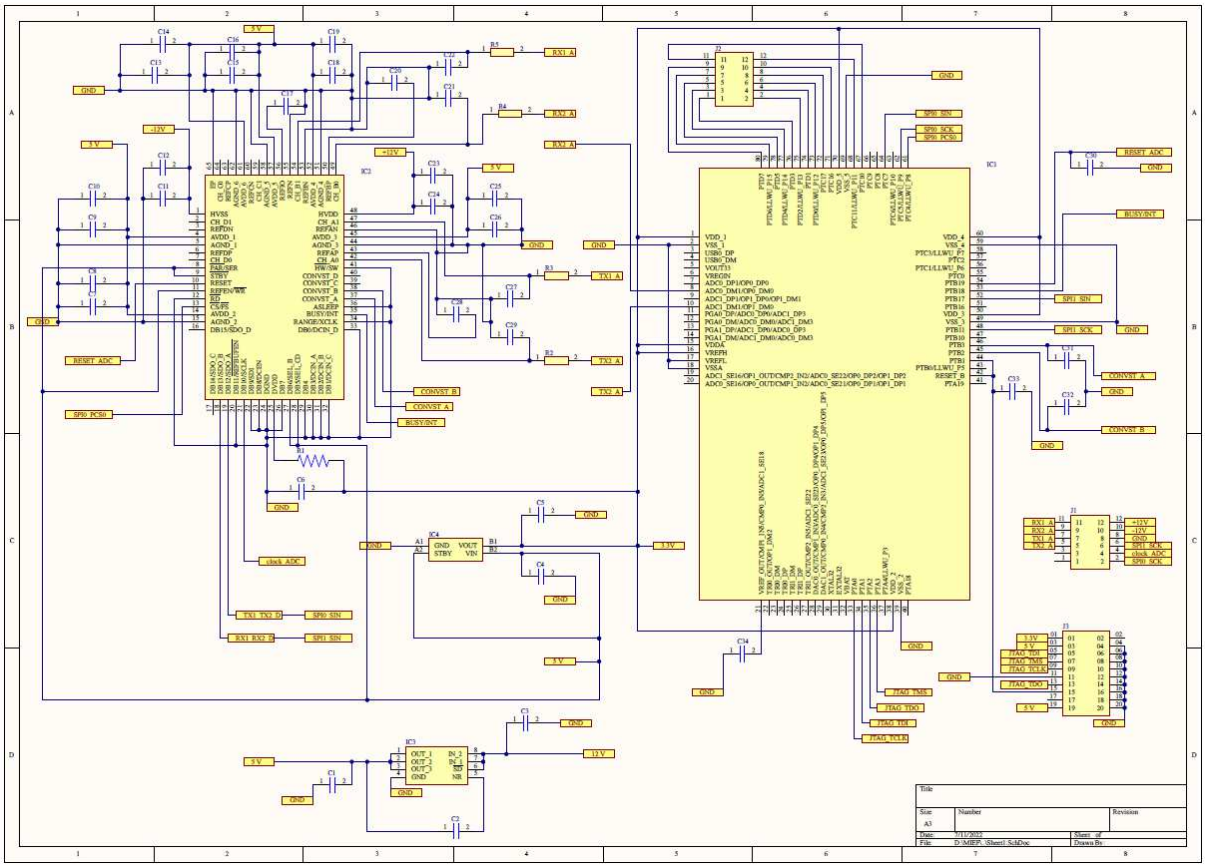


Figura 5 - Esquemático do circuito desenvolvido.

O microcontrolador possui os protocolos de comunicação SPI, I2C e UART. Foi escolhido o protocolo de comunicação SPI para enviar os sinais convertidos em digitais pelo ADC para o microcontrolador por ser o mais rápido e melhor para comunicação entre um baixo número de dispositivos periféricos. O SPI (*Serial Peripheral Interface*) é um protocolo de comunicação síncrono utilizado para curtas distâncias e serve para efetuar a comunicação entre um *master* e vários *slaves*, utilizando quatro ligações [10]. Neste caso, o microcontrolador será o *master* e só haverá um *slave* que será o ADC. O SPI tem quatro ligações principais, SPIx_SIN, SPIx_SOUT, SPIx_SCK e SPIx_PCSy, sendo que 'x' toma os valores de 0 ou 1 consoante o canal de SPI a ser utilizado e 'y' toma vários valores caso se utilize vários slaves, embora neste caso só se utilize um, logo só se utiliza o SPIx_PCS0. O SPIx_SIN é a comunicação do *slave* para o *master*, neste caso vai ser o envio dos dados do ADC para o microcontrolador. O SPIx_SOUT é a comunicação do *master* para o *slave*, neste caso não queremos enviar dados do microcontrolador para o ADC por isso não vai ser utilizado. O SPIx_SCK é o *clock* gerado pelo *master* de modo que o *master* e o *slave* funcionem ao mesmo ritmo, ou seja, sincronizados. O SPIx_PCSy representa o *chip select*, isto é, em caso de múltiplos *slaves*, o *chip select* vai dar a informação de qual *slave* é que o *master* vai receber a informação, para este circuito utiliza-se o *chip select* de um só canal porque queremos que ambos os canais do SPI recebam a informação ao mesmo tempo e basta um *chip select* de um dos canais do SPI para dar esta informação.

O JTAG (*Joint Test Action Group*) é uma interface que serve para programar e testar os microcontroladores quando colocados no PCB. Será com recurso a esta interface que o microcontrolador será programado e testado [11]. O JTAG utiliza quatro ligações, JTAG_TMS, JTAG_TDO, JTAG_TDI e JTAG_TCLK. O JTAG_TMS (*test mode select*) é controlado pela tensão neste pino e serve para controlar as ações do JTAG. O JTAG_TDO (*test data-out*) representa os dados que saem do microcontrolador. O JTAG_TDI (*test data-in*) representa os dados que são enviados para o microcontrolador. O JTAG_TCLK (*test clock*) representa a velocidade de funcionamento. Os pinos do JTAG foram ligados a um conector. Posteriormente, será ligado a este conector, montado no PCB, um adaptador que fará a ligação ao computador.

Foram utilizados três conectores, J1 para ligar à placa -12 V, 12 V e o *ground* e os sinais dos elétrodos. Foi utilizado o conector, J2, com ligação a pinos não utilizados do microcontrolador, caso seja necessário efetuar algum teste. No conector J3 encontra-se ligado o JTAG.

Os pinos RESET, CONVST_A e CONVST_B do ADC são pinos de input do ADC e estão ligados a pinos de output do microcontrolador. O RESET serve para reiniciar a aquisição de dados do ADC. Quando os pinos CONVST_A e CONVST_B recebem um sinal a 1 do microcontrolador começa a aquisição de dados dos canais A e B, respetivamente, por parte do ADC. O pino BUSY/INT do ADC é um pino de *output* que está ligado a um pino de *input* do microcontrolador e cuja função é enviar informação sobre se o ADC está ou não a funcionar.

Ambos os *clocks* dos canais do protocolo de comunicação SPI estão ligados ao conector J1 para que através de um *jumper* se escolha qual dos *clocks* a ser testado e utilizado. No entanto, é suposto ambos os *clocks* serem iguais.

Tal como já foi referido, foi necessário adicionar um ADC externo para converter os quatro sinais analógicos em digitais. Como o microcontrolador tem dois canais de ADC, estes dois canais serão utilizados para converter os sinais Tx2 e Rx2 com o objetivo de efetuar a comparação com os dados obtidos pelo ADC externo.

O ADC ADS8528SRGCT é um *simultaneous sampling* ADC. Significa que permite a aquisição de vários sinais ao mesmo tempo, sendo assim mais rápido na aquisição e uma boa escolha para a aplicação do projeto. Este ADC recebe os sinais nos canais escolhidos, mas envia os dados para o microcontrolador em pares de sinais, ou seja, o ADC recebe os sinais nos canais CH_A0 e CH_A1, mas vai enviar estes dois sinais para o microcontrolador através de um só canal, o DB12/SDO_A, por exemplo. O ADC pode funcionar em *hardware mode* ou *software mode*. Neste caso, foi escolhido o *hardware mode* por ser mais fácil fazer as ligações aos pinos do que programar o ADC.

O esquemático foi validado através da ferramenta '*Validate PCB Project*' que se encontra na opção '*Project*'. Esta ferramenta serve para verificar se todas as ligações estão corretas e se existe algum tipo de sobreposição ou pinos no ar, i.e., não ligados. Ao utilizar esta ferramenta e não aparecendo nenhum aviso de erro no ecrã depreendemos que não há problemas e o circuito está pronto para avançar para a próxima etapa, que é o PCB.

Os passos seguintes representam a forma como se processa a aquisição dos dados/sinais.

Passos para a aquisição e envio de dados:

Passo 1: O microcontrolador envia um sinal digital a 1 através dos pinos PTB2 e PTB3 para os pinos CONVST_A e CONVST_B do ADC para os canais A e B (CH_A0, CH_A1, CH_B0, CH_B1) começarem a receber os dados.

Passo 2: Os canais A e B (CH_A0, CH_A1, CH_B0, CH_B1) do ADC recebem os dados.

Passo 3: Os dados são guardados nos buffers do ADC enquanto não são transferidos para o microcontrolador.

Passo 4: O microcontrolador através do pino PTC4/LLWU_P8 manda um sinal a 0 para o pino FS para o ADC enviar os dados para o microcontrolador.

Passo 5: O ADC através dos pinos DB12/SDO_A (TX1 e TX2) e DB13/SDO_B (RX1 e RX2) envia os dados para os pinos PTC7 e PTB17 do microcontrolador.

Passo 6: O microcontrolador recebe os dados.

Passo 7: O microcontrolador faz o tratamento dos dados.

Passo 8: O microcontrolador transmite os dados para o satélite que por sua vez enviará para o controlo da missão na Terra.

Passo 9: Volta ao passo 1.

3.8. PCB

O PCB foi desenhado utilizando o programa *Altium Designer*.

É possível ver o esquemático do PCB no apêndice A.2.

De modo a construir um PCB totalmente funcional e sem falhas, há regras fundamentais a ter em atenção.

As principais regras são:

- As pistas (*traces*) que fazem a ligação de sinais diferentes não se podem cruzar para não sobrepor os sinais.

- As pistas não podem fazer ângulos de 90 graus, porque isso leva ao aquecimento nos cantos e introduz maiores gradientes de temperatura.

- As pistas devem ser o mais diretas e curtas possível para minimizar a resistividade total da pista.

- Respeitar a distância mínima entre componentes, pistas e vias.

- Os condensadores das alimentações (*decoupling capacitors*) devem ser colocados o mais próximo possível dos pinos das respetivas alimentações dos componentes porque funcionam como filtros para estabilizar a tensão.

- A largura das pistas da alimentação deve ser superior à das pistas que fazem a ligação dos sinais, porque nessas pistas a corrente é mais elevada necessitando de dissipar mais calor.

- Seguir as indicações e requisitos do fabricante escolhido para o fabrico do PCB.

Como o circuito é relativamente pouco complexo, será feito um PCB de somente duas camadas (*double layer*) - uma camada superior e outra inferior (*top and bottom layer*). Num PCB de duas camadas, as camadas estão expostas ao ar, logo a dissipação de calor nestas camadas será maior.

O tamanho e o peso do PCB não são, para já, requisitos importantes para esta etapa do projeto, apenas serão relevantes ao construir a versão final. O tamanho desta versão não *rad hard* poderia ter sido reduzido, mas tal como referido acima, não é um requisito principal nesta etapa do projeto.

Para desenhar o PCB no Altium Designer foi criado um PCB para onde foi importado o esquemático, isto é, foram importados todos os componentes e as respetivas ligações. Posteriormente, os componentes foram colocados em posições estratégicas, de modo a minimizar o comprimento das pistas. As pistas foram colocadas através da funcionalidade '*auto-route*' do Altium Designer que conseguiu completar 100% das ligações, sendo necessário apenas alguns ajustes de modo a otimizar as ligações e o espaço.

No desenho do PCB encontramos ao lado dos condensadores um círculo amarelo que representa o polo positivo do condensador (sempre que a polaridade do mesmo seja relevante). O microcontrolador, o ADC, os conectores e os reguladores de tensão também têm um círculo amarelo ou um traço amarelo que serve para indicar a posição do pino número um. Estes círculos têm como função mostrar a orientação dos componentes e são úteis na fase de montagem, nomeadamente se esta for manual. As resistências não necessitam deste tipo de informação pelo facto de que ligar numa direção ou na outra será igual.

O Altium Designer inclui regras padrão já definidas pelo que foi necessário alterar certas regras de modo que o *software* não acuse erros no PCB desenvolvido. A empresa JLCPCB, fabricante do PCB, tem um valor mínimo fixo para o tamanho e grossura dos caracteres, logo os caracteres que servem de identificação de componentes no PCB têm de ter um tamanho de 40 mil[3] e uma grossura de 6 mil [12]. A distância entre componentes foi alterada para 6 mil, visto que 10 mil é muito e os componentes ficariam muito afastados uns dos outros, tornando o PCB muito grande.

[3] Mil - é a unidade de medida de comprimento do sistema inglês de medidas e representa um milésimo de uma polegada, 1 mil equivale a 0,0254 mm, sendo a unidade que o Altium Designer utiliza por defeito.

Para calcular a largura das pistas que fazem as ligações das alimentações utilizaram-se as seguintes fórmulas:

Primeiro, é calculada a área:

$$Área[mils^2] = \left(Corrente[Amps] \div \left(k \times \left(Temp_{Aum[°C]}\right)^b\right)\right)^{\frac{1}{c}} \quad (3)$$

Depois, é calculada a largura:

$$Largura[mils] = Área[mils^2] \div \left(Espessura[oz] \times 1{,}378\left[\frac{mils}{oz}\right]\right) \quad (4)$$

Para camadas internas IPC-2221: k=0,024, b=0,44, c=0,725;

Para camadas externas IPC-2221: k=0,048, b=0,44, c=0,725;

Onde k, b e c são constantes resultantes do ajuste de curvas às curvas IPC-2221.

Os valores da corrente e da espessura utilizados foram 1 A e 1 mil, respetivamente. Foram escolhidos estes valores para obter resultados para casos extremos, porque na verdade a corrente que sai dos reguladores de tensão é 0,5 A e a espessura da camada é 1,4 mil. Foi considerada a temperatura ambiente de 25°C e um aumento de temperatura de 10°C. A largura das pistas das alimentações (+12 V, -12 V, 5 V, 3,3 V, 0) é 16,3 mil enquanto o resto das pistas tem uma largura de 10 mil (esta última é a largura padrão do Altium Designer).

Foram criados pontos de teste ao longo do PCB para tornar possível a visualização de alguns sinais importantes através de um osciloscópio, ou ver o valor da tensão através de um multímetro, para verificar se é igual ao esperado e está tudo a funcionar como era suposto. Os pontos de testes são criados através da colocação de uma via por cima da pista por onde passa o sinal, permitindo assim verificar o sinal que passa na pista em questão. Todos os pontos de testes foram identificados. A Tabela 6 mostra a identificação de cada ponto de teste e a que sinal corresponde.

Tabela 6 – Pontos de teste.

Sinal	Identificação
RX1	TP1
RX2	TP2
TX1	TP3
TX2	TP4
clock_ADC	TP5
SPI_PCS0	TP6
JTAG_TCLK	TP7
CONVST_A	TP8
CONVST_B	TP9

Foram feitas várias tentativas até encontrar a configuração ideal.

Visto que o circuito é menor que o tamanho padrão do Altium Designer, o tamanho do PCB foi reduzido para 2885 mil por 2875 mil, o que corresponde a 7,3279 cm por 7,3025 cm. O PCB tem um total de 46 componentes.

O PCB foi validado pela ferramenta disponibilizada pelo Altium Designer, '*Design Rule Check*', que se encontra na opção '*Tools*' e permite detetar qualquer violação das regras pré-definidas, de modo a corrigir os eventuais erros. Após não se verificar nenhuma violação das regras são produzidos os *gerber files*, que são os ficheiros que contêm a informação para fabricação do PCB.

Depois de terem sido produzidos os *gerber files* reparou-se que faltou definir a *keep out layer* e as *mechanical layers*. Estas camadas servem para definir as fronteiras dos componentes bem como as dimensões do PCB. Depois de definidas estas camadas foram gerados novos *gerber files*.

Foi feito o pedido de fabricação à empresa JLCPCB, empresa chinesa especializada na fabricação de PCBs, onde foram anexados os *gerbers files*. A empresa respondeu dizendo que faltava a *drill layer* para puderem começar o fabrico. Foi necessário gerar outro tipo de ficheiros, os *NC Drill Files*. Após se enviar de novo os *gerber files* e os *NC Drill Files* a empresa fabricou o PCB. O PCB só pode ser construído (Figura 6) quando os ficheiros de fabrico *gerber file*s e os *NC Drill Files* estão disponíveis, porque estes contêm informação crucial para o fabrico (e.g., furação) da placa.

A Figura 6 mostra o PCB que foi encomendado à empresa JLCPCB. Mostra as pistas, as vias e onde vão ser soldados os componentes. Após a chegada da placa (sem componentes) à Active Space Technologies, esta foi observada ao microscópio de modo a tentar descobrir alguma falha física, sendo também testada com recurso a um multímetro para detetar possíveis curto-circuitos.

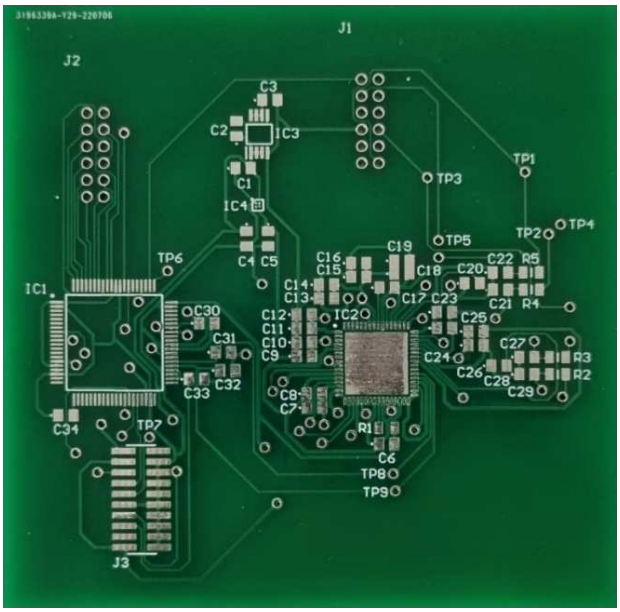


Figura 6 - PCB sem componentes.

Na figura 7 podemos ver o PCB com os componentes soldados. Os componentes foram colocados e soldados na Active Space Technologies. Notar que a maioria dos componentes é muito pequena e têm de ser colocados na placa com pasta e depois soldados em forno. Posteriormente, por vezes, são dados retoques com ferro de soldar.

Após a colocação dos componentes, o PCB foi de novo observado ao microscópio com o intuito de se procurar possíveis falhas na solda e se alguns pinos foram soldados entre eles por acidente.

Alguns testes com multímetro são novamente realizados para detetar, por exemplo, curto-circuitos ou ligações indevidas nos pinos de comunicação. Os testes lógicos são efetuados posteriormente, durante o desenvolvimento do *software*.

A lista dos componentes utilizados na montagem do PCB encontra-se no Apêndice A.3.

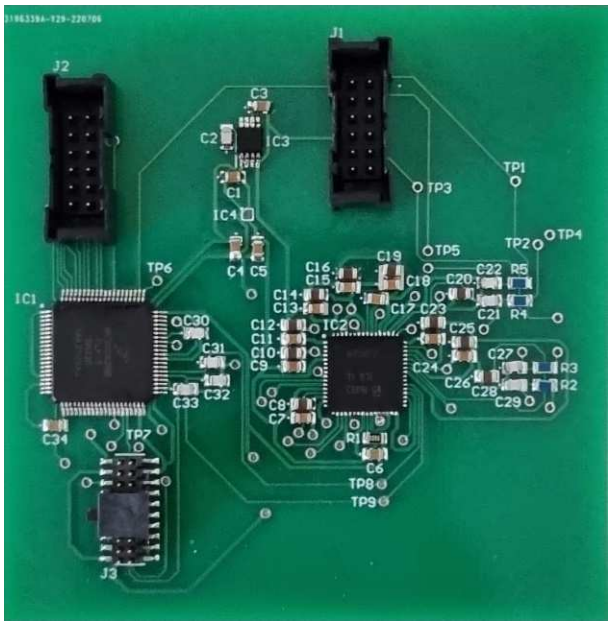


Figura 7 - PCB com componentes.

4. *Software*

4.1. Introdução ao desenvolvimento do *software* e IDE

O IDE utilizado para o desenvolvimento do *software* foi o MCUXpresso, que é o IDE da empresa do microcontrolador, pois permite configurar mais facilmente o microcontrolador, nomeadamente os pinos e os periféricos que vão ser utilizados.

IDE significa *Integrated Development Environment* e serve para programadores desenvolverem código de *software* mais facilmente dado que o IDE reúne todas as ferramentas necessárias para o seu desenvolvimento. Um IDE contém principalmente um editor de código, um compilador e um *debugger* [13].

O *software* é desenvolvido na linguagem de programação C, que é compatível com o MCUXpresso e mais adequada para a programação de microcontroladores.

O software vai ser carregado para o microcontrolador através do protocolo de comunicação JTAG.

4.2. Requisitos/funcionamento do *software*

O *software* desenvolvido serve para controlar o ADC e tratar os dados enviados por este para o microcontrolador.

Cada pino do microcontrolador tem várias funções disponíveis pelo que é necessário escolher a configuração necessária. Através da opção '*pins*' na opção '*ConfigTools*' no IDE foram definidas as configurações dos pinos que vão ser utilizados como GPIO, SPI e JTAG.

O *software* terá de enviar sinais do microcontrolador para o ADC, para este começar a recolha dos dados e enviá-los para o microcontrolador. Aquando da receção desses dados, o *software* guardá-los-á em vetores para, de seguida, através de FFT obter as amplitudes e as fases dos quatro sinais recolhidos pelo ADC.

O ADC vai enviar os dados recebidos nos pinos CH_A0 e CH_A1 pelo pino SDO_A e os dados recebidos nos pinos CH_B0 e CH_B1 pelo pino SDO_B que estão ligados aos pinos SPI0_SIN e SPI1_SIN do microcontrolador. O canal SDO_A envia primeiro os dados de CH_A0 e de seguida os dados de CH_A1 e o canal SDO_B envia primeiro os dados de CH_B0 e de seguida os dados de CH_B1. Como cada canal de emissão envia dados de dois canais de receção, vai ser preciso dividir no *software* o vetor que guarda os dados

recebidos. Em cada canal, SDO_A e SDO_B, vão ser recebidos 32 bits e depois divididos em duas metades, correspondendo os primeiros 16 bits a CH_A0 ou a CH_B0 e os segundos 16 bits a CH_A1 ou a CH_B1, dependendo do canal.

O IDE permite utilizar o protocolo de comunicação SPI de duas formas. Uma maneira é com funções próprias do MCUXpresso, a outra é através de funções do CMSIS. Foi utilizado o CMSIS pela simplicidade de utilização. CMSIS significa *Common Microcontroller Software Interface Standard*, é um conjunto de funções compatível com todos os processadores ARM *Cortex* e uma interface genérica de modo a ser possível reutilizar *software*. Estas funcionalidades foram desenvolvidas pela Keil [14].

4.3. Desenvolvimento do *software*

Para desenvolver o código, foi necessário começar por ativar os pinos, o *clock* e os periféricos no IDE. De seguida, adicionam-se as drivers necessárias, tais como, a *driver* do CMSIS, porque vamos utilizar o CMSIS para fazer o protocolo de comunicação SPI, bem como a *driver* do GPIO porque temos pinos de entrada e saída que vamos utilizar e *drivers* do *clock*, das portas e do PIT (*Periodic Interrupt Timer*).

Para ajudar a desenvolver o código foram utilizados partes dos exemplos disponibilizados no MCUXpresso que é o IDE da NXP [15].

Para o microcontrolador receber os dados do ADC externo foi definido um protocolo de comunicação SPI, para isto foram utilizadas as funções do CMSIS já definidas, funções que permitem inicializar, ativar, desativar e desligar o protocolo bem como receber os dados. Tanto no site da MCUXpresso como no site da Keil podemos obter informação sobre as várias funções e parâmetros que cada função aceita, bem como a descrição do funcionamento de cada função [16], [17].

Os pinos de GPIO utilizados foram definidos como *input* ou *output* consoante a sua função e foram utilizadas funções do IDE para pôr o valor de cada pino de *output* a 0 ou a 1 ou ler o valor do pino de *input*.

O ADC envia pelo canal SDO_A para o microcontrolador 32 bits; desses 32 bits os primeiros 16 bits correspondem ao canal CH_A0 e os segundos 16 bits correspondem ao canal CH_A1 e da mesma maneira funciona o canal B. Logo será necessário dividir os dois *arrays* de 32 bits em quatro *arrays* de 16 bits correspondendo aos canais CH_A0, CH_A1, CH_B0 e CH_B1. Após a divisão, os 16 bits de cada canal representam um número em binário e vão ser transformados em decimal e guardados em *arrays* correspondentes a cada canal. O ADC apenas envia um ponto de cada vez por isso é preciso fazer um ciclo que repita o número de vezes do tamanho do *array* pretendido.

Para fazer a aquisição dos dados sempre à mesma cadência utilizou-se um PIT. Quando se define um *interrupt* para uma função específica, sempre que ocorre a condição definida, é "levantada" uma *flag*. Quando essa *flag* é levantada, o CPU para a execução do programa e é executado o *handler,* que é a função associada ao *interrupt*. Após a execução do *handler,* o CPU continua a execução do código no exato sítio onde parou [18]. PIT significa *Periodic Interrupt Timer* e permite fazer a aquisição de dados sempre com a mesma taxa de execução. Sempre que passa esse tempo, levanta uma *flag* que para a execução do programa e vai para o *handler* do PIT que simplesmente vai baixar a *flag* e mudar o valor de uma variável booleana. O intervalo de tempo é constante porque mesmo que a aquisição seja mais rápida o programa vai ficar preso dentro de um ciclo até passar o tempo definido.

O tempo a que foi feita cada aquisição será guardado em cada posição de um *array*, multiplicando o número da iteração do ciclo pelo intervalo de tempo definido.

Após o desenvolvimento do *software*, o código será enviado para o microcontrolador através da interface JTAG e posteriormente testado.

O funcionamento global do código desenvolvido consiste em ativar o ADC para fazer a aquisição de um ponto de cada sinal, ou seja, no total quatro pontos. Após a aquisição desses pontos o microcontrolador vai "pedir" para o ADC os enviar. Quando o microcontrolador recebe os pontos, vai recebê-los em pares e dividir os 32 bits em 16 bits de modo a passar para os quatro pontos e irá passá-los para decimal e guardar nos *arrays* correspondentes. Esta aquisição está desenvolvida dentro de um ciclo que repete estes passos para o número de vezes igual ao tamanho definido para os *arrays*.

4.4. Pseudocódigo

O pseudocódigo abaixo representa a implementação do que foi descrito na secção 5.3. e no apêndice B.1. encontra-se o código desenvolvido.

Includes

Defines

Variáveis

```
PIT_Handler {
        pitIsrFlag = true
```

```
        Baixar a flag do interrupt
}
SPI_Master_SignalEvents{}
Main{
        Inicialização da board
        Configuração dos GPIO
        Definição dos GPIO utilizados
        Inicialização do master do SPI0 e SPI1
        Inicialização e ativação do PIT

        Para i=0 até i=tamanho {
                pitIsrFlag = false;
                Começar o timer do PIT
                Convst_a=1
                Convst_b=1
                Enquanto busy==1 {
                        Recebe o valor de busy
                }

                Convst_a=0
                Convst_b=0
                Fs=0
                SPI0.receive
                SPI1.receive

                Enquanto true {
                        Se SPI0.getdatacount==32 e SPI1.getdatacount==32{
                                Para i=0 até i=16 {
                                        Dividir os 32 bits do canal SPI0 em dois arrays de 16 bits
```

```
                                        Dividir os 32 bits do canal SPI1 em dois arrays de 16 bits
                                }
                                Break
                        }
                }

                Para i=0 até i=16 {
                        Passar os 16 bits do array canal_A0 para decimal
                        Passar os 16 bits do array canal_A1 para decimal
                        Passar os 16 bits do array canal_B0 para decimal
                        Passar os 16 bits do array canal_B1 para decimal
                }

                canal_A0[i]=decimalA0;
                canal_B0[i]=decimalB0;
                canal_A1[i]=decimalA1;
                canal_B1[i]=decimalB1;
                tempo[i]=intervalo_tempo*i;

                Enquanto pitIsrFlag == false {
                        ;
                }
                Se i == tamanho {
                        Desativar o PIT
                }
        }
        Desligar o master do SPI0 e SPI1
}
```

5. Testes e validação

5.1. Validação do PCB

Para validar o PCB, utilizou-se um multímetro de modo a efetuar o teste de continuidade, isto é, para verificar se há ligação entre as duas pontas deste, sendo que, se houver ligação, o multímetro emite um sinal sonoro.

O teste de continuidade utiliza a bateria interna do multímetro. Colocam-se as pontas de prova deste entre dois pontos do PCB. Se houver ligação, é sinal de que o circuito está fechado e a resistência entre os dois pontos é muito baixa (e.g., inferior a 1 Ω), sendo emitido um sinal sonoro. Se entre os dois pontos não houver ligação, a resistência é muito grande e diz-se que é um circuito aberto [19].

O condensador C28 apresentou um defeito. A ponta esquerda está ligada ao *ground*, como seria suposto, e a ponta direita está ligada ao ADC, também como seria suposto. No entanto, a ponta direita também está ligada ao *ground*, o que não era suposto. Como a razão para este problema era desconhecida, foram efetuadas várias tentativas para perceber de onde apareceu este erro. Primeiro, o condensador foi retirado para verificar se não havia solda a ligar as duas pontas tendo-se verificado que não, logo o problema não seria daqui. De seguida, a ligação entre a ponta direita do condensador C28 e o ADC foi cortada de modo a perceber se era o pino do condensador ou o pino 43 do ADC que estava ligado ao *ground* e conclui-se, utilizando o mesmo teste do multímetro, que era o pino do ADC que estava ligado ao *ground*. Posteriormente, foi cortada a ligação ao pino 55 do ADC pois este pino também fazia a ligação ao *ground* por baixo do ADC. O problema podia estar na solda colocada por baixo do ADC que podia estar a conectar o pino 43 ao *ground*. Verificou-se que o problema não seria deste *ground* por isso a ligação do condensador C28 ao pino 43 do ADC e a ligação ao pino 55 foram restabelecidas através de um fio soldado a ligar o locais que foram cortados.

Como resultado, o condensador C28 foi perdido e substituído por um novo condensador diferente, mas com o mesmo valor de capacitância. No final, o motivo deste problema não foi inequivocamente descoberto.

Após o teste de continuidade, para comprovar se todas as ligações estavam operacionais, ligou-se a fonte de alimentação ao conector J1 de modo a alimentar o PCB e ver se a alimentação estava a ser feita normalmente. Verificou-se que estava em falta uma ligação no PCB, tendo-se confirmado que no Altium Designer esta ligação, dos +12 V do conector J1 para o regulador de tensão IC3, também estava em falta. A alternativa encontrada com a finalidade de remediar a falta desta ligação foi soldar um fio entre o pino dos +12 V do conector J1 da camada inferior e os pinos dos +12 V do regulador de tensão IC3 na camada superior, tendo-se obtido sucesso.

Utilizou-se de novo o multímetro para continuar a validação do PCB e verificar se a tensão correta chegava aos pinos corretos. Começou-se pelos +12 V e os -12 V e em

todos os condensadores que recebem estas tensões obtiveram-se valores de 11,96 V e -12,01 V, respetivamente. Os condensadores e os pinos do ADC que seriam supostos receber 5 V, recebiam 5,07 V. Estas pequenas diferenças nas tensões medidas em comparação com as tensões teóricas não são relevantes porque são mínimas. Outro problema apareceu quando se mediu a tensão nos condensadores e nos pinos do microcontrolador pois supostamente dever-se-ia medir 3,3 V, mas media-se 4,15 V. Pôs-se em dúvida o funcionamento do regulador de tensão IC4 por ser de reduzidas dimensões (tamanho da ordem de 2 mm) e extremamente difícil de soldar pelo que poderia estar mal soldado e a causar erros ao não converter os 5 V em 3,3 V. Não sendo possível testar o regulador de tensão IC4 isoladamente, para tentar resolver este problema escolheu-se outro regulador de tensão que converta de 5 V para 3,3 V e que tenha pinos altos de modo a ligá-lo numa *breadboard*, pois o PCB estava desenhado para o regulador de tensão IC4. Com o novo regulador de tensão ligado na *breadboard,* continuou-se a medir uma tensão de 4,15 V. Ao se testar o regulador de tensão isoladamente do PCB verificou-se que a tensão de saída era 3,29 V, o que indica o seu bom funcionamento, pois a diferença para os 3,3 V não é relevante. O novo regulador de tensão é o LF33ABV, e tal como o anterior, converte 5 V em 3,3 V além de também ter uma corrente de saída de 0,5 A.

Como o problema dos 4,15 V se mantinha, para tentar perceber a sua origem, foi cortada a alimentação dos 3,3 V ao ADC. Ao cortar esta ligação verificou-se que o circuito passava a ter uma alimentação de 3,3 V, como esperado, e o microcontrolador passou a ser alimentado com a tensão de alimentação correta, logo o problema seria do ADC. Dado que não se encontrou nenhum problema no desenho do circuito e tirar o ADC do PCB não seria opção, foi decidido que não se utilizaria o ADC nem o *software* desenvolvido pois o código englobava o uso do ADC. Como alternativa, decidiu-se utilizar apenas os dois canais do ADC interno do microcontrolador e desenvolver um novo *software*.

No final, apenas se utilizou o microcontrolador para receber dois sinais, um sinal transmissor e um sinal recetor. Tirando o ADC externo, que não será utilizado nesta placa, todas as ligações e tensões foram verificadas e estão corretas.

Como a origem dos problemas no condensador C28 e na alimentação de 3,3 V ao ADC não foi descoberta e não se encontrou nenhum erro no esquemático do circuito, acredita-se que os problemas verificados durante os testes se devam a problemas na soldadura, devido a serem componentes de dimensões muito reduzidas; notar que a estanhagem foi feita manualmente ou com recurso a um forno em detrimento do típico "*pick-and-place*" automático. A Figura 8 mostra o PCB com as várias correções realizadas.

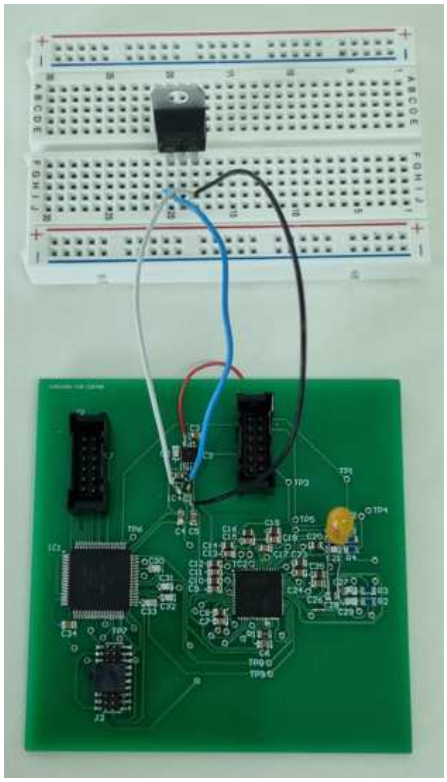


Figura 8 - PCB com a ligação em falta restaurada, o condensador novo e o regulador de tensão externo

5.2. Validação do *software*

Enquanto o PCB estava a ser fabricado e os componentes montados na placa, começou-se o desenvolvimento do *software*.

Como o PCB não estava a funcionar como era suposto e consequentemente se passou a utilizar apenas os dois canais do ADC interno do microcontrolador em vez do ADC externo, foi necessário desenvolver um novo *software*. O *software* desenvolvido para a utilização do microcontrolador com o ADC externo não pôde ser testado. Quando se verificou que seria necessário desenvolver um *software* novo, o *software* para o ADC externo deixou de ser desenvolvido e apenas fará a aquisição dos quatro *arrays* correspondentes aos quatro sinais uma única vez.

5.2.1. Software novo

O novo *software* receberá dois sinais, um sinal transmissor e um sinal recetor.

Foram utilizadas as funções do MCUXpresso para inicializar e ativar o ADC e fazer a aquisição dos dados. Cada canal de ADC irá guardar os dados nos *arrays* correspondentes. A taxa de amostragem a que é feita a aquisição dos dados será controlada novamente através do PIT. O código foi desenvolvido de modo a adquirir o número de pontos definido, isto é, só se fazia a aquisição dos pontos até os *arrays* estarem cheios.

5.2.2. Pseudocódigo

O pseudocódigo abaixo representa a implementação do que foi descrito na secção 6.2.1. e no apêndice B.2. encontra-se o código desenvolvido.

```
Includes
Defines
Variáveis
PIT_Handler {
        pitIsrFlag = true
        Baixar a flag do interrupt
}
Main {
        Inicialização da board;
        Configuração e inicialização do ADC0;
        Configuração e inicialização do ADC1;
        Configuração e inicialização do PIT;
        Enquanto i < tamanho {
                Se pitIsrFlag == true {
                        pitIsrFlag = false;
                        Configuração do canal de conversão do ADC0;
```

```
                    Configuração do canal de conversão do ADC1;
                    Canal0[i] = valor da conversão do ADC0;
                    Canal1[i] = valor da conversão do ADC1;
                    I++;
                }
        }
}
```

5.3. JTAG para SWD

Com a finalidade de carregar o código desenvolvido para o microcontrolador, pensou-se utilizar a interface JTAG. No entanto, a *probe* de JTAG disponível na Active Space Technologies não é compatível com o MCUXpresso pois este IDE apenas utiliza *probes* específicas. Como uma parte destas são muito dispendiosas, a alternativa encontrada foi utilizar a *probe* MCU-LINK que pertence à NXP. Apesar desta *probe* suportar JTAG e SWD, decidiu-se trocar a interface a ser utilizada para SWD pois serão necessárias menos ligações entre a *probe* e o PCB [20]. Esta decisão deveu-se ao facto de no PCB estar um conector para JTAG de 20 pinos e o MCU-Link ter um conector com 10 pinos, tendo sido por esse facto necessário soldar fios ao lado dos pinos do conector J3 de modo a ser possível fazer as ligações (Figura 9).

SWD significa *Serial Wire Debug* e é uma interface série que apenas necessita de duas ligações, SWDIO e SWDCLK, e serve para fazer a comunicação entre o computador e o microcontrolador, tal como o JTAG [21].

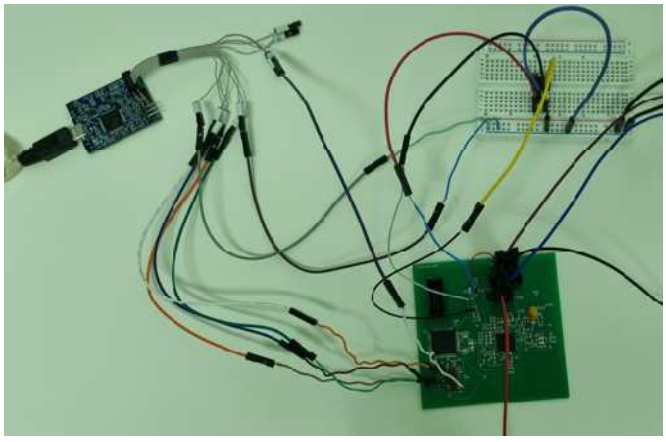


Figura 9 - Ligações do SWD.

5.4. Testes

Para iniciar os testes foi necessário estabelecer a ligação entre o microcontrolador e o computador onde se encontra instalado o IDE, tendo esta ligação sido feita através do protocolo SWD. Foi necessário atualizar o *firmware* da *probe*. O MCUXpresso consegue identificar a *probe* do MCU-Link assim que a ligação é efetuada e se faz o *debug*. Após estabelecida a ligação, foi ligada uma fonte de alimentação de 12 V ao PCB de modo a alimentar o microcontrolador.

No princípio foram testados os exemplos do MCUXpresso. Como os microcontroladores da série MK50DX256xxx10 não têm nenhuma placa de desenvolvimento foram utilizados os exemplos disponíveis para a placa twrk60d100m, que é a placa de desenvolvimento dos microcontroladores da série MK60DN512xxx10, visto os exemplos serem compatíveis com os microcontroladores da série MK50DX256xxx10. Inicialmente foi testado o exemplo '*twrk60d100m_demo_apps_hello_world*' de modo a se verificar a ligação entre o IDE e o microcontrolador e de seguida, o exemplo referente ao ADC interno do microcontrolador, '*twrk60d100m_driver_examples_adc16_polling*', para se perceber se as ligações estavam corretas, se havia boa comunicação e se os canais do ADC estavam funcionais. Escolhido o exemplo, seleciona-se a opção '*debug*' no IDE. Após o início do *debug* escolhe-se a opção '*resume*' para que o código seja executado. Os PRINTF do programa são vistos na consola. Quando se experimentou o primeiro exemplo para receber dados do ADC não foi possível visualizar os PRINTF na consola pelo facto de que quando se cria o projeto em '*Project Options*', em '*SDK Debug Console*', a predefinição é UART e tem de se mudar para *Semihost*. Após a mudança para *Semihost* já foi possível observar os valores na consola.

O MCUXpresso tem uma opção que permite tornar as variávies do código em variáveis globais e visualizar os seus valores e as mudanças de valor em tempo real. Não se utilizou esta opção porque a variação do valor das variáveis tem de ser lenta o suficiente para se poder notar a diferença. Por exemplo, no caso do ADC, se a aquisição for muito rápida, só conseguimos ver o último valor lido, pelo que para se notar alguma diferença, a aquisição deve ser feita, no mínimo, em intervalos de 1 segundo.

A seguir a testar os exemplos ligou-se o gerador de sinais ao PCB, nos pinos 9 e 5 do conector J1, sendo que cada pino estava ligado a um canal diferente. Para verificar que a onda que o gerador de sinais emitia era igual ao que o ADC recebia, utiliza-se a função PRINTF para apresentar os valores na consola e copiou-se os valores para o Excel, onde se construiu um gráfico. Na Figura 10 podemos ver as ligações do computador, do gerador de sinais e da fonte de alimentação ao PCB.

Os canais do ADC interno têm ambos 12 bits de resolução, o que significa que recebem valores até $2^{12} = 4096$.

O ADC não é capaz de fazer a conversão de valores negativos, facto que levou a que fosse necessário adicionar um *offset* com o objetivo de tornar a onda positiva na sua

totalidade. Esta solução é particularmente importante para funções sinusoidais, uma vez que as ondas quadradas e dente de serra (ou triangular) são normalmente definidas com mínimo em zero.

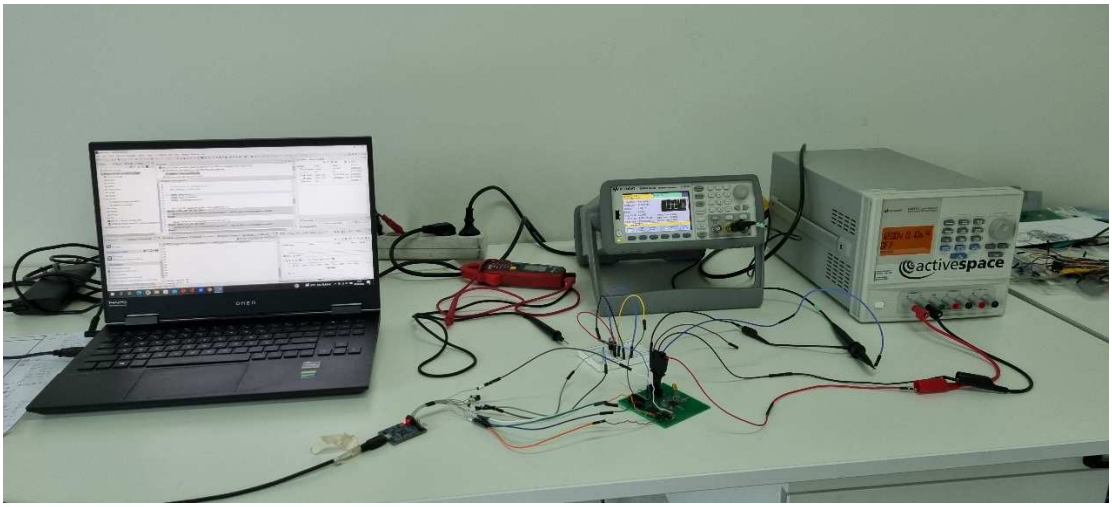


Figura 10 – Ligações do computador, do gerador de sinais e da fonte de alimentação ao PCB.

Começou-se por testar o canal 0 do ADC que corresponde ao pino 9 do conector J1. Escolheu-se uma onda triangular (dente de serra) no gerador de sinais e obteve-se o gráfico da Figura 11.

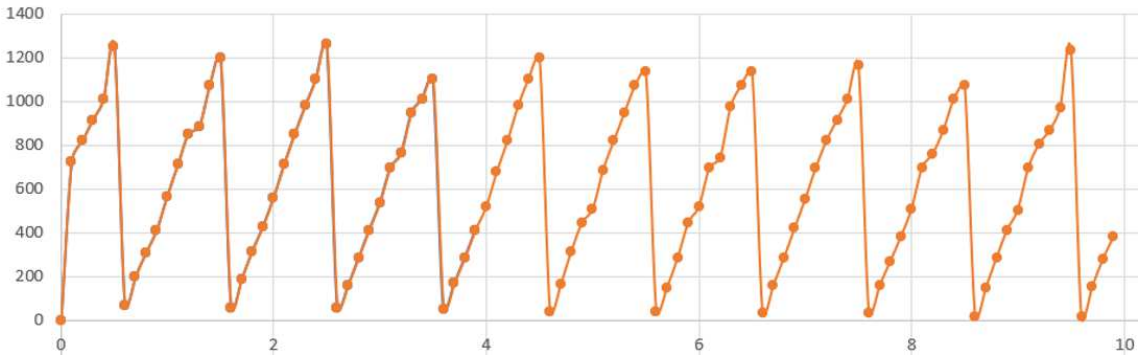


Figura 11 – Exemplo do canal0 a fazer a aquisição de uma onda triangular.

Neste caso, a amplitude e frequência não são relevantes, trata-se de um teste. Podemos observar que não é exatamente uma onda triangular como esperado. As pequenas variações devem-se ao ruído por o circuito não estar numa situação ideal, todos as ligações e fios soldados contribuíram para estas variações, provocando assim o ruído. Portanto, todas as aquisições dos sinais terão ruído.

Definiram-se duas ondas sinusoidais com amplitude de 500 mVpp, *offset* de 260 mV, tamanho da aquisição de 20 pontos e intervalo de aquisição de 0,1 s. Como o ADC não foi capaz de receber os números negativos, foi definido um offset para a onda ser positiva, o zero da onda passou a estar nos 250 mV, logo a onda passa do intervalo 250 mV a 250 mV para 0 V a 500 mV, que no gráfico corresponde ao intervalo 0 a 1300 com o zero nos 650. Na Figura 12 vemos a laranja uma onda com frequência de 2 Hz e a azul uma onda com frequência de 1 Hz. Da figura 12 podemos concluir que ambos os canais do ADC estão a funcionar corretamente.

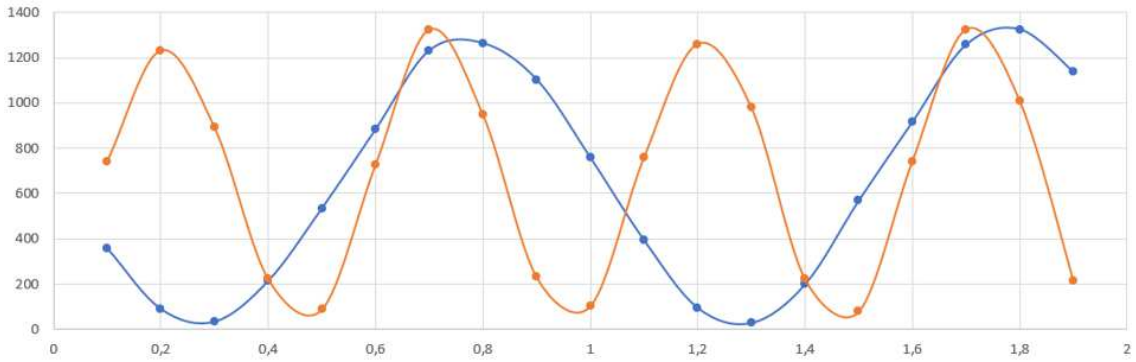


Figura 12 – Exemplo dos dois canais a fazer a aquisição dos dados.

É necessário verificar o sincronismo dos dois canais do ADC porque é preciso que ambos os canais façam a aquisição ao mesmo tempo. Foram escolhidas três frequências para os sinais, 1 kHz, 10 kHz e 50 kHz; para cada uma destas frequências foram feitas dez testes a uma frequência de amostragem de 500 kHz e um tamanho de cada array de 7000 pontos. Depois cada grupo de testes de cada frequência foi analisado em Matlab. Primeiro foram calculados a correlação e através de FFT as diferenças de fase dos sinais e de seguida foi calculada a média destes dois valores e o desvio padrão. Na tabela 7 podemos ver a correlação e a diferença de fase para cada frequência.

Tabela 7 - Correlação e diferença de fase para 1 kHz, 10 kHz e 50 kHz

Frequência	1 kHz	10 kHz	50 kHz
Correlação	0,9956 ± 0,0033	0.7290 ± 0.0605	0.5571 ± 0.0030
Diferença de fase	-5.0463 ± 1.5707	42.8397 ± 5.5791	55.2135 ± 0.2336

Podemos concluir que quando a frequência diminui, maior é a correlação e menor é a diferença de fase, como era esperado. Quanto menor for a frequência maior é o sincronismo entre canais do ADC.

Na figura 13 podemos verificar visualmente as conclusões tiradas.

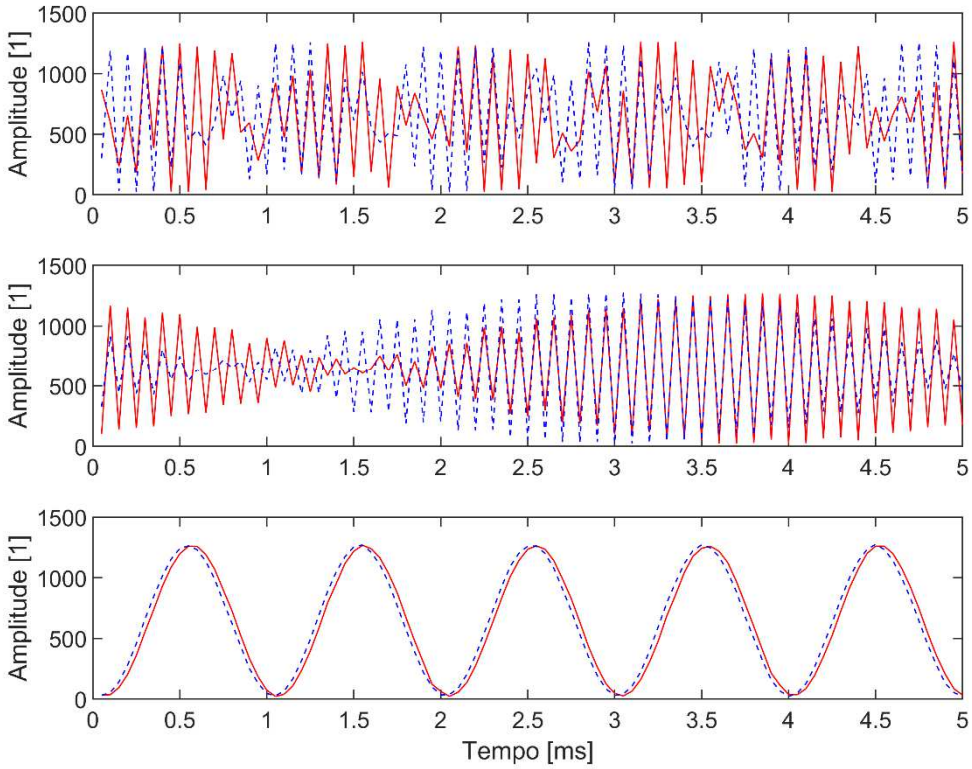


Figura 13 - Ondas com frequências de 50 kHz, 10 kHz e 1 kHz, respetivamente de cima para baixo

Um parâmetro importante a ter em consideração quando queremos fazer a aquisição dos sinais é o número de pontos que podemos guardar na memória disponível no microcontrolador. Quando se usa a opção *build* do projeto é possível ver na consola a quantidade de memória ocupada nas diferentes memórias disponíveis, RAM e *flash*. Foram experimentados vários tamanhos para os dois *arrays* e chegou-se à conclusão que o tamanho máximo que cada *array* pode ter é de 3552 posições. Ao se descobrir o tamanho máximo dos *arrays* notou-se também que o microcontrolador possui duas memórias RAM de 32 kbytes separadas e que apenas guardava os valores numa delas. A Figura 14 mostra as diferentes memórias do microcontrolador e as suas capacidades.

```
Memory region         Used Size  Region Size  %age Used
   PROGRAM_FLASH:      12956 B       256 KB      4.94%
       SRAM_UPPER:        32 KB        32 KB    100.00%
       SRAM_LOWER:         0 GB        32 KB      0.00%
         FLEX_RAM:         0 GB         4 KB      0.00%
Finished building target: MK50D10_Project_semADC.axf
```

Figura 14 - Diversas memórias e os seus respetivos tamanhos.

Como podemos ver na Figura 14, apenas a SRAM_UPPER é ocupada, chegando aos 100 % da capacidade quando os dois *arrays* que guardam os dados do ADC têm um tamanho de 3552 posições. Os dados guardados nos *arrays* são do tipo int, logo cada posição do *array* ocupa 32 bits. Como a resolução do ADC é apenas de 12 bits podemos mudar o tipo de dados de int para int16_t, passando de 32 bits para 16 bits e sendo assim possível guardar o dobro dos dados, podendo cada *array* ter agora 7104 posições. Não esquecer que se o tratamento dos dados tivesse sido realizado, não seria possível ter tantas posições, embora continue disponível uma RAM de 32 kbytes. Não foi utilizada a SRAM_LOWER por falta de tempo para tratar da localização em que cada variável era guardada.

Para descobrir a frequência de amostragem máxima foram feitas várias tentativas e encontrou-se o intervalo de tempo mínimo entre aquisições de pontos que corresponde a 2 µs; abaixo deste valor o ADC não recebe dados. Sendo a frequência o inverso do período, ficamos a saber que a frequência de amostragem máxima do ADC é 500 kHz, isto é, o ADC consegue receber 500 mil amostras por segundo.

O teorema de Nyquist diz nos que para fazer a aquisição de um sinal é necessário utilizar uma frequência de amostragem que seja no mínimo o dobro da frequência da onda em aquisição, isto é, para fazer a aquisição de um sinal é preciso no mínimo ter dois pontos por período da onda. Como o ADC tem uma frequência de amostragem máxima de 500 kHz, a frequência máxima que a onda pode ter para se conseguir adquirir é de 250 kHz. Logo, o ADC interno do microcontrolador apenas faz a aquisição de ondas DC até ondas com frequência de 250 kHz.

Como os elétrodos transmissores irão emitir entre 1 kHz e 10 kHz, a frequência de amostragem terá de ser de, pelo menos, 20 kHz; utilizando os 500 kHz de frequência de amostragem máxima do ADC, cada onda conterá pelo menos 25 pontos.

Contudo, será necessário fazer um estudo mais aprofundado sobre que frequência de amostragem a utilizar, porque a frequência máxima e o tamanho máximo do *array* podem não ser os ideais, e com diferentes frequências de amostragem podemos ter uma melhor definição do sinal, mas sempre com a limitação do tamanho máximo dos *arrays*. Pode ser necessário considerar mais pontos por período para ter uma melhor resolução da onda, mas para isso seria necessário aumentar o intervalo de aquisição, tudo dependendo da frequência da onda e da resolução pretendidas.

6. Discussão e conclusão

Com este trabalho, desenvolveu-se o módulo digital do TPIP, isto é, o PCB e o *software* que permite a sua validação. Este novo sensor permitirá obter melhores dados sobre fenómenos de ablação em escudos térmicos.

A profundidade do desenvolvimento do *software* e dos respetivos testes alcançada nesta dissertação foi em parte condicionada relativamente à ambição inicial deste trabalho, decorrente dos longos tempos de espera de envio de peças e componentes, da montagem do PCB e da resolução de problemas inesperados do trabalho experimental.

6.1. Conclusão

O desenvolvimento de melhores sensores ajudará na construção e desenvolvimento de escudos térmicos que protegem os veículos espaciais durante a entrada ou reentrada atmosférica. Neste trabalho, propôs-se continuar o desenvolvimento de um novo tipo de sensor que através de um modo de atuação diferente dos sensores já existentes, contribui para o avanço do estudo de processos de ablação em sistemas de proteção térmica de sondas de reentrada.

O passo inicial para desenvolver este novo tipo de sensor foi escolher um microcontrolador, tendo-se optado por VA41630 da Vorago Technologies. Como o desenvolvimento do novo tipo de sensor ainda está no início, e dado que os microcontroladores *rad hard* são dispendiosos, para começar a montagem e realizar os primeiros testes foi escolhido um microcontrolador não *rad hard*, mas com características semelhantes, o MK50DX256CLK10 da NXP Semiconductor. No entanto, foi necessário adicionar um ADC externo.

Com a realização dos primeiros testes de validação do PCB, para verificar se as ligações estavam bem feitas e se as tensões corretas chegavam aos pinos certos, foram descobertos problemas no condensador C28 que tinha ambos os pinos ligados ao *ground* e era suposto só ter um e o pino de alimentação de 3,3 V do ADC externo estava a fornecer uma tensão de 4,15 V o que provocava que todos os pinos que deviam receber 3,3 V recebessem os 4,15 V. Quanto aos problemas relacionados com o ADC externo, não foi possível resolver por não se encontrar a causa exata, pelo que se excluiu este ADC do circuito. Foi então necessário escrever um novo código.

Com o novo código, e sem o ADC externo, o funcionamento do microcontrolador, dos canais do ADC do microcontrolador e do *software* foi o esperado. Foi possível verificar

que havia a presença de ruído nos sinais adquiridos pelo ADC do microcontrolador. Isto deve-se ao facto de as ligações entre o gerador de sinais e o PCB e entre o PCB e o computador ainda não serem as ideais, visto que as conexões entre os fios e entre os fios e os conectores não são as melhores para comunicações.

Infelizmente, devido aos prazos de entrega de material, não foi possível continuar o trabalho, mas o que foi possível realizar até aqui é um bom ponto de partida para as etapas seguintes do projeto do desenvolvimento do TPIP.

Apesar de um dos requisitos ser a aquisição de quatro sinais através de um ADC externo e no fim só ter sido possível obter a aquisição de dois sinais através do ADC interno, podemos ver que a aquisição dos dois sinais foi efetuada com sucesso, o que é um bom ponto de partida, pois, não contando com o ADC externo, foi possível desenvolver um PCB funcional. O *software* de validação também será um bom ponto de partida para o desenvolvimento do módulo de *software*, o qual realiza a aquisição dos sinais a intervalos de tempo definidos, o que será um requisito para o *software* final.

6.2. Trabalho futuro

O trabalho futuro a desenvolver passará por corrigir os problemas do PCB e utilizar o software desenvolvido para validar o funcionamento com o ADC externo, de modo a ser possível receber os quatro sinais e completar o software para poder fazer o tratamento dos dados, nomeadamente o cálculo da FFT. De seguida, fazer os testes funcionais com o intuito de verificar a variação entre os sinais de transmissão e receção e a estabilidade do sinal. Também será necessário continuar os testes simulando os processos de ablação com o objetivo de se recriar um ambiente mais real. Por fim, analisar os dados obtidos e construir o modelo de voo do instrumento, englobando os elétrodos, unidade analógica, unidade digital, software de comunicação e de processamento de sinal.

Referências

[1] T. Oishi, E. R. Martinez, and J. A. Santos, “Development and Application of a TPS Ablation Sensor for Flight,” *Member AIAA 3 Mechanical Engineer*, pp. 229–233, 2008.

[2] Martinez E, Venkatapathy E, and Oishi T, “CURRENT DEVELOPMENTS IN FUTURE PLANETARY PROBE SENSORS FOR TPS”.

[3] E. Martinez and T. Oishi, “CURRENT DEVELOPMENTS IN SENSORS FOR THERMAL PROTECTION SYSTEMS.”

[4] F. S. Milos, Y. K. Chen, T. H. Squire, and R. A. Brewer, “Analysis of Galileo Probe heatshield ablation and temperature data,” *J Spacecr Rockets*, vol. 36, no. 3, pp. 298–306, 1999, doi: 10.2514/2.3465.

[5] D. Bose, J. A. Santos, E. Rodriguez, T. White, M. Olson, and M. Mahzari, “Mars Science Laboratory heat shield instrumentation and arc jet characterization,” 2013. doi: 10.2514/6.2013-2778.

[6] “HEAT Sensor for Thermal Protection Systems.” [Online]. Available: http://technology.nasa.gov/

[7] D. Bose, T. White, M. Mahzari, and K. Edquist, “Reconstruction of aerothermal environment and heat shield response of mars science laboratory,” in *Journal of Spacecraft and Rockets*, 2014, vol. 51, no. 4, pp. 1174–1184. doi: 10.2514/1.A32783.

[8] O. Uyanna and H. Najafi, “Thermal protection systems for space vehicles: A review on technology development, current challenges and future prospects,” *Acta Astronautica*, vol. 176. Elsevier Ltd, pp. 341–356, Nov. 01, 2020. doi: 10.1016/j.actaastro.2020.06.047.

[9] “What is a Printed Circuit Board (PCB)? - Printed Circuits LLC.” https://www.printedcircuits.com/what-is-a-pcb/ (acedido em 7 Setembro 2022).

[10] “Introduction to SPI Interface | Analog Devices.” https://www.analog.com/en/analog-dialogue/articles/introduction-to-spi-interface.html (acedido em 7 Setembro 2022).

[11] “What is JTAG and how can I make use of it? - XJTAG Tutorial.” https://www.xjtag.com/about-jtag/what-is-jtag/ (acedido em 7 Setembro 2022).

[12] “PCB Manufacturing & Assembly Capabilities - JLCPCB.” https://jlcpcb.com/capabilities/Capabilities (acedido em 7 Setembro 2022).

[13] “What is an IDE or Integrated Development Environment? | Veracode.” https://www.veracode.com/security/integrated-development-environment (acedido em 7 Setembro 2022).

[14] “How CMSIS Technology Works [And Why it Makes Our Lives Easier].” https://content.ccontrols.net/blog/cmsis (acedido em 7 Setembro 2022).

[15] “MCUXpresso IDE for NXP MCUs | Linux, Windows and iOS | NXP Semiconductors.” https://www.nxp.com/design/software/development-software/mcuxpresso-software-and-tools-/mcuxpresso-integrated-development-environment-ide:MCUXpresso-IDE (acedido em 7 Setembro 2022).

[16] “MCUXpresso SDK API Reference Manual: SPI CMSIS driver.” https://mcuxpresso.nxp.com/api_doc/dev/1589/group__spi__driver.html (acedido em 7 Setembro 2022).

[17] “SPI Interface.” https://www.keil.com/pack/doc/CMSIS/Driver/html/group__spi__interface__gr.html (acedido em 7 Setembro 2022).

[18] “Interrupts - GeeksforGeeks.” https://www.geeksforgeeks.org/interrupts/ (acedido em 7 Setembro 2022).

[19] “How to Test for Continuity with a Digital Multimeter | Fluke.” https://www.fluke.com/en-us/learn/blog/digital-multimeters/how-to-test-for-continuity (acedido em 7 Setembro 2022).

[20] “MCU Link JTAG/SWD Debug Probe | NXP Semiconductors.” https://www.nxp.com/design/microcontrollers-developer-resources/mcu-link-debug-probe:MCU-LINK (acedido em 7 Setembro 2022).

[21] “SWD - SEGGER Wiki.” https://wiki.segger.com/SWD (acedido em 7 Setembro 2022).

Anexos

Anexo A

A.1. Pinos ADC

HVSS – alimentação - ligado

CH_D1 – canal D1 – não utilizado

REFDN – referência canal D – não utilizado

AVDD – alimentação – ligado

AGND – alimentação – ligado

REFDP – referência canal D – não utilizado

CH_D0 – canal D0 – não utilizado

PAR/SER – escolher entre interface *parallel* ou *serial* – ligado a 5V

STBY – *standby mode* – ligado a +5V

RESET – *reset input* – ligado ao microcontrolador

REFEN/WR – *internal reference* – ligado a +5V

RD – ligado ao *ground* em interface *serial*

CS/FS – inicialização da transferência de dados – ligado ao microcontrolador

AVDD – alimentação - ligado

AGND – alimentação - ligado

DB15/SDO_D – dados do canal D – não utilizado

DB14/SDO_C – dados do canal C – não utilizado

DB13/SDO_B – dados do canal B – ligado ao microcontrolador (RX1 e RX2)

DB12/SDO_A – dados do canal A – ligado ao microcontrolador (TX1 e TX2)

DB11/REFBUFEN – *reference buffer* – ligado ao *ground*

DB10/SCLK – *clock* – não utilizado

DB9/SDI – ligado ao *ground*

DB8/DCEN – ativar *daisy chain* – ligado ao *ground*

DGND – alimentação - ligado

DVDD – alimentação - ligado

DB7 – ligado ao *ground* em interface *serial*

DB6/SEL_B – ativar canal B – ligado a +5V (SEL_B=1 e SEL_CD=1 -> dados do canal B em SDO_B)

DB5/SEL_CD – ativar canal C e D – ligado a +5V

DB4 – ligado ao *ground* em interface *serial*

DB3/DCIN_A – *daisy chain* canal A – ligado ao *ground*

DB2/DCIN_B – *daisy chain* canal B – ligado ao *ground*

DB1/DCIN_C – *daisy chain* canal C – ligado ao *ground*

DB0/DCIN_D – *daisy chain* canal D – ligado ao *ground*

RANGE/XCLK – *analog input voltage range* – ligado ao *ground*

BUSY/INT – ligar ao microcontrolador

ASLEEP – *Auto-sleep* – ligado ao *ground*

CONVST_A – inicialização da receção de dados do canal A – ligado ao microcontrolador

CONVST_B – inicialização da receção de dados do canal B – ligado ao microcontrolador

CONVST_C – inicialização da receção de dados do canal C – não utilizado

CONVST_D – inicialização da receção de dados do canal D – não utilizado

HW/SW – escolher entre *hardware* ou *software mode* (utilizar *hardware mode*) – ligado ao *ground*

CH_A0 – canal A0 – ligado TX2

REFAP – referência canal A – ligado

AGND – alimentação - ligado

AVDD – alimentação - ligado

REFAN – referência canal A – ligado

CH_A1 – canal A1 – ligado TX1

HVDD – alimentação - ligado

CH_B0 – canal B0 – ligado RX2

REFBP – referência canal B – ligado

AGND – alimentação - ligado

AVDD – alimentação - ligado

REFBN – referência canal B – ligado

CH_B1 – canal B1 – ligado RX1

REFN – tensão de referência – ligado

REFIO – referência negativa – ligado

AVDD – alimentação - ligado

AGND – alimentação - ligado

CH_C1 – canal C1 – não utilizado

REFCN – referência canal C – não utilizado

AVDD – alimentação - ligado

AGND – alimentação - ligado

REFCP – referência canal C – não utilizado

CH_C0 – canal C0 – não utilizado

A.2. Pinos microcontrolador

VDD – alimentação – ligado

VSS – alimentação – ligado

USB0_DP – porta usb – não utilizado

USB0_DM – porta usb – não utilizado

VOUT33 – não utilizado

VREGIN – usb *regulator* – não utilizado

ADC0_DP1/OP0_DP0 – ADC – não utilizado

ADC0_DM1/OP0_DM0 – ADC – RX

ADC1_DP1/OP1_DP0/OP1_DM1 – ADC – não utilizado

ADC1_DM1/OP1_DM0 – ADC – TX

PGA0_DP/ADC0_DP0/ADC1_DP3 – *programmable gain amplifier* – não utilizado

PGA0_DM/ADC0_DM0/ADC1_DM3 – *programmable gain amplifier* – não utilizado

PGA1_DP/ADC1_DP0/ADC0_DP3 – *programmable gain amplifier* – não utilizado

PGA1_DM/ADC1_DM0/ADC0_DM3 – *programmable gain amplifier* – não utilizado

VDDA – alimentação analógica – ligado

VREFH – tensão de referência do ADC – ligado

VREFL – tensão de referência do ADC – ligado

VSSA – alimentação analógica – ligado

ADC1_SE16/OP1_OUT/CMP2_IN2/ADC0_SE22/OP0_DP2/OP1_DP2 – ADC – não utilizado

ADC0_SE16/OP0_OUT/CMP1_IN2/ADC0_SE21/OP0_DP1/OP1_DP1 – ADC – não utilizado

VREF_OUT/CMP1_IN5/CMP0_IN5/ADC1_SE18 – tensão de referência interna do ADC – ligado ao *ground*

TRI0_OUT/OP1_DM2 – amplificador de transimpedância – não utilizado

TRI0_DM – não utilizado

TRI0_DP – não utilizado

TRI1_DM – não utilizado

TRI1_DP – não utilizado

TRI1_OUT/CMP2_IN5/ADC1_SE22 – não utilizado

DAC0_OUT/CMP1_IN3/ADC0_SE23/OP0_DP4/OP1_DP4 – DAC – não utilizado

DAC1_OUT/CMP0_IN4/CMP2_IN3/ADC1_SE23/OP0_DP5/OP1_DP5 – DAC – não utilizado

XTAL32 – não utilizado

EXTAL32 – não utilizado

VBAT – não utilizado

PTA0 – JTAG_TCLK

PTA1 – JTAG_TDI

PTA2 – JTAG_TDO

PTA3 – JTAG_TMS

PTA4/LLWU_P3 – não utilizado

VDD – alimentação – ligado

VSS – alimentação – ligado

PTA18 – não utilizado

PTA19 – não utilizado

RESET_b – *reset* – PTB1 – restaura todas as configurações para os valores padrão

PTB0/LLWU_P5 – não utilizado

PTB1 – GPIO – RESET_b

PTB2 – GPIO – CONVST_A

PTB3 – GPIO – CONVST_B

PTB10 – SPI1_PCS0 – ligado

PTB11 – SPI1_SCK – ligado

VSS – alimentação – ligado

VDD – alimentação – ligado

PTB16 – SPI1_SOUT – não utilizado

PTB17 – SPI1_SIN – ligado

PTB18 – GPIO – BUSY/INT

PTB19 – GPIO – RESET_ADC

PTC0 – não utilizado

PTC1/LLWU_P6 – não utilizado

PTC2 – não utilizado

PTC3/LLWU_P7 – não utilizado

VSS – alimentação – ligado

VDD – alimentação – ligado

PTC4/LLWU_P8 – SPI0_PCS0 – ligado

PTC5/LLWU_P9 – SPI0_SCK – ligado

PTC6/LLWU_P10 – SPI0_SOUT – não utilizado

PTC7 – SPI0_SIN – ligado

PTC8 – não utilizado

PTC9 – não utilizado

PTC10 – não utilizado

PTC11/LLWU_P11 – não utilizado

VSS – alimentação – ligado

VDD – alimentação – ligado

PTC16 – não utilizado

PTC17 – não utilizado

PTD0/LLWU_P12 – não utilizado

PTD1 – não utilizado

PTD2/LLWU_P13 – não utilizado

PTD3 – não utilizado

PTD4/LLWU_P14 – não utilizado

PTD5 – não utilizado

PTD6/LLWU_P15 – não utilizado

PTD7 – não utilizado

Apêndices

Apêndice A - Esquemáticos

A.1. Esquemático do circuito

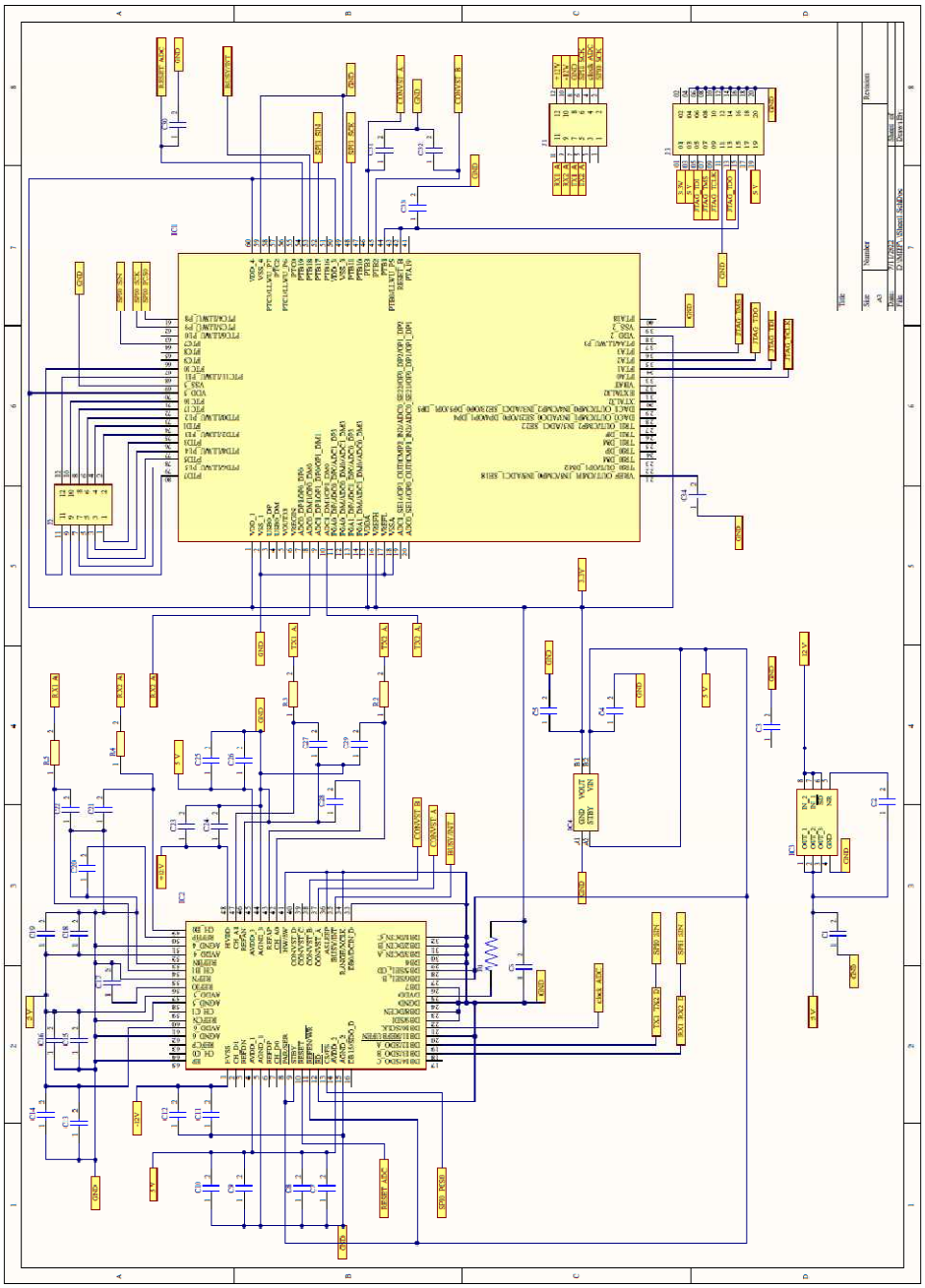


Figura 15 - Esquemático do circuito

A.2. Esquemático do PCB

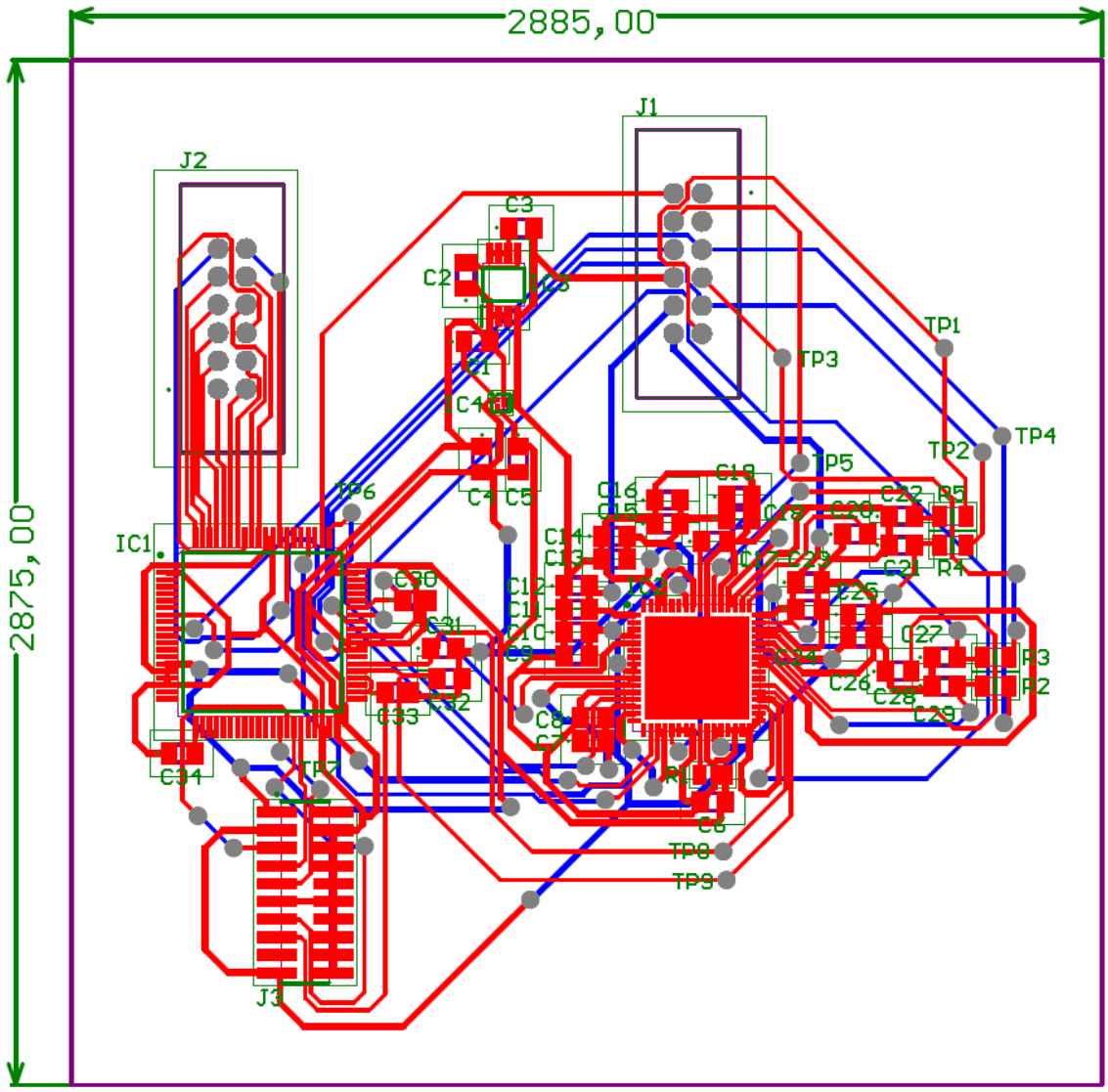


Figura 16 - Esquemático do PCB

A.3. Lista de componentes

Tabela 8 – Componentes utilizados no circuito.

Componente	**Referência**	**Marca**
Microcontrolador não rad hard (x1)	MK50DX256CLK10	NXP Semiconductors
ADC (x1)	ADS8528SRGCT	Texas Instruments
Regulador de tensão (5V -> 3.3V) (x1)	BU33SA5WGWZ-E2	ROHM Semiconductor
Regulador de tensão (12 V -> 5V) (x1)	ADP3335ARMZ-5-R7	Analog Devices
Conector macho (x1)	DF51A-12DP-2DSA	Hirose Electric
Conector fêmea (x1)	DF51-12DS-2C	Hirose Electric
Conector JTAG (x1)	FTSH-110-01-L-DV-K	Samtec
Resistência 100 kΩ (x4)	TNPU0805100KAZEN00	Vishay / Draloric
Resistência 10 Ω (x1)	RQ73C2A10RBTD	TE Connectivity / Holsworthy
Condensador 0,1 µF (x9)	C0805C104K8RACAUTO	KEMET
Condensador 0,47 µF (x1)	C0805X474K5RAC7210	KEMET
Condensador 820 pF (x4)	C0805C821F5GACTU	KEMET
Condensador 1 µF (x5)	C0805C105J4RACTM	KEMET
Condensador 10 µF (x10)	C0805X106J9RACTU	KEMET
Condensador 30 pF (x4)	C0805C300F5GACTU	KEMET
Condensadores 100 pF (x1)	C0805C101F1GACTU	KEMET

Apêndice B – Códigos de software

B.1. Código com o ADC externo

```
#include "board.h"
#include "peripherals.h"
#include "pin_mux.h"
#include "clock_config.h"
#include "MK50D10.h"
#include "fsl_gpio.h"
#include <time.h>
#include "fsl_dspi.h"
#include <math.h>
#include "fsl_pit.h"
#include "Driver_SPI.h"
#include "fsl_clock.h"
#include "fsl_common.h"
#include "fsl_port.h"
#include "fsl_dspi_cmsis.h"
#include "fsl_device_registers.h"


#define DRIVER_MASTER_SPI0 Driver_SPI0
#define DRIVER_MASTER_SPI1 Driver_SPI1
#define TRANSFER_SIZE 32U         /*! Transfer dataSize */
#define TRANSFER_BAUDRATE 500000U /*! Transfer baudrate - 500k*/
#define PIT_HANDLER PIT0_IRQHandler
#define PIT_IRQ_ID PIT0_IRQn
/* Get source clock for PIT driver */
#define PIT_SOURCE_CLOCK CLOCK_GetFreq(kCLOCK_BusClk)

/* Variables */
uint8_t masterRxData0[TRANSFER_SIZE] = {0U};
uint8_t masterRxData1[TRANSFER_SIZE] = {0U};
#define tamanho  60U //definir tamanho
uint8_t canal_A0[tamanho] = {0U};
uint8_t canal_A1[tamanho] = {0U};
uint8_t canal_B0[tamanho] = {0U};
uint8_t canal_B1[tamanho] = {0U};
uint8_t canal_A0_t[tamanho] = {0U};
uint8_t canal_A1_t[tamanho] = {0U};
uint8_t canal_B0_t[tamanho] = {0U};
uint8_t canal_B1_t[tamanho] = {0U};
int decimalA0 = 0;
int decimalB0 = 0;
```

```
int decimalA1 = 0;
int decimalB1 = 0;
int intervalo_tempo = 1; //tempo para o interrupt contar
uint8_t tempo[tamanho] = {0U};
volatile bool pitIsrFlag = false;
int busy = 0;
volatile bool isTransferCompleted0;
volatile bool isTransferCompleted1;


/* PIT0_IRQn interrupt handler */
void PIT_HANDLER(void) {

    /* Clear interrupt flag.*/
    PIT_ClearStatusFlags(PIT, kPIT_Chnl_0, kPIT_TimerFlag);
    pitIsrFlag = true;
}




void DSPI_MasterSignalEvent_t0(uint32_t event)
{
     /* user code */
     if(ARM_SPI_EVENT_TRANSFER_COMPLETE){
          PRINTF("Data incoming");

     }
    isTransferCompleted0 = true;
    PRINTF("This is MasterSignalEvent.\r\n");
}

void DSPI_MasterSignalEvent_t1(uint32_t event)
{
    /* user code */
    isTransferCompleted1 = true;
    PRINTF("This is MasterSignalEvent.\r\n");
}



int main(void) {

    /* Init board hardware. */
    BOARD_InitBootPins();
    BOARD_InitBootClocks();
    BOARD_InitBootPeripherals();
```

```
    //Define a digital input pin configuration,
    gpio_pin_config_t input_config =
    {
      kGPIO_DigitalInput,
      0,
    };

    //Define a digital output pin configuration,
    gpio_pin_config_t output_config =
    {
      kGPIO_DigitalOutput,
      0,
    };

    GPIO_PinInit(GPIOB, 2U, &output_config); //convst_b
    GPIO_PinInit(GPIOB, 3U, &output_config); //convst_a
    GPIO_PinInit(GPIOB, 1U, &output_config); //reset
    GPIO_PinInit(GPIOB, 18U, &input_config); //busy
    GPIO_PinInit(GPIOB, 19U, &output_config); //reset_adc
    GPIO_PinInit(GPIOC, 4U, &output_config); //fs



    /*SPI0 master init*/
    DRIVER_MASTER_SPI0.Initialize(DSPI_MasterSignalEvent_t0);
    DRIVER_MASTER_SPI0.PowerControl(ARM_POWER_FULL);
    DRIVER_MASTER_SPI0.Control(ARM_SPI_MODE_MASTER,
TRANSFER_BAUDRATE);
    /*SPI1 master init*/
    DRIVER_MASTER_SPI1.Initialize(DSPI_MasterSignalEvent_t1);
    DRIVER_MASTER_SPI1.PowerControl(ARM_POWER_FULL);
    DRIVER_MASTER_SPI1.Control(ARM_SPI_MODE_MASTER,
TRANSFER_BAUDRATE);

    /* Structure of initialize PIT */
    pit_config_t pitConfig;

     PIT_GetDefaultConfig(&pitConfig);

     /* Init pit module */
     PIT_Init(PIT0_IRQn, &pitConfig);

     /* Set timer period for channel 0 */
     PIT_SetTimerPeriod(PIT0_IRQn, kPIT_Chnl_0,
USEC_TO_COUNT(intervalo_tempo, PIT_SOURCE_CLOCK));
```

```
      /* Enable timer interrupts for channel 0 */
      PIT_EnableInterrupts(PIT0_IRQn, kPIT_Chnl_0,
kPIT_TimerInterruptEnable);

      /* Enable at the NVIC */
      EnableIRQ(PIT0_IRQn);


    for (int i = 0; i<tamanho; i++) {

        pitIsrFlag = false;

        /* Start channel 0 */
        PIT_StartTimer(PIT0_IRQn, kPIT_Chnl_0);

        GPIO_WritePinOutput(GPIOB, 3U, 1); //convst_a = 1
        GPIO_WritePinOutput(GPIOB, 2U, 1); //convst_b = 1


        while(busy==1){
           busy = GPIO_ReadPinInput(GPIOB, 18U);
        }

        GPIO_WritePinOutput(GPIOB, 3U, 0); //convst_a = 0
        GPIO_WritePinOutput(GPIOB, 2U, 0); //convst_b = 0

        GPIO_WritePinOutput(GPIOC, 4U, 0); //fs = 0

        /* Start master0 transfer */
        DRIVER_MASTER_SPI0.Receive(masterRxData0,
TRANSFER_SIZE);
        DRIVER_MASTER_SPI1.Receive(masterRxData1,
TRANSFER_SIZE);

        while (true) {
             if (DRIVER_MASTER_SPI0.GetDataCount == 32 &&
DRIVER_MASTER_SPI1.GetDataCount == 32){


          for (i=0;i<16;i++){
                      canal_A0_t[i]=masterRxData0[i];
                      canal_A1_t[i]=masterRxData0[i+16];
                      canal_B0_t[i]=masterRxData1[i];
                      canal_B1_t[i]=masterRxData1[i+16];
                }
                break;
            }
        }
```

```
        for (int j = 0; j<16;j++){
             //array invertido
             decimalA0+=canal_A0_t[15-j]*pow(2,15-j);
             decimalB0+=canal_B0_t[15-j]*pow(2,15-j);
             decimalA1+=canal_A1_t[15-j]*pow(2,15-j);
             decimalB1+=canal_B1_t[15-j]*pow(2,15-j);
        }


        canal_A0[i]=decimalA0;
        canal_B0[i]=decimalB0;
        canal_A1[i]=decimalA1;
        canal_B1[i]=decimalB1;

        tempo[i]=intervalo_tempo*i;

        while(pitIsrFlag == false){
           ;
        }

        if(i==tamanho){
           /* Disable timer interrupts for channel 0 */
           PIT_DisableInterrupts(PIT0_IRQn, kPIT_Chnl_0,
!kPIT_TimerInterruptEnable);
        }

    }



    /* Master0 power off */
    DRIVER_MASTER_SPI0.PowerControl(ARM_POWER_OFF);
    DRIVER_MASTER_SPI1.PowerControl(ARM_POWER_OFF);

    /* Master0 uninitialize */
    DRIVER_MASTER_SPI0.Uninitialize();
    DRIVER_MASTER_SPI1.Uninitialize();


}
```

B.2. Código sem o ADC externo

```
#include "fsl_debug_console.h"
#include "board.h"
#include "fsl_adc16.h"
#include "fsl_pit.h"
#include "pin_mux.h"
#include "clock_config.h"

/*******************************************************************
 * Definitions
*******************************************************************/
#define DEMO_ADC16_BASE_0 ADC0
#define DEMO_ADC16_CHANNEL_GROUP_0 0U
#define DEMO_ADC16_USER_CHANNEL_0 20U

#define DEMO_ADC16_BASE_1 ADC1
#define DEMO_ADC16_CHANNEL_GROUP_1 0U
#define DEMO_ADC16_USER_CHANNEL_1 20U

#define PIT_HANDLER PIT0_IRQHandler
#define PIT_IRQ_ID PIT0_IRQn
/* Get source clock for PIT driver */
#define PIT_SOURCE_CLOCK CLOCK_GetFreq(kCLOCK_BusClk)


/*******************************************************************
 * Variables
*******************************************************************/
volatile bool pitIsrFlag = false;
int i = 0U;
#define tamanho  100U
double intervalo_tempo = 2;
int16_t canal0[tamanho] = {0};
int16_t canal1[tamanho] = {0};


/*******************************************************************
 * Code
*******************************************************************/

void PIT_HANDLER(void)
{
    /* Clear interrupt flag.*/
    PIT_ClearStatusFlags(PIT, kPIT_Chnl_0, kPIT_TimerFlag);
    pitIsrFlag = true;
}
```

```
/*!
 * @brief Main function
 */
int main(void)
{

    BOARD_InitPins();
    BOARD_BootClockRUN();
    BOARD_InitDebugConsole();

    adc16_config_t adc16ConfigStruct_0;
    adc16_channel_config_t adc16ChannelConfigStruct_0;

    adc16_config_t adc16ConfigStruct_1;
    adc16_channel_config_t adc16ChannelConfigStruct_1;

    ADC16_GetDefaultConfig(&adc16ConfigStruct_0);
#ifdef BOARD_ADC_USE_ALT_VREF
    adc16ConfigStruct_0.referenceVoltageSource =
kADC16_ReferenceVoltageSourceValt;
#endif

    ADC16_GetDefaultConfig(&adc16ConfigStruct_1);
#ifdef BOARD_ADC_USE_ALT_VREF
    adc16ConfigStruct_1.referenceVoltageSource =
kADC16_ReferenceVoltageSourceValt;
#endif

    ADC16_Init(DEMO_ADC16_BASE_0, &adc16ConfigStruct_0);
    ADC16_EnableHardwareTrigger(DEMO_ADC16_BASE_0, false);

    ADC16_Init(DEMO_ADC16_BASE_1, &adc16ConfigStruct_1);
    ADC16_EnableHardwareTrigger(DEMO_ADC16_BASE_1, false);


    adc16ChannelConfigStruct_0.channelNumber =
DEMO_ADC16_USER_CHANNEL_0;
adc16ChannelConfigStruct_0.enableInterruptOnConversionCompleted
= false;
#if defined(FSL_FEATURE_ADC16_HAS_DIFF_MODE) &&
FSL_FEATURE_ADC16_HAS_DIFF_MODE
    adc16ChannelConfigStruct_0.enableDifferentialConversion =
false;
#endif
```

```
    adc16ChannelConfigStruct_1.channelNumber =
DEMO_ADC16_USER_CHANNEL_1;
adc16ChannelConfigStruct_1.enableInterruptOnConversionCompleted
= false;
#if defined(FSL_FEATURE_ADC16_HAS_DIFF_MODE) &&
FSL_FEATURE_ADC16_HAS_DIFF_MODE
    adc16ChannelConfigStruct_1.enableDifferentialConversion =
false;
#endif

    /* Structure of initialize PIT */
    pit_config_t pitConfig;

    /*
    * pitConfig.enableRunInDebug = false;
    */
     PIT_GetDefaultConfig(&pitConfig);

     /* Init pit module */
     PIT_Init(PIT, &pitConfig);

     /* Set timer period for channel 0 */
     PIT_SetTimerPeriod(PIT, kPIT_Chnl_0,
USEC_TO_COUNT(intervalo_tempo, PIT_SOURCE_CLOCK));

     /* Enable timer interrupts for channel 0 */
     PIT_EnableInterrupts(PIT, kPIT_Chnl_0,
kPIT_TimerInterruptEnable);

     /* Enable at the NVIC */
     EnableIRQ(PIT_IRQ_ID);

     /* Start channel 0 */
     PRINTF("\r\nStarting PIT");
     PIT_StartTimer(PIT, kPIT_Chnl_0);

     while (i<tamanho)
     {

          if (true == pitIsrFlag)
          {

               pitIsrFlag = false;

               ADC16_SetChannelConfig(DEMO_ADC16_BASE_0,
DEMO_ADC16_CHANNEL_GROUP_0, &adc16ChannelConfigStruct_0);
               ADC16_SetChannelConfig(DEMO_ADC16_BASE_1,
DEMO_ADC16_CHANNEL_GROUP_1, &adc16ChannelConfigStruct_1);
```

```
                canal0[i] =
ADC16_GetChannelConversionValue(DEMO_ADC16_BASE_0,
DEMO_ADC16_CHANNEL_GROUP_0);
                canal1[i] =
ADC16_GetChannelConversionValue(DEMO_ADC16_BASE_1,
DEMO_ADC16_CHANNEL_GROUP_1);
                i++;
            }
      }

      PRINTF("\ncanal0\n");
      for (int j=0;j<tamanho;j++){
            PRINTF("%d\n",canal0[j]);
      }
      PRINTF("\ncanal1\n");
      for (int j=0;j<tamanho;j++){
            PRINTF("%d\n",canal1[j]);
      }

      PRINTF("acabou");

}
```