



Universidade de Brasília – UnB
Faculdade UnB Gama – FGA
Engenharia Aeroespacial

On-Board Computer: Projeto de um Computador de Bordo para Pequenos Satélites

Autor: Guilherme Silva Lionço
Orientador: Prof. Dr. Giancarlo Santilli

Brasília, DF
2018



Guilherme Silva Lionço

On-Board Computer: Projeto de um Computador de Bordo para Pequenos Satélites

Monografia submetida ao curso de graduação em Engenharia Aeroespacial da Universidade de Brasília, como requisito parcial para obtenção do Título de Bacharel em Engenharia Aeroespacial.

Universidade de Brasília – UnB

Faculdade UnB Gama – FGA

Orientador: Prof. Dr. Giancarlo Santilli

Coorientador: Prof. Dr. Leonardo Aguayo

Brasília, DF

2018

Guilherme Silva Lionço

On-Board Computer: Projeto de um Computador de Bordo para Pequenos Satélites/ Guilherme Silva Lionço. – Brasília, DF, 2018-

87 p. : il. (algumas color.) ; 30 cm.

Orientador: Prof. Dr. Giancarlo Santilli

Trabalho de Conclusão de Curso – Universidade de Brasília – UnB
Faculdade UnB Gama – FGA , 2018.

1. Palavra-chave01. 2. Palavra-chave02. I. Prof. Dr. Giancarlo Santilli. II.
Universidade de Brasília. III. Faculdade UnB Gama. IV. On-Board Computer:
Projeto de um Computador de Bordo para Pequenos Satélites

Guilherme Silva Lionço

On-Board Computer: Projeto de um Computador de Bordo para Pequenos Satélites

Monografia submetida ao curso de graduação em Engenharia Aeroespacial da Universidade de Brasília, como requisito parcial para obtenção do Título de Bacharel em Engenharia Aeroespacial.

Trabalho aprovado. Brasília, DF, 02 de julho de 2018

Prof. Dr. Giancarlo Santilli
Orientador

Prof. Dr. Leonardo Aguayo
Coorientador

Prof. Dr. Cristian Vendittozzi
Convidado 1

Brasília, DF
2018

“Nunca guarde o conhecimento.

Guardado enaltece o ego, partilhado enobrece o ser.

Guardado morre inerte, partilhado vive ativo.

Guardado vale apenas àquilo que é, partilhado vale aquilo em que pode se tornar.

Transmitir o que se sabe é eternizar a existência.”

(Fernando Pinheiro)

Resumo

Pesquisadores da Universidade de Brasília (UnB) e do Instituto de Aviação da Polônia (ILOT) estão estudando a viabilidade de uma missão CubeSat 3U, para demonstrador tecnológico. Alguns estudos já estão sendo realizados, com o intuito de oferecer soluções para essa futura missão. O presente projeto de pesquisa visa a construção de um Computador de Bordo (OBC) para essa futura missão. Foi utilizado a metodologia de *Co-Design*, que permitiu o desenvolvimento do *hardware* e *software* simultaneamente. Durante a concepção do projeto teórico, foi escolhido o MSP432P111 como o microcontrolador e alguns dispositivos para o *hardware* do sistema. Já para o *software* embarcado foi definido como sistema operacional o FreeRTOS. Não foi possível terminar todas as atividades previstas no cronograma, pois houveram alguns imprevistos durante a realização do projeto. Os objetivos não atingidos durante o TCC1 serão finalizados durante o período de férias.

Palavras-chaves: OBC. CubeSat. Computador de Bordo. Nano Satélites.

Abstract

Researchers at the University of Brasília (UnB) and the Aviation Institute of Poland (ILOT) are studying the feasibility of a CubeSat 3U mission, as a technology demonstrator. Some studies are already being carried out, in order to offer solutions for this future mission. The present research project is aimed at the construction of an Onboard Computer (OBC) for this future mission. The co-Design methodology was used, which allowed the development of hardware and software simultaneously. During the design of the theoretical project, MSP432P111 was chosen as the microcontroller and some devices for the hardware of the system. For the embedded software, the FreeRTOS was defined as an operating system. It was not possible to finish all the activities foreseen in the schedule, as there were some unforeseen events during the project. The goals not reached during TCC1 will be finalized during the mid-year vacation.

Key-words: OBC. CubeSat. On-Board Computer. Nanosatellites.

Lista de ilustrações

Figura 1 – Cronograma do Trabalho de Pesquisa.	19
Figura 2 – Categoria de Pequenos Satélites e alguns exemplos.	21
Figura 3 – Lançamento de pequenos satélites entre 1995 e 2014.	22
Figura 4 – Algumas variações de CubeSat.	23
Figura 5 – Especificações Estruturais para um CubeSat 3U+.	24
Figura 6 – Arquitetura de um Satélite.	26
Figura 7 – Estrutura 3U Padrão NanoAvionics.	26
Figura 8 – Principais componentes de um EPS.	27
Figura 9 – Componentes AODCS da empresa Clyde Space.	27
Figura 10 – Componentes TT&C da empresa EnduroSat.	28
Figura 11 – Computador de bordo da empresa EnduroSat.	28
Figura 12 – Arquitetura do OBDH do FloripaSat.	30
Figura 13 – Arquitetura do OpenOBC.	31
Figura 14 – Placa Eletrônica - iOBC.	31
Figura 15 – Arquitetura do FM430.	32
Figura 16 – PCB Padrão PC104 8-bits.	34
Figura 17 – Classificação dos efeitos da Radiação Espacial.	35
Figura 18 – Performance dos processadores CORTEX-M.	42
Figura 19 – Arquitetura do microcontrolador MSP432P4111.	43
Figura 20 – Esquemático Eletrônico de uma aplicação usual do INA193A-EP.	47
Figura 21 – Esquemático Eletrônico de uma aplicação usual do MPU9250.	48
Figura 22 – Dimensões do módulo PC/104-Plus em polegadas e milímetros.	49
Figura 23 – Arquitetura de abstração de Camadas.	52
Figura 24 – Estados das Tasks no FreeRTOS	57
Figura 25 – Exemplo Timer UML.	60
Figura 26 – UML da Camada de Serviço do Sistema.	61
Figura 27 – Arquitetura prévia do OBC.	63
Figura 28 – Esquemático prévio do OBC.	64
Figura 29 – Esquemático do MPU9250.	87
Figura 30 – Esquemático do STWD100.	87

Lista de tabelas

Tabela 1 – Microcontroladores selecionados.	33
Tabela 2 – Efeitos Acumulativos causada pela radiação espacial.	36
Tabela 3 – Efeitos de Evento Único causados pela radiação espacial.	36
Tabela 4 – Requisitos do Hardware.	39
Tabela 5 – Microcontroladores selecionados.	41
Tabela 6 – Pontuação de cada microcontrolador.	41
Tabela 7 – Comparação dos tipos de Memória.	45
Tabela 8 – Especificação das memórias da Unidade de Armazenamento.	46
Tabela 9 – Informações técnicas do INA193A-EP.	47
Tabela 10 – Informações técnicas sobre sensor MPU9250.	47
Tabela 11 – Informações técnicas sobre sensor TMP422-EP.	48
Tabela 12 – Requisitos do <i>software</i> emabarcado.	51
Tabela 13 – Lista de todas as APIs disponíveis no pacote DriverLib.	55
Tabela 14 – APIs da categoria Task Control e Task Creation.	58
Tabela 15 – Perfil do Diagrama FunctionalC.	60
Tabela 16 – Análise de OBCs em Missões Passadas	75
Tabela 17 – Requisitos do OBC.	78
Tabela 18 – Modos de Operação e Consumo do MSP432P4111	79
Tabela 19 – EPS - SID 97 (0x61)	81
Tabela 20 – EPS Min/Max - SID 81 (0x51)	82
Tabela 22 – COM - SID 65 (0x41)	83
Tabela 23 – Payload - SID 113 (0x71)	83
Tabela 24 – ADCS - SID 17 (0x11)	84
Tabela 26 – EPS archive - temperatures - SID 98 (0x62)	85
Tabela 27 – My caption	85
Tabela 28 – EPS archive - voltages - SID 100 (0x64)	85
Tabela 29 – PAYLOAD - Image Sensor	86
Tabela 30 – Quantidade total de dados.	86

Sumário

1	INTRODUÇÃO	17
1.1	Contexo e Justificativa	17
1.2	Objetivos	17
1.3	Metodologia	18
2	REFERENCIAL BIBLIOGRÁFICO	21
2.1	Pequenos Satélites	21
2.2	Padrão CubeSat	23
2.3	Principais Subsistemas de um CubeSat	25
2.4	Visão Geral de um OBC	28
2.4.1	Computadores de Bordo Existentes	29
2.5	Padrão PC/104	33
2.6	Ambiente Espacial	34
2.6.1	Radiação Espacial	35
2.6.2	Categoria de Componentes	36
3	DESENVOLVIMENTO DO HARDWARE	39
3.1	Requisitos e Funcionalidades	39
3.2	Análise do Microcontrolador	39
3.3	Microcontrolador MSP432P4111	42
3.3.1	Unidade de Memória	43
3.3.2	Portas e Periféricos	44
3.3.3	Consumo e Modos de Operação	44
3.4	Unidade de Armazenamento	44
3.5	Periféricos	46
3.5.1	Sensor de Corrente	46
3.5.2	Sensor Inercial	47
3.5.3	Sensor de Temperatura	48
3.6	Interfaces	48
3.7	Dimensões da Placa	49
3.8	Sistema para Mitigar Falhas	50
3.8.1	Watchdog Externo	50
4	DESENVOLVIMENTO DO SOFTWARE	51
4.1	Requisitos e Funcionalidades	51
4.2	Arquitetura do Software	52

4.3	DriverLib	54
4.4	FreeRTOS	56
4.4.1	APIs do FreeRTOS	56
4.5	Desenvolvimento da Camada de Serviços do Sistema	59
5	RESULTADOS PRELIMINARES	63
6	TRABALHOS FUTUROS	65
7	REFERÊNCIAS	67
	APÊNDICES	73
	APÊNDICE A – ANÁLISE DAS MISSÕES ANTERIORES	75
	APÊNDICE B – REQUISITOS DO OBC	77
	APÊNDICE C – MODOS DE OPERAÇÃO	79
	APÊNDICE D – ESTIMATIVA DE ARMAZENAMENTO DE DADOS	81
	APÊNDICE E – ESQUEMÁTICO ELETRÔNICO	87

1 INTRODUÇÃO

1.1 Contexo e Justificativa

O Instituto de Aviação Varsóvia (ILOT, do polonês *Instytut Lotnictwa*) e a Universidade de Brasília (UNB) decidiram firmar uma cooperação tecnologia, em novembro de 2014. Entre várias oportunidades, essa cooperação possibilita estágios para alunos de graduação em Engenharia Aeroespacial no ILOT. Uma nova proposta está sendo discutida entre os pesquisadores das duas instituições, visando a criação de uma missão 3U CubeSat.

Essa missão será destinada à validação de conceitos e teste de componentes, enquadrando-se como demonstrador tecnológico. Os objetivos iniciais, levantados pelos pesquisadores, são: uso de câmeras para monitoramento das calotas polares, com o intuito de estudar os efeitos da poluição; uso de um *Pulsed Plasma Thruster* (PPT) para estudo de controle orbital; uso de um acelerômetro para mapeamento do campo gravitacional terrestre.

Um projeto desse porte abrirá várias oportunidades de pesquisa na área de subsistemas satelitais, sensoriamento remoto, controle orbital, entre outros. Alguns estudos já estão sendo realizados, com o intuito de oferecer soluções para essa futura missão. O presente trabalho faz parte dessa série de pesquisas e tem o intuito de realizar a construção de um computador de bordo (OBC, do inglês *On-Board Computer*) para CubeSats.

1.2 Objetivos

Este trabalho tem como objetivo principal o desenvolvimento de um OBC para o controle e gerenciamento de um CubeSat. Os objetivos específicos são:

- Controle de uma Câmera CMOS (do inglês *Complementary Metal-Oxide-Semiconductor*);
- Controle de um PPT;
- Garantir uma subsistema com um alto nível de confiança, mesmo não utilizando dispositivos resistentes à radiação;
- Divulgação da pesquisa na plataforma GitHub.

1.3 Metodologia

Inicialmente será realizada uma pesquisa bibliográfica, para reunir informações que darão base ao projeto. Nessa fase será analisado: ambiente espacial; pequenos satélite, mais especificamente os CubeSats; OBCs (*hardware* e *software*) de missões passadas e disponíveis à venda.

Após a pesquisa bibliográfica, será realizado o levantamento dos requisitos da missão com o intuito de definir as funcionalidades e delimitar o escopo da pesquisa.

Logo em seguida, será feito o projeto preliminar do sistema. Para a parte de *hardware*, será escolhido: o microcontrolador, sensores, memória; entre outros. Já para o *software*, será escolhido: a arquitetura, linguagem de programação, sistema operacional; etc. Vale ressaltar que optou-se por uma concepção conjunta (do inglês *Co-Design*) do *hardware* e *software*, em que o desenvolvimento de ambos acontece simultaneamente.

Em paralelo ao estudo preliminar, serão realizados testes em *protoboard*, visando a validação de alguns subsistemas separadamente. Esses testes são de extrema importância, pois, aumentam as chances do sistema funcionar nas fases de prototipagem final. Nessa fase, os códigos para cada subsistema serão testados e, em seguida, unidos em um único código.

Por fim, após a validação do sistema em *protoboard*, será feito o projeto final do sistema. Para a parte de *hardware*, serão entregues os esquemáticos, *layout* da PCB (do inglês *Printed Circuit Board*), BOM (do inglês *Bill of Materials*) e simulações de circuitos analógicos. Para a parte de *software*, será entregue a arquitetura do sistema, diagrama UML (do inglês *Unified Modeling Language*) e fluxograma das rotinas.

O TCC2 visará a criação da placa de circuito impresso, prototipagem, teste da placa, alterações finais e, por fim, a entrega do produto final. A Figura 30 mostra o fluxograma para o projeto do OBC.

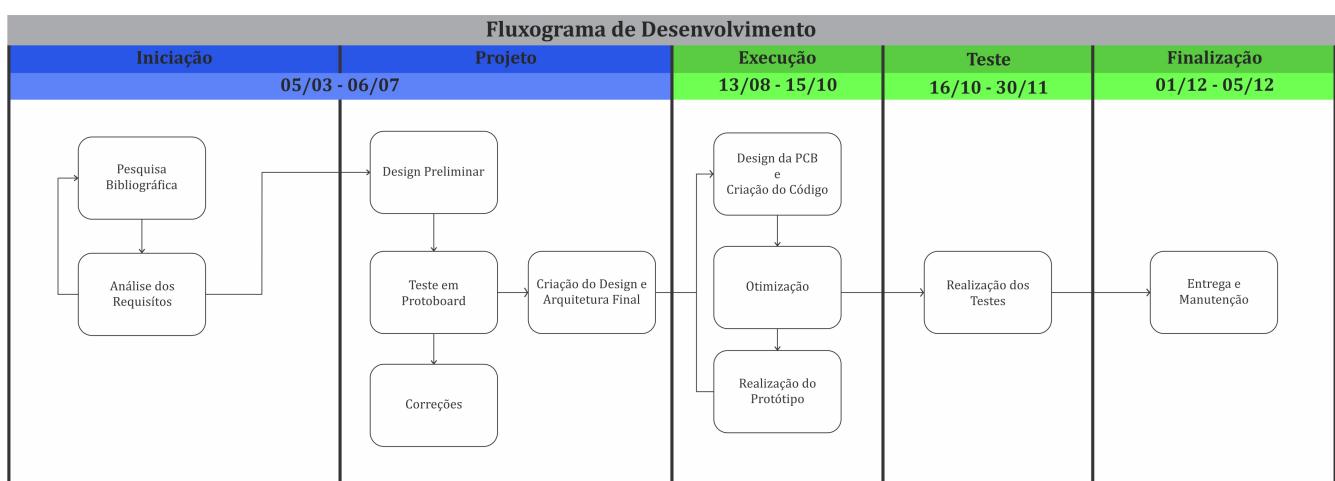


Figura 1 – Cronograma do Trabalho de Pesquisa.

2 REFERENCIAL BIBLIOGRÁFICO

Esta seção visa oferecer uma introdução aos conceitos básicos para o entendimento deste trabalho de pesquisa. Inicialmente será mostrado a categoria de pequenos satélites e a concepção do padrão CubeSat. Logo em seguida, será exposto os principais subsistemas de um CubeSat, dando ênfase no computador de bordo. Será mostrado a importância do barramento PC104 e as interferências do ambiente espacial em sistemas embarcados.

2.1 Pequenos Satélites

O termo “pequenos satélites” é designado para caracterizar satélites que possuem massa úmida (massa do satélite mais massa do propelente) inferior a 500 kg. Esta definição cumpre a terminologia estabelecida pelo *Small Space Technology Program* (SSTP) da NASA (FROST; AGASID, 2015). As subcategorias de pequenos satélites, e alguns exemplos de espaçonaves já lançadas, podem ser observadas na figura abaixo.

Fonte:(BARNHART, 2008, pág.42).

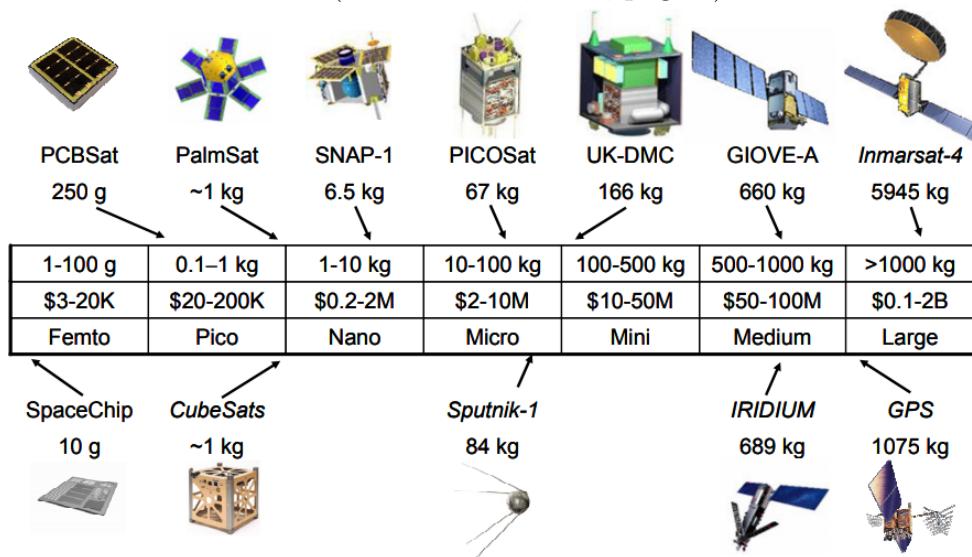


Figura 2 – Categoria de Pequenos Satélites e alguns exemplos.

As espaçonaves são normalmente agrupadas conforme sua massa. Como visto na figura acima, pequenos satélites podem ser divididos nas seguintes subcategorias: minissatélites com massa entre 100-500kg, microsatélites com massa entre 10-100kg, nanosatélites com massa entre 1-10kg, picosatélites com massa entre 0.1-1kg e femtosatélites com massa entre 1-100g. No limite superior dessa categoria, existem minisatélites como o Globalstar, satélite de telecomunicações da empresa Globalstar LP, que possui massa úmida de 450

kg (GLOBALSTAR, L.P, 2000). Já no extremo inferior, existem projetos como o KickSat que colocou em órbita 128 femtosatélites com 5g (GRIFFITHS, 2017).

Nas últimas décadas, houve um aumento no lançamento de nanosatélites. Esse crescimento pode ser observado na Figura 3, que mostra a quantidade de pequenos satélites colocados em órbita entre os anos de 1995 e 2014. Fatores como a miniaturização da eletrônica, otimização dos veículos lançadores e crescimento de empresas no setor ajudaram nesse crescimento (FACCHINETTI et al., 2016).

Fonte:(WEKERLE; FILHO, 2017, pág.3).

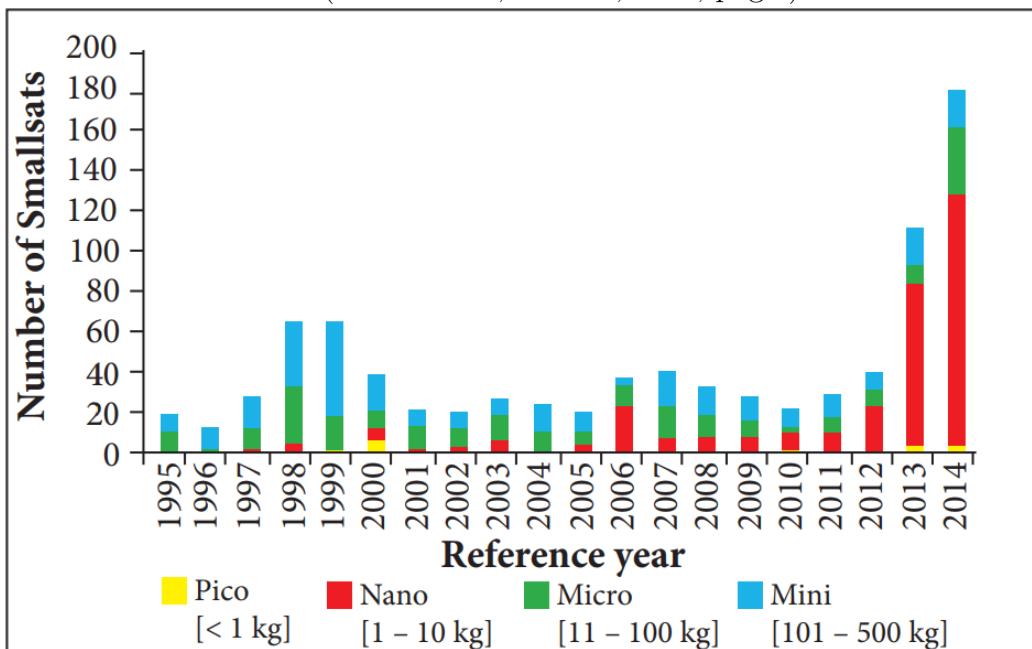


Figura 3 – Lançamento de pequenos satélites entre 1995 e 2014.

Fazendo uma análise minuciosa nos sites Nanosatellite Database¹ e Space Launch Report², sites que armazenam dados de missões satelitais, observa-se que em 2007 cerca de 2,6% (9 de 348 satélites) eram CubeSats. Já em 2017 o numero de lançamentos passou para 57,6% (295 de 512 satélites). Tendo como base esses valores, pode-se afirmar que os CubeSats ajudaram na popularização da categoria de nanosatélites.

De forma resumida, o CubeSat é um satélite cúbico de 10cm de aresta e que pesa apenas alguns quilogramas, sendo muito utilizado no meio acadêmico. Eles podem ser compostos por um único cubo (uma unidade ou 1U) ou vários cubos combinados formando, por exemplo, unidades 3U e 6U. A Figura 4 mostra as configurações mais usuais para uma CubeSat (FROST; AGASID, 2015).

¹ KULU, Erik. Nanosatellite Database by Erik. 2018. Disponível em: <<http://www.nanosats.eu/>>. Acesso em: 14 maio 2018.

² N2YO. Browse Satellites by Launch date. 2018. Disponível em:<<https://www.n2yo.com/browse/>>. Acesso em: 14 maio 2018.

Fonte:(MABROUK, 2017, pág.1).

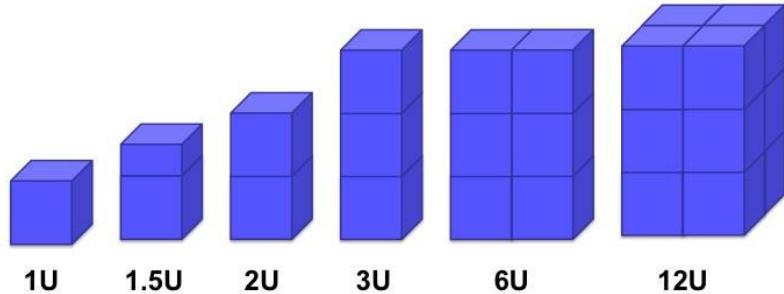


Figura 4 – Algumas variações de CubeSat.

2.2 Padrão CubeSat

O padrão CubeSat começou como um esforço colaborativo entre Jordi Puig-Suari, professor na Universidade Politécnica da Califórnia (CalPoly), e Bob Twiggs, professor na Universidade de Stanford. A intenção original era fornecer para a comunidade científica universitária meios mais acessíveis ao espaço (DEEPAK; TWIGGS, 2012).

Twiggs retrata o contexto histórico do desenvolvimento do conceito CubeSat em detalhes no seu artigo “*Origin of CubeSat*” (TWIGGS, 2008). Em resumo, o abandono radical do design tradicional de satélites, ocasionado pelo CubeSat, começou após o lançamento bem-sucedido do *Orbiting Picosatellite Automatic Launcher* (OPAL), projeto liderado por Twiggs, que pôs em órbita seis *hockey* picosatélites. Inspirado por esse sucesso, Twiggs explorou um *design* maior e mais cúbico para suportar mais capacidade. Ele encontrou o modelo perfeito para seu novo *design* em uma loja varejo, uma embalagem cúbica de 10 cm para ursinhos de pelúcia. O nanosatélites resultante, foi um cubo medindo 10 cm de aresta e pesando apenas 1 kg, nomeado CubeSat (DEEPAK; TWIGGS, 2012).

Mais tarde naquele ano, Twiggs e Puig-Suari disponibilizaram as especificações para o Padrão CubeSat. Em 2003, Puig-Suari e CalPoly desenvolveram o *Poly-PicoSatellite Orbital Deployer* (P-POD) para lançar até três CubeSats 1U. As primeiras missões CubeSat começaram no mesmo ano do desenvolvimento do P-POD (DEEPAK; TWIGGS, 2012).

Com o passar dos anos, a documentação do *CubeSat Standard* sofreu algumas alterações. A versão mais atualizada pode ser encontrada em alguns sites, como o Cubesat.org³. A Figura 5 mostra as dimensões padronizadas de um CubeSat 3U+ (CUBESAT PROGRAM, 2014). Um ponto a ser mencionado é que o 3U+ foi criado para atender as

³ CubeSat INFO. 2018. Disponível em:<<https://goo.gl/P8vZYy>>. Acesso em: 14 maio 2018.

demandas de sensoriamento remoto, possibilitando colocar uma câmera sem prejudicar o espaço interno da plataforma.

Fonte:(CUBESAT PROGRAM, 2014, pág.31).

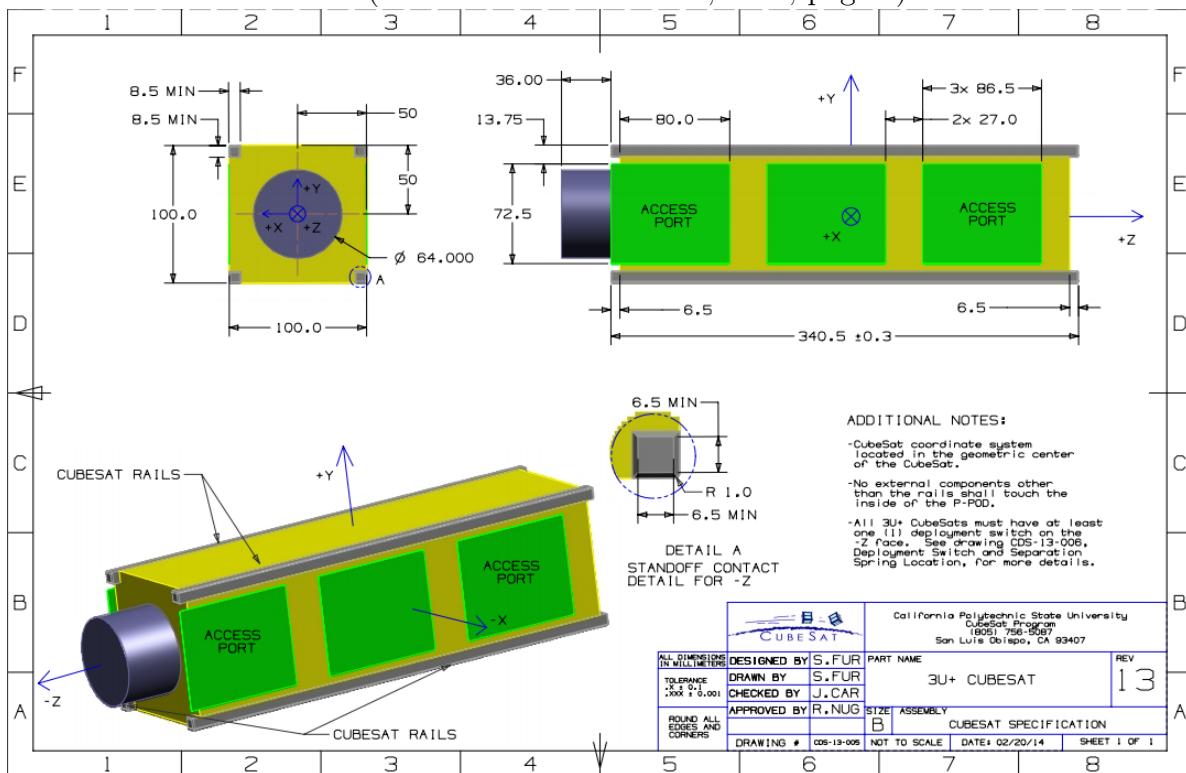


Figura 5 – Especificações Estruturais para um CubeSat 3U+.

Graças aos CubeSats, muitas universidades possuem seu próprio programa espacial. Porém o uso dessa tecnologia não é exclusiva apenas para grandes universidades; escolas secundárias, ensino médio e escolas primárias puderam desenvolver suas próprias missões (DEEPAK; TWIGGS, 2012). Um exemplo de projeto é o UbatubaSat, CubeSat feito pelos estudantes da escola municipal Tancredo de Almeida Neves, em Ubatuba-SP (FIORAVANTI, 2011).

Essa facilidade de acesso ao espaço é consequência de algumas características que o padrão CubeSat dispõem. De acordo com a empresa ISIS (2018), esses aspectos podem ser sintetizados em sete atributos:

- **Modularidade:** Por possuir um tamanho normatizado, tornou-se possível desenvolver módulos funcionais padronizados, como subsistemas de energia, comunicação, entre outros. Esse aspecto fez com que soluções de prateleira (COTS, do inglês *Commercial Off-The-Shelf*) fossem amplamente utilizadas no projeto de algumas missões, agilizando e barateando o desenvolvimento do CubeSat;
- **Lançamento:** Os CubeSats são tipicamente lançados no espaço como cargas contidas em contêineres padronizados, por exemplo, P-POD, visando a redução da

complexidade e custo do lançamento;

- **Custo:** Geralmente as missões CubeSat são desenvolvidas usando um "orçamento baixo", comparado às missões tradicionais;
- **Componentes:** Componentes que não possuem o selo “*Space Qualified*” são frequentemente utilizados e aceitos em missões CubeSat, permitindo uma abordagem de baixo custo e de rápida implementação. O fato de muitas missões CubeSat estarem em órbita baixa e possuírem um tempo de vida curto faz com que o uso desses componentes se torne viável;
- **Desenvolvimento:** Devido à baixa complexidade e escopo limitado, é comum usar metodologias de projeto menos formais. Em algumas circunstâncias, é possível trabalhar em equipes compactas, eliminando a necessidade de pacotes de documentação e outras despesas gerais. Obviamente, um nível mínimo de formalidade e documentação é sempre necessário;
- **Risco:** Tipicamente, a abordagem utilizada em missões CubeSats correm um risco técnico mais elevado oferecendo em troca: custo menor, implementação mais rápida, aplicação mais inovadora, ou um conjunto desses elementos;
- **Nichos de Aplicação:** O benefício do padrão CubeSats não é possuir boa performance em termo de largura de banda ou resolução espacial, pois não é compatível com as restrições estruturais impostas pelo padrão. Entretanto, quando usado em redes ou constelações, os CubeSats são capazes de fornecer uma resolução temporal a preços acessíveis.

2.3 Principais Subsistemas de um CubeSat

Os CubeSats são formados por duas partes principais, que são: a Carga Útil (do inglês *Payload*) e a plataforma (chamada em inglês de *Bus*). A Carga Útil é definida de acordo com a missão do satélite, por exemplo: satélites para observação da Terra possuem como carga útil uma câmera; satélites para telecomunicações possuem um rádio; etc. A Plataforma tem o objetivo de abrigar a Carga Útil para que ela funcione normalmente no ambiente espacial. A Plataforma é geralmente composta por: Sistema Estrutural (SS, do inglês *Structural System*), Sistema de Potência Elétrica (EPS, do inglês *Electrical Power System*), Sistema de Controle e Determinação de Órbita & Atitude (AODCS, do inglês *Attitude and Orbit Determination and Control System*), Controle de Telecomando e Telemetria (TT&C, do inglês *Telemetry Telecomand Control*) e o Computador de Bordo (OBC, do inglês *On-Board Computer*).

A Figura 6 visa ilustrar esses subsistemas (ADDAIM; KHERRAS; ZANTOU, 2010).

Fonte:(ADDAIM; KHERRAS ; ZANTOU, 2010, adaptado pág.6).

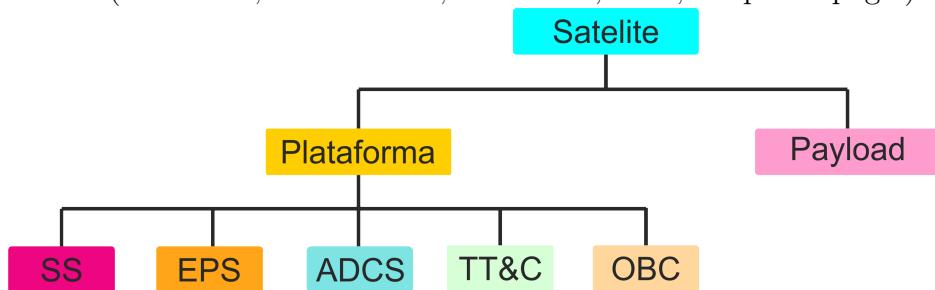


Figura 6 – Arquitetura de um Satélite.

Abaixo há uma breve descrição dos subsistemas e alguns exemplos de componentes comerciais.

Descrição dos Subsistemas:

- **SS:** Esse subsistema visa proteger o satélite dos efeitos do ambiente espacial e servir como interface com o veículo lançador. O SS deve ser leve e resistente, sem contar que deve blindar ao máximo a radiação espacial. Geralmente o material utilizado na estrutura é alguma liga de alumínio. A Figura 7 mostra o *Structure Subsystem* da NanoAvionics⁴.

Fonte:(NANO AVIONICS, 2018, pág.1).



Figura 7 – Estrutura 3U Padrão NanoAvionics.

- **EPS:** O *Electrical Power System* tem o objetivo de fornecer energia elétrica à todos os subsistemas, inclusive a carga útil. Ele é responsável pela geração, armazenamento e distribuição da energia elétrica. Os principais elementos de um EPS são: baterias, painéis solares e switchs. A Figura 8 é mostra os principais elementos de um EPS.

⁴ Empresa NanoAvionics. 2018. Disponível em:<<https://n-avionics.com/>>. Acesso em: 30 maio 2018.

Fonte:(CUBESTAR, 2018, pág.1).



Figura 8 – Principais componentes de um EPS.

- **AODCS:** Esse subsistema tem o objetivo de garantir que câmeras e antenas estejam apontados para os alvos predeterminados. Os principais componentes do AODCS são: rodas de reação, sensores iniciais, fotodiodos e magnetorques. Recentemente estão sendo utilizados módulos PPT (*Pulsed Plasma Thruster*) para compensar o arrasto atmosférico. A Figura 9 é mostra o AODCS (esquerda) e PPT (direita) da empresa Clyde Space⁵.

Fonte: (CLYDE SPACE, 2018, pág.1).

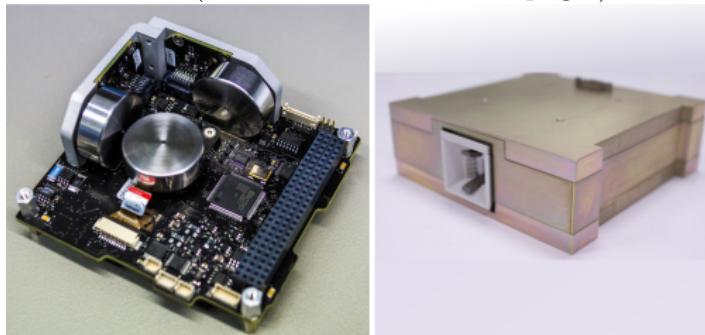


Figura 9 – Componentes AODCS da empresa Clyde Space.

- **TT&C:** Esse subsistema serve como interface entre a estação de solo e o Satélite. Ele envia pacotes de telemetria e dados da *Payload*, e recebe telecomandos da estação terrestre. Os componentes principais desse subsistema são: transceptor e antena. A Figura 10 mostra uma antena (esquerda) e um módulo de telecomunicações (direita) da empresa Endurosat⁶.

⁵ Empresa Clyde Space. 2018. Disponível em:<<https://www.clyde.space/>>. Acesso em: 30 maio 2018.

⁶ Empresa EnduroSat. 2018. Disponível em:<<https://www.endurosat.com/>>. Acesso em: 30 maio 2018.



Figura 10 – Componentes TT&C da empresa EnduroSat.

- **OBC:** O satélite precisa de um subsistema capaz de gerenciar todos os outros subsistemas. O OBC tem o intuito de garantir o funcionamento de todos os subsistemas de acordo com os telecomandos, provindos da estação de solo, e do software embarcado. Os principais componentes são: microcontrolador, sistema operacional e módulo de armazenamento. A Figura 11 mostra o OBC da empresa EnduroSat.

Fonte:(ENDUROSAT, 2018b, pág.1).

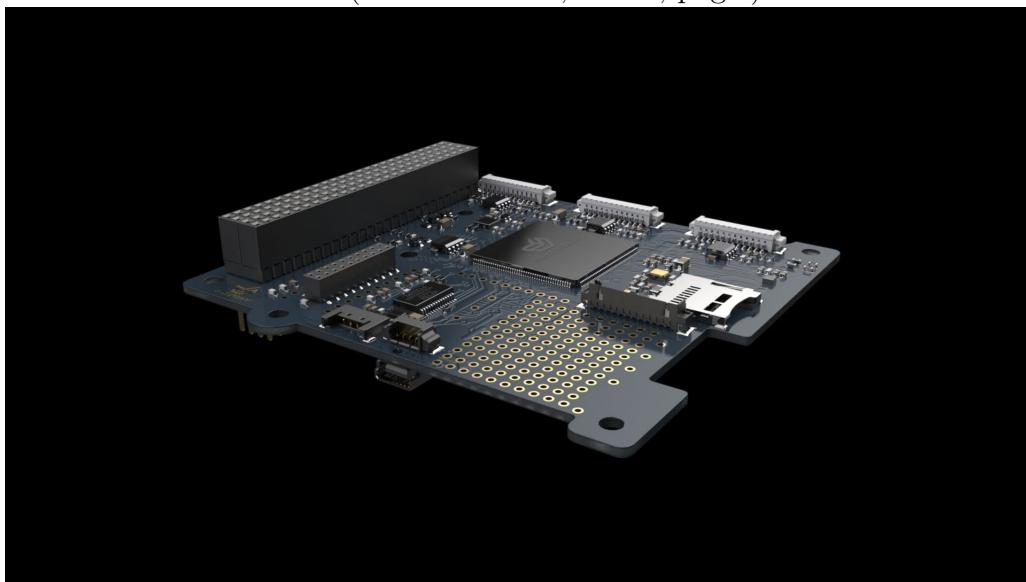


Figura 11 – Computador de bordo da empresa EnduroSat.

2.4 Visão Geral de um OBC

O Computador de Bordo é considerado o cérebro do CubeSat e consiste, essencialmente, de um microcontrolador conectado à outros subsistemas por barramentos e periféricos de dados. Normalmente, um Sistema Operacional de Tempo Real (RTOS, do inglês *Real-Time Operating System*) controla todas as aplicações que são executadas no microcontrolador (LUMBWE, 2013).

As funções mais importantes de um OBC são exemplificadas abaixo (STRAS et al., 2003):

- Capturar e armazenar dados da telemetria e *Payload*;
- Codificar e transmitir dados à estação de solo;
- Processar os telecomandos provenientes da estação de solo, compensando o tempo atraso no canal de uplink;
- Monitorar os subsistemas do Satélite.

2.4.1 Computadores de Bordo Existentes

Essa avaliação tem como objetivo mostrar ao leitor alguns OBCs, utilizados em missões CubeSat, e destacar as características mais relevantes. Uma coleta mais detalhada foi realizada e se encontra no Apêndice [A](#).

• **FloripaSat OBDH**

O projeto FloripaSat, da Universidade Federal de Santa Catarina (UFSC), possui um computador de bordo que é responsável pela sincronização de ações e fluxo de dados entre os subsistemas (*Payload* e EPS) e com a Segmento Terrestre. Esse subsistema é composto por seis submodulos: CPU (MCU: CPU + RAM + *Program Flash*), memória não volátil, *Drivers* de controle, unidade de medição inercial (IMU), Sensores de Corrente (sensor de Sol) e interfaces de comunicação (FLORIPASAT,2016).

O sistema responsável pela execução do *firmware* consiste de SoC que contém uma CPU, Memória RAM e Flash (usado para armazenamento de programas e status dos registradores). O Microcontrolador escolhido foi o MSP430F6659IPZ da Texas Instruments. Tal microcontrolador, conhecido por ter um baixo consumo, possui sete modos de operação de consumo, 4 timers de 16-bits, 12 AD/DA de 12-bits, 6 interfaces de comunicação serial, bloco de real-time clock (RTC) e mais de 74 pinos de I/O. O clock de operação do OBDH é de 32MHz (FLORIPASAT,2016).

Provendo um sistema de redundância, há um monitor de tensão com um *Watch-dog Timer* (FLORIPASAT,2016). A Figura [12](#) mostra a arquitetura do FloripaSat OBDH.

Fonte: (FLORIPASAT,2016, pág.45).

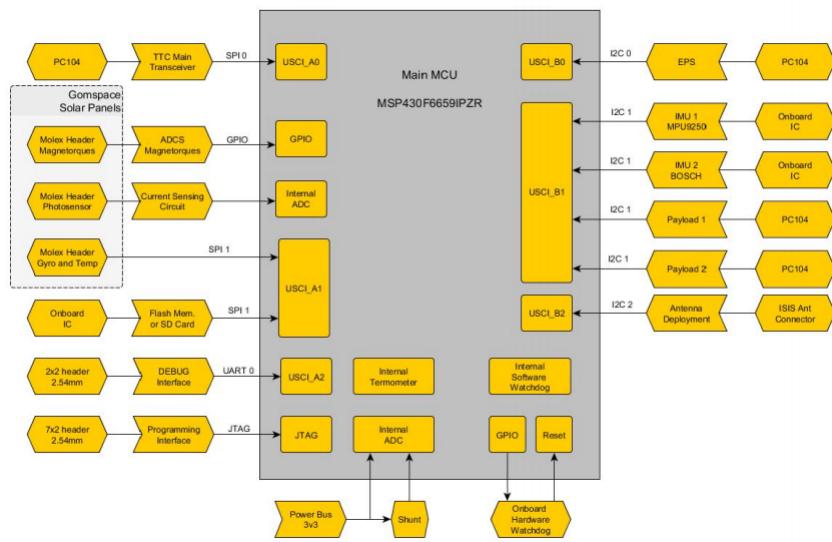


Figura 12 – Arquitetura do OBDH do FloripaSat.

• Open OBC

Esse é um computador open source para CubeSat, desenvolvido pela Universidade Federal do Ceará (UFC) com o do Instituto Nacional de Pesquisas Espaciais (INPE), tendo como pontos fortes o baixo custo e a alta confiabilidade. A arquitetura utilizada no Open OBC contém: processador TMS570LS0432 da fabricante Texas Instruments.

A arquitetura proposta utiliza o processador TMS570LS0432 do fabricante Texas Instruments, o qual possui: núcleo ARM Cortex-R4 em duas CPUs; detecção e correção de falhas em suas memórias RAM e ROM internas; *hardware BIST* (Auto-teste interno de fábrica) tanto na CPU quanto na memória RAM; e outras características de segurança como o monitoramento do clock e da tensão de alimentação. Uma memória *Flash* externa foi utilizada para armazenamento de código e dados. Foram disponibilizadas duas interfaces I2C para a comunicação com os subsistemas existentes em um CubeSat, sendo uma exclusiva para comunicação com o Transponder e outra comum para os demais. A arquitetura é complementada por uma interface UART para diagnóstico e depuração, sinais PWM para acionamento das bobinas de torque e entradas ADC para medição da intensidade da luz solar nas faces do satélite. Estão previstos ainda um cartão MicroSD para armazenamentos de dados e uma interface CAN para tráfego de informações transmitidas em tempo real, garantindo assim um controle rígido de erros e a recepção de mensagens (OPENOBC, 2017).

Fonte: (OPENOBC, 2017, pág.28).

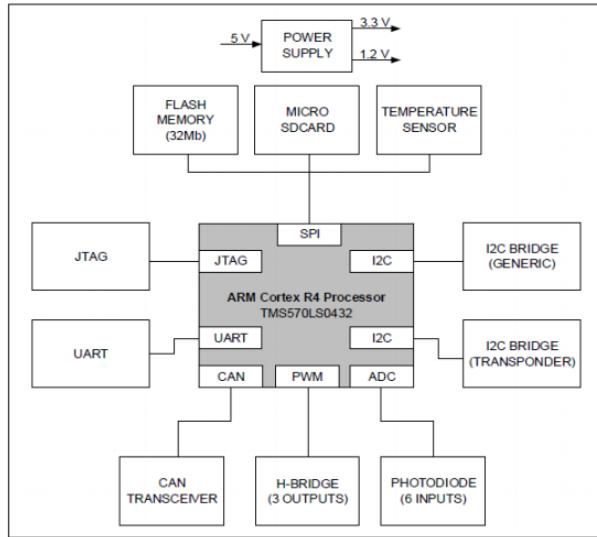


Figura 13 – Arquitetura do OpenOBC.

- **ISIS On Board Computer (iOBC)**

O iOBC, do fabricante ISIS (Innovative Solution In Space), é um computador de bordo com alto desempenho baseado em um processador ARM9 com velocidade de clock de 400MHz e oferece uma infinidade de interfaces padronizadas. Tem a capacidade de modularidade, permitindo a adição de eletrônicos ou interfaces específicas da missão, tornando o iOBC um ótimo computador de bordo para inúmeras missões. A arquitetura utiliza o processador AT91SAM9G20 do fabricante Microchip, o qual possui: núcleo ARM9 32-bit com 400MHz (ISIS, 2016).

A fabricante não disponibilizou a arquitetura do iOBC. A Figura 14 mostra a placa eletrônica do iOBC.

Fonte: Empresa ISIS⁷.



Figura 14 – Placa Eletrônica - iOBC.

• FM430 Flight Module

O FM430, da fabricante Pumpkin Inc, é uma solução compacta para sistemas ambientais difíceis. Possui como microcontrolador o MSP430F1612 de 16-bit, da fabricante Texas Instruments, com velocidade de clock de 7.3728 MHz. 50-60kB ROM e 2-10kB de RAM, 48 pinos I/O, 2 USART, 2 SPI, 1 I2C, 12-bit A/D e D/A, sensor de temperatura. SD Card para armazenamento (32MB – 2GB). Uma porta USB (Universal Serial Bus) e um conector de fonte de alimentação externa para facilitar a configuração pré-lançamento. A unidade do microcontrolador consome mais de 100 mW de potência (PUMPKIN,2008).

Fonte:(PUMPKIN, 2008, pág.6).

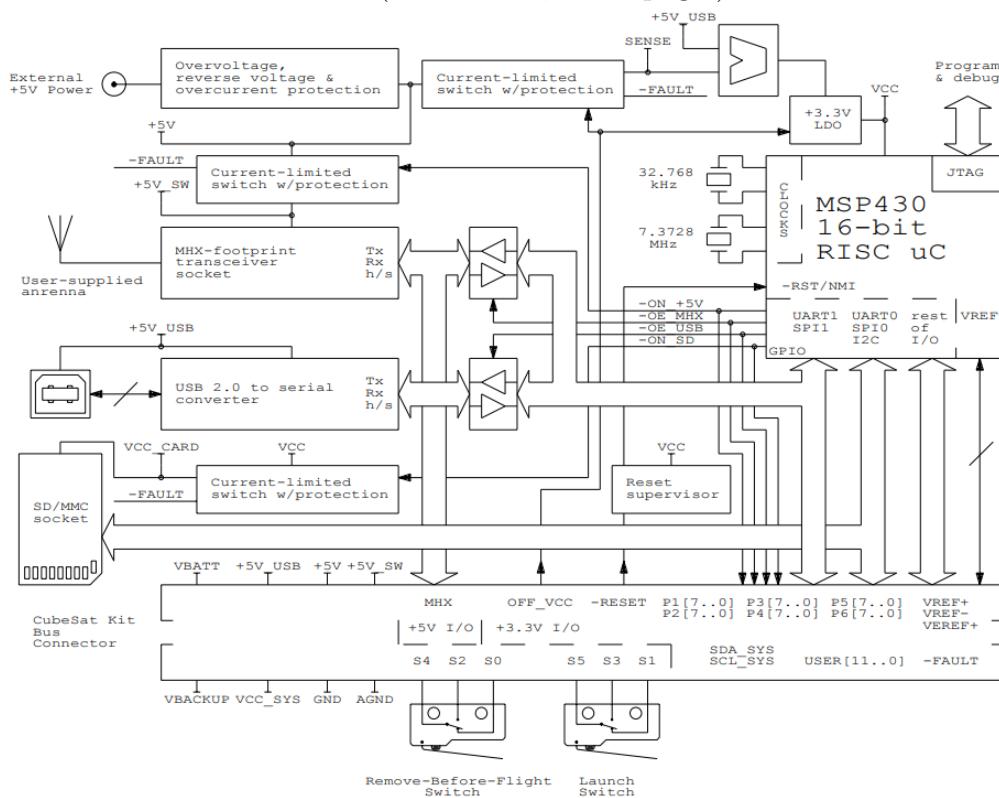


Figura 15 – Arquitetura do FM430.

Tabela Comparativa entre os microcontroladores

Tabela 1 – Microcontroladores selecionados.

	Projetos			
Recursos	FloripaSat OBDH	Open OBC	iOBC	FM430 Flight Mode
Processador	MSP430F6659IPZ	TMS570LS0432	AT91SAM9G20	MSP430F1612
Fabricante	Texas Instruments	Texas Instruments	Micro Chip	Texas Instruments
Clock	8 MHz	80 MHz	400 MHz	7.3728 MHz
Watchdog Timer	Sim	Sim	Sim	Sim
Memória Flash	512 KB	16 KB	X	55 KB
Memória RAM	66 KB	32 KB	32 KB	5 KB
EEPROM	X	16 KB	X	X
I2C	3	0	2	1
Canais ADC	16 canais de 12bit	6 canais de 12bit	6 canais de 10bit	8 canais de 12bit
CAN	X	X	X	X
SPI	3	2	2	2
PWM	5 canais	0	6	1
UART	3	1	7	2

2.5 Padrão PC/104

O PC/104 é o padrão de placas eletrônicas mais utilizado na indústria e em missões CubeSat. O PC/104 Embedded Consortium (2008) definiu as restrições mecânicas e elétricas para uma placa padrão de circuito impresso. As principais restrições podem ser encontradas a seguir:

- Cada placa deve ter uma forma de 90x96mm;
- A potência por módulo deve estar entre 1-2W, limitando a corrente do barramento para 4mA.
- Os módulos devem ser de 8-bits ou 16-bits, correspondendo ao barramento PC e PC/AT, respectivamente;
- O espaçamento entre as placas não deve exceder 15,24mm;
- Os conectores do barramento podem ser do tipo “empilhado” ou “não-empilhado” dependendo do projeto;
- A altura dos componentes não deve exceder 11,05mm;
- A uso do barramento deve estar de acordo com a Tabela 2:*8-bit and 16-bit ISA Bus Signal Assignments* do PC/104 Standard⁸.

⁸ PC/104 Standard, version 2.6. 2018. Disponível em:<https://pc104.org/wp-content/uploads/2015/02/PC104_Spec_v2_6.pdf>. Acesso em: 03 julho 2018.

A tecnologia PC/104 é vantajosa para missões CubeSat devido a padronização de dimensões, barramentos, interfaces mecânicas e elétricas. Esses fatores acarretam na redução de custos, riscos e tempo envolvidos no projeto, sem contar na versatilidade de integrar outras soluções presentes no mercado, que utilizam o mesmo padrão (JANES,2006). A Figura 16 mostra as dimensões mecânicas para uma placa PC/104 de 8-bits.

Fonte: (PC/104 Embedded Consortium, 2008, pág.15).

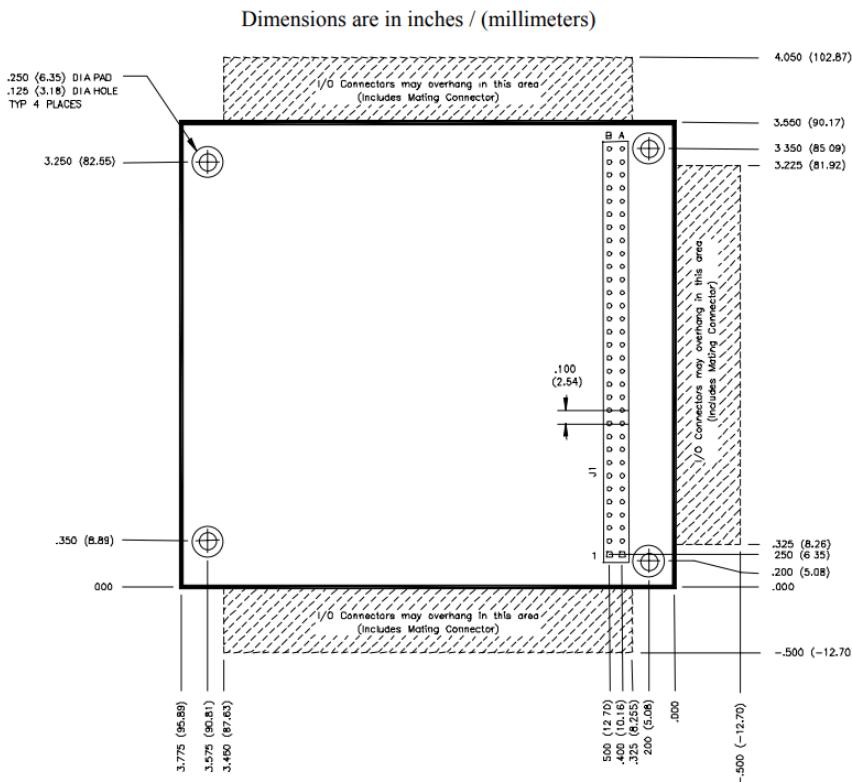


Figura 16 – PCB Padrão PC104 8-bits.

2.6 Ambiente Espacial

Um problema a ser considerado no desenvolvimento de um de um satélite, é o efeito do ambiente espacial nos subsistemas. Os problemas podem variar entre mal funcionamento operacional até danos físicos. Geralmente essas considerações são feitas para missões LEO (do inglês *Low Earth Orbit*) e Deep Space (do inglês *Espaço Profundo*) de longa duração (FROST; AGASID, 2015).

De acordo com Finckenor e Groh (2015), vários testes foram realizados na ISS (o inglês *International Space Station*) para estudar a degradação dos materiais no ambiente espacial. Os principais efeitos são o vácuo, radiação ultravioleta, radiação espacial, plasma, temperaturas extremas, fadiga térmica e impacto de lixo espacial.

Dentre os efeitos citados acima, o mais importante, para projetos da eletrônica

embarcada é a radiação espacial (FROST; AGASID, 2015). Uma breve introdução sobre o tema e os possíveis efeitos no sistema embarcado será dado nas seções a seguir.

2.6.1 Radiação Espacial

A radiação no espaço é formada por partículas emitidas de várias fontes, dentro e fora do sistema solar. Os raios cósmicos (radiações primárias) interagem com gases e outras substâncias em altas altitudes, produzindo a radiação secundária. A combinação das radiações primárias e secundárias formam o ambiente de radiação espacial. As partículas desse ambiente podem causar degradação e falha dos sistemas embarcados (NASAa, 1996).

A quantidade de radiação que incide nos satélites depende da altitude e do tipo de órbita. A maioria dos CubeSats é colocado em órbitas LEO (altitude entre 100-1.000 km), recebendo uma dose de radiação de aproximadamente 0,1 krad/ano. Em missões com duração típica de 3-5 anos, a dose de radiação é menor que 0,5 krad/ano (PETKOV, 2003).

De acordo com Petkov (2003), os efeitos provocados por essas radiações podem ser classificados de acordo com a Figura 17.



Figura 17 – Classificação dos efeitos da Radiação Espacial.

- **Efeito Acumulativo**

O Total de Dose Ionizante (TID, do inglês *Total Ionizing Dose*) e o Dano de Deslocamento (DD, do inglês *Displacement Damage*) são efeitos permanentes causados pela radiação ao longo do tempo. Com isso, o tempo da missão é um fator que intensifica tais efeitos. Abaixo há uma breve descrição sobre tais efeitos (PETKOV, 2003).

Tabela 2 – Efeitos Acumulativos causada pela radiação espacial.

Efeito	Descrição
Total Ionizing Dose (TID)	Incidência de radiação sobre o material. Ao longo do tempo ocasiona mudança nas características elétricas, levando à falhas
Displacement Damage (DD)	Deslocamento dos átomos em semicondutores, levando à alteração das características elétricas do dispositivo

- **Efeito de Evento Único**

Efeitos de Evento Único (SEE, do inglês Single Event Effect) são efeitos ocorridos em dispositivos eletrônicos devido a passagem de uma única partícula pelo sistema. Essa passagem pode mudar um elemento bi estável ou estado digital de uma memória, acarretando em efeitos temporários e permanentes (PETKOV, 2003).

Tabela 3 – Efeitos de Evento Único causados pela radiação espacial.

Efeito	Descrição
Single Event Upset (SEU)	Alteração temporária do estado do latch ou da memória
Single Event Latchup (SEL)	Falha permanente causada pela quantidade de corrente drenada durante o estado de latchup
Single Event Gate Rupture (SEGR)	Falha permanente do dispositivo, mais comum em transistores de potência
Single Event Burnout (SEB)	Falha permanente do dispositivo, mais comum em transistores de potência
Enhanced Low Dose Rate Sensitivity (ELDRS)	Falha permanente do dispositivo causada pela radiação em baixas doses

2.6.2 Categoria de Componentes

A resistência a radiação dos componentes é um fator muito importante para estimar a taxa de falha e a vida útil do sistema. Há três categorias que os componentes eletrônicos podem se enquadrar, em relação a resistência a radiação: *Comercial* (COTS), *Rad Tolerant* e *Rad-Hard*. Abaixo há uma breve descrição de cada uma, juntamente com os níveis de TID, SEU e taxa de SEU (NASA, 1999).

- *Comercial*:

- O processo de design não garante resistência à radiação.
- Pouco controle da radiação.
- Níveis de resistência:
 - * Dose Total: 2 a 10 krad (típico).
 - * Limite de SEU: $5 \frac{MeV}{mg.cm^2}$.
 - * Taxa de erro SEU: $10^{-5} \frac{erros}{bit.dia}$ (típico).

- O cliente realiza testes de radiação e assume todos os riscos.
 - Avaliação e risco do cliente.
- *Rad Tolerant:*
 - O design garante resistência à radiação até um certo nível.
 - Pouco controle da radiação.
 - Níveis de resistência:
 - * Dose Total: 20 a 50 krad (típico).
 - * Limite de SEU: $20 \frac{MeV}{mg.cm^2}$.
 - * Taxa de erro SEU: 10^{-7} - $10^{-8} \frac{\text{erros}}{\text{bit.dia}}$.
 - Geralmente testado apenas para falha funcional.
 - Avaliação e risco do cliente.
 - *Rad-Hard:*
 - Projetado e processado para um nível de dureza específico
 - Vários testes realizados com o substrato do chip
 - Níveis de resistência:
 - * Dose Total: > 200 krad a > 1 Mrad.
 - * Limite de SEU: $80-150 \frac{MeV}{mg.cm^2}$.
 - * Taxa de erro SEU: 10^{-10} - $10^{-12} \frac{\text{erros}}{\text{bit.dia}}$.
 - Geralmente testado apenas para falha funcional.
 - Avaliação e risco do cliente.

3 DESENVOLVIMENTO DO HARDWARE

Esse capítulo descreverá o desenvolvimento do *hardware*. Inicialmente será mostrado os requisitos e funcionalidades consideradas para o sistema. O passo seguinte é um dos mais importantes pois mostrará o processo de seleção do microcontrolador, componente que afetará restante do projeto. Em seguida será exposto os componentes adicionais que formarão o OBC, como: memória, sensores, etc. Por fim, será apresentado alguns mecanismos de mitigação de falhas e os esquemáticos do sistema.

3.1 Requisitos e Funcionalidades

Para o desenvolvimento de qualquer projeto, a definição dos requisitos é essencial, pois essa etapa influenciará as demais fases e definirá as expectativas das partes interessadas. Até o presente momento, poucos requisitos foram delimitados, pois o projeto da missão CubeSat ainda está em fase de discussão.

A abordagem utilizada para contornar essa situação foi adicionando requisitos de CubeSats já lançados com alguns requisitos já especificados para a missão. No Apêndice B há uma explicação sobre essa abordagem.

Com base no Apêndice B, foi possível delimitar as funcionalidades para o *hardware* do sistema. Esses requisitos podem ser vistos na tabela abaixo.

Tabela 4 – Requisitos do Hardware.

Número do Requisito	Descrição do Requisito
OBC-H-R1	OBC deve realizar a aquisição dos seguintes dados: Temperatura; Tensão e Corrente consumidas pelo sistema; Sensor Inercial; Dados dos demais subsistemas do CubeSat; Payload (Imagens da Câmera).
OBC-H-R2	Possuir sistema de proteção contra travamentos.
OBC-H-R3	Os componentes devem suportar a faixa de temperatura das Órbitas Baixas e Heliosíncronas.
OBC-H-R4	O OBC deve possuir interfaces condizente com cada subsistema do satélite.
OBC-H-R5	Possuir concepção versátil.
OBC-H-R6	Possuir soluções que protejam o sistema contra falhas.
OBC-H-R7	Possuir armazenamento não volátil de dados.

3.2 Análise do Microcontrolador

O microcontrolador é um dos componente mais importante do OBC, pois ele executará o *software* destinado ao gerenciamento do satélite. No processo de seleção do mi-

crocontrolador, utilizou-se as seguintes características como parâmetro de escolha:

- **Baixo Consumo:** A energia disponível em um Cubesat depende da área de cobertura e eficiência dos painéis solares. De acordo com Wiley J. e Richard Wertz (1992), os painéis solares podem fornecer, aproximadamente, $100\text{W}/\text{m}^2$. Supondo que a futura missão tenha painéis solares fixados na superfície da plataforma 3U, um lado do CubeSat teria $0,03\text{m}^2$ (3W) de cobertura. Supondo também que essa potência seja dividida igualmente entre os subsistemas (TT&C, AODCS, OBC, EPS e PAYLOAD), o OBC teria disponível até 0,6W.
- **Conversores AD:** O OBC contará com muitos sensores (corrente, luminosidade, etc), sendo assim, conversores AD são importante para aferir os dados dos sensores.
- **Portas I/O:** Possuir GPIO (do inglês *General-Purpose Input/Output*) se torna importante para a comunicação com outros subsistemas do satélite.
- **Interfaces Seriais (UART, I2C, SPI):** Os protocolos seriais são os mais utilizados entre os protocolos de comunicação digital. Com isso, possuir pinos dedicados à esses protocolos ajuda no desenvolvimento do OBC.
- **PWM:** Para o controle de atitude será necessário utilizar motores de passo, consequentemente há a necessidade de pinos que gerem sinais PWM.
- **Frequência de Operação:** Em aplicações embarcadas, a performance do microcontrolador é normalmente proporcional à frequência do clock.
- **Faixa de Temperatura:** Segundo CZERNIK (2004), a estimativa de temperatura em uma órbita LEO pode variar de -80°C (eclipse), e 53°C (insolação). Já para órbitas Sun-Síncronas, a estimativa de variação é de -27°C à 43°C (FRIEDEL, Jonas; MCKIBBON, Sean, 2011). O microcontrolador deverá suportar essas temperaturas.

A partir das características acima, procurou-se por microcontroladores das fabricantes mais usuais, como a Microchip e Texas Instruments. Optou-se pela seleção de microcontroladores que possuíssem uma placa de desenvolvimento, para viabilizar o tempo de implementação. Os dispositivos utilizados na escolha são expostos na Tabela 5.

Tabela 5 – Microcontroladores selecionados.

Microcontrolador	Consumo	AD	I/O	InterfaceSerial	PMW	Watchdog Timer	Frequência	Faixa de Temperatura [°C]
MSP432P4111	25mW @ 48MHz	24 AD	84	4 UART/SPI, 4 I2C	Sim	sim	48MHz	-40 a 85
MC9S12XHZ512	30,9mW @40MHz	16 AD	57	3 UART/2 SPI, 1 I2C	sim	sim	40MHz	-40 to 105
MSP430F5529	30,3mW @ 25MHz	12 AD	63	2 UART/2 SPI, 1 I2C	sim	sim	25MHz	-40 a 85
RM42L432	480mW @ 100MHz	16 AD	45	2 UART/1 SPI, 2 I2C	sim	sim	100MHz	-40 to 105
ATSAMD21J18	5mW @ 48MHz	20 AD	52	6 UART/SPI/I2C	sim	sim	48MHz	-40 a 85
ADuCM360	18,15mW @16MHz	4 AD	19	1 UART, 1 SPI, 2 I2C	sim	sim	16MHz	-40 a 125

Após a seleção dos microcontroladores, foi colocado pesos para cada característica de acordo com a importância no sistema. Vale a pena ressaltar que os dados acima foram retirados dos datasheet de cada microcontrolador.

Tabela 6 – Pontuação de cada microcontrolador.

Microcontrolador	Consumo	AD	I/O	InterfaceSerial	PMW	Watchdog Timer	Frequência	Faixa de Temperatura [°C]	SOMA
PESO	5	3	3	4	3	3	5	4	30
MSP432P4111	5	3	3	3,5	3	3	2,4	3,03	25,93
MC9S12XHZ512	2,83	2	2,04	3,7	3	3	2	3,52	22,09
MSP430F5529	1,8	1,5	2,25	3	3	3	1,25	3,03	18,83
RM42L432	0,46	2	1,61	3,7	3	3	5	3,52	22,29
ATSAMD21J18	5	2,5	1,86	4	3	3	2,4	3,03	24,79
ADuCM360	1,93	0,5	0,68	3,5	3	3	0,8	4	17,41

A partir da tabela acima pode-se observar que microcontroladores MSP432P4111 e o ATSAMD21J18 possuem as maiores nota. Ambos contam com processadores da família ARM Cortex M, sendo o Microchip da categoria M0+ e o TI da categoria M4F. Alguns pontos podem ser levantados em comparação aos dois microcontroladores:

- O microcontrolador da Texas Instruments possui 32 GPIOs e 4 ADC a mais que o microcontrolador da Microchip.
- De acordo com a Figura 18, o MSP432P4111 (M4F) possui uma performance superior ao ATSAMD21J18 (M0+), aproximadamente 37%.
- Outro item que não foi colocado na tabela de comparação é a memória SRAM. O MSP432P4111 conta com 64 KB de SRAM contra 32 KB do ATSAMD21J18.

A partir da análise acima, pode-se concluir que o MSP432P4111 é a melhor opção entre os microcontroladores analisados. Ele pode ser considerado uma ótima ferramenta

Fonte:(ARM Microcontrollers, 2018, pág.1).

Cortex-M

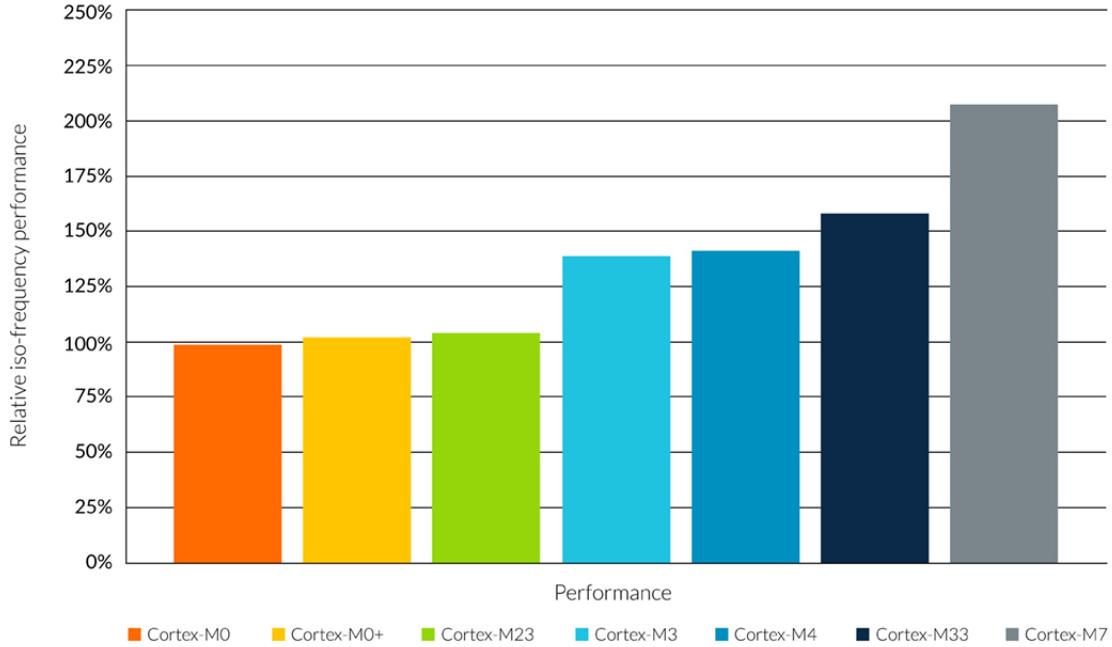


Figura 18 – Performance dos processadores CORTEX-M.

em aplicações de baixo consumo, missões 1U, ou aplicações que há a necessidade de uma boa performance, missões 2U e 3U. Sendo assim o microcontrolador selecionado para o OBC é o MSP432P4111.

3.3 Microcontrolador MSP432P4111

As características do MSP432P4111 serão apresentadas nessa seção. Será mostrado o processador, unidade de memória interna, periféricos e consumo de energia em cada modo de operação. As referências utilizadas para essa seção foram o DataSheet do microcontrolador (Texas Instruments, 2018) e o livro *Microcontroller Engineering with MSP432* (BAI, 2016). A Figura 19 mostra o diagrama de blocos do MSP432P4111.

O MSP432P4111 possui como microprocessador o ARM Cortex-M4F. Esse microprocessador possui arquitetura *Reduced Instruction Set Computing* (RISC) com 32-bit de instrução, podendo operar em frequências acima de 48MHz. Ele foi projetado para aplicações que exigem baixa potência, alta eficiência, boa capacidade de processamento de sinais, baixo custo e fácil usabilidade. Esse microcontrolador não possui nenhuma unidade de armazenamento, entretanto, ele oferece interfaces para memórias ROM e SRAM externas. Devido ao tamanho de instrução, 32-bit, a capacidade máxima pesquisável para a memória é de 4GB, podendo haver mais de uma unidade de armazenamento com esse tamanho (Texas Instruments, 2018).

Fonte: (Texas Instruments, 2018, pág.3).

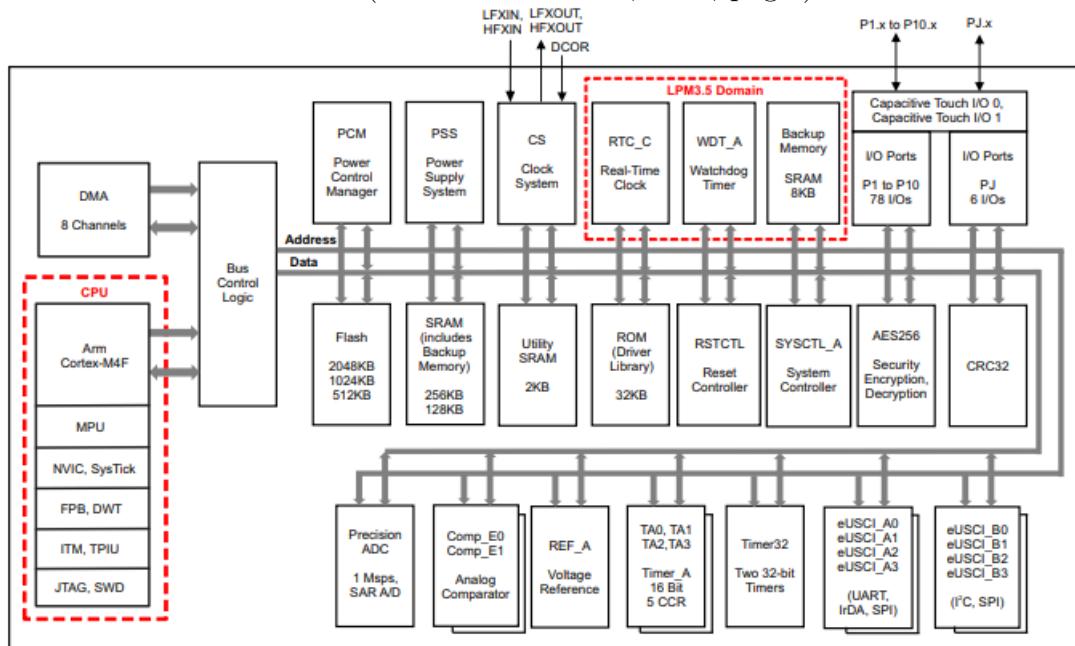


Figura 19 – Arquitetura do microcontrolador MSP432P4111.

O barramento interno é de 32-bit, também chamado de *Advanced Microcontroller Bus Architecture* (AMBA). Esse tipo de barramento oferece eficiência de operação e baixo consumo. O barramento principal entre o MCU e os componentes externos é o *Advanced High Performance Bus* (AHB), oferecendo interface com a memória e os periféricos (BAI, 2018).

Mesmo o microprocessador possuindo o *System Control Block*, que informa se houve algum erro na operação do sistema, não é tolerante à falhas.

3.3.1 Unidade de Memória

O MSP432P4111 possui os seguintes componentes de memória:

- 2048KB Memoria Flash principal;
- 32KB Memória Flash de informação (área utilizada para *Bootloader*, *TVL* e *Flash MailBox*);
- 256KB SRAM (incluindo 8KB de memória de backup);
- 32KB ROM carregada com as bibliotecas MSP432TM *Peripheral Driver*.

Esse microcontrolador suporta um endereçamento de 4GB dividido em oito zonas de 512MB. A memória de 2048KB serve para armazenar o programa do usuário.

3.3.2 Portas e Periféricos

O microcontrolador MSP432P4111 possui 11 portas de propósito geral (GPIO), cada porta pode variar de 1 a 10. Para o controle de saída e entrada utiliza-se os registradores: PxIN, PxOUT, PxDIR, PxREN e PxDS. Para selecionar as portas, utiliza-se os registradores: PxSEL0, PxSEL1 e PxSEL2. Por fim, para configurar as interrupções das portas, utiliza-se os registradores: PxIFG, PxIES, PxIE e PxIV. O valor de ‘x’ indica o número da porta, 1 a 10 (BAI, 2018).

O microcontrolador conta com os seguintes periféricos:

- 4 UART, com frequência de clock até 7 MHz;
- 4 I2C, com frequência de clock até 1MHz;
- 8 SPI;
- 24 ADC canais de 14-bit;
- 84 I/O;
- 4 Temporizadores de 16-bit (Captura, Comparação e PWM);

3.3.3 Consumo e Modos de Operação

O microcontrolador suporta alguns modos de operação que permitem otimizar a potência consumida pelo sistema a cada aplicação. O *Power Control Manager* (PCM) é responsável por gerenciar os modos de baixo consumo de cada área do sistema.

No Apêndice C há uma tabela que mostra todos os modos de operação, com uma descrição e o consumo em cada modo.

3.4 Unidade de Armazenamento

Geralmente, em projetos de sistemas embarcados, após a escolha do microcontrolador é realizada a escolha da memória. Mesmo o microprocessador possuindo 2048KB de memória Flash e 256KB de SRAM, será implementado um banco de memória externo que será destinada ao armazenamento do dados de telemetria e da *payload*.

Diante da grande variedade de memórias disponíveis, a escolha do tipo de memória para esta aplicação se tornou uma tarefa exaustiva. Nesse processo de escolha, foi utilizada uma tabela comparativa, presente no *Small Spacecraft Technology State of the Art* (FROST; AGASID,2015) e ilustrada abaixo.

De acordo com a Tabela 7, as memória MRAM e FERAM (ou FRAM) são as que mais resistem a radiação (TID), o que as caracterizam como ótimas alternativas

Tabela 7 – Comparaçao dos tipos de Memória.

Característica	SRAM	DRAM	Flash	MRAM	FRAM	CRAM/PCM
Não-Volátil	Não	Não	Sim	Sim	Sim	Sim
Tensão de operação, + - 10%	3.3 - 5 V	3.3 V	3.3 e 5 V	3.3 V	3.3 V	3.3 V
Organização bits/die	512k x 8	16M x 8	16M x 8; 32M x 8	128k x 8	16k x 8	-
Retenção de Dados (@70° C)	N/A	N/A	10 anos	10 anos	10 anos	10 anos
Resistência (Ciclo de Deletar/Escrita)	Ilimitado	Ilimitado	10^6	10^{13}	10^{13}	10^{13}
Tempo de Acesso	10 ns 200ms escrita; 2ms para deletar	25 ns	50 ns depois de uma pagina lida;	300 ns	300 ns	100 ns
Radiação (TID)	1Mrad	59krad	30krad	1Mrad	1 Mrad	1 Mrad
SEU rate (relativo)	zero	Alto	zero (celulas); Baixo -Medio (dispositivos eletronicos)	zero	zero	zero
Faixa de Temperatura	Padrão Militar	Industrial	Comercial	Padrão Militar	Padrão Militar	Padrão Militar
Potência	500 mW	300 mW	30 mW	900 mW	270 mW 1.5 MB	-
Pacote	4MB	128 MB	128 - 256 MB	1 MB (pacote com 12 chips)		

Fonte: (FROST; AGASID, 2015, pág.97).

para a unidade de armazenamento. Comparando-se essas memórias, pôde-se observar-se que ambas possuem as seguintes características em comum: não-volatilidade, tensão de operação, retenção de dados, ciclos de operação, tempo de acesso, TID, SEU e faixa de temperatura.

Dentre as características distintas entre as memórias MRAM e FRAM, observa-se que a memória do MRAM gasta três vezes mais energia que a memória do FRAM. Como o quesito consumo elétrico é primordial em aplicações espaciais, a memória FRAM foi escolhida para compor a unidade de armazenamento. Essa memória será destinada para arquivar o software embarcado.

Para escolher o tamanho da memória FRAM, há a necessidade de estimar o tamanho do software embarcado. Esse software depende de vários fatores, como: redundâncias, funcionalidades implementadas, segurança, algoritmos, etc, o que torna essa estimativa muito complicado. Sendo assim, definiu-se o espaço como o dobro da memória SRAM do microcontrolador, resultando em 512KB.

Em relação ao espaço para a telemetria e dados dos subsistemas, foi realizado uma estimativa, utilizando os parâmetros de armazenamento de dados da missão *SWIS-SCube*. Esses parâmetros foram adaptados ao contexto do projeto, adicionando os dados proveniente da Câmera. Esses parâmetros se encontram no Apêndice D.

A partir dessa estimativa, a quantidade de dados armazenado em 1 dia seria de 176MB, 10MB proveniente de telemetria e 166MB da *Payload* (imagens). Desse modo, um cartão de 512MB seria mais que o suficiente para armazenar os dados provenientes

do sistema em 2 dias. Entretanto, como não se sabe a quantidade de estações de solo disponíveis para descarregar os dados, optou-se por escolher o tamanho máximo que o microcontrolador pode suportar, que no caso é de 4GB.

A memória Flash é a única que possui uma arquitetura que comporta 4GB de capacidade. Com isso, foi selecionado um cartão de memória com 4GB de capacidade para armazenar os dados da telemetria e da *Payload*.

Para a escolha dos fornecedores, escolheu-se o site *DigiKey*¹. Esse site possui filtros que ajudaram na seleção dos itens. O preço foi o fator decisório na seleção. As memórias escolhidas e suas características são mostradas na Tabela 8.

Tabela 8 – Especificação das memórias da Unidade de Armazenamento.

PRODUTO	Fornecedor	Temp OP	Corrente	Vin	Capacidade	Protocolo
FM25V05-GTR	Cypress Semiconductor Corporation	-40°C ~85°C	300uA	2 V ~3.6 V	512Kb (64K x 8)	SPI
SQF-MSDM1-4G-21E	Advantech Corp	-40°C ~85°C	150uA	2.7 V ~3.6 V	4GB	SPI

3.5 Periféricos

Essa seção visa mostrar os sensores utilizados para compor o sistema e ajudar o microcontrolador a atingir os requisitos exigidos na Seção 3.1.

3.5.1 Sensor de Corrente

Com o intuito de cumprir o requisito de leitura de corrente, escolheu-se o sensor INA193A-EP. Esse sensor possui qualificação de alta confiabilidade (Hi-Rel, do inglês High Reliability), ideal para aplicações onde há grande variação de temperatura. O INA193A-EP suportar valores de -16V a 80V e um máximo de 1,3A.

A figura 20 mostra o esquemático eletrônico de uma aplicação usual do INA193A-EP.

¹ Disponível em: <<https://www.digikey.com>>

Fonte:(TEXAS INSTRUMENTS, 2018d, pág.2)

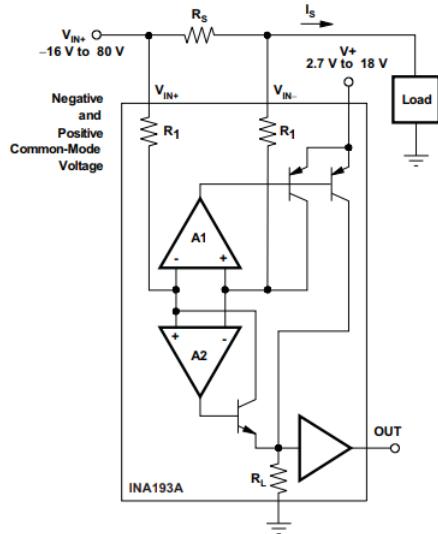


Figura 20 – Esquemático Eletrônico de uma aplicação usual do INA193A-EP.

A tabela 9 mostra as especificações do INA193A-EP. Essas informações foram retiradas da ficha técnica do sensor.

Tabela 9 – Informações técnicas do INA193A-EP.

Produto	Fornecedor	Temp. Op [°C]	Corrente [mA]	Vin [V]	Erro [%]
INA193A-EP	Texas Instruments	-55~105	1.3mA	2.7 ~18	2.2

3.5.2 Sensor Inercial

O sensor inercial será responsável por realizar as medidas iniciais nos 3 eixos, dados que possibilitará o controle das rodas de reação do AODCS. O MPU9250, fabricado pela *TKD ENVISENSE*, foi escolhido para realizar essa tarefa. Esse sensor possui acelerômetro, magnetômetro e giroscópio nos 3 eixos, além de um sensor de temperatura (INVENSENSE, 2018).

A Tabela 10 mostra as especificações do MPU9250. Essas informações foram retiradas da ficha técnica do sensor.

Tabela 10 – Informações técnicas sobre sensor MPU9250.

Produto	Fornecedor	Temp. Op. [°C]	Corrente [mA]	Vin [V]	Resolução [bit]	Protocolo
MPU-9250	TDK InvenSense	-40 ~85	3.2mA	2.4 ~3.6	Giro./Ace. ->16 Mag. ->14	SPI/I2C

A figura 21 mostra o esquemático eletrônico de uma aplicação usual do MPU9250, utilizando protocolo I2C(esquerda) e SPI (direita).

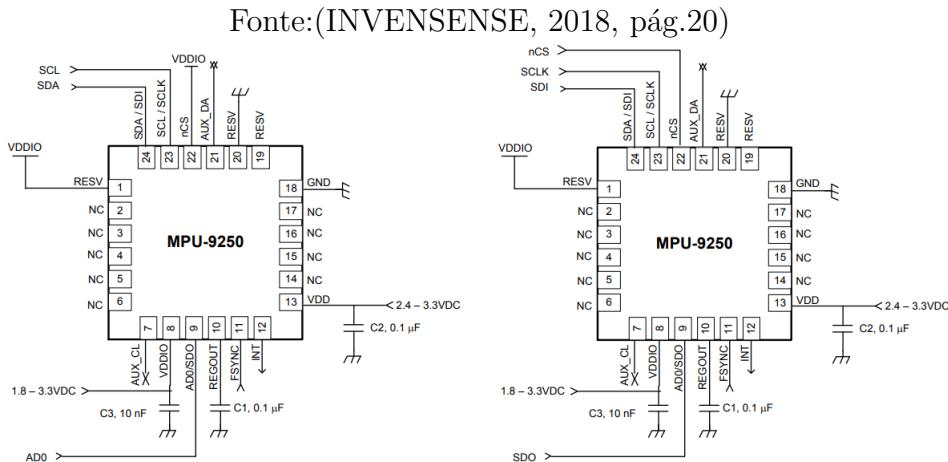


Figura 21 – Esquemático Eletrônico de uma aplicação usual do MPU9250.

3.5.3 Sensor de Temperatura

Mesmo o MPU9250 possuindo um sensor de temperatura, foi escolhido outro para realizar a medida da temperatura do OBC. Possuir um sensor de temperatura dedicado à essa função é ideal, pois as funções primárias do MPU9250 podem interferir no valor das medidas. Por exemplo, caso o sensor inercial esteja sendo utilizado por um período muito longo, a temperatura aferida será acima da temperatura correta.

O sensor de temperatura escolhido foi o TMP422-EP, da fabricante Texas Instruments, dividido sua confiabilidade HI-REL. O dispositivo comunica através do protocolo de comunicação o SPI, sendo capaz de medir temperaturas com uma precisão de 2 °C em uma faixa de temperatura de -55°C a 125°C (TEXAS INSTRUMENTS, 2008).

A tabela 11 mostra as especificações do TMP422-EP. Essas informações foram retiradas da ficha técnica do sensor.

Tabela 11 – Informações técnicas sobre sensor TMP422-EP.

Produto	Fornecedor	Temp. Op. [°C]	Corrente [uA]	Vin [V]	Resolução [bit]	Protocolo
TMP422-EP	Texas Instruments	-55 ~127	60	2.7 ~5.5	12	SPI

3.6 Interfaces

Essa seção visa descrever as interfaces que possibilitam a comunicação do OBC entre os demais subsistemas do satélite, durante a missão, e também com o usuário, a fase de programação e debug.

Vale a pena ressaltar que devido ao tempo, essa parte será melhor descrita no TCC2. Os pontos a serem discutidos no TCC2 são:

- Barramento PC104;
- USB (Universal Synchronous Bus);
- JTAG (Joint Test Action Group).

3.7 Dimensões da Placa

Como foi mencionado anteriormente, o formato da placa seguirá o padrão *PC/104-Plus*. Esse padrão é comumente visto em missões CubeSat. A placa possui dimensões físicas de 90.17 x 95.89 mm, possuindo 4 furos M3 nas extremidades da placa. A Figura 22 ilustra as dimensões da placa estabelecida pelo *PC/104 Consortium*.

Fonte:(PC/104 EMBEDDED CONSORTIUM, 2008, pág.20)

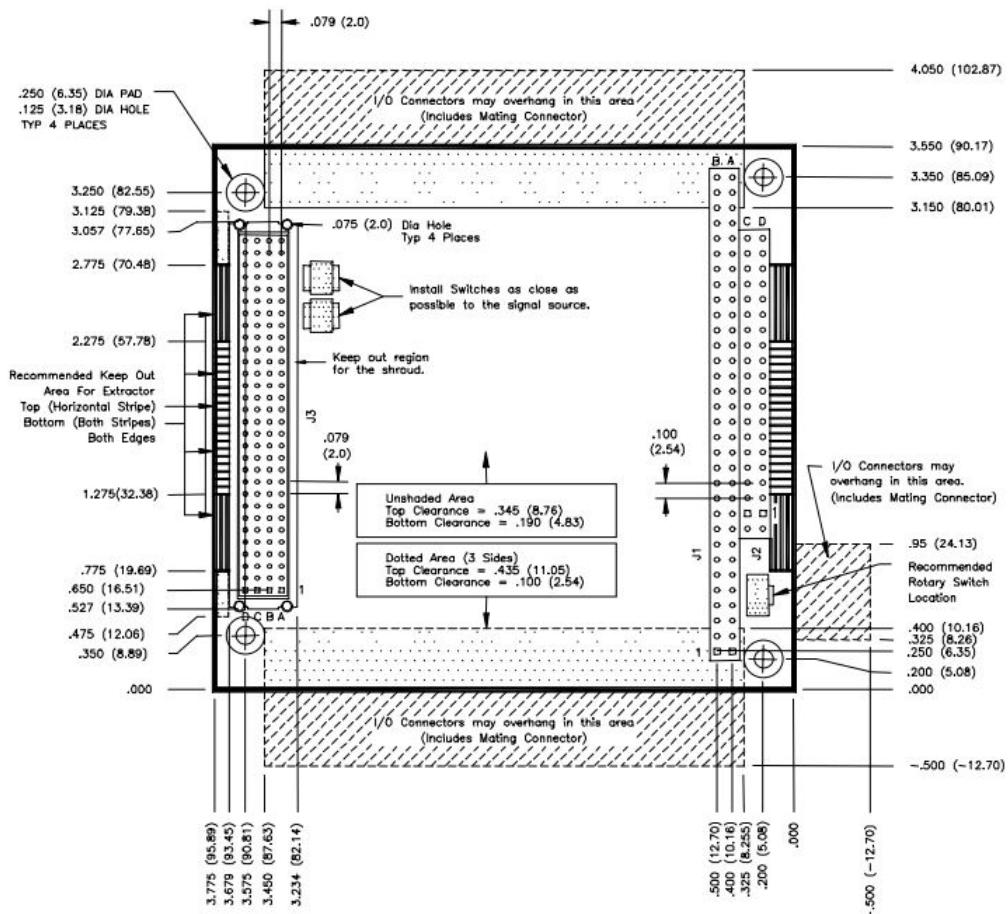


Figura 22 – Dimensões do módulo PC/104-Plus em polegadas e milímetros.

3.8 Sistema para Mitigar Falhas

De acordo com Frost e Agasid (2015), missões de longa duração necessitam de mecanismos que diminuam o risco devido à radiação espacial. Como o microcontrolador e os componentes selecionados no projeto não possuem classificação *Rad-Hard*, optou-se pela utilização de mecanismos que atenuassem as falhas.

A partir de uma breve pesquisa, levantou-se as seguintes estratégias:

- Proteção Metálica (EMI *shielding*);
- Proteção contra corrente excessiva (*OverCurrent*);
- Código Corretor;
- *Watchdog* externo.

A discussão sobre as estratégias acima serão abordadas realizadas no TCC2. O único ponto a ser discutido no TCC1 será o *Watchdog* externo.

3.8.1 *Watchdog* Externo

Conforme Frost e Agasid (2015), o *Watchdog* é normalmente utilizado para monitorar o estado do microcontrolador, evitando o travamento do sistema. Basicamente, um *Watchdog* externo é um contador regressivo que, ao final da contagem, reinicia o microcontrolador em caso de um evento SEE.

O microcontrolador já possui um *Watchdog* interno que reiniciará o microcontrolador em caso de travamento. O uso de um contador externo trará mais robustez ao sistema.

O contador utilizado para o sistema de *Watchdog* externo foi o STWD100, o mesmo utilizado pelo projeto do Botma (2011). Esse dispositivo é produzido pela fabricante STMicroelectronics e possui três tempos para *reset* (3.4 ms, 6.3 ms, 102 ms e 1.6 s).

A escolha do tempo de *reset* será feita no TCC2. Nessa fase, o desenvolvimento do *software* embarcado estará em sua fase final e informações do código, como o tempo para executar todas as *Tasks*, estarão disponível.

4 DESENVOLVIMENTO DO SOFTWARE

Este capítulo visa descrever o desenvolvimento do *software* embarcado. Inicialmente será mostrado os requisitos e funcionalidades consideradas no desenvolvimento do sistema. Logo em seguida, será exposto a arquitetura utilizada para a aplicação. Após essa etapa, será mostrado com mais detalhes os *softwares* FreeRTOS, DriverLib e a camada de serviço do cliente.

Tendo em vista que a missão CubeSat ainda está em fase de discussão, a abordagem a ser utilizada no desenvolvimento é realizar um *software* modular que, a medida que os requisitos se tornem definitivos, as funcionalidades sejam implementadas. Com isso, o *software* a ser desenvolvido servirá para teste do *hardware* e deverá ser atualizado para a missão final.

4.1 Requisitos e Funcionalidades

Para o desenvolvimento de qualquer projeto, a definição dos requisitos é essencial, pois essa etapa influenciará todo o projeto e definirá as expectativas das partes interessadas. Como mencionado acima, poucos requisitos foram delimitados devido à fase inicial da missão CubeSat.

A abordagem utilizada para contornar essa situação foi adicionando requisitos de CubeSats já lançados com alguns requisitos já especificados para a missão, mesma abordagem utilizada na Seção 3.

Com base no Apêndice B, foi possível delimitar as funcionalidades para o *hardware* do sistema. Esses requisitos podem ser vistos na tabela abaixo.

Tabela 12 – Requisitos do *software* emabarcado.

Número do Requisito	Descrição do Requisito
OBC-SW-R1	O OBC deve armazenar os seguintes dados a cada um segundo: Imagens da Carga Útil e informações temporais e espaciais das imagens; Temperatura do sistema; Tensão e Corrente consumidas pelo sistema; Atitude do CubeSat; Resposta de cada subsistema.
OBC-SW-R2	O OBC deve controlar os subsistemas do CubeSat.
OBC-SW-R3	O OBC deve realizar um log de eventos do sistema.
OBC-SW-R4	O OBC deve ter um controle da referência temporal, com uma precisão de 500ms.
OBC-SW-R5	O OBC deve realizar o pacote de telemetria/payload e enviar dados para o subsistema de TT&C, durante uma janela de transmissão.
OBC-SW-R6	O OBC deve identificar e executar os comandos recebidos da Estação Terrestre.
OBC-SW-R7	O OBC deve alternar os modos de operação de acordo com a potência na bateria.
OBC-SW-R8	O OBC deve possuir um sistema anti travamento.

4.2 Arquitetura do Software

A arquitetura de um software é um dos principais pontos a serem definidos em um projeto de *software*, pois a arquitetura influenciará na manutenibilidade e reusabilidade do sistema. Existem várias arquitetura de *software*, sendo as mais usuais: Arquitetura Baseada em Camadas (LBA, do inglês *Layered-based Architecture*), Arquitetura Orientada em Serviços (SOA, do inglês *Service Oriented Architecture*) e a Arquitetura Baseada em Desenvolvimento (CBD, do inglês *Component-based Development*). Dentre essas, a arquitetura que melhor atende os padrões de modularidade é a Arquitetura em Camadas, pois abstrai o *software* embarcado de acordo com sua proximidade com o *hardware* (BACELO, 2010).

A arquitetura em camadas (do inglês *Layered Architecture*) é muito utilizada em sistemas embarcados, pois ajuda na abstração de alguns componentes e interfaces. Isso faz com que o usuário não precise ter noção de partes muito específicas do sistema, facilitando na usabilidade e na atualização do *software*. A figura abaixo mostra uma arquitetura em camadas de abstração, a mesma que será utilizada no sistema.

Fonte: (EBRARY, 2018, adaptado pág.1).

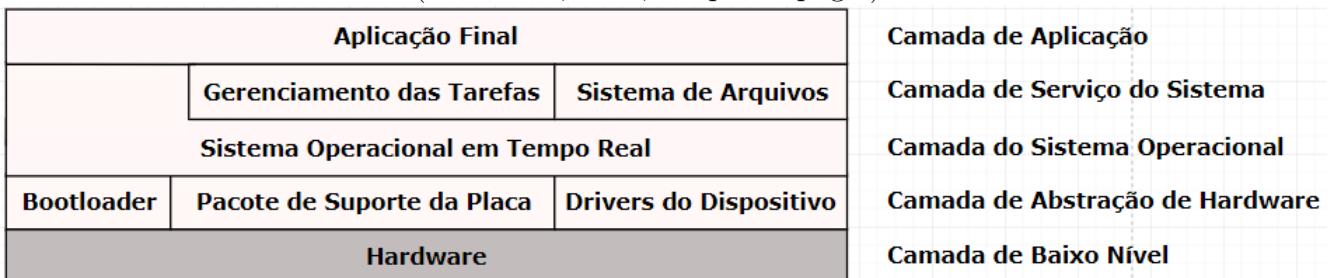


Figura 23 – Arquitetura de abstração de Camadas.

Abaixo há uma breve descrição de cada camada e sua aplicação no sistema final.

- **Camada de Abstração de Hardware**

A camada de abstração de *hardware* (HAL, do inglês *Hardware Abstraction Layer*), é uma camada de abstração de *software* entre o *hardware* do sistema embarcado e o sistema operacional. Em geral, o HAL inclui o bootloader, o pacote de suporte de placa, drivers e outros componentes. Ela é a camada de mais baixo nível do *software* embarcado (EBRARY, 2018).

No sistema, a camada de HAL é desempenhada pelo pacote *Driver Library (Driver-Lib)*, desenvolvido pela Texas Instruments, que tem o intuito de facilitar o desenvolvimento de projetos embarcados e ajudar na portabilidade dos códigos. Utilizando esse pacote, o desenvolvedor não necessita saber o que acontece a nível de regis-

dor, tornando o desenvolvimento mais amigável e rápido (TEXAS INSTRUMENTS, 2018).

O *DriverLib* inclui as Interface de Programação de Aplicativos (APIs, do inglês *Application Programming Interface*) das funcionalidades de ADC, Interrupção, UART, entre outros. Esse pacote pode ser também embutido no código final ou não. O MSP432P4111 possui em sua memória ROM uma cópia do *DriverLib*, fazendo com que o pacote não interfira no tamanho do código final.

- **Camada do Sistema Operacional**

Um Sistema Operacional de Tempo Real (RTOS, do inglês *Real-Time Operating System*) é um sistema de *software* que tem a habilidade de prover um serviço em um tempo pré-determinado. O RTOS gerencia uniformemente os recursos da camada inferior, camada HAL, oferecendo esses recursos em forma de serviços (EBRARY,2018).

Escolher um RTOS é uma tarefa importante para dar suporte à interrupções, temporizadores, comunicação entre tarefas, sincronização, gerenciamento de memória, múltiplo acesso, prioridade de execução e escalonamento de tarefas (BASKIYAR ; MEGHANATHAN, 2005).

Analizando o Apêndice A, pôde-se afirmar que a maioria das missões utilizavam RTOS, sendo os mais comuns o FreeRTOS e o Linux. Dentre esses, apenas o FreeRTOS possui suporte para o MSP432P4111 pois o Linux necessita de um *hardware* com maior velocidade de clock e maior memória RAM. Tendo esse cenário, o sistema operacional escolhido para o sistema foi o FreeRTOS e ele será explicado mais adiante.

- **Camada de Serviço do Sistema**

A Camada de Serviço do Sistema (CSS) é uma interface que o sistema operacional fornece à Camada de Aplicação Final. Usando essa interface, os aplicativos podem acessar vários serviços fornecidos pelo sistema operacional. Essa camada geralmente inclui o sistema de arquivos, o gerenciador de tarefas, temporizadores, etc (EBRARY,2018).

Nesse caso, o FreeRTOS já oferece APIs para que auxiliam o acesso de suas funcionalidades. Entretanto a CSS não estaria completa só com essas APIS, com isso será há a necessidade de implementar mais funcionalidades nessa camada, levando em consideração os requisitos da Tabela 12. Mais adiante, será explicado a proposta de implementação.

- **Camada de Aplicação**

A Camada de Aplicação Final (CAF) possui a maior hierarquia da Arquitetura em Camadas. Ela implementa as funcionalidades e tarefas do sistema. De uma forma geral, os níveis abaixo tem o objetivo de auxiliar a CAF (EBRARY, 2018).

Como essa camada realizará tarefas específicas da missão, a necessidade de requisitos da aplicação deverão ser listados. Com isso, a CAF será desenvolvida em versões futuras do *software* embarcado, implementando as funcionalidades exigidas pela missão.

4.3 DriverLib

Como mencionado na seção anterior, o *DriverLib* é um conjunto de APIs utilizado para controlar, configurar e manipular os periféricos do microcontrolador, MSP432P411. Além de deixar o código mais intuitivo, esse pacote auxilia a criação de um código de fácil portabilidade entre as plataformas da família MSP432 e MSP430.

Utilizar o *DriverLib* como camada de HAL é vantajoso em virtude de ser uma solução testada por profissionais e bem documentada. O Guia do Usuário do *DriverLib* apresenta um exemplo onde compara a configuração do *MasterClock* à nível de registradores, Código 4.1, e utilizando a API-*CS_initClockSignal()* do *DriverLib*, Código 4.2. Com esse exemplo fica evidente o grau de abstração e facilidade que o *DriverLib* oferece.

Como mencionado na seção anterior, o *DriverLib* é um conjunto de APIs utilizado para controlar, configurar e manipular os periféricos do microcontrolador, MSP432P411. Além de deixar o código mais intuitivo, esse pacote auxilia a criação de um código de fácil portabilidade entre as plataformas da família MSP432 e MSP430.

Código 4.1 – Configurando o MasterClock a nível de registrador

```

1 int main(void){
2 //...
3 CSKEY = 0x695A;
4 CSCTL |= SELM_1 | DIVM_2;
5 SKEY = 0;
6 //...
7 }
```

Código 4.2 – Configurando o MasterClock com a API do DriverLib

```

1 int main(void){
2 //...
3 CS_initClockSignal(CS_MCLK, CS_VLOCK_SELECT, CS_CLOCL_DIVIDER_32);
4 //...
5 }
```

A tabela abaixo mostra as vinte e cinco APIs do pacote, assim como uma breve descrição sobre cada API. Caso o leitor queira se aprofundar na leitura poderá consultar a referência (TEXAS INSTRUMENTSc, 2015).

Tabela 13 – Lista de todas as APIs disponíveis no pacote DriverLib.

API	Descrição	API	Descrição
ADC4	Permite o controlar os conversores Analógico Digital.	PMAP	Essa API permite configurar o modulo Port Mapping Controller. Esse modulo é responsável por reconfigurar as funções digitais de cada porta.
AES256	Permite a criptografia e descriptografia de dados de 128bits, de acordo com o padrão (AES256)	PSS	Permite a configuração das várias entradas de alimentação do microcontrolador, de modo a otimizar a eficiência energética.
COMP_E	Essa API fornece um conjunto de funções para inicializar os módulos COMP_E, de comparação de dois sinais de entrada analógicos.	REF_A	Permite configurar e ativar o uso da tensão de referência REF_A
CRC32	Permite fornecer um conjunto de funções para a verificação de dados. Essas funções são úteis quando há a necessidade de verificar a acurácia de um dado recebido em um canal de comunicação.	ResetCtl	Permite configurar e manipular as funções do reset do microcontrolador, tanto soft reset quanto hard reset.
CS	Permite controlar o sistema de clock do microcontrolador.	RTC_C	Essa API fornece um conjunto de funções para controlar o Real Time Clock (RTC_C).
DMA	Essa API permite controlar o Direct Memory Access (DMA) do microcontrolador, permitindo transferir blocos de dados sem a necessidade de utilizar o processamento do microcontrolador.	SPI	Essa API fornece o controle do barramento eUSCI_A/eUSCI_B, no modo SPI, permitindo a configuração da frequência de transmissão, envio/recebimento de dados, status, etc.
FlashCtl	Permite o controlar o processo de gravar, apagar e configurar a memória interna do processador.	SysCtl	Essa API junta os módulos do sistema que não se encaixam em nenhum periférico específico.
FPU	Essa API fornece métodos para manipular o comportamento da Unidade de Ponto Flutuante (FPU do inglês Floating-Point Unit) do processador Cortex-M.	SysTick	O SysTick é um temporizador simples que fornece uma interrupção periódica para RTOS, mas ele pode ser usado para outros fins de temporização.
GPIO	Permite configurar e ativar os pinos de entrada/saída do microcontrolador e configurar as interrupções.	Timer32	Permite a configuração do Timer32 (contador de 32 bits).
I2C	Essa API fornece o controle do barramento eUSCI_B, no modo I2C, permitindo a configuração da frequência de transmissão, envio/recebimento de dados, status, etc.	Timer_A	Essa API permite configurar o modulo TimerA. Esse modulo é um temporizador/contador de 16 bits, suportando múltiplos modos captura/comparação, PWM e temporização de intervalos.
NVIC	Permite controlar o Nested Vectored Interrupt Controller (NVIC). Esse modulo ativa, desativa, regista e configura as prioridades das interrupções do microcontrolador.	UART	Essa API fornece o controle do barramento serial USCI, permitindo a configuração da frequência de transmissão, envio/recebimento de dados, status, etc.
MPU	Essa API fornece funções para configurar o Memory Protection Unit (MPU). O MPU é acoplado ao núcleo do processador Cortex-M e fornece um meio de estabelecer permissões de acesso à regiões da memória.	WDT_A	Permite o controle do Watchdog padrão do sistema.
PCM	Permite o gerenciamento dos estados de energia do microcontrolador.		

4.4 FreeRTOS

O *FreeRTOS* é um kernel (gerenciador) utilizado em aplicações embarcadas que necessitam de aplicação em tempo real, sendo normalmente empregado em CubeSats. Esse kernel, desenvolvido e mantido pela Real Time Engineers Ltd, é distribuído gratuitamente sobre a licença *General Public License* (GPL) (BARRY, 2016). O *FreeRTOS* foi desenvolvido para ser pequeno, portátil e escalável; de acordo com o site oficial¹ o kernel possui uma imagem típica de 6K a 12K bytes.

O FreeRTOS é utilizado tanto em sistemas Não Críticos de Tempo Real (Soft-RTOS), quanto em sistemas Críticos de Tempo Real (Hard-RTOS). Aplicações Soft-RTOS possuem um tempo específico para a realização de uma tarefa, mas o não cumprimento do prazo de execução não causa uma falha no sistema. Por exemplo, caso as luzes internas do carro não acenderem no instante em que a porta for aberta, o usuário não estaria em risco. Já as aplicações Hard-RTOS, o não cumprimento do prazo resulta em falha do sistema. Por exemplo, se o sensor de colisão do carro atrasar a sua resposta, o usuário estaria em grande risco (BARRY, 2016).

No caso de aplicações aeroespaciais, o sistema deve ser projetado para ser um Hard-RTOS pois o travamento do *software* embarcado ocasionaria em falha total do sistema. Foi percebido que todas as missões pesquisadas, Apêndice B, possuíam algum tipo de sistema operacional para gerenciar as tarefas embarcadas.

No FreeRTOS, cada tarefa em execução é chamada de ‘task’. Para manter um padrão, será utilizada essa nomenclatura. No contexto do projeto, o uso das tasks é fundamental para criar um certo nível de abstração e garantir o requisito de Hard-RTOS. Por exemplo, a leitura da tensão da bateria é de extrema importância, então ela possuirá maior prioridade em relação a task de envio de dados; pois se o satélite consumir por completo a carga da bateria, ocasionando na perda do satélite .

Uma task só pode estar em quatro estados: *Ready*, *Running*, *Suspend* e *Blocked*. As APIs são encarregadas de alterar os estados de cada *task*. Os estados são melhores ilustrados na figura abaixo, onde há evidenciado as APIs responsáveis pela mudança.

4.4.1 APIs do FreeRTOS

As APIs do FreeRTOS foram criadas para facilitar o desenvolvimento, fazendo uma interface entre o usuário e o kernel. De acordo com o site oficial há 145 APIs que são agrupadas nas seguintes categorias:

- *Task Creation*, permite que o usuário crie ou elimine uma *task* do sistema.

¹ Disponível em <<https://www.freertos.org>>. Acesso em: 18/06/2018

Fonte: (LIN,2010, adaptado pág.20)

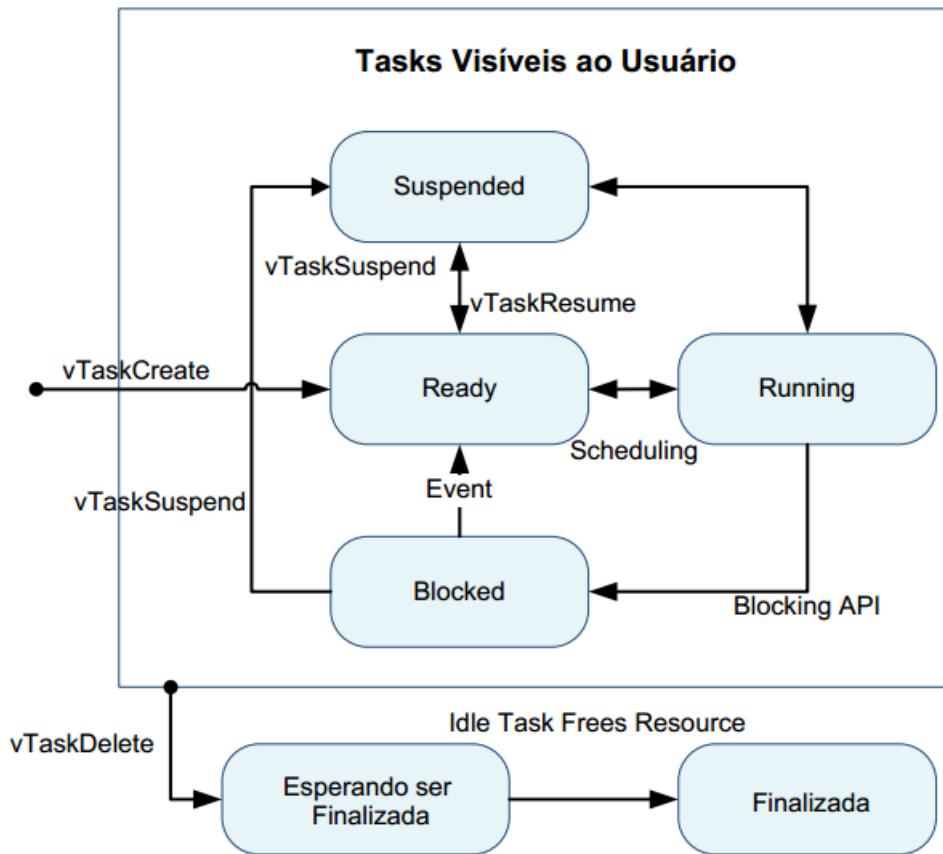


Figura 24 – Estados das Tasks no FreeRTOS

- *Task Control*, permite a troca dos estados das *task*.
- *Task Utilities*, possui APIs que permitem o usuário pegar alguma informação sobre as *task*, por exemplo, quais *task* estão em execução.
- *RTOS Kernel Control*, essa API oferece controle sobre o *kernel*, permitindo o usuário começar o *task scheduler*, suspender as *task* em execução, entre outros.
- *Direct To Task Notifications*, permite a troca de mensagens entre as *task*.
- *FreeRTOS-MPU Specific*, possui 3 APIs que permitem o usuário realizar o controle do *kernel* MPU.
- *Queues*, permite a comunicação entre as tarefas e interrupções a partir de uma fila.
- *Queue Sets*, permite o controle mais geral das filas, por exemplo, bloqueio de acesso das tarefas às filas.
- *Semaphore/Mutexes*, permite que usuário use semáforos binários para sincronização.
- *Software Timers*, permite a utilização de temporizadores, muito utilizado para contabilizar o tempo gasto entre as *task*.

- *Event Group*, essas APIs são flags que indicam se um evento ocorreu ou não, sendo muito utilizado para a gerar logs do sistema.
- *Co-routines*, permite a criação de rotinas, uma alternativa para tarefas com pouca prioridade.

Descrever cada API do *kernel* seria um tarefa cansativa e desnecessária, uma vez que há o manual de referência que explica cada uma e ainda mostra alguns exemplos. Com isso, abaixo há uma tabela que contém a descrição das funcionalidades das APIs das categorias *Task Creation* e *Task Control*. Essas categorias foram escolhidas para serem tratadas no texto, pois a manipulação das tarefas é a ação mais usual em um projeto com o FreeRTOS.

Abaixo há uma breve descrição sobre as APIs das categorias *Task Creation* e *Task Control*.

Tabela 14 – APIs da categoria Task Control e Task Creation.

Categoria	Nome da API	Funcionalidade
Task Creation	xTaskCreate()	Criação de uma task com sua prioridade. Se a prioridade for maior que a task em execução, a task criada começa sua execução. A API retorna um handler para a tarefa.
	xTaskCreateStatic()	Cria uma task, semelhante ao xTaskCreate. Entretanto, o tamanho destinado para a RAM é definido pelo usuário e é alocado estaticamente. A API retorna um handler para a tarefa.
	vTaskDelete()	Realiza a remoção da tarefa. Se a task estiver em execução, o kernel busca uma tarefa de maior prioridade para ser executada.
Task Control	vTaskDelay()	Coloca a task, que está em execução, no modo Blocked e aciona o timer para despertar.
	vTaskDelayUntil()	Coloca uma tarefa em execução no estado de Blocked e configura o tempo de resumo.
	uxTaskPriorityGet()	Retorna a prioridade da task a partir do seu handler.
	vTaskPrioritySet()	Define uma nova prioridade para a task.
	vTaskSuspend()	Coloca a task em modo Suspended e localiza outra tarefa com prioridade maior para ser executada.
	vTaskResume()	Coloca a tarefa suspensa para o estado de Ready.
	xTaskResumeFromISR()	Coloca a tarefa suspensa no estado de Ready quando o Scheduler estiver parado.
	xTaskAbortDelay()	Força uma tarefa a sair do modo Blocked e entrar no Ready

4.5 Desenvolvimento da Camada de Serviços do Sistema

A Linguagem de Modelagem Unificada (UML, do inglês *Unified Modeling Language*) foi escolhida para a estruturação da CSS. Essa ferramenta permite representar graficamente as características de um sistema, como: requisitos, especificações, estruturas, comportamentos, entre outros (BOOSH, 2005). Escolheu-se essa linguagem devido os seguintes fatores:

- rápido entendimento do *software* a partir do diagrama UML;
- fácil manutenção do *software*;
- portabilidade do *software* para várias plataformas e linguagens;
- linguagem de modelagem amplamente utilizada na engenharia de *software*.

Com as características acima, fica evidente que a UML ajuda na padronização, rápida compreensão e escalabilidade do sistema. Tais pontos são primordiais para versões futuras do *software* embarcado, pois os desenvolvedores não perderam tanto tempo para entender o sistema ou até mesmo tendo que reescrever o sistema novamente. Entretanto, alguns pontos devem ser levados em consideração antes de usar a UML para modelar *software* embarcado.

O uso da UML ocorre majoritariamente em *softwares* que possuem programação Orientada a Objetos (OO), pois essa modelagem foi baseada no paradigma de orientação a objetos. No contexto do projeto, o intuito é utilizar programação estrutural. Algumas extensões da UML, também chamados de perfis da UML, permitem implementar funcionalidades adicionais para um propósito específico.

De acordo com Douglass (2009), o *FunctionalC* é a extensão da UML que permite a modelagem de sistemas baseados na linguagem C. Os diagramas primários desta extensão são mostrados na Tabela 15.

Tabela 15 – Perfil do Diagrama FuncionalC.

Tipo de Diagrama	Diagrama FuncionalC	Diagrama UML	Descrição
Requisitos	Diagrama de Caso de Uso	Diagrama de Caso de Uso	Representa os usos do sistema com relação aos atores
Estrutura	Diagrama de Construção	Diagrama de Componente	Mostra o conjunto de artefatos construídos a partir dos arquivos de origem, como executáveis e bibliotecas
	Gráfico de Chamadas	Diagrama de Classes	Mostra as chamadas e suas sequências entre conjuntos de funções
	Diagrama de Arquivos	Diagrama de Classes	Mostra o conjunto de arquivos .c e .h e suas relações
Comportamento	Diagrama do Código	nenhum	Mostra o código-fonte gerado
	Diagrama de Mensagens	Diagrama Sequencial	Mostra sequências de chamadas e eventos enviados entre um conjunto de arquivos, incluindo valores de parâmetros passados
	Maquina de Estados	Diagrama de Estados	Mostra a máquina de estado para arquivos e como suas funções e ações incluídas são executadas à medida que eventos (síncronos ou assíncronos) são recebidos
	Fluxograma	Diagrama de Atividades	Detalha o fluxo de controle para uma função ou caso de uso

Fonte: (DOUGLASS, pag 3,2009).

Para ilustrar como é realizada a implementação de um diagrama UML em linguagem C, o considera-se um arquivo Timer, que tem o objetivo de atualizar o tempo. Ele possui como atributo os segundos e minutos, bem como métodos e operações, como: Reset() e Tick(). A função reset poderia inicializar as variáveis para zero, e a função tick poderia incrementar o tempo em um segundo. Em UML esse elemento seria representado conforme a Figura 25. Em C++ seria realizado uma classe, mas em C seria realizado uma **struct**, conforme mostra o Código 4.3.

Fonte: (DOUGLASS, 2009, pág.5).

<Timer>
+ mins : int
+ secs : int
- Tick(): void
- Reset(): void

Figura 25 – Exemplo Timer UML.

Código 4.3 – Transcrição do Timer UML para código em C.

```

1 extern int mins;
2 extern int secs;
3
4 /*## operation Reset() */
5 void Reset();
6
7 /*## operation tick() */
8 void tick();
9
10}

```

Fonte: (DOUGLASS, 2009, pág.5).

• Desenvolvimento da UML

Analizando os requisitos do sistema, tabela 12, foi possível abstrair as funções principais exigidas para o software: Controle dos Subsistemas, Controle das Tarefas e Armazenamento dos dados. Essas funcionalidades foram transformadas em três pacotes: **ControleTarefas** (Controle das Tarefas), **Sistema** (Controle dos Subsistemas) e **ArmazenamentoDados** (Armazenamento de dados). A UML do CSS é mostrado na Figura 26.

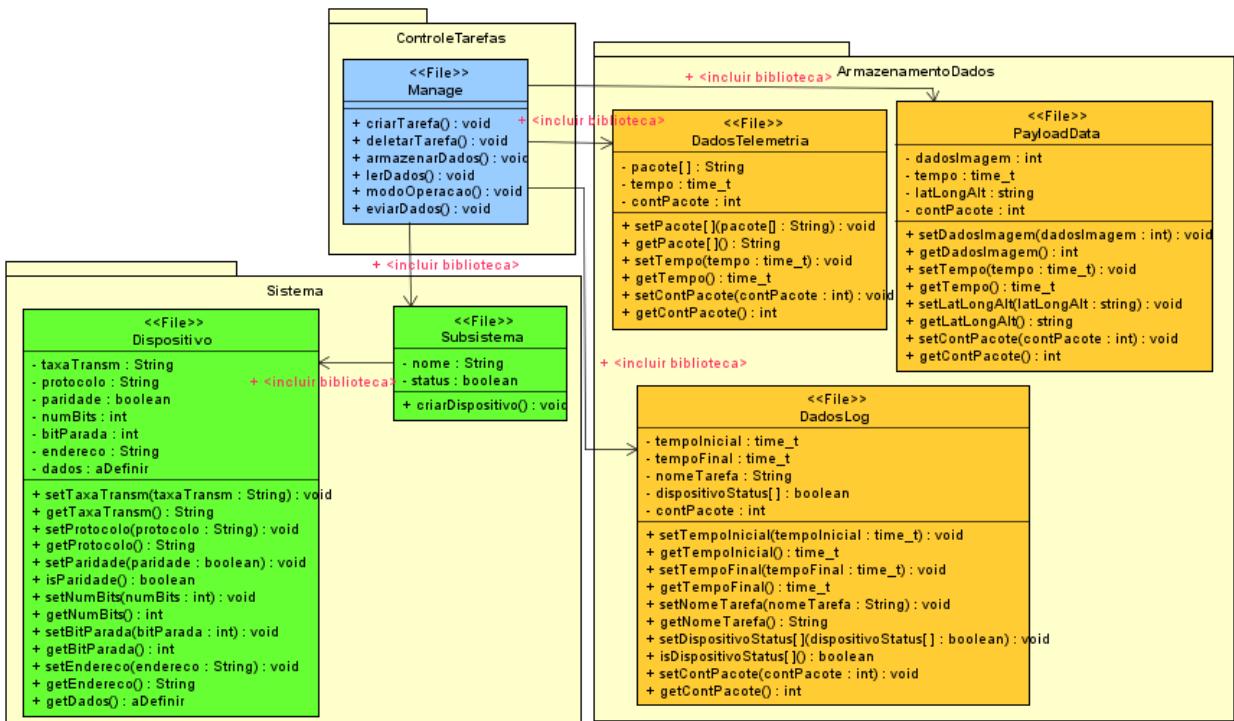


Figura 26 – UML da Camada de Serviço do Sistema.

5 RESULTADOS PRELIMINARES

Para o desenvolvimento do *hardware*, quase todos os componentes foram definidos. Em relação às interfaces, há a necessidade de definir os pinos que serão conectados ao barramento ISA. A arquitetura prévia do OBC pode ser vista na Figura 28. Os pontos de interrogação serão resolvidos após a definição das interfaces.

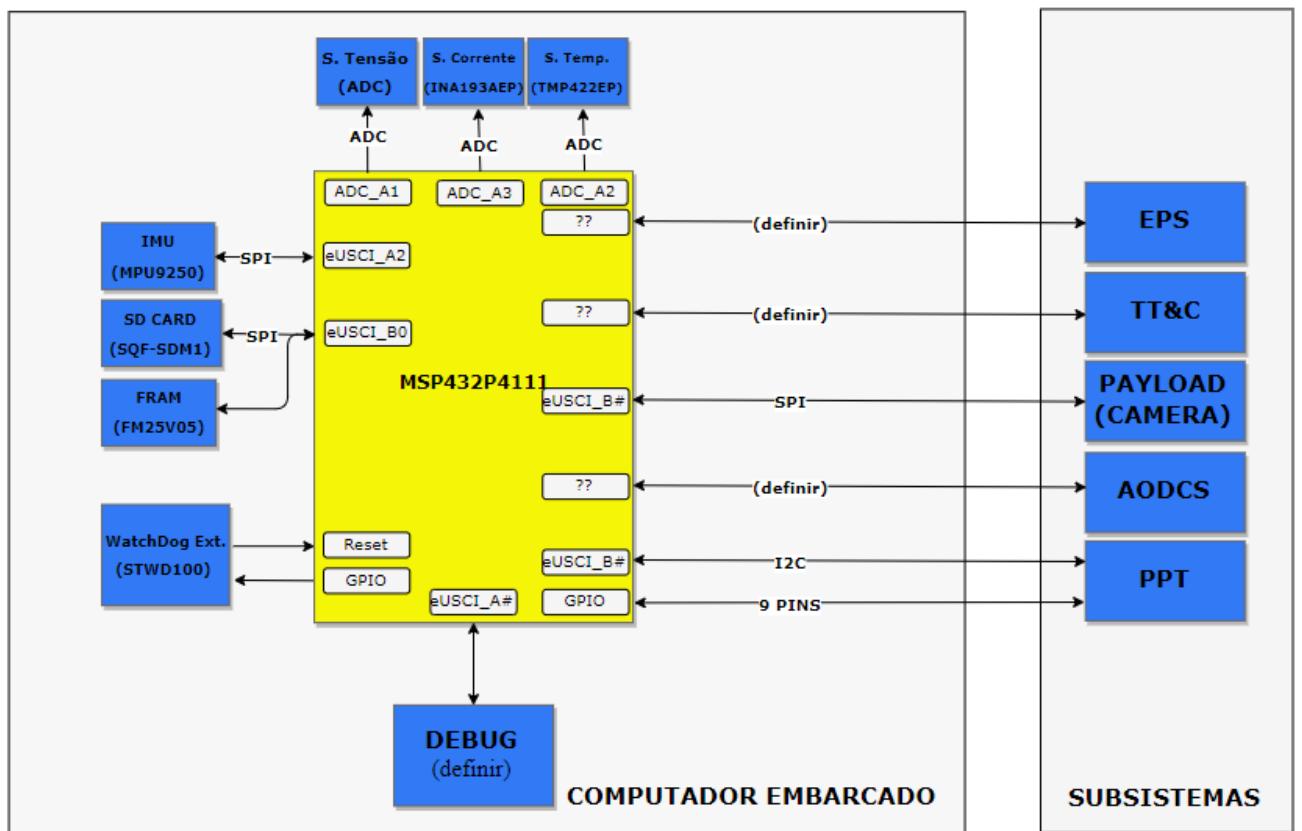


Figura 27 – Arquitetura prévia do OBC.

Em relação aos esquemáticos, alguns componentes já foram desenhados no KiCad, como o microprocessador, sensor inercial e o *Watchdog Externo*. A Figura 28 mostra a visão geral do esquemático prévio do sistema. A resolução está pequena, caso o leitor queira olhar com mais detalhe, no Apêndice E há imagens de cada componente do esquemático.

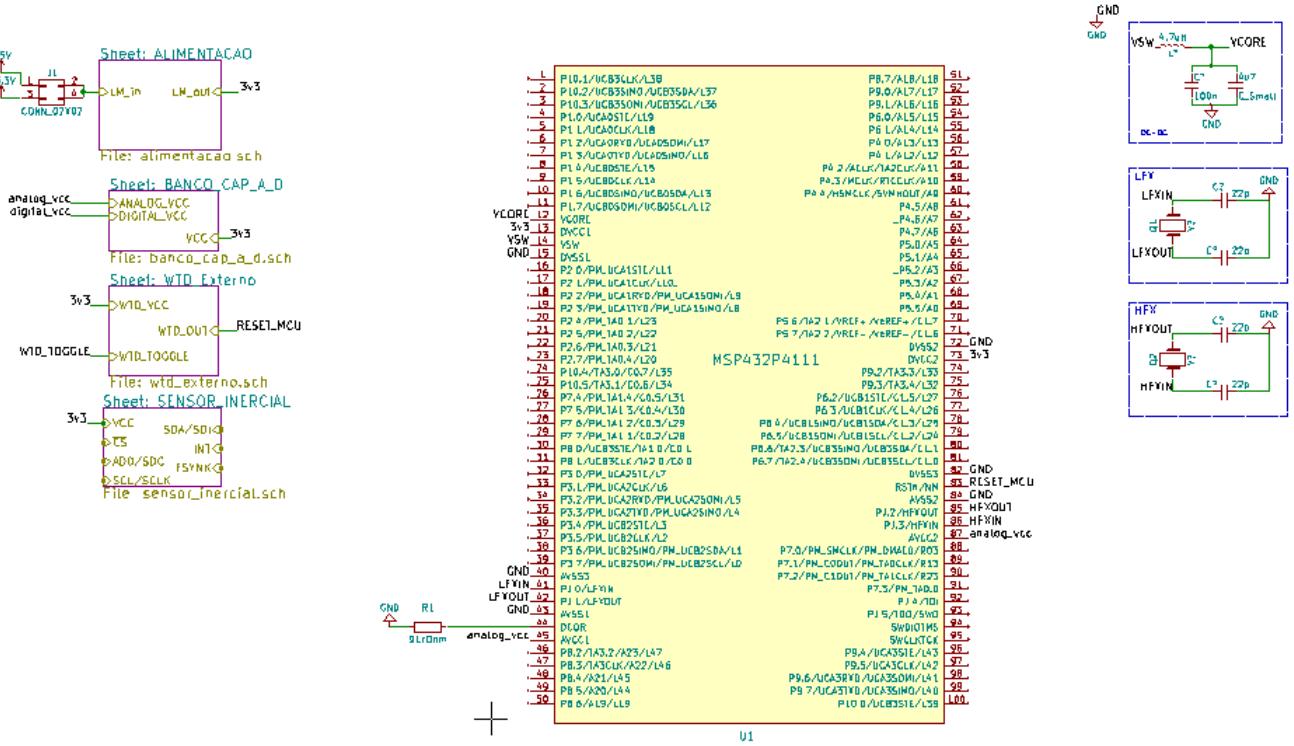


Figura 28 – Esquemático prévio do OBC.

Não foi possível cumprir o cronograma previsto na Seção 1.3, devido a alguns imprevistos. Alguns fatores não foram cogitados durante a elaboração do cronograma, como: o atraso de alguns componentes comprados no exterior, atraso do desenvolvimento do software devido a falta de experiência com o FreeRTOS, entre outros. Para recuperar o tempo perdido, o autor utilizará o período das férias para realizar as atividades ausentes.

6 TRABALHOS FUTUROS

O objetivo do TCC1 era realizar o projeto teórico do OBC para que o TCC2 fosse dedicado à elaboração do *layout* da PCB e implementação do *software* embarcado. Para evitar que haja atrasos no projeto, o autor realizará as atividades restantes no período das férias.

Apesar dos atrasos, a realização do projeto até o TCC2 é plausível. Apenas alguns pontos, como *shielding* metálico e códigos corretores, não sejam implementados devido o tempo escasso. O cronograma será mantido, pois o início do TCC2 ocorre no primeiro dia de aula (13/08/2018), restando um mês para completar as atividades restantes.

7 REFERÊNCIAS

ADDAIM, Adnane ; KHERRAS , Abdelhaq ; ZANTOU, El Bachir. **Design of Low-cost Telecommunications CubeSat-class Spacecraft.** Centre For Space Research And Studies, EMI, Marocos: [s.n.], 2010. 6 p. Disponível em: <goo.gl/ciGc2d>. Acesso em: 05 abr. 2018.

ARM MICROCONTROLLERS. **Processors Cortex-m Series.** 2018. Disponível em: <<https://www.arm.com/products/processors/cortex-m>>. Acesso em: 18 jun. 2018.

BACELO , Ana Paula Terra . **Arquitetura de Software:** conceitos e tendências. PU-CRS: [s.n.], 2010. 38 p. Disponível em: <https://www.inf.pucrs.br/jornada.facin/jafacin_2010/palestras/ArquiteturaDeSoftware.pdf>. Acesso em: 06 jun. 2018.

BAI, Ying . **Microcontroller Engineering with MSP432:** Fundamentals and Applications. [S.l.]: Taylor & Francis Group, And Informa Business, 2016. 817 p.

BARNHART, David J. **Very Small Satellite Design for Space Sensor Networks.** United Kingdom: Faculty Of Engineering And Physical Sciences - Faculty Of Engineering And Physical Sciences, 2008. 233 p. Disponível em: <<http://www.dtic.mil/cgi/tr/fulltext/u2/a486188.pdf>>. Acesso em: 31 mar. 2018.

BARRY, Richard. **Mastering the FreeRTOS™ Real Time Kernel:** A Hands-On Tutorial Guide. [S.l.]: Real Time Engineers Ltd., 2016. 399 p. Disponível em: <https://www.freertos.org/Documentation/161204_Mastering_the_FreeRTOS_Real_Time_Kernel-A_Hands-On_Tutorial_Guide.pdf>. Acesso em: 17 maio 2018.

BARRY, Richard. **Mastering the FreeRTOS™ Real Time Kernel:** A Hands-On Tutorial Guide. [S.l.]: Real Time Engineers Ltd., 2016. 399 p. Disponível em: <https://www.freertos.org/Documentation/161204_Mastering_the_FreeRTOS_Real_Time_Kernel-A_Hands-On_Tutorial_Guide.pdf>. Acesso em: 17 maio 2018.

BASKIYAR , S. ; MEGHANATHAN, N. . **A Survey of Contemporary Real-time Operating Systems** . Auburn University: [s.n.], 2005. 8 p. Disponível em: <<http://www.eng.auburn.edu/~baskiyar/MyArticles/Survey-of-RTOS-Informatica.pdf>>. Acesso em: 06 jun. 2018.

BOTMA, Pieter Johannes . **The Design and Development of an ADCS OBC for a CubeSat.** 2011. 114 p. Mestrado (Mestrado em Engenharia) - Faculdade de Engenharia, Universidade de Stellenbosch, África do Sul, 2011. Disponível em: <https://scholar.sun.ac.za/bitstream/handle/10019.1/18040/botma_design_2011.pdf?sequence=2&isAllowed=y>. Acesso em: 10 maio 2018.

- CLYDE SPACE. **High-Precision Attitude Determination and Control System (ADCS).** 2018. Disponível em: <goo.gl/rALuMR>. Acesso em: 03 jun. 2018.
- CUBESAT PROGRAM. **CubeSat Design Specification Rev. 13.** California: California Polytechnic, 2014. 42 p. Disponível em: <https://static1.squarespace.com/static/5418c831e4b0fa4ecac1bacd/t/56e9b62337013b6c063a655a/1458157095454/cds_rev13_final2.pdf>. Acesso em: 31 mar. 2018.
- CUBESTAR. **Electronic Power System.** 2018. Disponível em: <http://cubestar.no/index.php?p=1_19_Electronic-Power-System>. Acesso em: 03 jun. 2018. CZERNIK, Sylwia. **Design of the Thermal Control System for Compass-1.** 2004. 84 p. Tese (Graduação em Ciências Aplicadas)- University of Applied Sciences Aachen, Alemanha, 2004. Disponível em: <http://www.crn2.inpe.br/conasat1/projetos_cubesat/subsistemas/TCS/COMPASS-1%20-%20TCS%20-%20Design%20of%20the%20Thermal%20Control%20System.pdf>. Acesso em: 17 jun. 2018.
- DEEPAK, Ravi A.; TWIGGS, Robert J. **Thinking Out of the Box: Space Science Beyond the CubeSat.** 1. ed. Virginia - US: Journal Of Small Satellites, 2012. 3 e 4 p. v. 1. Disponível em: <<http://www.jossonline.com/wp-content/uploads/2014/12/0101-Thinking-Outside-the-Box-Space-Science-Beyond-the-CubeSat.pdf>>. Acesso em: 25 mar. 2018.
- DENIS, Amandine et al. **QB50 - System Requirements and Recommendations.** [S.l.: s.n.], 2015. 59 p. Disponível em: <https://www.qb50.eu/index.php/tech-docs/category/QB50_Systems_Requirements_issue_76e8e.pdf?download=89:qb50-docs>. Acesso em: 06 jun. 2018.
- DOUGLASS, Bruce Powel. **UML for the C programming language..** Estados Unidos: IBM Corporation, 2009. 12 p. Disponível em: <<http://c3328005.r5.cf0.rackcdn.com/08a0b2bb-a705-4dc7-a058-6a20fcc9e3a0.pdf>>. Acesso em: 10 jun. 2018.
- EBRARY. **Typical Software Architecture.** 2018. Disponível em: <https://ebrary.net/22045/computer_science/typical_software_architecture>. Acesso em: 06 jun. 2018.
- EICKHOFF, Jens . **An Introduction to Onboard Computers, Onboard Software and Satellite Operations.** 1. ed. [S.l.]: Springer-Verlag Berlin Heidelberg, 2012. 282 p. v. 1.
- ENDUROSAT. **CUBESAT OBC.** 2018b. Disponível em: <<https://www.endurosat.com/products/cubesat-onboard-computer-obc/>>. Acesso em: 03 jun. 2018.
- ENDUROSAT. **CUBESAT UHF ANTENNA.** 2018a. Disponível em: <<https://www.endurosat.com/products/cubesat-uhf-antenna/>>. Acesso em: 03 jun. 2018.
- FACCHINETTI, Giovanni et al. **SMALL SATELLITES: Economic Trends.** Italy: [s.n.], 2016. 102 p. Disponível em: <<http://www.defencesa.com/upload/Facchinetti%20G>>.

%20Small%20Satellites%20Economic%20Trends%20Dec%202016-FINAL.pdf>. Acesso em: 02 jun. 2018.

FINCKENOR, Miria M. ; GROH, Kim K. **Space Environmental Effects**. Washington: NASA ISS Program Science Office, 2015. 40 p. Disponível em: <https://www.nasa.gov/sites/default/files/files/NP-2015-03-015-JSC_Space_Environment-ISS-Mini-Book-2015-508.pdf>. Acesso em: 08 abr. 2018.

FIORAVANTI, Carlos . **Uma escola em órbita:** Professores e estudantes constroem satélite no litoral paulista. São Paulo: FAPESP, 2011. 1 p. Disponível em: <<http://revistapesquisa.fapesp.br/wp-content/uploads/2012/05/040-041-180.pdf>>. Acesso em: 25 mar. 2018.

FRIEDEL, Jonas; MCKIBBON, Sean. **Thermal Analysis of the CubeSat CP3 Satellite**. San Luis Obispo, CA: [s.n.], 2011. 23 p. Disponível em: <<http://digitalcommons.calpoly.edu/cgi/viewcontent.cgi?article=1054&context=aerosp>>. Acesso em: 17 jun. 2018.

FROST, Chad; AGASID, Elwood. **Small Spacecraft Technology State of the Art**. California: NASA Ames Research Center, 2015. 17 p. Disponível em: <https://www.nasa.gov/sites/default/files/atoms/files/small_spacecraft_technology_state_of_the_art_2015_tagged.pdf>. Acesso em: 25 mar. 2018.

GLOBALSTAR, L.P. EMPRESA. **Description of the Globalstar System**. California: [s.n.], 2000. 47 p. Disponível em: <<https://gsproductsupport.files.wordpress.com/2009/04/description-of-the-globalstar-system-gs-tr-94-0001-rev-e-2000-12-07.pdf>>. Acesso em: 31 mar. 2018.

GRIFFITHS, Ian Michael. **Location techniques for pico- and femto-satellites, with applications for space weather monitoring**. 2017. 142 p. Thesis (Doctor of Philosophy)- University of Leicester, England, 2017. Disponível em: <<https://lra.le.ac.uk/bitstream/2381/39973/1/2017GriffithsIMPhD.pdf>>. Acesso em: 31 mar. 2018.

INVENSENSE. MPU-9250 Product Specification Revision 1.1. 2018. Disponível em: <<https://www.invensense.com/wp-content/uploads/2015/02/PS-MPU-9250A-01-v1.1.pdf>>. Acesso em: 18 jun. 2018.

ISIS. **Innovative Solutions In Space**. 2016. Disponível em: <<https://www.isispace.nl/cubesats/>>. Acesso em: 31 nov. 2018.

JANES, Michael . **The PC/104 Technology In Embedded System Design**. Canada: [s.n.], 2006. 22 p. Disponível em: <<http://nparc.nrc-cnrc.gc.ca/eng/view/fulltext/?id=22caabb7-7f63-42b1-96fa-47bcd465d87b>>. Acesso em: 08 abr. 2018.

LIN, Yuhui. **Formal Analysis of FreeRTOS**. 2010. 177 p. Tese (Mestrado em Engenharia de Software)- University of York, Departamento de Ciência da Computação, Estados Unidos, 2010. Disponível em: <<http://goo.gl/R397j8>>. Acesso em: 18 jun. 2018.

LUMBWE, LWABANJI TONY. **Development of an onboard computer (OBC) for a CubeSat.** Bellville: Faculty Of Engineering At The Cape Peninsula University Of Technology, 2013. 178 p. Disponível em: <http://etd.cput.ac.za/bitstream/handle/20.500.11838/1172/Lumbwe_T_Final2013.pdf?sequence=1&isAllowed=y>. Acesso em: 06 abr. 2018.

MABROUK, Elizabeth . **What are SmallSats and CubeSats?**. Disponível em: <<https://www.nasa.gov/content/what-are-smallsats-and-cubesats>>. Acesso em: 25 mar. 2018.

MASUTTI, Davide et al. **QB50 - System Requirements and Recommendations.** [S.l.: s.n.], 2014. 53 p. Disponível em: <https://www.qb50.eu/index.php/tech-docs/category/QB50_system_requirements_issue_606e0.pdf?download=58:qb50-docs>. Acesso em: 06 jun. 2018.

NANO AVIONICS. **Standard Structure.** 2018. Disponível em: <<https://n-avionics.com/cubesat-components/structures-and-deployable-mechanisms/cubesat-structure/>>. Acesso em: 03 jun. 2018.

NASA. **Space Radiation Effects on Electronic Components in Low-Earth Orbit.** 1999. Disponível em: <<http://llis.nasa.gov:80/lesson/824>>. Acesso em: 03 fev. 2018.

NASA. **SPACE RADIATION EFFECTS ON ELECTRONIC COMPONENTS IN LOW-EARTH ORBIT.** Johnson Space Center (JSC).: [s.n.], 1996. 7 p. Disponível em: <<https://pdfs.semanticscholar.org/a13e/52893d0fa3d08b2ce9e03b0b7e9592848a4f.pdf>>. Acesso em: 26 maio 2018.

PC/104 Embedded Consortium (a). **PC/104 Embedded Consortium.** 2.6. ed. [S.l.: s.n.], 2008. 25 p. Disponível em: <https://pc104.org/wp-content/uploads/2015/02/PC104_Spec_v2_6.pdf>. Acesso em: 06 abr. 2018.

PC/104 Embedded Consortium (b). **PC/104-Plus Specification - Version 2.3.** [S.l.: s.n.], 2008. 33 p. Disponível em: <https://pc104.org/wp-content/uploads/2015/02/PC104_Plus_v2_32.pdf>. Acesso em: 10 maio. 2018.

PETKOV, Mihail P. **The Effects of Space Environments on Electronic Components.** Pasadena, CA, United States: Jet Propulsion Lab.; California Inst. Of Tech., 2003. 36 p. Disponível em: <<https://trs.jpl.nasa.gov/bitstream/handle/2014/7193/03-0863.pdf?sequence=1&isAllowed=y>>. Acesso em: 26 maio 2018.

PUMPKIN. **CubeSat Kit FM430 Flight Module.** C. rev. San Francisco: [s.n.], 2008. 14 p. Disponível em: <http://www.cubesatkit.com/docs/datasheet/DS_CSKit_FM430_710-00252-C.pdf>. Acesso em: 20 abr. 2018.

RAZZAGHI, Elyas. **Design and Qualification of On-Board Computer for Aalto-1 CubeSat.** 2012. 77 p. Master degree (Master of Science Space Engineering)- Department

of Computer Science, Electrical and Space Engineering, Luleå University of Technology, Sweden, 2012. Disponível em: <<http://www.diva-portal.org/smash/get/diva2:1022951/FULLTEXT02.pdf>>. Acesso em: 10 jun. 2018.

SELVA, Daniel; KREJCI, David. **A survey and assessment of the capabilities of Cubesats for Earth observation.** A: Acta Astronautica, 2012. 50-68 p. Disponível em: <<https://www.sciencedirect.com/science/article/pii/S0094576511003742>>. Acesso em: 11 jun. 2018.

STRAS, Luke et al. **The Design and Operation of The Canadian Advanced Nanospace eXperiment (CanX-1).** Canada: University Of Toronto Institute For Aerospace Studies, 2003. 11 p. Disponível em: <<http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.557.7343&rep=rep1&type=pdf>>. Acesso em: 02 jun. 2018.

TEXAS INSTRUMENTS. (d) CURRENT SHUNT MONITORS –16-V to 80-V COMMON MODE RANGE. 2018. Disponível em: <<http://www.ti.com/lit/ds/symlink/ina193a-ep.pdf>>. Acesso em: 18 jun. 2018

TEXAS INSTRUMENTS (a) . **MSP432P411x, MSP432P401x SimpleLink™ Mixed-Signal Microcontrollers.** 2018. 214 p. Disponível em: <<http://www.ti.com/lit/ds/symlink/msp432p4111.pdf>>. Acesso em: 17 maio 2018.

TEXAS INSTRUMENTS (b). **1.5C ACCURATE PROGRAMMABLE DIGITAL TEMPERATURE SENSORS WITH SPI™ INTERFACE.** USA: , 2008. 17 p. Disponível em: <<http://www.ti.com/lit/ds/symlink/tmp122-ep.pdf>>. Acesso em: 04 jun. 2018.

TEXAS INSTRUMENTS(c). **MSP432® Peripheral Driver Library: User’s Guide.** 1. ed. Estados Unidos: 2015. 387 p. Disponível em: <https://e2e.ti.com/cfs-file/_key/communityserver-discussions-components-files/166/MSP432_5F00_DriverLib_5F00_Users_5F00_Guide_2D00_MSP432P4xx_2D00_2_5F00_20_5F00_00_5F00_08.pdf>. Acesso em: 18 jun. 2018.

TEXAS INSTRUMENTS (d). **MSP Driver Library.** 2018. Disponível em: <<http://www.ti.com/tool/MSPDRIVERLIB>>. Acesso em: 06 jun. 2018.

TWIGGS, Robert. Origin of CubeSat. In: HELVAJIAN, HENRY ; JANSON, SIEGFRIED W. **SMALL SATELLITES: PAST, PRESENT, AND FUTURE.** 1. ed. El Segundo, CA: He Aerospace Corporation, 2008. p. 151-153. v. 1.

WEKERLE, Timo ; FILHO, José Bezerra Pessoa . **Status and Trends of Smallsats and Their Launch Vehicles — An Up-to-date Review.** São Paulo: Departamento de Ciência e Tecnologia Aeroespacial - Instituto Tecnológico de Aeronáutica - Divisão de Engenharia Aeronáutica e Mecânica, 2017. 18 p. v. 3. Disponível em: <<http://www.scielo.br/pdf/jatm/v9n3/2175-9146-jatm-09-03-0269.pdf>>. Acesso em: 31 mar. 2018.

WILEY J., Larson; RICHARD WERTZ, James. **Space Mission Analysis and Design. Second edition.** 3. ed. Estados Unidos: Microcosm, 1992. 301–352 p. Disponível em: <<http://deseng.ryerson.ca/~fil/I/Papers/DamianPapers/SMAD.pdf>>. Acesso em: 18 jun. 2018.

Apêndices

APÊNDICE A – Análise das Missões Anteriores

Analisou-se algumas missões CubeSat, com o intuito de extrair informações que pudessem ser úteis no desenvolvimento do OBC. Para a seleção, tentou-se buscar missões que tiveram êxito em sua operação e que foram destinadas à observação da Terra. A tabela abaixo mostra as características mais marcantes dos OBCs e de suas missões.

Tabela 16 – Análise de OBCs em Missões Passadas

Nome	Missão				Microcontrolador												Informações Adicionais		
	Informações Gerais				Especificações Gerais				Conversores		Interface e Portas								
	Nome	Tipo de Missão	Unidade	TRL	Microcontrolador	Consumo	Barramento de Dados [bit]	Frequência [MHz]	AD	DA	I/O	Interface Serial	I2C	Modos de Operação	Sistema Operacional	Watchdog Timer	Faixa de Temperatura [°C]		
Hawksat-1	Demostrador Tecnológico	1U	9	MSP430F1612	12mW @ 8MHz	16	8	8	2	48	2 USART, 2 SPI,	1	2	não	sim	-40 a 80			
SwissCube	Observação da Terra	1U	9	AT91M55800A ARM	140mW @ 33MHz	32	33	8	2	58	3 USART, 2 SPI,	0	5	não	sim	-40 a 85			
Aalto-1	Observação da Terra	1U	9	AT91RM9200 ARM	80mW @ 80MHz	32	180	-	-	122	3 USART, 1 SPI,	0	2	Linux	sim	-40 a +85			
GOMX-1	Demonstrador Tecnológico	2U	9	Nanomind A712D AT91M55800A	293,7mW @ 40MHz	32	40	6	2	50	USART, SPI,	1	5	Linux	sim	-40 a +85			
OpenOBC	x	x	5/4	TMS570LS0432ARM Cortex-R4	405mW @ 80MHz	32	80	16	0	45	1 USART, 2 SPI,	2	2	FreeRTOS	sim	-40 a +85			
FloripaSat	Projeto Universitário	1U	5/4	MSP430F6659IPZ	0,7mW @ 8MHz	16	16	16	0	74	3 USART, 3 SPI,	3	8	FreeRTOS	sim	-40 a +85			
AAUSAT	Observação da Terra	3U	9	C161PI	150mW @ 25MHz	16	25	4	0	76	2 USART, 2 SPI,	2	2	RTX166	sim	-40 a +85			
ATMOCUBE	Observação da Terra	1U	6	PIC 16F877	1,6mW @ 4MHz	8	20	8	0	14	1 USART, 1 MSSP	0	2	PIC 16F877	sim	-55 a +125			
DTUSat	Demonstrador Tecnológico	1U	9	Atmel AT91M40800	150mW @ 40MHz	32	40	0	0	32	2 USART	0	Frequência de operação variável		não	sim	-40 a +85		
CP2	Demonstrador Tecnológico Estudo das Instabilidades do Plasma	1U	9	PIC18F6720	63mW @ 25MHz	16	25	12	0	52	2 USART, 3 SPI	1	Frequência de operação variável		não	sim	-40 a +85		
RAX-1	Instabilidades do Plasma	1U	9	MSP430F1612	12mW @ 8MHz	16	8	8	2	48	2 USART, 2 SPI,	1	2	não	sim	-40 a 80			

A partir do gráfico acima foi possível afirmar os seguintes pontos:

1. A maioria dos microcontroladores eram baseados em processadores de arquitetura ARM.
2. Missões 1U possuíam velocidade de clock reduzida. Isso porque a energia a bordo disponível não permite a uso de velocidade de processamento alta.
3. Em relação às entradas e periféricos, todos os computadores de bordo possuem pinos para I/O que podem ser programados de acordo com a necessidade da missão, deixando a design muito mais versátil. Sobre a quantidade de periféricos, todos possuem interfaces serial.
4. 55% dos OBCs analisados utilizavam um sistema Operacional de Tempo Real (RTOS, do inglês Real Time Operating System), sendo o Linux e o FreeRTOS os mais utilizados.
5. Na maioria dos casos, a temperatura de operação do OBC era delimitada pela faixa de tolerância dos componentes comerciais (COTS, Cost Of The Shelf).

6. 80% das missões analisadas eram 1U.
7. A porcentagem de microcontroladores com 16 e 32-bit é a mesma, 54,5%.
8. Em média, o consumo pela frequência de clock é de 2,916mW/MHz.

APÊNDICE B – Requisitos do OBC

Como a missão ainda está em fase de discussão, o escopo não foi totalmente delimitado, consequentemente, poucos requisitos foram definidos. Com o intuito de reduzir o escopo e oferecer insumos durante o projeto do OBC, buscou-se a documentação de requisitos das missões CubeSat já lançadas. Foi possível achar duas missões que possuíam uma documentação sistematizada, que foram o QB50 (DENIS et al., 2015) e o Aalto-1 (RAZZAGHI, 2012).

Os requisitos do QB50 e Aalto-1, específicos ao computador de bordo, foram adicionados aos requisitos prévios do projeto, e são mostrados na tabela 17. Nessa tabela, a divisão dos requisitos ocorre da seguinte maneira: [1] são os da própria missão, [2] QB50 e [3] Aalto-1.

Tabela 17 – Requisitos do OBC.

Número do Requisito	Descrição do Requisito
OBC-R1	O OBC deve controlar uma câmera CMOS e armazenar as imagens provenientes desse dispositivo em uma memória não volátil [1]
OBC-R2	O OBC deve armazenar um arquivo que contenha a geolocalização e tempo de captura das imagens [1]
OBC-R3	O OBC deve controlar um PPT (do inglês Pulsed Pulsed Plasma Thruster) [1]
OBC-R4	O OBC deve gerenciar todas as Payloads transportadas pelo CubeSat [3]
OBC-R5	O OBC deve possuir interface com todos os subsistemas do CubeSat [2][3]
OBC-R6	O OBC deve controlar todas as atividades embarcadas, exceto o controle de atitude [3]
OBC-R7	O OBC deve ler os dados de cada subsistema a cada 1 segundo [2][3]
OBC-R8	O OBC deve armazenar o tempo, modo de operação do satélite, tensão e corrente do EPS, temperatura do TT&C e EPS [2]
OBC-R9	O OBC deve possuir um sensor inercial e armazenar os dados provenientes desse componente [1]
OBC-R10	OBC deve armazenar um registro de eventos [2]
OBC-R11	O OBC deve ter uma referência temporal com precisão de 500ms, para o armazenamento dos dados. Os tempos relativos devem ser contados e armazenados de acordo com a referência de 01.01.2000 00:00:00 UTC [2]
OBC-R12	OBC deve enviar os dados armazenados quando o satélite entrar em uma janela de transmissão e, ao mesmo tempo, decodificar e processar telecomandos enviados pela estação terrestre [2][3]
OBC-R13	O software embarcado deve checar telecomandos indesejados, dados e mensagens inconsistência, rejeitando entradas ilegais [2]
OBC-R14	Deve ser implementado um comando que permite a limpeza da memória não volátil do OBC [2]
OBC-R15	OBC deve ser possuir técnicas de atualização de software e capacidade de Boot Loader [3]
OBC-R16	OBC deve possuir todas as tarefas das missão [3]
OBC-R17	O software embarcado deve proteger-se contra loops infinitos não intencionais, erros computacionais e possíveis travamentos [2]
OBC-R18	O software embarcado deve possuir um Sistema de Operação em Tempo Real (do inglês Real-Time Operating System) oferecendo opções de prioridade de tarefas [3]
OBC-R19	O OBC deve ser projetado para durar mais que dois anos [3]
OBC-R20	O OBC deve ser projeto para ser o mais versátil possível [1]

APÊNDICE C – Modos de Operação

Tabela 18 – Modos de Operação e Consumo do MSP432P4111

MODOS DE OPERAÇÃO	DESCRIÇÃO	FREQUÊNCIA	POTENCIA[mW]
AM_LDO_VCORE0	Modo ativo baseado em LDO, desempenho médio, nível de tensão do núcleo 0	0 - 24MHz	13,6
LPM0_LDO_VCORE0	O mesmo que AM_LDO_VCORE0, exceto que a CPU está desligada (sem execução do código)	0 - 24MHz	3,36
AM_LDO_VCORE1	Modo ativo baseado em LDO, desempenho máximo, nível de tensão do núcleo 1	0 - 48MHz	25,28
LPM0_LDO_VCORE1	O mesmo que AM_LDO_VCORE1, exceto que a CPU está desligada (sem execução de código)	0 - 48MHz	5,12
AM_DCDC_VCORE0	Modo ativo baseado em DC / DC, desempenho médio, nível de tensão do núcleo 0	0 - 24MHz	8,48
LPM0_DCDC_VCORE0	O mesmo que AM_DCDC_VCORE0, exceto que a CPU está desligada (sem execução de código)	0 - 24MHz	2,72
AM_DCDC_VCORE1	Modo ativo baseado em DC / DC, desempenho máximo, nível de tensão do núcleo 1	0 - 48MHz	9,28
LPM0_DCDC_VCORE1	O mesmo que AM_DCDC_VCORE1, exceto que a CPU está desligada (sem execução de código)	0 - 48MHz	3,84
AM_LF_VCORE0	Modo ativo de baixa frequência baseado em LDO, nível de tensão do núcleo 0	0 - 128KHz	0,96
LPM0_LF_VCORE0	O mesmo que AM_LF_VCORE0, exceto que a CPU está desligada (sem execução de código)	0 - 128KHz	0,64
AM_LF_VCORE1	Modo ativo de baixa frequência baseado em LDO, nível de tensão do núcleo 1	0 - 128KHz	1,6
LPM0_LF_VCORE1	O mesmo que AM_LF_VCORE1, exceto que a CPU está desligada (sem execução de código)	0 - 128KHz	0,8
LPM3_VCORE0	Modo de baixa potência baseado em LDO com retenção de estado total, nível de tensão do núcleo 0. Além de RTC_C e WDT_A, outros periféricos podem estar operacionais com um clock externo ou interno de baixa frequência até 128 kHz.	0 - 128KHz	0,0256
LPM3_VCORE1	Modo de baixa potência baseado em LDO com retenção de estado total, nível de tensão do núcleo 1. Além de RTC_C e WDT_A, outros periféricos podem estar operacionais com um clock externo ou interno de baixa frequência até 128 kHz.	0 - 128KHz	0,01552
LPM4_VCORE0	Modo de baixa potência baseado em LDO com retenção de estado total, nível de tensão do núcleo 0. Os periféricos podem ser operados a partir de clocks externos de até 128 kHz.	0 - 128KHz	0,00992
LPM4_VCORE1	Modo de baixa potência baseado em LDO com retenção de estado total, nível de tensão do núcleo 1. Os periféricos podem ser operados a partir de clocks externos de até 128 kHz.	0 - 128KHz	0,01296
LPM3.5	Modo de baixa potência baseado em LDO, nível de tensão do núcleo 0, sem retenção de registros periféricos, RTC_C e WDT_A podem estar ativos	0 - 32.768KHz	0,01184
LPM4.5	Tensão do núcleo desligada, ativação somente através de reset de pino ou I/O com capacidade de ativação	-	0,0007328

APÊNDICE D – Estimativa de Armazenamento de Dados

Tabela 19 – EPS - SID 97 (0x61)

Description	Bits	Units
COM last report time	32	s
ADCS last report time	32	s
CDMS last report time	32	s
Payload last report time	32	s
Battery 1 voltage	8	V
Battery 1 redundancy voltage	8	V
Battery 2 voltage	8	V
Battery 2 redundancy voltage	8	V
Battery 1 temperature	8	°C
Battery 2 temperature	8	°C
Digital power bus voltage	8	V
Analog power bus voltage	8	V
External temperature	8	°C
Frame temperature	8	°C
Microcontroller temperature	8	°C
Board temperature	8	°C
Motherboard temperature	8	°C
Solar cell -X current	8	A
Solar cell +X current	8	A
Solar cell -Y current	8	A
Solar cell +Y current	8	A
Solar cell -Z current	8	A
Solar cell +Z current	8	A
Face -X temperature	8	°C
Face +X temperature	8	°C
Face -Y temperature	8	°C
Face +Y temperature	8	°C
Face -Z temperature	8	°C
Face +Z temperature	8	°C
Payload enable/disable	1	
ADCS enable/disable	1	
ADS 1/2 status	1	
Payload status	1	
ADCS status	1	
CDMS status	1	
Beacon status	1	
COM status	1	
Payload error flag	1	
ADCS error flag	1	
CDMS error flag	1	
COM error flag	1	
EPS error flag	1	
Spare 2 bits (not used)	2	
Spacecraft mode	1	
Error code	8	
Software watchdog timeout	8	ms
TOTAL	360	bits
	45	bytes

Tabela 20 – EPS Min/Max - SID 81 (0x51)

Description	Bits	Units
Battery 1 temperature minimum	8	°C
Battery 1 temperature maximum	8	°C
Battery 2 temperature minimum	8	°C
Battery 2 temperature maximum	8	°C
External temperature minimum	8	°C
External temperature maximum	8	°C
Frame temperature minimum	8	°C
Frame temperature maximum	8	°C
Microcontroller temperature minimum	8	°C
Microcontroller temperature maximum	8	°C
Board temperature minimum	8	°C
Board temperature maximum	8	°C
Motherboard temperature minimum	8	°C
Motherboard temperature maximum	8	°C
Face -X temperature minimum	8	°C
Face -X temperature maximum	8	°C
Face +X temperature minimum	8	°C
Face +X temperature maximum	8	°C
Face -Y temperature minimum	8	°C
Face -Y temperature maximum	8	°C
Face +Y temperature minimum	8	°C
Face +Y temperature maximum	8	°C
Face -Z temperature minimum	8	°C
Face -Z temperature maximum	8	°C
Face +Z temperature minimum	8	°C
Face +Z temperature maximum	8	°C
Solar cell -X current minimum	8	A
Solar cell -X current maximum	8	A
Solar cell +X current minimum	8	A
Solar cell +X current maximum	8	A
Solar cell -Y current minimum	8	A
Solar cell -Y current maximum	8	A
Solar cell +Y current minimum	8	A
Solar cell +Y current maximum	8	A
Solar cell -Z current minimum	8	A
Solar cell -Z current maximum	8	A
Solar cell +Z current minimum	8	A
Solar cell +Z current maximum	8	A
Battery 1 voltage minimum	8	V
Battery 1 voltage maximum	8	V
Battery 2 voltage minimum	8	V
Battery 2 voltage maximum	8	V
Digital power bus voltage minimum	8	V
Digital power bus voltage maximum	8	V
TOTAL		352 bits
		44 bytes

Tabela 22 – COM - SID 65 (0x41)

Description	Bits	Units
Microcontroller temperature	8	°C
Board temperature	8	°C
Beacon board temperature	8	°C
Maximum length of telemetry frames I-Field	8	bytes
Number of flags between two frames transmission	8	flags
Timeout of virtual channel 1 (real-time acks)	8	s
Timeout of virtual channel 2 (archived acks)	8	s
Timeout of virtual channel 4 (payload data)	8	s
Timeout of virtual channel 6 (archived HK)	8	s
Timeout of virtual channel 7 (real-time HK)	8	s
General timeout of reception	8	s
General timeout of transmission	8	s
TX DAC low value	12	V
TX DAC high value	12	V
TOTAL	120	bits
	15	bytes

Tabela 23 – Payload - SID 113 (0x71)

Description	Bits	Units
Detector temperature	8	°C
Microcontroller temperature	8	°C
Board temperature	8	°C
Current mode of the camera	1	
Read/write error of internal registers of the detector	1	
Image present in SRAM and ready to be transmitted	1	
Spare 1 bit (not used)	1	
Current program location being executed	4	
TOTAL	32	bits
	4	bytes

Tabela 24 – ADCS - SID 17 (0x11)

Description	Bits	Units
Sun Sensor Face X-, Angle A1 Measurement	12	mV
Sun Sensor Face X-, Reference R1 Measurement	12	mV
Sun Sensor Face X-, Angle A2 Measurement	12	mV
Sun Sensor Face X-, Reference R2 Measurement	12	mV
Sun Sensor Face X+, Angle A1 Measurement	12	mV
Sun Sensor Face X+, Reference R1 Measurement	12	mV
Sun Sensor Face X+, Angle A2 Measurement	12	mV
Sun Sensor Face X+, Reference R2 Measurement	12	mV
Sun Sensor Face Y-, Angle A1 Measurement	12	mV
Sun Sensor Face Y-, Reference R1 Measurement	12	mV
Sun Sensor Face Y-, Angle A2 Measurement	12	mV
Sun Sensor Face Y-, Reference R2 Measurement	12	mV
Sun Sensor Face Y+, Angle A1 Measurement	12	mV
Sun Sensor Face Y+, Reference R1 Measurement	12	mV
Sun Sensor Face Y+, Angle A2 Measurement	12	mV
Sun Sensor Face Y+, Reference R2 Measurement	12	mV
Sun Sensor Face Z-, Angle A1 Measurement	12	mV
Sun Sensor Face Z-, Reference R1 Measurement	12	mV
Sun Sensor Face Z-, Angle A2 Measurement	12	mV
Sun Sensor Face Z-, Reference R2 Measurement	12	mV
Sun Sensor Face Z+, Angle A1 Measurement	12	mV
Sun Sensor Face Z+, Reference R1 Measurement	12	mV
Sun Sensor Face Z+, Angle A2 Measurement	12	mV
Sun Sensor Face Z+, Reference R2 Measurement	12	mV
Spare 1 bit (not used)	1	
Gyroscope X On/Off Flag	1	
Gyroscope Y On/Off Flag	1	
Gyroscope Z On/Off Flag	1	
Sun Sensor X- On/Off Flag	1	
Sun Sensor X+ On/Off Flag	1	
Sun Sensor Y- On/Off Flag	1	
Sun Sensor Y+ On/Off Flag	1	
Sun Sensor Z- On/Off Flag	1	
Sun Sensor Z+ On/Off Flag	1	
Magnetotorquer X Current Sign	1	
Magnetotorquer Y Current Sign	1	
Magnetotorquer Z Current Sign	1	
Magnetotorquer X On/Off Flag	1	
Magnetotorquer Y On/Off Flag	1	
Magnetotorquer Z On/Off Flag (LSB)	1	
Magnetometer X Measurement	16	uT
Magnetometer Y Measurement	16	uT
Magnetometer Z Measurement	16	uT
Gyroscope X Measurement	16	mrad/s
Gyroscope Y Measurement	16	mrad/s
Gyroscope Z Measurement	16	mrad/s
Bdot Gain	16	
Bdot Lambda	16	
Bdot Rotation Speed of Command	16	rad/s
Magnetotorquer X Offset	8	uA
Magnetotorquer Y Offset	8	uA
Magnetotorquer Z Offset	8	uA

Tabela 26 – EPS archive - temperatures - SID 98 (0x62)

Description	Bits	Units
Battery 1 temperature	8	°C
Battery 2 temperature	8	°C
External temperature	8	°C
Frame temperature	8	°C
Board temperature	8	°C
Motherboard temperature	8	°C
Face -X temperature	8	°C
Face +X temperature	8	°C
Face -Y temperature	8	°C
Face +Y temperature	8	°C
Face -Z temperature	8	°C
Face +Z temperature	8	°C
TOTAL	96	bits
	12	bytes

Tabela 27 – My caption

Description	Bits	Units
Solar cell -X current	8	A
Solar cell +X current	8	A
Solar cell -Y current	8	A
Solar cell +Y current	8	A
Solar cell -Z current	8	A
Solar cell +Z current	8	A
TOTAL	48	bits
	6	bytes

Tabela 28 – EPS archive - voltages - SID 100 (0x64)

Description	Bits	Units
Battery 1 voltage	8	V
Battery 2 voltage	8	V
Digital power bus voltage	8	V
Analog power bus voltage	8	V
TOTAL	32	bits
	4	bytes

Tabela 29 – PAYLOAD - Image Sensor

Description	Bits	Units
Horário de Captura	32	s
Lat/Long	32	°
Pixels	921600	JPG
TOTAL	921664	bits
	115208	bytes

Tabela 30 – Quantidade total de dados.

TELEMETRIA		PAYLOAD	
210	bytes	115208	bytes
105	bytes/s	1920,1	bytes/s
9072000	Bytes/dia	165899520	Bytes/dia
9,072	MBytes/dia	165,89952	MBytes/dia
10 MBytes/dia		166 MBytes/dia	
TOTAL: 176 MBytes/dia			

APÊNDICE E – Esquemático Eletrônico

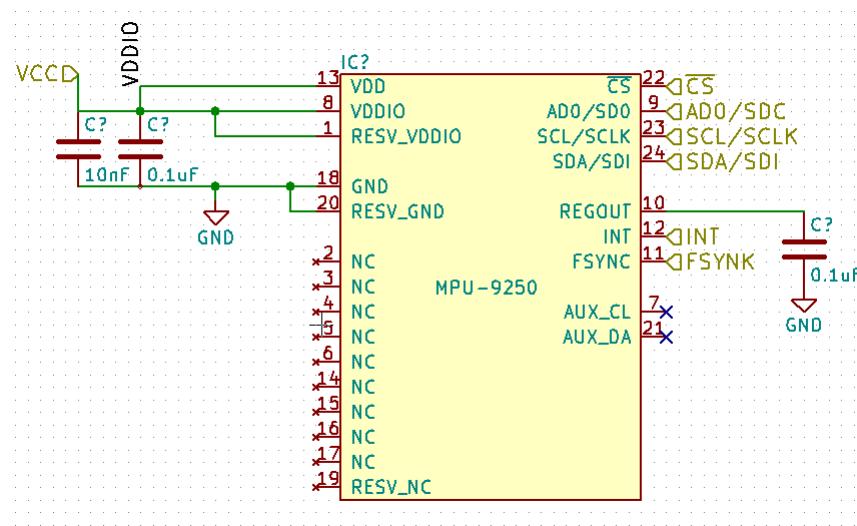


Figura 29 – Esquemático do MPU9250.

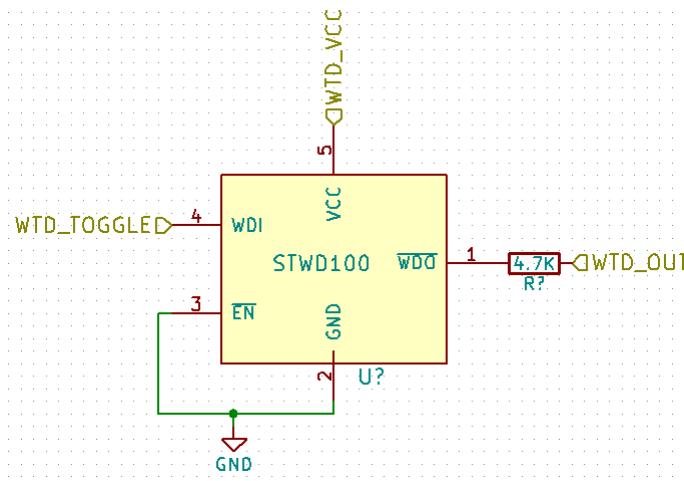


Figura 30 – Esquemático do STWD100.