



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

## **Desenvolvimento de um Computador de Bordo para Pequenos Satélites**

**Autor:** Guilherme Silva Lionço  
**Orientador:** Prof. Dr. Leonardo Aguayo

**Brasília, DF**  
**2019**





Guilherme Silva Lionço

## **Desenvolvimento 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. Leonardo Aguayo

Coorientador: Prof. Dr. Giancarlo Santilli

Brasília, DF

2019



---

Guilherme Silva Lionço

Desenvolvimento de um Computador de Bordo para Pequenos Satélites/ Guilherme Silva Lionço. – Brasília, DF, 2019-

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

Orientador: Prof. Dr. Leonardo Aguayo

Trabalho de Conclusão de Curso – Universidade de Brasília – UnB  
Faculdade UnB Gama – FGA , 2019.

1. Palavra-chave01. 2. Palavra-chave02. I. Prof. Dr. Leonardo Aguayo. II. Universidade de Brasília. III. Faculdade UnB Gama. IV. Desenvolvimento de um Computador de Bordo para Pequenos Satélites

CDU 02:141:005.6

---

Guilherme Silva Lionço

## **Desenvolvimento 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, 15 de julho de 2019 – Data da aprovação do trabalho:

---

**Prof. Dr. Leonardo Aguayo**  
Orientador

---

**Prof. Dr. Giancarlo Santilli**  
Coorientador

---

**Prof. Dr. William Reis Silva**  
Convidado

Brasília, DF  
2019

### **Dedicatória.**

*Dedico este trabalho primeiramente à Deus, que me guiou durante toda minha jornada.  
Aos meus pais, irmãos e à toda minha família que, com muito carinho e apoio, não  
mediram esforços para que eu chegasse até esta etapa de minha vida. Aos meus amigos,  
que tornaram a graduação mais agradável e divertida.*



# Agradecimentos

Agradeço à equipe da Percepio, que forneceram uma licença para a utilização do Tracealyzer. Agradeço aos professores Leonardo Aguayo e Giancarlo Santilli por acreditarem e incentivarem meu projeto de graduação. E por fim, agradeço minha irmã, Juliana Lionço, por reler inúmeras versões dessa monografia.



*“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 microcontrolador e outros dispositivos para compor o *hardware* do OBC. Já para o *software* embarcado foi definido como sistema operacional o FreeRTOS. Durante os testes em protoboard foi possível verificar: o consumo do microcontrolador; modos de operação do software embarcado; a aquisição e armazenamento de dados; etc. Alguns requisitos estabelecidos no projeto não foram cumpridos devido à complexidade do projeto. Os pontos que não foram desenvolvidos ou aprofundados durante a pesquisa foram levantados e estão presentes na última seção.

**Palavras-chave:** OBC. CubeSat. Computador de Bordo. Nano Satélites.



# Abstract

Researchers from the University of Brasília (UnB) and the Polish Aviation Institute (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 is aimed at the construction of an Onboard Computer (OBC) for this future mission. During the development of the OBC, it was used the co-design methodology, which allowed for the development of hardware and software at the same time. During the design of the theoretical project, it was chosen the microcontroller and another devices to compose the OBC's *hardware*. For the embedded software, the FreeRTOS operating system was defined as the operating system. During the protoboard test, it was possible to verify: the consumption of the microcontroller; modes of operation of the embedded software; the acquisition and data storage; etc. Some requirements established at the beginning were not fulfilled due to the complexity of the project. Points that were not developed during the search were raised and are present in the last section.

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



# **Lista de ilustrações**

Figura 1 – Categoria de Pequenos Satélites e alguns exemplos. . . . .	23
Figura 2 – Lançamento de pequenos satélites entre 1995 e 2014. . . . .	24
Figura 3 – Algumas variações de CubeSat. . . . .	25
Figura 4 – Especificações Estruturais para um CubeSat 3U+. . . . .	26
Figura 5 – Arquitetura de um Satélite. . . . .	28
Figura 6 – Estrutura 3U Padrão NanoAvionics. . . . .	28
Figura 7 – Principais componentes de um EPS. . . . .	29
Figura 8 – Componentes AODCS da empresa Clyde Space. . . . .	29
Figura 9 – Componentes TT&C da empresa EnduroSat. . . . .	30
Figura 10 – Computador de bordo da empresa EnduroSat. . . . .	30
Figura 11 – Arquitetura do OBDH do FloripaSat. . . . .	32
Figura 12 – Arquitetura do OpenOBC. . . . .	33
Figura 13 – Placa Eletrônica - iOBC. . . . .	33
Figura 14 – Arquitetura do FM430. . . . .	34
Figura 15 – PCB Padrão PC104 8-bits. . . . .	36
Figura 16 – Classificação dos efeitos da Radiação Espacial. . . . .	37
Figura 17 – Performance dos processadores CORTEX-M. . . . .	44
Figura 18 – Arquitetura do microcontrolador MSP432P4111. . . . .	45
Figura 19 – Esquemático Eletrônico de uma aplicação usual do ACS711. . . . .	49
Figura 20 – Esquemático Eletrônico de uma aplicação usual do MPU9250. . . . .	49
Figura 21 – Esquemático Eletrônico de uma aplicação usual do MCP9700. . . . .	50
Figura 22 – Dimensões do módulo PC/104-Plus em polegadas e milímetros. . . . .	51
Figura 23 – Esquemático Eletrônico do OBC. . . . .	53
Figura 24 – PCB - Vista Isométrica. . . . .	54
Figura 25 – Arquitetura de abstração em Camadas. . . . .	56
Figura 26 – Estados das Tasks no FreeRTOS . . . . .	60
Figura 27 – Exemplo Timer UML. . . . .	64
Figura 28 – Maquina de Estados da Camada de Serviço. . . . .	65
Figura 29 – Fluxograma de Inicialização do sistema. . . . .	66
Figura 30 – Diagrama de Arquivos do Sistema. . . . .	67
Figura 31 – Protoboard com os componentes COTS e a LaunchPad. . . . .	69
Figura 32 – Formato do Pacote de Telemetria. . . . .	70
Figura 33 – Dados da telemetria armazenados no cartão de memoria. . . . .	70
Figura 34 – Comparação entre os Modos de hibernação (azul) e Nominal (amarelo). . . . .	71
Figura 35 – Comparação entre os Modos de hibernação (azul) e Pouca Bateria (amarelo). . . . .	72

Figura 36 – Snapshoot no Tracealyzer durante a inicialização do sistema. . . . .	73
Figura 37 – Snapshoot no Tracealyzer alguns segundos após a inicialização do sistema. . . . .	73
Figura 38 – Snapshoot no Tracealyzer do sistema executando NM_MODE. . . . .	74
Figura 39 – Snapshoot no Tracealyzer do sistema executando BLLM1_MODE. . . . .	74
Figura 40 – Snapshoot no Tracealyzer do sistema executando HB_MODE. . . . .	75
Figura 41 – Snapshoot no Tracealyzer do sistema executando em todos os modos. . . . .	75
Figura 42 – Procedimento para simulação de travamento do <i>TT&amp;CTask</i> . . . . .	76
Figura 43 – Event Log no Tracealyzer do <i>WatchDogTask</i> sendo reinicializado. . . . .	77
Figura 44 – Grafico da CPU no Tracealyzer do <i>WatchDogTask</i> sendo reinicializado. . . . .	77
Figura 45 – Valores de FLASH e RAM alocados para o <i>FreeRTOS</i> . . . . .	78
Figura 46 – Valores de FLASH e RAM alocados para o <i>software</i> embarcado. . . . .	78
Figura 47 – Vista superior e inferior da PCB. . . . .	79
Figura 48 – Esquemático Eletrônico do OBC. . . . .	105
Figura 49 – Esquemático Regulador de Tensão. . . . .	106
Figura 50 – Esquemático Banco de Capacitores. . . . .	107
Figura 51 – Esquemático Watchdog Externo. . . . .	108
Figura 52 – Esquemático Sensor de Corrent. . . . .	109
Figura 53 – Esquemático Armazenamento de Dados. . . . .	110
Figura 54 – Esquemático Sensor Inercial. . . . .	111
Figura 55 – Esquemático Medidor de Tensão. . . . .	112
Figura 56 – Esquemático JTAG. . . . .	113
Figura 57 – Esquemático Sensor de Temperatura. . . . .	114
Figura 58 – PCB - Vista Isométrica. . . . .	114
Figura 59 – PCB - Plano de Cobre Superior [3V3]. . . . .	115
Figura 60 – PCB - Plano de Cobre Inferior [GND]. . . . .	115
Figura 61 – <i>Bill of Material</i> do OBC . . . . .	117
Figura 62 – Fluxograma de Inicialização. . . . .	119
Figura 63 – Fluxograma de Inicialização do Hardware. . . . .	119
Figura 64 – Fluxograma de Inicialização do Clock. . . . .	120
Figura 65 – Fluxograma de criação das <i>Tasks</i> . . . . .	120
Figura 66 – Fluxograma do <i>Watchdog Task</i> . . . . .	121
Figura 67 – Fluxograma do <i>DataStorage Task</i> . . . . .	121
Figura 68 – Fluxograma do <i>HouseKeeping Task</i> . . . . .	121

# Listas de tabelas

Tabela 1 – Tabela comparativa dos OBCs.	35
Tabela 2 – Efeitos Acumulativos causada pela radiação espacial.	38
Tabela 3 – Efeitos de Evento Único causados pela radiação espacial.	38
Tabela 4 – Requisitos do Hardware.	41
Tabela 5 – Microcontroladores selecionados.	43
Tabela 6 – Pontuação de cada microcontrolador.	43
Tabela 7 – Comparação dos tipos de Memória.	47
Tabela 8 – Especificação das memórias da Unidade de Armazenamento.	48
Tabela 9 – Informações técnicas do ACS711.	48
Tabela 10 – Informações técnicas sobre sensor MPU9250.	49
Tabela 11 – Informações técnicas sobre sensor MCP9701T.	50
Tabela 12 – Pinos JTAG no MSP432P4111.	51
Tabela 13 – Requisitos do <i>software</i> embarcado.	55
Tabela 14 – Lista de todas as APIs disponíveis no pacote DriverLib.	59
Tabela 15 – APIs da categoria <i>Task Creation</i> e <i>Task Control</i> .	62
Tabela 16 – Configurações realizadas nos arquivos de <i>porting</i>	63
Tabela 17 – Perfil do Diagrama FunctionalC.	64
Tabela 18 – Análise de OBCs em Missões Passadas	93
Tabela 19 – Requisitos do OBC.	96
Tabela 20 – Modos de Operação e Consumo do MSP432P4111	97
Tabela 21 – EPS - SID 97 (0x61)	99
Tabela 22 – EPS Min/Max - SID 81 (0x51)	100
Tabela 24 – COM - SID 65 (0x41)	101
Tabela 25 – Payload - SID 113 (0x71)	101
Tabela 26 – ADCS - SID 17 (0x11)	102
Tabela 28 – EPS archive - temperatures - SID 98 (0x62)	103
Tabela 29 – My caption	103
Tabela 30 – EPS archive - voltages - SID 100 (0x64)	103
Tabela 31 – PAYLOAD - Image Sensor	104
Tabela 32 – Quantidade total de dados.	104



# Sumário

<b>1</b>	<b>INTRODUÇÃO</b>	<b>21</b>
1.1	Contexto e Justificativa	21
1.2	Objetivos	21
1.3	Metodologia	22
<b>2</b>	<b>REFERENCIAL TEÓRICO</b>	<b>23</b>
2.1	Pequenos Satélites	23
2.2	Padrão CubeSat	25
2.3	Principais Subsistemas de um CubeSat	27
2.4	Visão Geral de um OBC	31
2.4.1	Computadores de Bordo Existentes	31
2.5	Padrão PC/104	35
2.6	Ambiente Espacial	36
2.6.1	Radiação Espacial	37
2.6.2	Categoria de Componentes	38
<b>3</b>	<b>DESENVOLVIMENTO DO HARDWARE</b>	<b>41</b>
3.1	Requisitos e Funcionalidades	41
3.2	Análise do Microcontrolador	42
3.3	Microcontrolador MSP432P4111	44
3.3.1	Unidade de Memória	45
3.3.2	Portas e Periféricos	46
3.3.3	Consumo e Modos de Operação	46
3.4	Unidade de Armazenamento	46
3.5	Periféricos	48
3.5.1	Sensor de Corrente	48
3.5.2	Sensor Inercial	48
3.5.3	Sensor de Temperatura	50
3.6	Interface de Programação/ <i>Debug</i>	50
3.7	Dimensões da Placa	51
3.8	Sistema para Mitigar Falhas	52
3.8.1	Watchdog Externo	52
3.9	Placa de Circuito Impresso	53
<b>4</b>	<b>DESENVOLVIMENTO DO SOFTWARE</b>	<b>55</b>
4.1	Requisitos e Funcionalidades	55

4.2	<b>Arquitetura do Software</b>	56
4.3	<b>DriverLib</b>	58
4.4	<b>FreeRTOS</b>	59
4.4.1	APIs do FreeRTOS	61
4.4.2	Configuração do FreeRTOS	62
4.5	<b>Desenvolvimento da Camada de Serviços do Sistema</b>	63
4.5.1	Desenvolvimento da UML	65
4.6	<b>Desenvolvimento do Código</b>	67
5	<b>RESULTADOS E DISCUSSÃO</b>	69
5.1	<b>Aquisição e Armazenamento de Dados</b>	69
5.2	<b>Consumo</b>	70
5.3	<b>Modos de Operação</b>	72
5.4	<b>Sistema Antitravamento</b>	76
5.5	<b>Alocação de Memória do FreeRTOS e da CSS</b>	78
5.6	<b>Printed Circuit Board</b>	79
6	<b>CONCLUSÃO E RECOMENDAÇÕES</b>	81
6.1	<b>Conclusão</b>	81
6.2	<b>Recomendações</b>	82
7	<b>REFERÊNCIAS</b>	85
	<b>APÊNDICES</b>	91
	<b>APÊNDICE A – ANÁLISE DAS MISSÕES ANTERIORES</b>	93
	<b>APÊNDICE B – REQUISITOS DO OBC</b>	95
	<b>APÊNDICE C – MODOS DE OPERAÇÃO</b>	97
	<b>APÊNDICE D – ESTIMATIVA DE ARMAZENAMENTO DE DADOS</b>	99
	<b>APÊNDICE E – ESQUEMÁTICO ELETRÔNICO</b>	105
	<b>APÊNDICE F – BILL OF MATERIAL</b>	117
	<b>APÊNDICE G – FLUXOGRAMAS DA CAMADA DE SERVIÇO</b>	119

# 1 Introdução

## 1.1 Contexto e Justificativa

O Instituto de Aviação Varsóvia (ILOT, do polonês *Instytut Lotnictwa*) e a Universidade de Brasília (UNB) decidiram firmar uma cooperação tecnológica, 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.

Com o propósito de oferecer uma base para o desenvolvimento do OBC, nos próximos capítulos será abordado um pouco melhor o contexto dos pequenos satélites e sua importância para o ambiente acadêmico.

## 1.2 Objetivos

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

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

### 1.3 Metodologia

Inicialmente foi realizada uma pesquisa bibliográfica para reunir informações que dariam base ao projeto. Nessa fase foi analisado: ambiente espacial; pequenos satélite, mais especificamente os CubeSats; OBCs (*hardware* e *software*) de missões passadas e disponíveis à venda. Após a pesquisa bibliográfica, foi realizado o levantamento dos requisitos da missão com o intuito de definir as funcionalidades e delimitar o escopo da pesquisa.

Logo em seguida, foi feito o projeto preliminar do sistema. Para a parte de *hardware*, foi escolhido: o microcontrolador, sensores, memória; entre outros. Já para o *software*, escolheu-se: 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, foram realizados testes em *protoboard*, visando a validação de alguns subsistemas separadamente. Nessa fase, os códigos para cada subsistema foram testados e, em seguida, unidos em um único código.

Por fim, após a validação do sistema em *protoboard*, realizou-se o projeto final do sistema. Para a parte de *hardware*, foram realizados os esquemáticos, *layout* da PCB (do inglês *Printed Circuit Board*) e o BOM (do inglês *Bill of Materials*). Ja para a parte de *software*, foi feito a arquitetura do sistema, diagrama UML (do inglês *Unified Modeling Language*), fluxograma das rotinas e código embarcado.

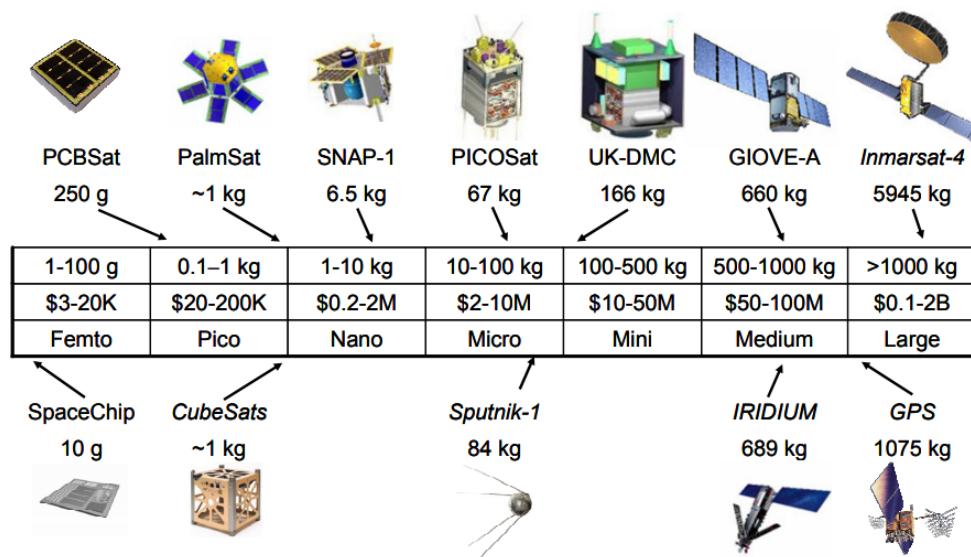
## 2 Referencial Teórico

Esta seção visa oferecer uma os 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, serão expostos 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.

**Figura 1** – Categoria de Pequenos Satélites e alguns exemplos.



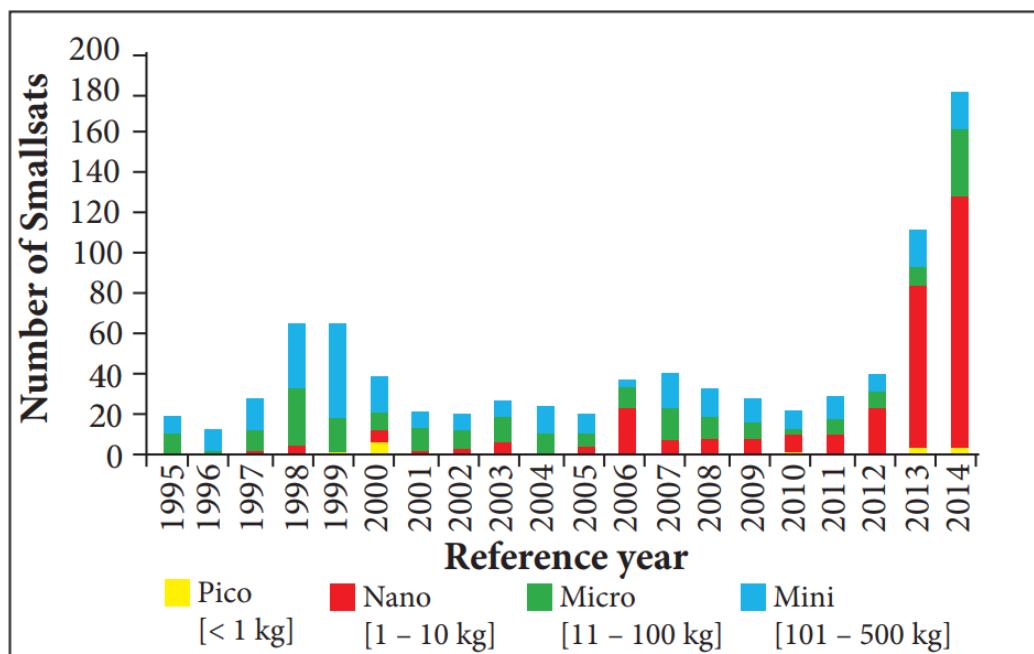
Fonte:(BARNHART, 2008, pág.42).

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 450kg

(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 2, 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).

**Figura 2** – Lançamento de pequenos satélites entre 1995 e 2014.



Fonte:(WEKERLE; FILHO, 2017, pág.3).

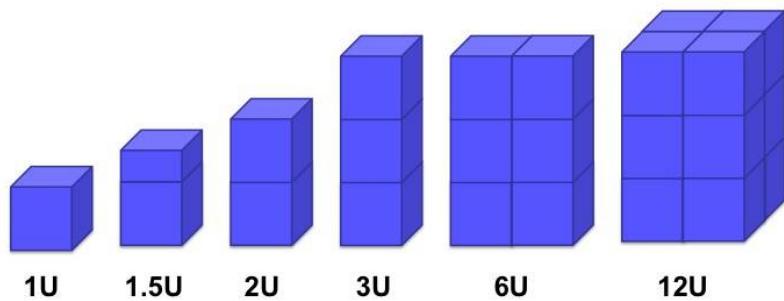
Fazendo uma análise minuciosa nos sites Nanosatellite Database<sup>1</sup> e Space Launch Report<sup>2</sup>, sites que armazenam dados de missões satelitais, observa-se que em 2007 cerca de 2,6% (9 de 348 satélites) eram CubeSats. Já em 2017 o número de lançamentos passou para 57,6% (295 de 512 satélites). Tendo como base esses valores, pode-se afirmar que os CubeSats ajudaram a popularização da categoria de nanosatélites.

De forma resumida, o CubeSat é um satélite cúbico de 10 cm de aresta e que pesa apenas alguns quilogramas, sendo muito utilizado no meio acadêmico como demonstrador tecnológico. 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 3 mostra as configurações mais usuais de um CubeSat (FROST; AGASID, 2015).

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

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

**Figura 3** – Algumas variações de CubeSat.



Fonte:(MABROUK, 2017, pág.1).

## 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).

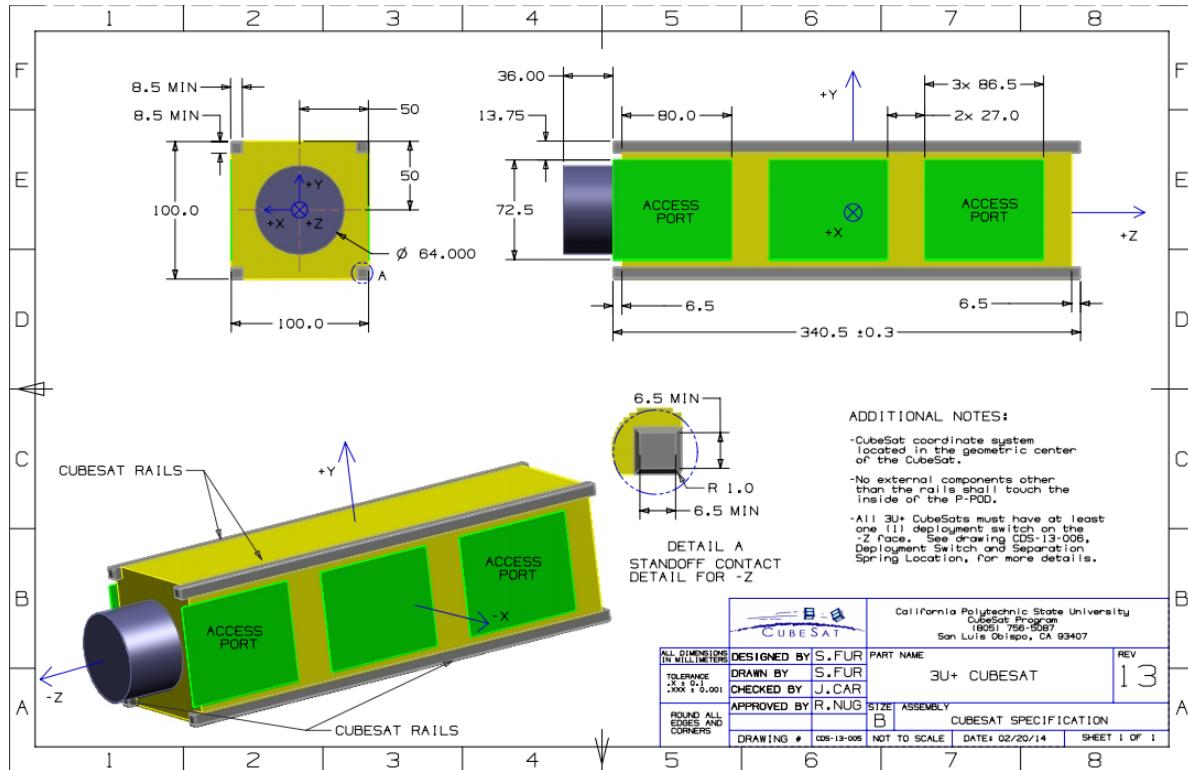
Mais tarde naquele ano, Twiggs e Puig-Suari disponibilizaram as especificações para o Padrão CubeSat. Em 2003, Puig-Suari e CalPoly desenvolveram o *Poly-PicoSatellite Orbital Deployer* (P-POD) para lançar até três CubeSats 1U. As primeiras missões CubeSat começaram no mesmo ano do desenvolvimento do 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.org<sup>3</sup>. A Figura 4 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 demandas de sensoriamento remoto, possibilitando colocar uma câmera sem prejudicar o

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

espaço interno da plataforma.

**Figura 4 – Especificações Estruturais para um CubeSat 3U+.**



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

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;

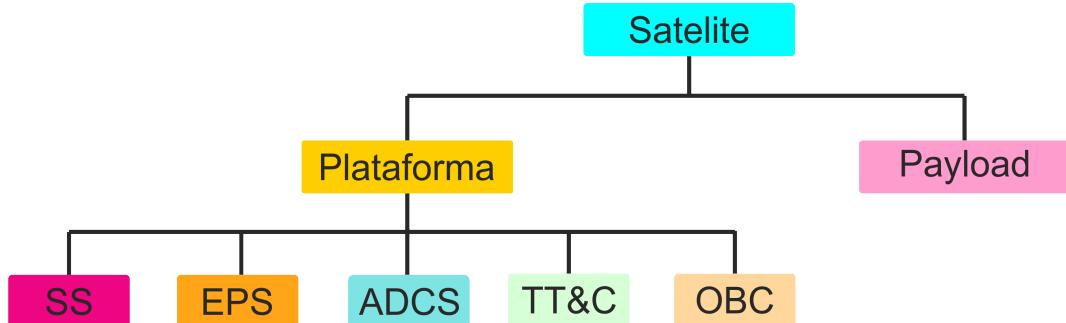
- **Custo:** Geralmente as missões CubeSat são desenvolvidas usando um "orçamento baixo", comparado às missões tradicionais;
- **Componentes:** Componentes que não possuem o selo “*Space Qualified*” são frequentemente utilizados e aceitos em missões CubeSat, permitindo uma abordagem de baixo custo e 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 5 visa ilustrar esses subsistemas (ADDAIM; KHERRAS; ZANTOU, 2010).

**Figura 5 – Arquitetura de um Satélite.**



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

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 6 mostra o *Structure Subsystem* da NanoAvionics<sup>4</sup>.

**Figura 6 – Estrutura 3U Padrão NanoAvionics.**



Fonte:(NANO AVIONICS, 2018, págs.1).

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

- **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 7 é mostra os principais elementos de um EPS.

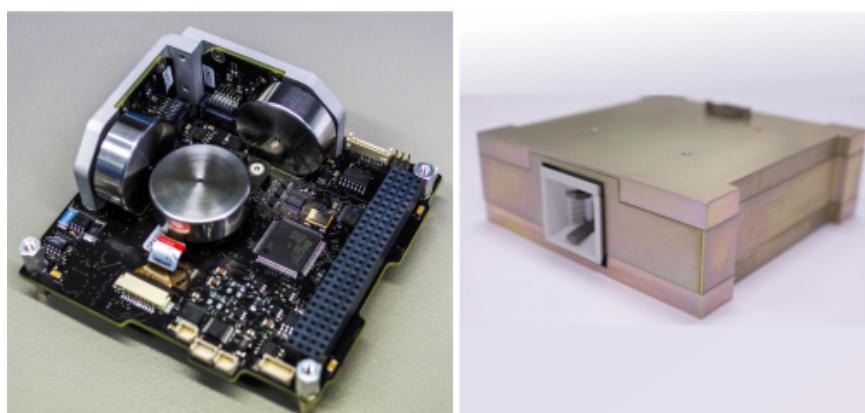
**Figura 7** – Principais componentes de um EPS.



Fonte:(CUBESTAR, 2018, pág.1).

- **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 inerciais, fotodiodos e magnetorques. Recentemente estão sendo utilizados módulos PPT (*Pulsed Plasma Thruster*) para compensar o arrasto atmosférico. A Figura 8 é mostra o AODCS (esquerda) e PPT (direita) da empresa Clyde Space<sup>5</sup>.

**Figura 8** – Componentes AODCS da empresa Clyde Space.

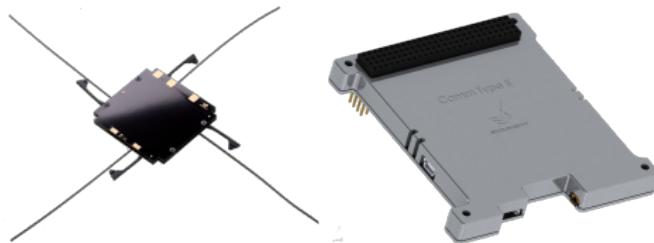


Fonte: (CLYDE SPACE, 2018, pág.1).

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

- **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 9 mostra uma antena (esquerda) e um módulo de telecomunicações (direita) da empresa Endurosat<sup>6</sup>.

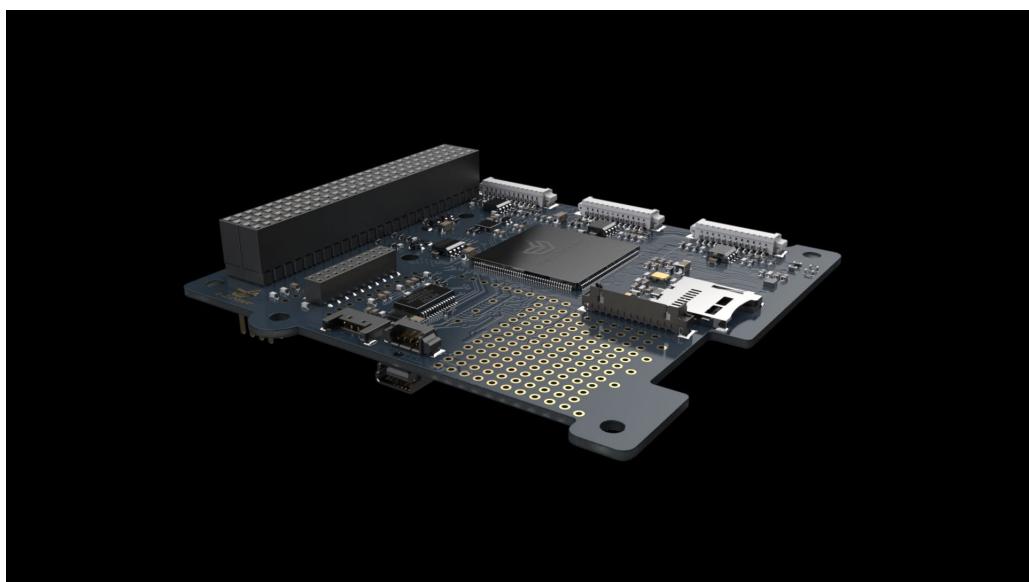
**Figura 9 – Componentes TT&C da empresa EnduroSat.**



Fonte:(ENDUROSAT, 2018a).

- **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 10 mostra o OBC da empresa EnduroSat.

**Figura 10 – Computador de bordo da empresa EnduroSat.**



Fonte:(ENDUROSAT, 2018b, pág.1).

---

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

## 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 apresentar ao leitor alguns OBCs, utilizados em missões CubeSat, e destacar as características mais relevantes. Ao final dessa seção há o resumo das características mais relevantes destes OBcs. Uma coleta mais detalhada foi realizada e se encontra no Apêndice A.

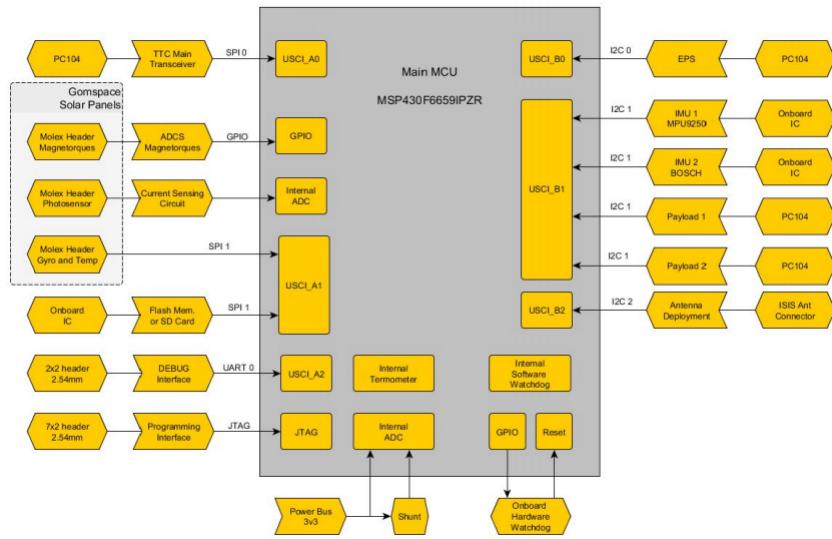
- **FloripaSat OBDH**

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 *Watchdog Timer* (FLORIPASAT,2016). A Figura 11 mostra a arquitetura do FloripaSat OBDH.

**Figura 11** – Arquitetura do OBDH do FloripaSat.



Fonte: (FLORIPASAT,2016, pág.45).

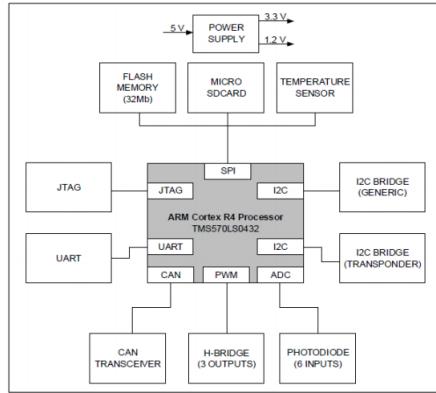
### • Open OBC

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, ga-

rantindo assim um controle rígido de erros e a recepção de mensagens (OPENOBC, 2017).

**Figura 12** – Arquitetura do OpenOBC.



Fonte: (OPENOBC, 2017, pág.28).

- **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 13 mostra a placa eletrônica do iOBC.

**Figura 13** – Placa Eletrônica - iOBC.

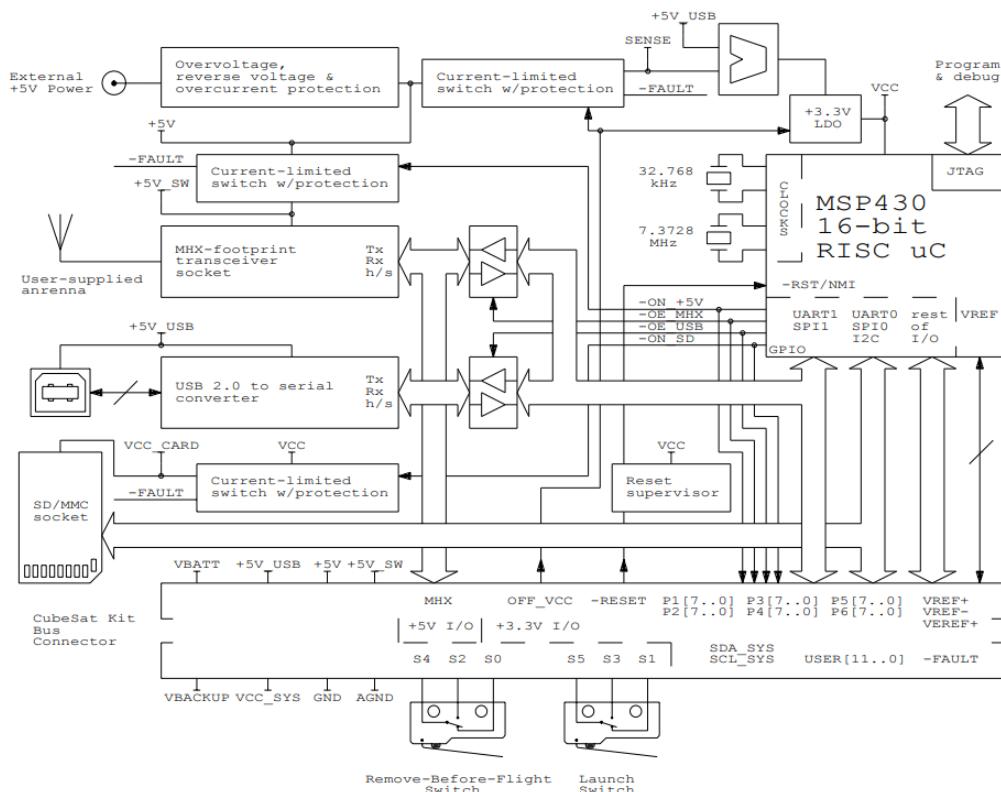


Fonte: Empresa ISIS<sup>7</sup>.

### • 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).

**Figura 14** – Arquitetura do FM430.



Fonte:(PUMPKIN, 2008, pág.6).

- **Tabela Comparativa entre os Computadores de Bordo**

As características mais relevantes dos OBCs, mostrados acima, são mostrados na Tabela 1.

**Tabela 1** – Tabela comparativa dos OBCs.

	<b>Projetos</b>			
Projeto/OBC	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

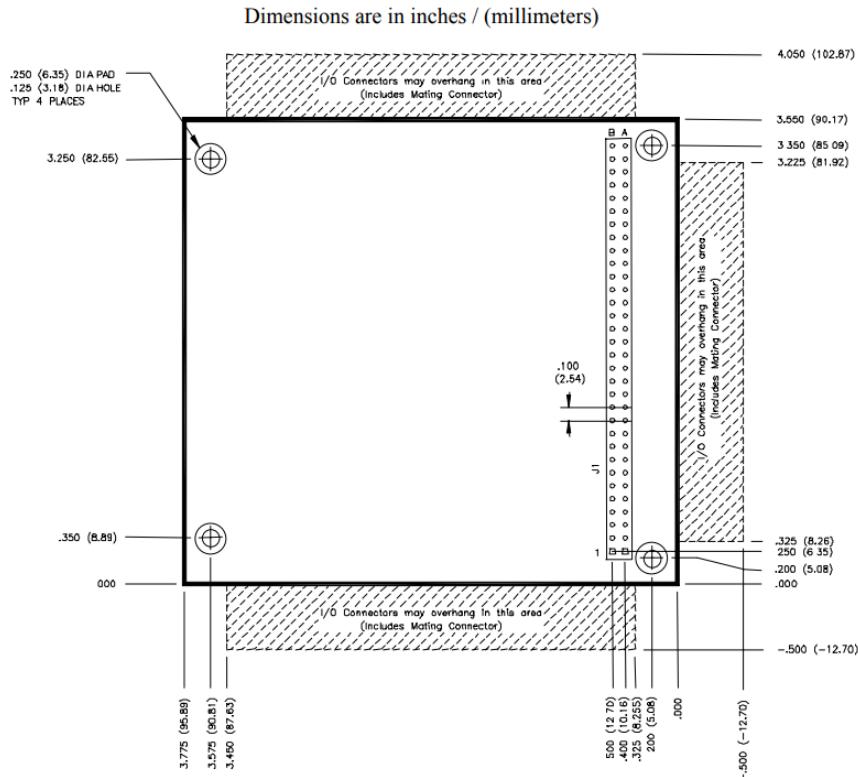
O PC/104 é o padrão de placas eletrônicas mais utilizado na indústria e em missões CubeSat. O PC/104 Embedded Consortium (2008) definiu as restrições mecânicas e elétricas para uma placa padrão de circuito impresso. As principais restrições podem ser encontradas a seguir:

- 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 Standard<sup>8</sup>.

<sup>8</sup> PC/104 Standard, version 2.6. 2018. Disponível em:<[https://pc104.org/wp-content/uploads/2015/02/PC104\\_Spec\\_v2\\_6.pdf](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 15 mostra as dimensões mecânicas para uma placa PC/104 de 8-bits.

**Figura 15** – PCB Padrão PC104 8-bits.



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

## 2.6 Ambiente Espacial

Um problema a ser considerado no desenvolvimento 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 16.

**Figura 16** – 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).

- **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 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

**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:  $10^{-7}$ - $10^{-8} \frac{erros}{bit.dia}$ .
  - 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:  $10^{-10}$ - $10^{-12} \frac{erros}{bit.dia}$ .
    - Geralmente testado apenas para falha funcional.
    - Avaliação e risco do cliente.



# 3 Desenvolvimento do Hardware

Esse capítulo descreverá o desenvolvimento do *hardware*. Inicialmente será apresentado 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, serão expostos os componentes adicionais que formarão o OBC, como: memória, sensores, etc. Por fim, será apresentado o esquemático do OBC e o *layout* da placa de circuito impresso.

## 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. Durante a fase inicial do projeto, poucos requisitos foram especificados, pois o acordo da missão CubeSat estava em fase de discussão.

A abordagem utilizada para contornar essa situação foi adicionar requisitos de CubeSats já lançados à 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 4.

**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 heliossíncrona.
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 componentes mais importantes do OBC, pois ele executará o *software* destinado ao gerenciamento do satélite. No processo de seleção do microcontrolador, 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  $0,03\text{m}^2$  ( $3\text{W}$ ) 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  $0,6\text{W}$ .
- **Conversores AD:** O OBC contará com muitos sensores (corrente, luminosidade, etc), sendo assim, conversores AD são importantes 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  $-80^\circ\text{C}$  (eclipse), e  $53^\circ\text{C}$  (insolação). Já para órbitas Sun-Síncronas, a estimativa de variação é de  $-27^\circ\text{C}$  à  $43^\circ\text{C}$  (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	Interface Serial	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.

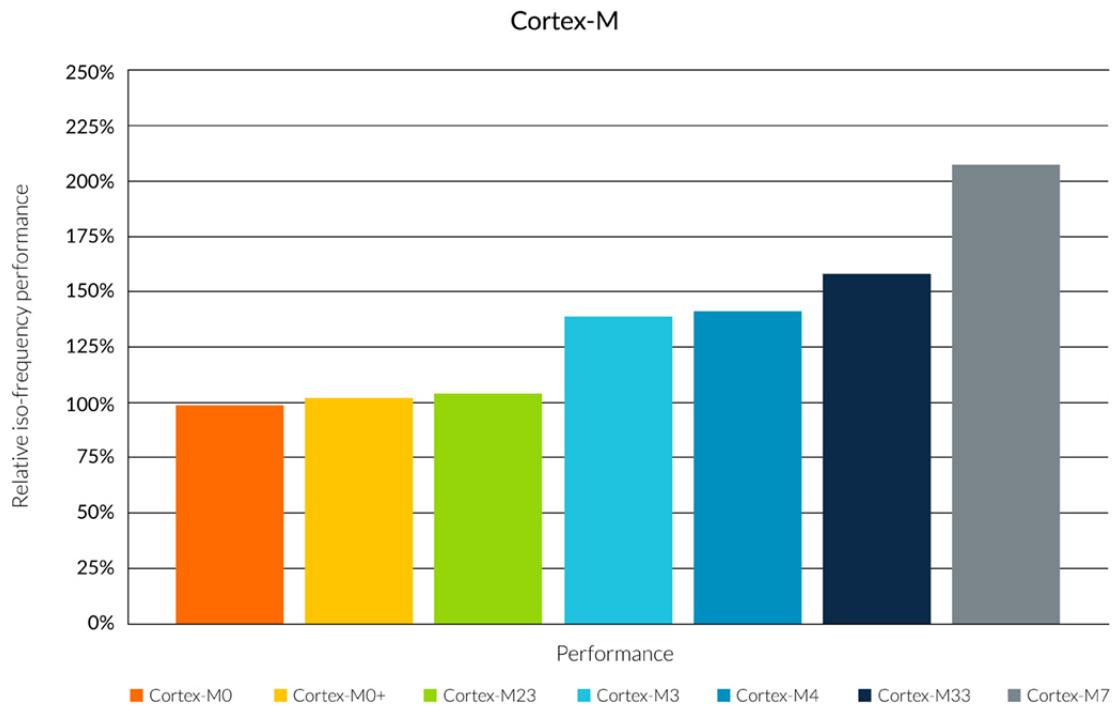
Microcontrolador	Consumo	AD	I/O	Interface Serial	PMW	Watchdog Timer	Frequência	Faixa de Temperatura [°C]	SOMA
PESO	5	3	3	4	3	3	5	4	30
MSP432P4111	5	3	3	3,5	3	3	2,4	3,03	25,93
MC9S12XHZ512	2,83	2	2,04	3,7	3	3	2	3,52	22,09
MSP430F5529	1,8	1,5	2,25	3	3	3	1,25	3,03	18,83
RM42L432	0,46	2	1,61	3,7	3	3	5	3,52	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 notas. 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 17, 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

**Figura 17** – Performance dos processadores CORTEX-M.



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

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.

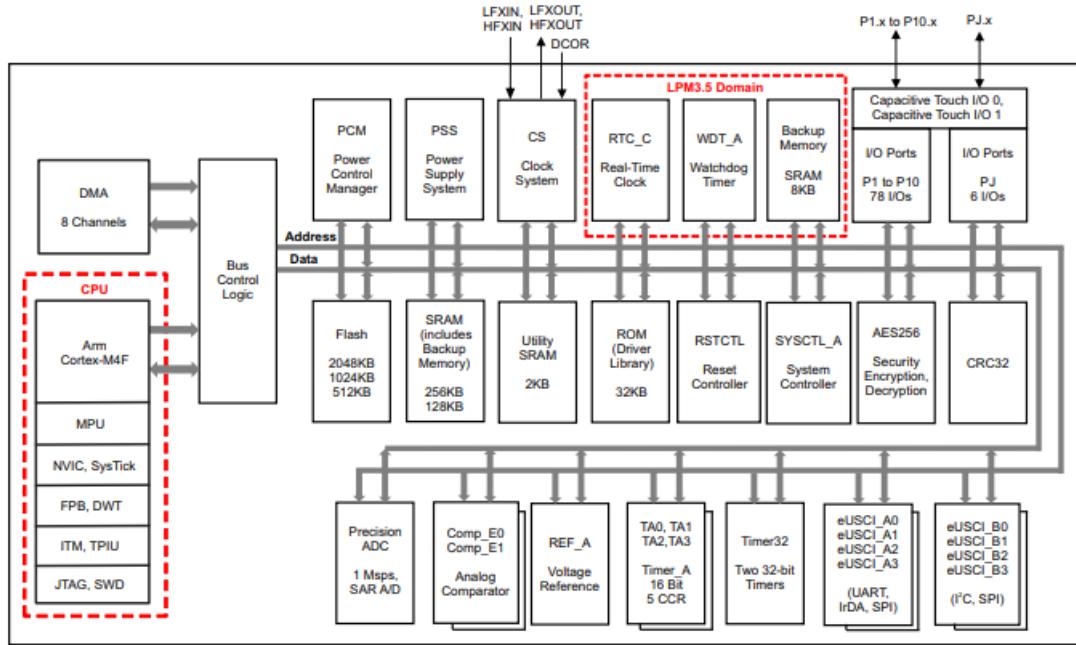
### 3.3 Microcontrolador MSP432P4111

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 18 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).

O barramento interno é de 32-bit, também chamado de *Advanced Microcontroller*

**Figura 18 – Arquitetura do microcontrolador MSP432P4111.**



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

*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 Memória 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 MSP432<sup>TM</sup> *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 manipula-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, Comparaçao 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 para a

**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
Resistencia (Ciclo de Deletar/Escrta)	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).

unidade de armazenamento. Comparando-se essas memórias, pôde-se observar 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 complicada. 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. A estimativa da quantidade de dados se encontra 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*<sup>1</sup>. 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 Semi. Corp.	-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 ACS711. Esse sensor possui um diferencial, ele conta com um sistema que protege o circuito contra sobre corrente, em caso de panes elétricas e tempestades solares. O ACS711 suporta valores de 3 a 5V de tensão e -40 a 125°C de temperatura.

A figura 19 mostra o esquemático eletrônico de uma aplicação usual do ACS711.

A tabela 9 mostra as especificações do ACS711. Essas informações foram retiradas da ficha técnica do sensor.

**Tabela 9** – Informações técnicas do ACS711.

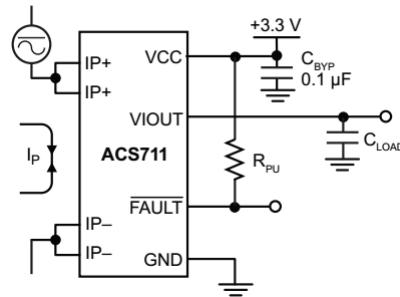
Produto	Fornecedor	Temp Op [°C]	Overcurrent limit [A]	Vin [V]	Erro [%]
ACS711	Allegro	-40~125	100	3~5	5

### 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, fabri-

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

**Figura 19** – Esquemático Eletrônico de uma aplicação usual do ACS711.



Fonte:(ALLEGRO, 2017, pág.1)

cado 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).

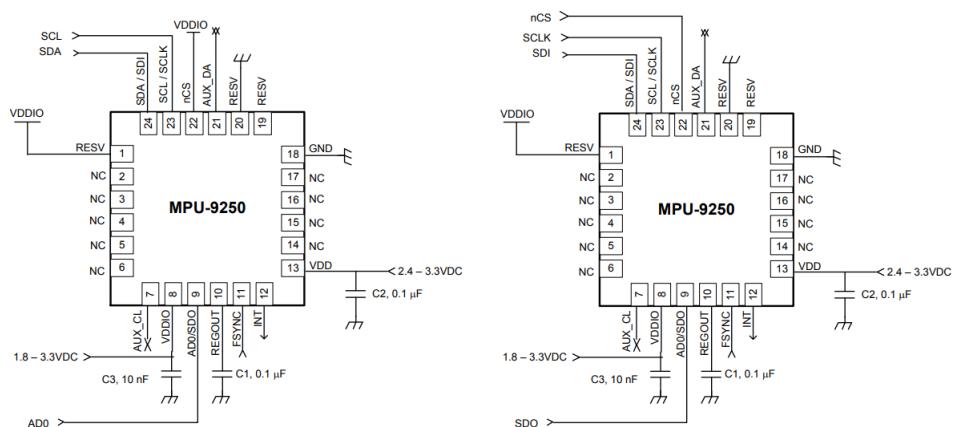
A Tabela 10 mostra as especificações do MPU9250. Essas informações foram retiradas da ficha técnica do sensor.

**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./Acel. ->16 Mag. ->14	SPI/I2C

A figura 20 mostra o esquemático eletrônico de uma aplicação usual do MPU9250, utilizando protocolo I2C(esquerda) e SPI (direita).

**Figura 20** – Esquemático Eletrônico de uma aplicação usual do MPU9250.



Fonte:(INVENSENSE, 2018, pág.20)

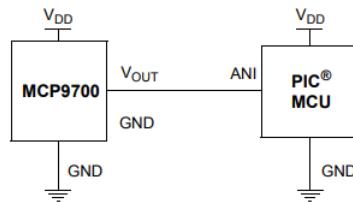
### 3.5.3 Sensor de Temperatura

Mesmo o MPU9250 e o MSP432P4111 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/MSP432P4111 podem interferir no valor das medidas. Por exemplo, o microcontrolador começará a esquentar após um período muito longo em modo de alta performance.

O sensor de temperatura escolhido foi o MCP9701T, da fabricante Microchip, devido sua fácil usabilidade e baixo consumo. O dispositivo comunica através de uma saída analógica, sendo capaz de medir temperaturas com uma precisão de 2 °C em uma faixa de temperatura de -40°C a 125°C (MICROCHIP, 2016).

A tabela 11 mostra as especificações do MCP9701T. Essas informações foram retiradas da ficha técnica do sensor.

**Figura 21** – Esquemático Eletrônico de uma aplicação usual do MCP9700.



Fonte:(MICROCHIP, 2016, pág.20)

**Tabela 11** – Informações técnicas sobre sensor MCP9701T.

Produto	Fornecedor	Temp. Op. [°C]	Corrente [uA]	Vin [V]	Resolução [bit]	Comunicação
MCP9701T	Microchip	-55 ~127	60	2.7 ~5.5	12	Analógico

## 3.6 Interface de Programação/Debug

O MSP432P4111 possui dois tipos de interface de programação: JTAG (*Joint Test Action Group*), SWD (*Serial Wire Debug*). Essas duas interfaces são utilizadas em conjunto para programar e "debugar" o microcontrolador<sup>2</sup>.

Os pinos destinados para a interface JTAG/SWD são mostrados na Tabela 12.

<sup>2</sup> Disponível em: <http://www.ti.com/lit/ug/slau747b/slau747b.pdf>

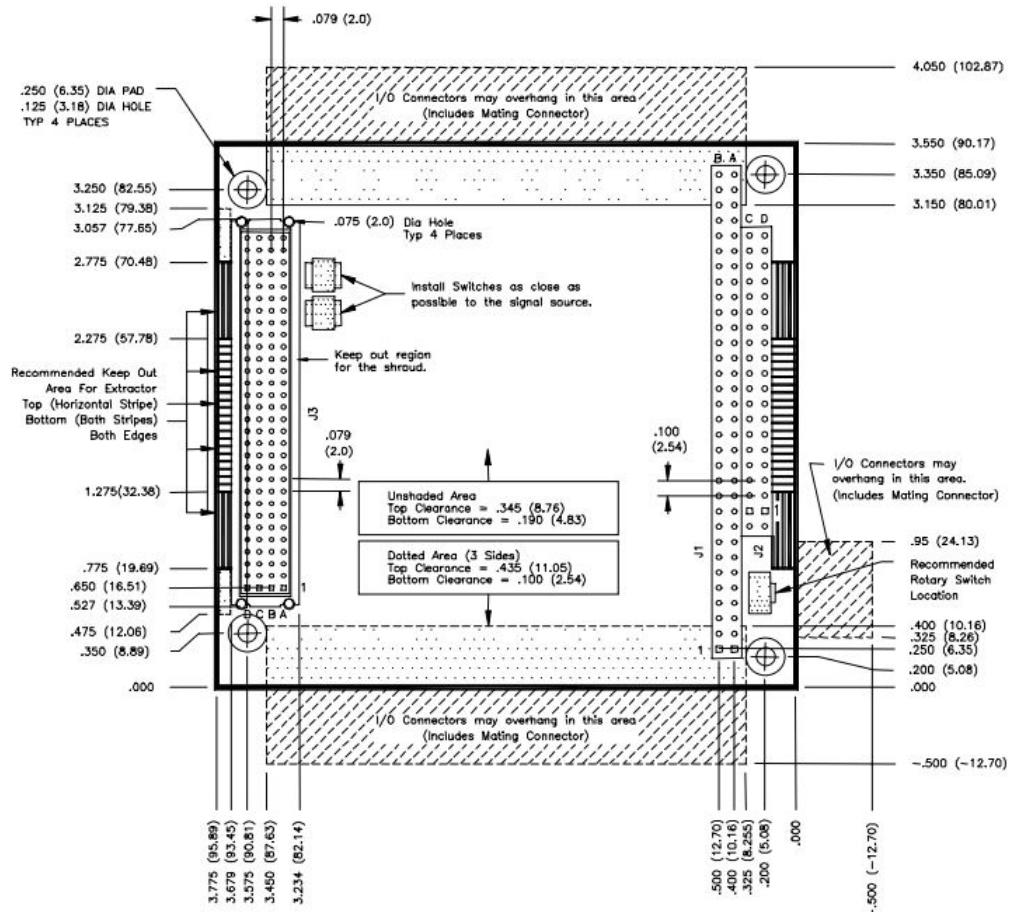
**Tabela 12** – Pinos JTAG no MSP432P4111.

Sinal	Descrição	Pino
TCK_SWCLK	JTAG clock input (TCK)	P.95
TMS_SWDIO	JTAG test mode select (TMS)	P.94
TDO_SWO	JTAG trace output (TWO)	P.93
TDI	JTAG test data input	P.92

### 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*.

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



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

## 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, levantaram-se as seguintes estratégias:

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

Dentre as estratégias acima, a única que será discutida é a *Watchdog* externo pois é uma solução simples e que não demanda muito tempo de implementação. As demais estratégias podem ser implementadas em uma segunda versão do computador de bordo.

### 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 sistema em caso de travamento. O uso de um contador externo trará mais robustez ao OBC.

O contador selecionado *Watchdog* externo foi o STWD100, mesmo contador utilizado pelo pesquisador Botma (2011) no desenvolvimento de um OBC. 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).

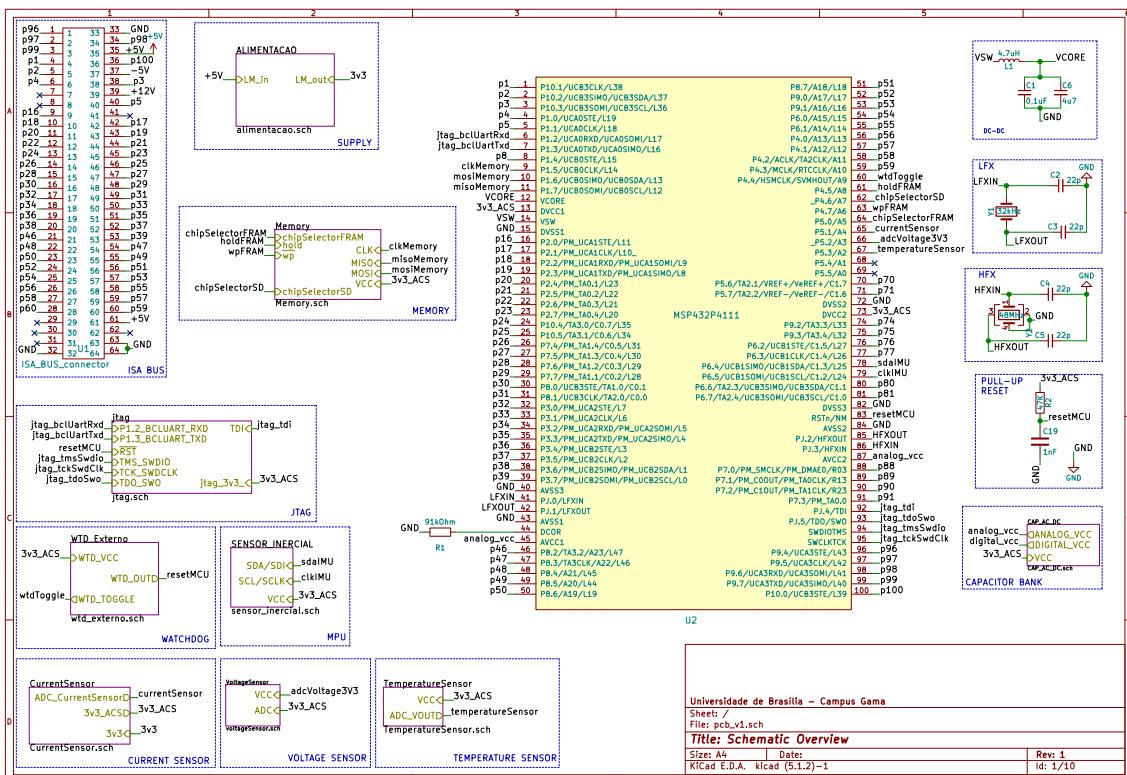
A escolha do tempo de *reset* será feita durante o teste do protótipo. Nessa fase, o *software* embarcado estará finalizado e possuirá o tempo de execução de todas as *Tasks*.

## 3.9 Placa de Circuito Impresso

Para a elaboração da placa de circuito impresso utilizou-se o software KiCad<sup>3</sup>. Essa ferramenta possui uma interface amigável e fácil de usar, sendo bem difundida na comunidade de engenharia eletrônica e *hobistas*. O fato do KiCad ser *Open Source* facilita a manutenção do OBC em futuras melhorias, pois não há a necessidade de comprar uma licença.

A Figura 23 mostra o esquemático eletrônico do OBC. Nele há todos os competentes/sensores mencionados nas seções anteriores. Foi utilizado o *datasheet* de cada componente para o desenvolvimento do esquemático. O Apêndice E possui todos os esquemáticos realizados.

**Figura 23 – Esquemático Eletrônico do OBC.**

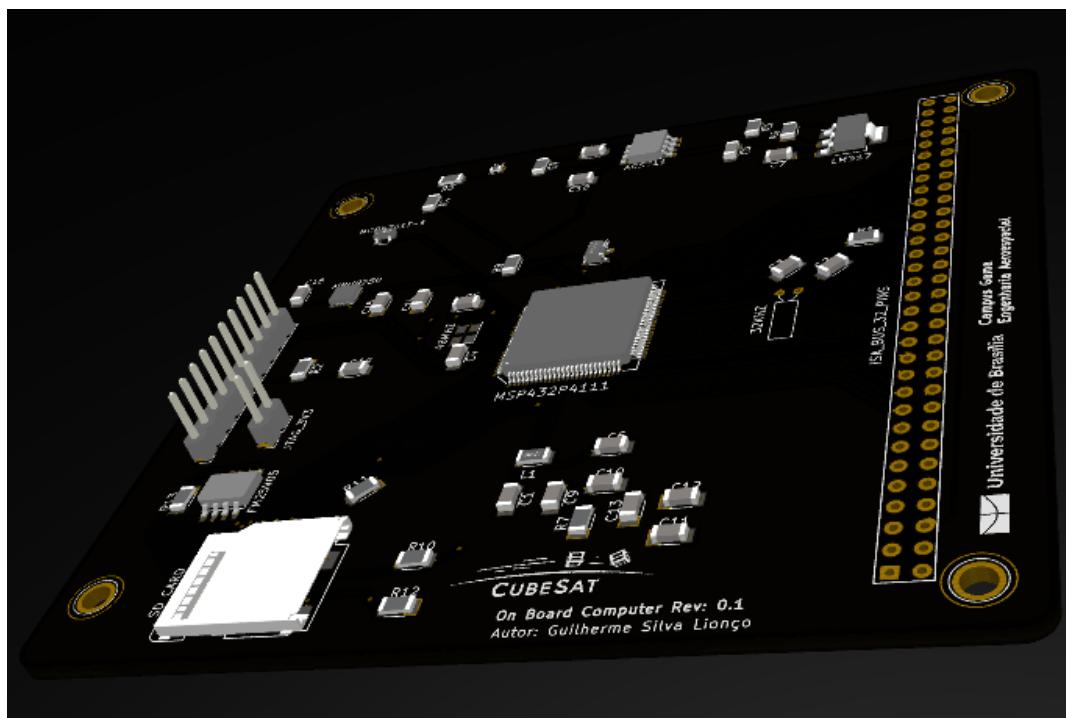


<sup>3</sup> Para mais detalhes, consulte: <<http://www.kicad-pcb.org/>>

Após a realização do esquemático, exportou-se o arquivo de conexões (.net) e o importou no ambiente "pcbnew" do KiCad. Esse ambiente permite realizar o roteamento das trilhas, criação de planos de alimentação, etc. Essa foi uma das etapas mais trabalhosas da pesquisa pois o roteamento é realizado manualmente.

Com o intuito de reduzir custo e aumentar a flexibilidade do projeto, os componentes SMD escolhidos possuíam tamanho 1206 (aproximadamente 3.2mm por 1.6mm). Essa escolha permite que os componentes sejam soldados manualmente sem ajuda de maquinário robotizado. A Figura 24 mostra PCB após o roteamento das trilhas.

**Figura 24 – PCB - Vista Isométrica.**



Alguns componentes não estão presentes na renderização 3D pois não foi possível desenhar-los, devido ao tempo limitado do projeto. No Apêndice E há as imagens do plano de terra e plano de alimentação da PCB.

A partir do BOM, realizou-se a cotação dos componentes na fornecedora online *DigiKey*. O valor ficou em US\$ 56,34 (BRL 216,73) e o frete em US\$40,00 (BRL 153,88). No Apêndice F há a lista do *Bill Of Material*.

# 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 estava em fase de discussão e poucos requisitos tinham sido especificados, utilizou-se uma abordagem modular no desenvolvimento do software embarcado. Sendo assim, a medida que os requisitos começaram a se tornar definitivos, as funcionalidades foram implementadas. Com isso, o *software* 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 13** – 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 arquiteturas 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.

**Figura 25** – Arquitetura de abstração em Camadas.



Fonte: (COHEN, 2014, adaptado pág.15).

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 (COHEN, 2014).

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

O *DriverLib* inclui a 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 Operationg 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 (COHEN, 2014).

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

Analisando 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 (COHEN, 2014).

Nesse caso, o FreeRTOS já oferece APIs que auxiliam o acesso de suas funcionalidades. Entretanto a CSS não estaria completa só com essas APIS, com isso há a necessidade de implementar mais funcionalidades nessa camada, levando em consideração os requisitos da Tabela 13. 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 (COHEN, 2014).

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 INSTRUMENTS, 2015).

**Tabela 14** – 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, portável e escalável; de acordo com o site oficial<sup>1</sup> 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.

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

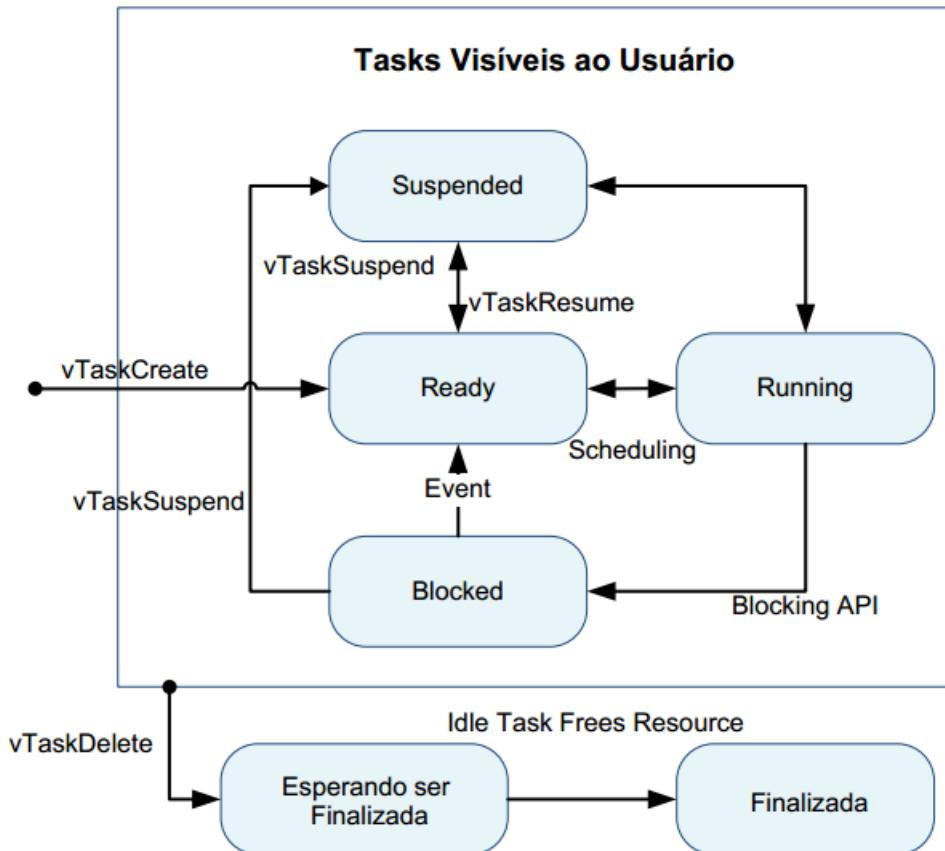
Por exemplo, se o sensor de colisão do carro atrasar a sua resposta, o usuário estaria em grande risco (BARRY, 2016).

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 a 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.

**Figura 26** – Estados das Tasks no FreeRTOS



Fonte: (LIN,2010, adaptado pág.20)

#### 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.
- *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 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 15** – APIs da categoria *Task Creation* e *Task Control*.

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

#### 4.4.2 Configuração do FreeRTOS

Como o *kernel* do FreeRTOS é comum para a maioria dos microcontroladores, o porting é essencial para adaptar o sistema para o tipo de microcontrolador usado. No site da empresa há um template dos arquivos **portmacro.h** e **FreeRTOSConfig.h** para o MSP432. Foram modificados poucos parâmetros, pois o aluno estava se adaptando à tecnologia. Na tabela 16 há os parâmetros modificados.

**Tabela 16** – Configurações realizadas nos arquivos de *porting*

Arquivo	Parâmetro	Configuração Realizada
portmacro.h	configCPU_CLOCK_HZ	48 MHz
portmacro.h	configTICK_RATE_HZ	100 KHz
FreeRTOSConfig.h	configUSE_TICKLESS_IDLE	Habilitado
FreeRTOSConfig.h	configUSE_COUNTING_SEMAPHORES	Habilitado
FreeRTOSConfig.h	configUSE_RECURSIVE_MUTEXES	Habilitado
FreeRTOSConfig.h	configUSE_TASK_NOTIFICATIONS	Habilitado
FreeRTOSConfig.h	configUSE_IDLE_HOOK	Habilitado
FreeRTOSConfig.h	configUSE_TICK_HOOK	Habilitado
FreeRTOSConfig.h	configPRE_SLEEP_PROCESSING( x )	vPreSleepProcessing( x )
FreeRTOSConfig.h	configPOST_SLEEP_PROCESSING( x )	vPostSleepProcessing( x )

## 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 17.

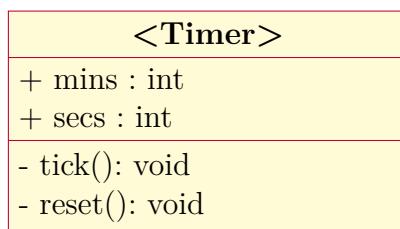
**Tabela 17** – Perfil do Diagrama FunctionalC.

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, 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 a cada segundo. Em UML esse elemento seria representado conforme a Figura 25. Em C++ seria realizado uma classe, mas em C seria implementado conforme mostra o Código 4.3.

**Figura 27** – Exemplo Timer UML.



Fonte: (DOUGLASS, 2009, pág.5).

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();

```

Fonte: (DOUGLASS, 2009, pág.5).

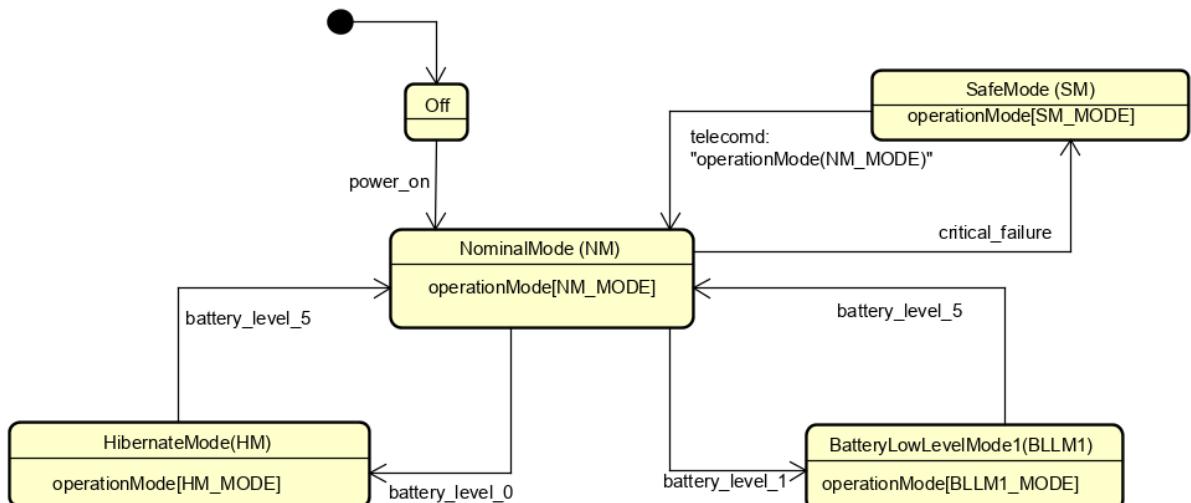
#### 4.5.1 Desenvolvimento da UML

Analizando os requisitos do sistema, Tabela 13, foi possível abstrair as funções principais exigidas para o software: Controle das Tarefas, Coleta de Dados e Armazenamento. Sendo assim, foram realizado a Máquina de Estados, Fluxograma e o Diagrama de Arquivos necessário para o software embarcado.

- **Máquina de Estados**

A máquina de estados desenvolvida levou em consideração os casos macros do CubeSat em sua órbita. Não foi levado em consideração os estágios de pré-lançamento e *deployment*. Foram considerados quatro modos de operação: **Nominal Mode**, **Safe Mode**, **Battery Low Level Mode** e **Hibernate Mode**. Esses modos são mostrados na Figura 28.

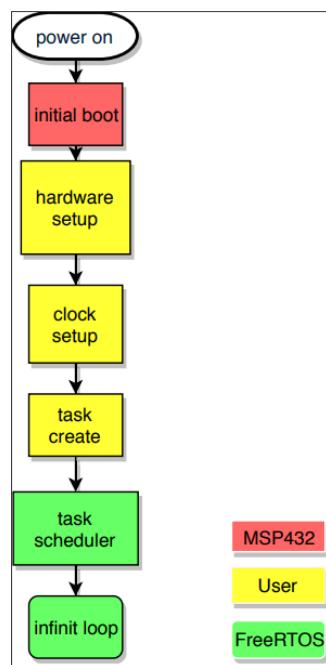
**Figura 28** – Maquina de Estados da Camada de Serviço.



- **Fluxograma**

Como o *kernel* escolhido foi o *FreeRTOS* as atividades do OBC foi divididas em *Tasks*. A CCS possui 8 rotinas, sendo: uma de controle (**TaskManager**); uma de coleta de dados (**HouseKeeping**); uma de armazenamento (**DataStorage**); uma de controle de travamento (**WatchDogTask**); cinco de referentes à camada de aplicação. A Figura 29 mostra o fluxograma de inicialização do sistema e no Apêndice G há os demais fluxogramas da CSS.

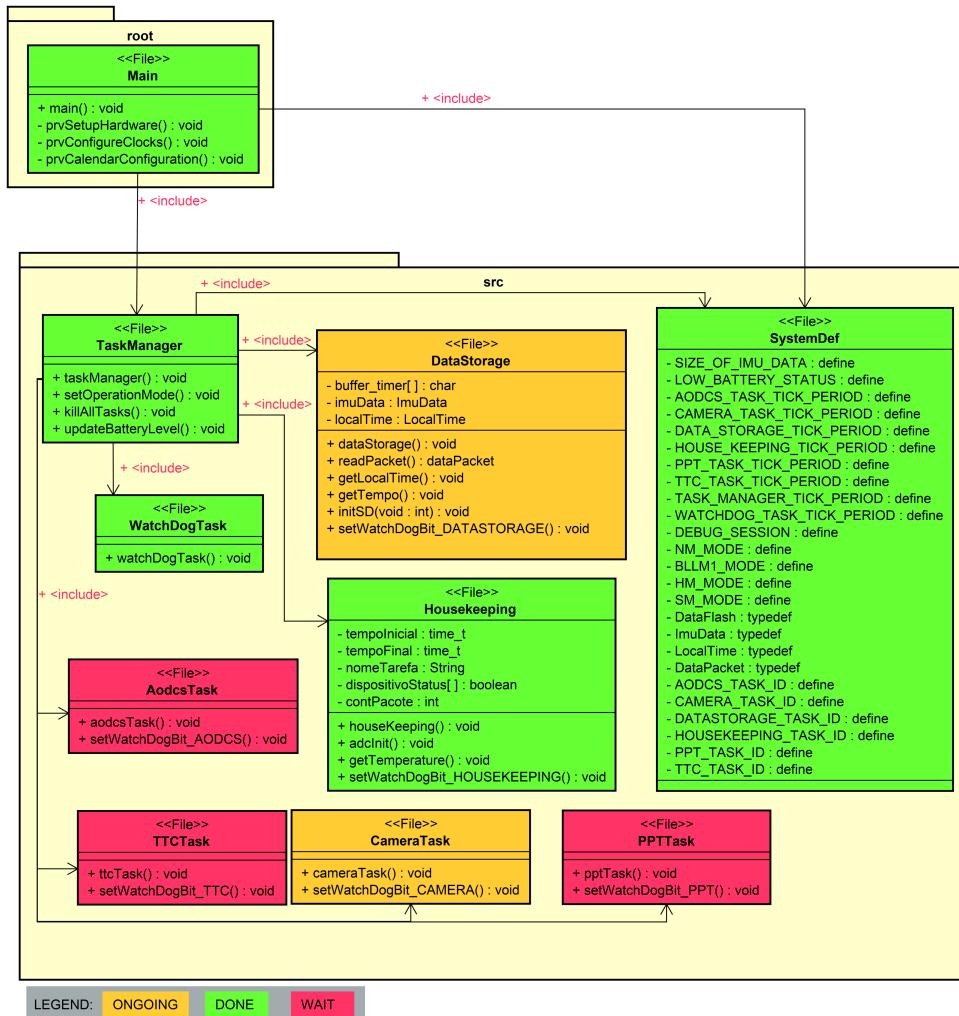
**Figura 29** – Fluxograma de Inicialização do sistema.



- **Diagrama de Arquivos**

O diagrama de arquivos é semelhante ao diagrama de classes em orientação a objetos. Devido a complexidade do sistema, alguns arquivos ainda estão sendo finalizados. A Figura 30 mostra o Diagrama de Arquivos.

Figura 30 – Diagrama de Arquivos do Sistema.



## 4.6 Desenvolvimento do Código

Com base na UML e no diagrama de interface do hardware foi realizado o código embarcado. O Ambiente de Desenvolvimento (IDE, do inglês *Integrated Development Environment*) escolhido foi o *Code Composer Studio*, devido a familiaridade do aluno com a ferramenta. Todo o código foi colocado no repositório<sup>2</sup> do aluno no GitHub.

<sup>2</sup> Código disponível em: [https://github.com/guilhermelionzo/firmware\\_onBoardComputer](https://github.com/guilhermelionzo/firmware_onBoardComputer)

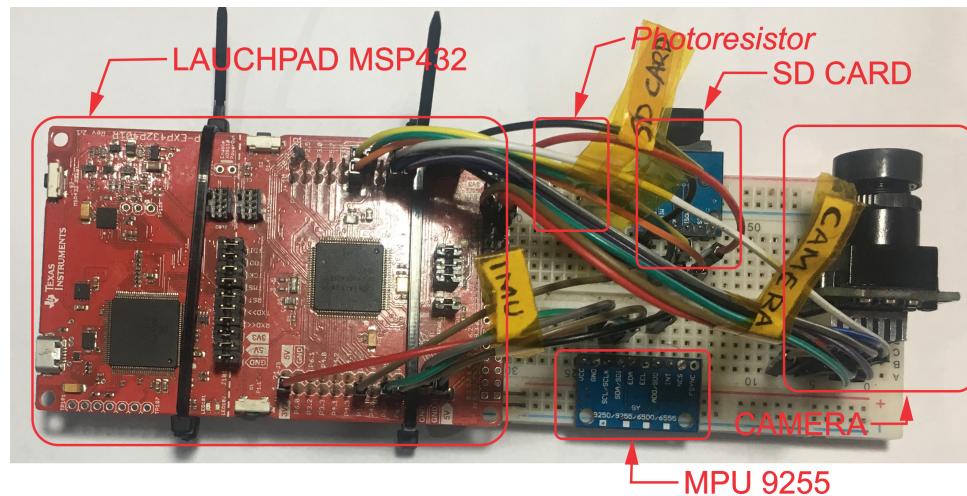


# 5 Resultados e Discussão

Durante o desenvolvimento do projeto as duas instituições estavam discutindo os acordos da parceria. Esse fato impossibilitou a captação de recursos para a compra dos componentes eletrônicos. Sendo assim, os resultados apresentados são pertinentes ao teste em protoboard, utilizando a LaunchPad do MSP432 e alguns módulos COTS.

Para simular o *EPS* do CubeSat, foi utilizado um *photoresistor* para indicar o nível de bateria. Essa abordagem condiz, de certa forma, com a realidade pois a incidência de luz interfere na quantidade de energia armazenada no *EPS*. A Figura 31 mostra a conexão dos componentes na LaunchPad. Nesta foto há os módulos COTS utilizados (SD Card, MPU9255 e *photoresistor*) e a câmera (*Payload*).

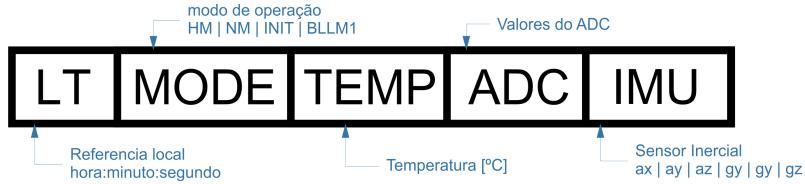
**Figura 31** – Protoboard com os componentes COTS e a LaunchPad.



## 5.1 Aquisição e Armazenamento de Dados

O serviço de armazenamento de aquisição e armazenamento de dados é feito pelas *Tasks HouseKeeping* e *DataStorage*, respectivamente. Durante os testes realizados, os dados foram salvados em formato de *ASIIIC*, para facilitar o debug. Vale a pena ressaltar que para a missão os dados serão armazenados conforme mostra o Apêndice B, pois o armazenamento será mais rápido e eficiente. O pacote de telemetria especificado para o teste seguiu o formato apresentado na Figura 32.

**Figura 32** – Formato do Pacote de Telemetria.



Durante os testes houveram algumas falhas para armazenar o tempo relativo no computador de bordo e alguns valores tiveram valor nulo. Não foi encontrar o motivo exato desse problema, mas uma possível causa é na conversão dos valores gerados pelo SysTick do sistema. A Figura 33 mostra os dados armazenados durante os testes.

**Figura 33** – Dados da telemetria armazenados no cartão de memoria.

Não foi possível realizar a aquisição das imagens provenientes da Câmera. O protocolo de controle da câmera (via I2C) foi desenvolvido, mas o protocolo de controle do FIFO (via SPI) está em fase de implementação. Durante os testes, não foi possível ler o valor do registrador teste do FIFO. Algumas possíveis causas desse problema seja a configuração da interface SPI, como por exemplo o clock, fase, polaridade e direção de transmissão.

## 5.2 Consumo

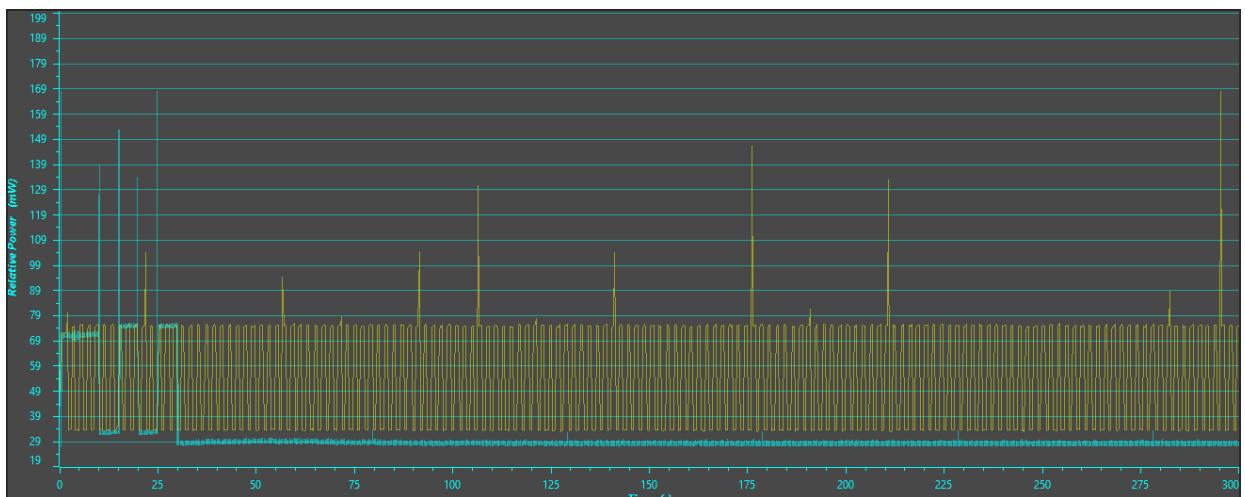
Um dos critérios mais importantes para o design de um OBC é o consumo energético em várias condições de operação. Ainda é mais importante nanosatélites onde a fonte de energia é limitada pela área dos painéis solares.

Os testes de consumo foram realizados utilizando a ferramenta *EnergyTracer* do *Code Composer Studio*. Essa ferramenta mede a corrente sendo consumida no barramento JTAG/SW, sendo assim ela permite calcular a corrente de todos os módulos/sensores alimentados pela launchpad.

Para simular o *EPS* do CubeSat, foi utilizado um *photoresistor* para indicar o nível de bateria. Houveram três baterias de teste de cinco minutos, um para cada estado. Os dados do consumo de cada um está presente na tabela 1.

Comparando o estado de nominal com o estado de hibernação, observa-se que houve uma economia de mais de 40%, aumentando a vida da bateria em dois dias. A Figura 34 mostra a comparação em forma gráfica, hibernação em azul e nominal em amarelo.

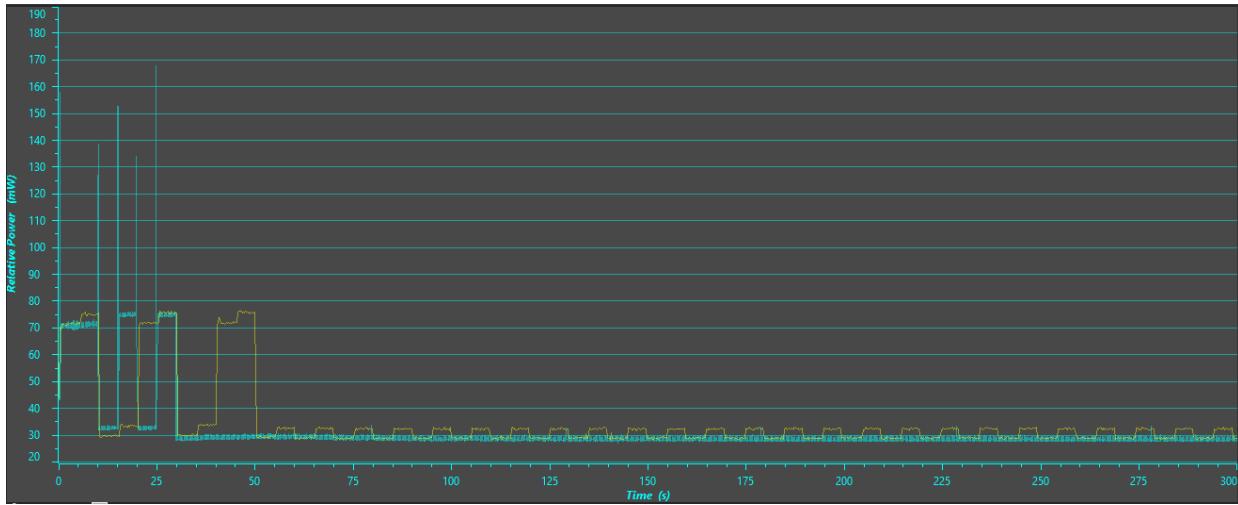
**Figura 34** – Comparação entre os Modos de hibernação (azul) e Nominal (amarelo).



Não foi possível concluir a mesma melhoria comparando o modo de baixo consumo com o modo de hibernação. Houve apenas uma economia de 9%, 0.6 dias. Esse fato ocorre por que o modo de hibernação não está completamente otimizado. A fonte de sincronismo durante esse estado ainda continua sendo 48MHz. O ideal seria utilizar o clock externo de 32KHz para realizar as interrupções do kernel. Infelizmente devido à inexperiência do aluno com o FreeRTOS, não foi possível adicionar uma segunda fonte de sincronismo no modo de baixo consumo. A Figura 35 mostra a comparação em forma gráfica, hibernação em azul e nominal em amarelo.

Apesar de não ter sido implementado o clock de baixa frequência para o modo de hibernação, é possível confrinhar que o software embarcado consegue alterar o modo de operação dependendo de uma fonte externa. Sendo assim, o requisito OBC-SW-R7 foi cumprido.

**Figura 35** – Comparação entre os Modos de hibernação (azul) e Pouca Bateria (amarelo).



### 5.3 Modos de Operação

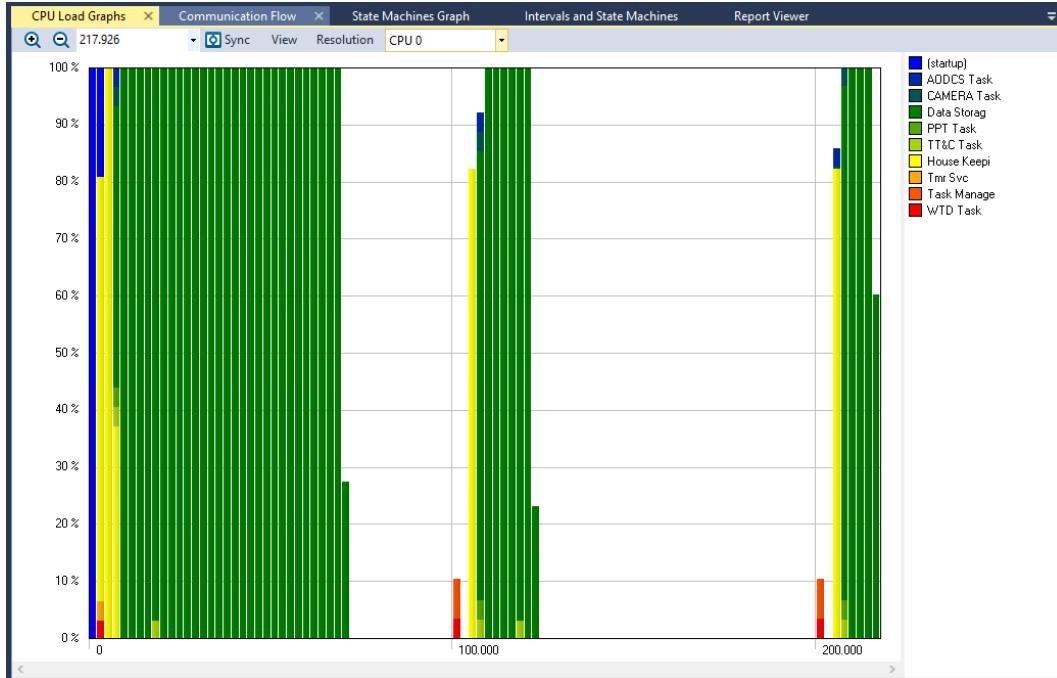
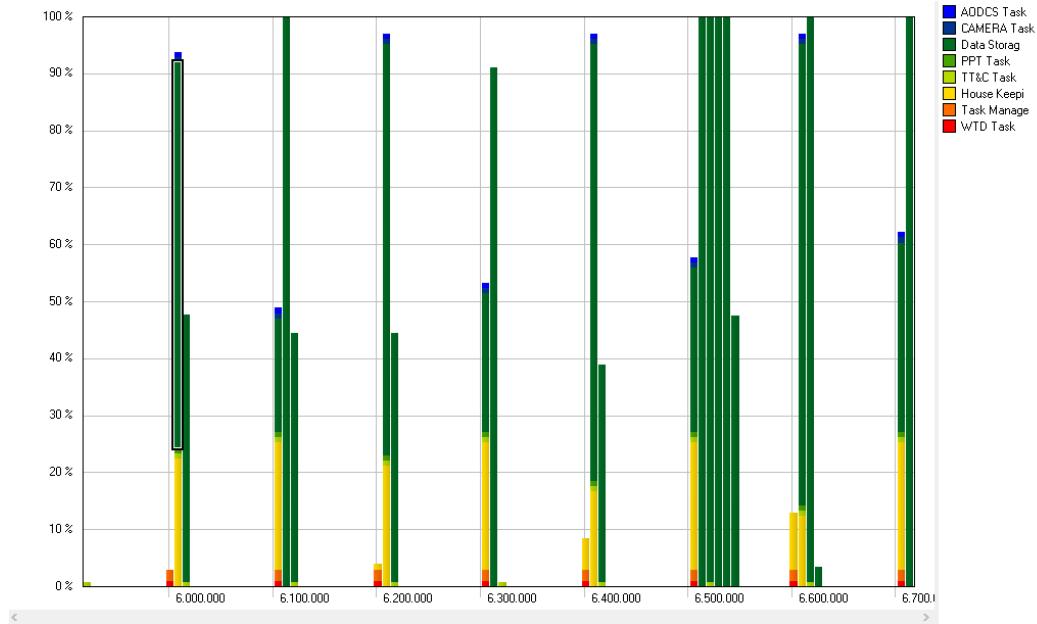
Para testar a máquina de estados do OBC, utilizou-se o software Tracealyzer, da empresa Percepio<sup>1</sup>. Essa ferramenta consiste em uma interface gráfica que interpreta os dados gerados durante a execução do FreeRTOS. Os dados são armazenados em um *buffer*, na memória RAM do microcontrolador, e após a solicitação do Tracealyzer são enviados para o computador. Utilizou-se a ferramenta em modo *screenshot*, o que consiste em pegar os dados do *buffer* e interpretar na interface gráfica. O modo *streaming*, que consiste em pegar os dados em modo contínuo, não foi utilizado pois não havia o cabo de *debug* J-link.

No decorrer dos testes foi possível observar vários fenômenos interessantes. Durante a inicialização do sistema, nota-se que há um grande consumo de CPU pelas *Tasks*, principalmente *DataStorage*, devido a montagem da imagem do SD Card. Isso acontece pois o sistema operacional está alocando memória para as *tasks*, *queues* e outros *handlers*. A Figura 36 mostra a inicialização do sistema.

Após alguns segundos foi possível notar que a task do *DataStorage* ainda consome muita CPU. Isso ocorre pois durante os testes os dados dos sensores foram convertidos e armazenados, o que consome muito processamento. É bem provável que salvar os dados em sua forma original diminuirá o poder de processamento em mais da metade. Figura 37

Após o período de inicialização, colocando o sistema em modo de alta performance (NM\_MODE) é possível ver que as *Tasks* se estabilizaram e começaram a diminuir o processamento do *kernel*, como é mostrado na Figura 38. O processamento médio ficou em torno de 8%. Nesse estado o *TaskManager* consome mais que os outras *Tasks*, em torno de 2%.

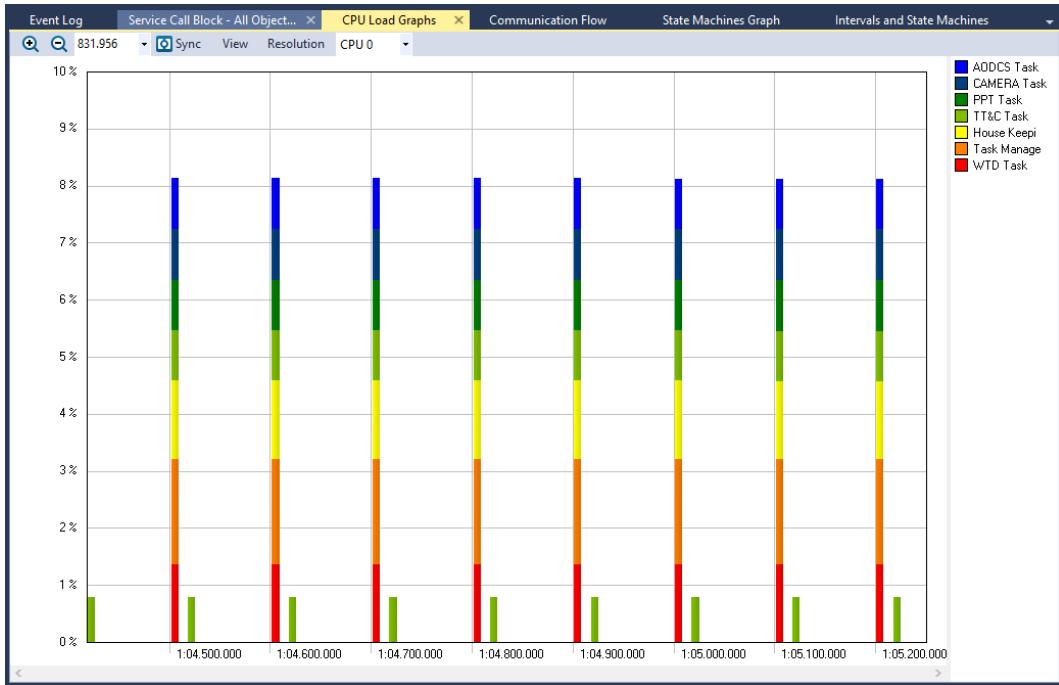
<sup>1</sup> Para mais detalhes da empresa consulte <<https://percepio.com/>>

**Figura 36** – Snapshot no Tracealyzer durante a inicialização do sistema.**Figura 37** – Snapshot no Tracealyzer alguns segundos após a inicialização do sistema.

Colocando o sistema em modo de baixo consumo é possível notar que há uma redução no tempo de execução das *Tasks*, que durante os testes esse tempo foi cinco vezes maior que o tempo de execução nominal, como mostrado na Figura 39. Também é visto que as *Tasks* com menos prioridade foram coladas em modo de sleep, restando apenas as *Tasks* que são de controle(*WTD Task* e *TaskManager*) e manipulação de dados (*HouseKeeping* e *DataManager*)

Como era de se esperar, durante o modo de hibernação apenas o *TaskManager* fica

**Figura 38** – Snapshoot no Tracealyzer do sistema executando NM\_MODE.

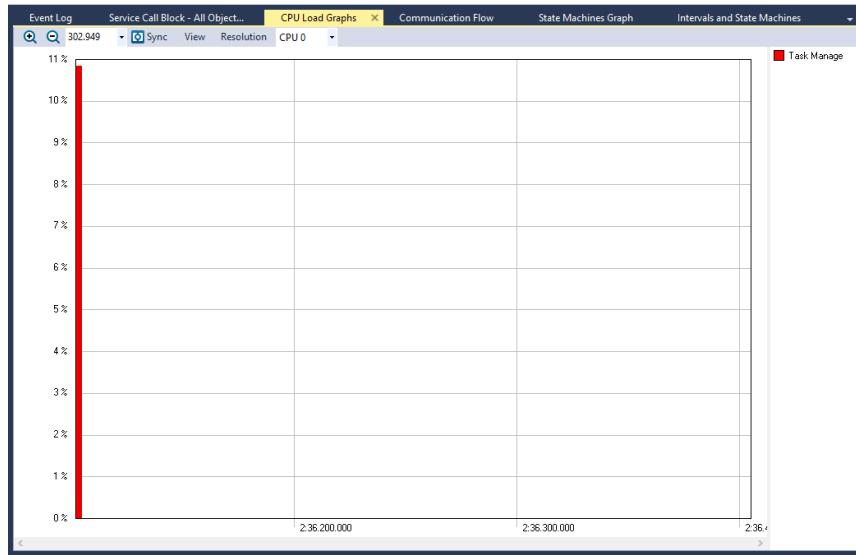


**Figura 39** – Snapshoot no Tracealyzer do sistema executando BLLM1\_MODE.



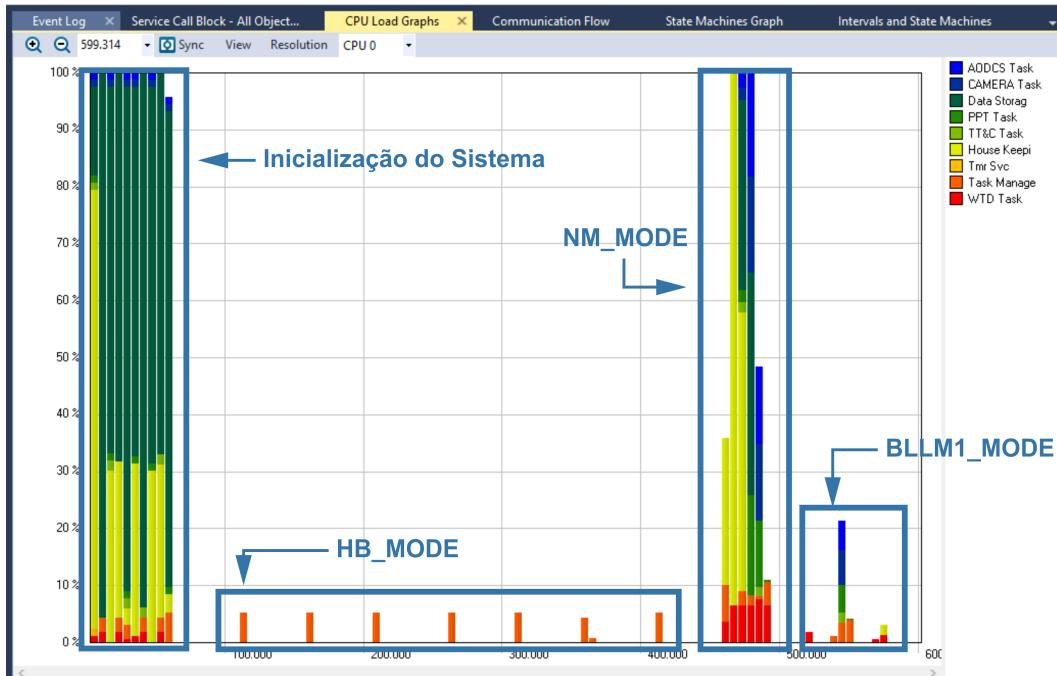
ativo e consumindo mais CPU do que o usual, quase de 11%. O *kernel* também executa essa *task* com um espaçamento cinco vezes maior que o normal. em 5 vezes ao normal. A Figura 40 mostra o gráfico de processamento do OBC em modo de hibernação.

**Figura 40 – Snapshot no Tracealyzer do sistema executando HB\_MODE.**



Por fim, alternou-se os modos de operação para que pudesse ter noção de como o sistema se comportaria em modo misto, saindo de um estado para o outro. O resultado mostrado na Figura 41 condiz com o esperado. Nos estágios iniciais há um grande uso de CPU, logo em seguida o sistema entra em modo de baixo consumo e apenas o *TaskManager* fica ativo e sendo executado mais lentamente. Após o modo de hibernação, o sistema entra em modo nominal e todas as tasks são executadas sem limite de CPU. E por fim, o sistema foi colocado em modo de baixo consumo, e apenas as *tasks* de controle(*WTD Task* e *TaskManager*) e manipulação de dados (*HouseKeeeping* e *DataManager*) foram executadas.

**Figura 41 – Snapshot no Tracealyzer do sistema executando em todos os modos.**



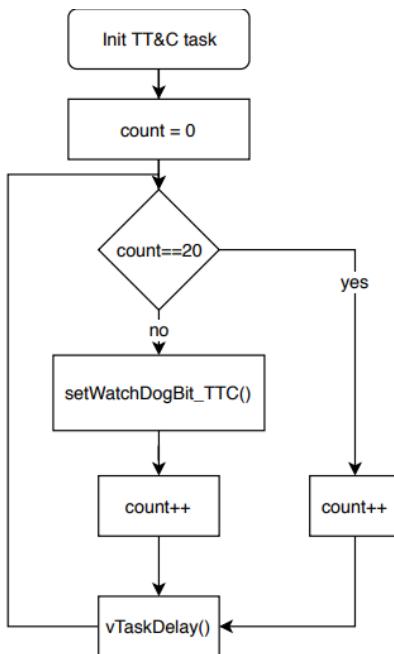
A utilização do programa *Tracealyzer* permitiu verificar a máquina de estados do *software* embarcado. Foi possível notar também que o *TaskManager* é capaz de gerenciar as *tasks* de acordo com algum *input* externo (Ex.: nível de bateria, luminosidade, etc). Também foi possível notar que o sistema em modo NM\_MODE ocupa 8 % de CPU, isso mostra que há aproximadamente 98 % de CPU ociosa que pode ser utilizada para processamento da Payload, TTC, PPT, etc.

## 5.4 Sistema Antitravamento

Uma das grandes preocupações no desenvolvimento do OBC estava na confiabilidade do sistema. O uso de componentes COTS põem em risco a vida útil da missão, pois tais componentes não possuem resistência à radiação. Sendo assim, foi utilizado no projeto três tipos de Watchdog: via software (*WatchDogTask*); interno (contador do MSP432); externo (STWD100).

Para testar a *WatchDogTask*, simulou-se um travamento na *task* do TT&C. Utilizou-se um contador que seria incrementado a cada loop. Se o contador fosse igual a 20, o TT&C não setaria o bit no *WATCHDOG\_EVENT\_GROUP*, simulando um travamento. Esse procedimento é mostrado na figura abaixo.

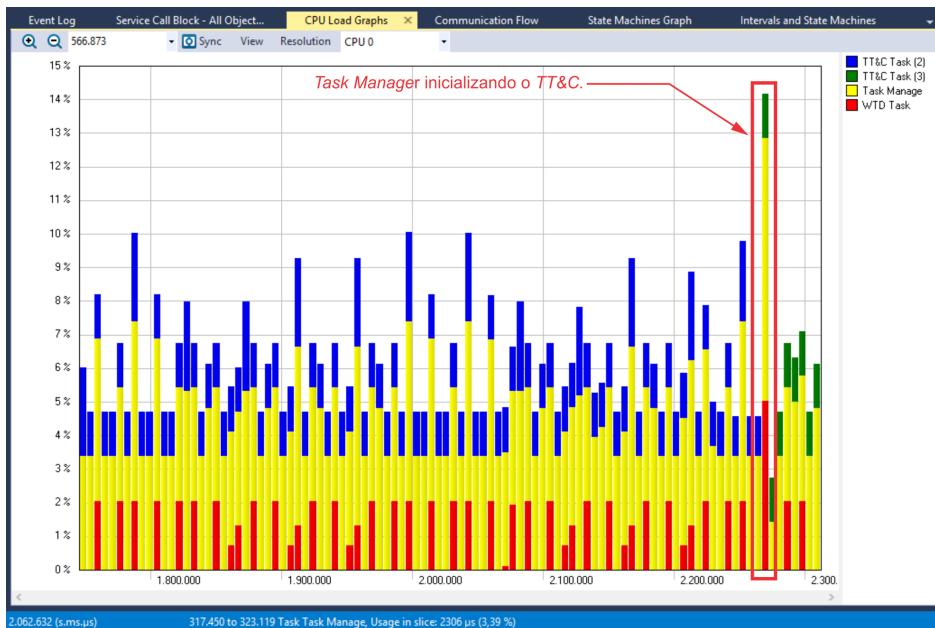
**Figura 42** – Procedimento para simulação de travamento do *TT&CTask*.



Como era de se esperar, a *WatchDogTask* conferiu os bits setados no handler e deletou a *TT&C Task*, conforme mostrado nas Figuras 43 e 44. Antes de ser delatada, a *task* do TT&C possuia a label **TT&C Task (2)** [azul] e, após ser reiniciada, mudou para **TT&C Task (3)** [verde].

**Figura 43 – Event Log no Tracealyzer do WatchDogTask sendo reinicializado.**

Timestamp	Actor	Event Text
2.270.136	WTD Task	free(0x20000A80) released 408 bytes
2.270.173	WTD Task	free(0x20000C18) released 96 bytes
2.270.213	WTD Task	vTaskDelete(TT&C Task (2))
2.270.213	TT&C Task (2)	Context switch on CPU 0 to TT&C Task (2)
2.270.213	WTD Task	Context switch on CPU 0 to WTD Task
2.270.278	WTD Task	xQueueSend(Queue #4)
2.270.306	WTD Task	vTaskDelayUntil(2283)
2.270.349	IDLE	Context switch on CPU 0 to IDLE
2.270.916	IDLE	OS Tick: 2270
2.271.916	IDLE	OS Tick: 2271
2.271.938	IDLE	Actor Ready: Task Manage
2.271.960	Task Manage	Context switch on CPU 0 to Task Manage
2.271.990	Task Manage	xQueueReceive(Queue #4)
2.272.015	Task Manage	malloc(408) returned 0x20000A80
2.272.053	Task Manage	malloc(96) returned 0x20000C18
2.272.131	Task Manage	xTaskCreate(TT&C Task (3))
2.272.144	Task Manage	Actor Ready: TT&C Task (3)
2.272.168	Task Manage	vTaskDelayUntil(2276)
2.272.210	TT&C Task (3)	Context switch on CPU 0 to TT&C Task (3)
2.272.242	TT&C Task (3)	vTaskDelayUntil(2276)

**Figura 44 – Grafico da CPU no Tracealyzer do WatchDogTask sendo reinicializado.**

No nível abaixo, há o *watchdog* do microcontrolador. Caso ocorra algum travamento no sistema operacional, o MSP432, a nível de *hardware*, reiniciará o sistema.

Por fim, na camada mais baixa há o *Watchdog* externo que resetará o OBC caso o MSP432 trave, trazendo mais robustez à solução. Não foi possível testar o componente STWD100 pois não ha o modulo COTS desse dispositivo, apenas sendo encontrado em montagem SMD.

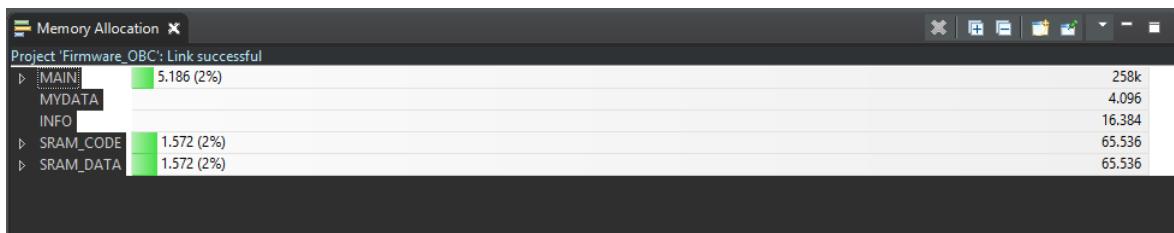
O uso do *Watchdog* em tres niveis trará mais confiabilidade ao OBC. Entretanto, essa solução só funcionará em casos de *Single Event Upset* (SEU), já em outros casos, como o SEL, é possível que haja danos permanentes no OBC.

## 5.5 Alocação de Memória do FreeRTOS e da CSS

Quantificar o espaço utilizado pelo *software* embarcado é extremamente importante, pois novas funcionalidades serão implementadas e o microcontrolador deverá possuir memória suficiente para acomodá-las. Para saber a memória utilizada pelo *kernel* e a camada de serviço do sistema, utilizou-se a ferramenta *Memory Allocation* da IDE do Code Composer.

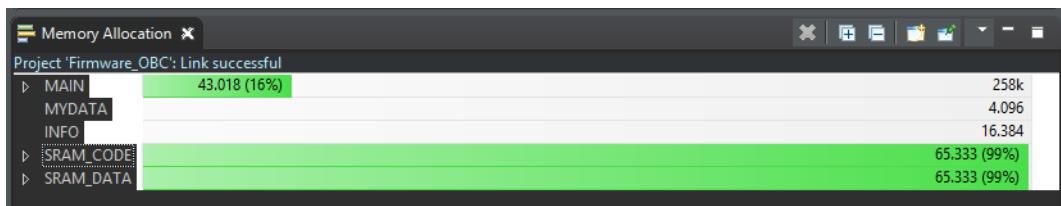
Desabilitando a inclusão do CLI, Tracealyzer e as *task* do CSS, o espaço ocupado pelo *FreeRTOS* foi de 5Kb para o código e 1,5Kb para a RAM. Isso mostra que o sistema operacional ocupa apenas 4% de memória FLASH e 2% de memória RAM, conforme é mostrado na Figura 45, possuindo um footprint pequeno. Esses valores seriam reduzido se fosse retirado algumas macros que não fazem parte do kernel. Esses valores comprovam que o *FreeRTOS* é uma ótima solução para sistemas que não possuem muita memória FLASH/RAM mas que necessitam de um sistema operacional de tempo real.

**Figura 45** – Valores de FLASH e RAM alocados para o *FreeRTOS*.



Quando há a inclusão de todas as bibliotecas, é percebido um aumento considerável na memória RAM, conforme mostrado na Figura 46. Esse aumento é devido ao uso do *Tracealyzer* que utiliza parte da memória RAM para o armazenamento das informações do sistema operacional. Mas como o *FreeRTOS* gerencia os recursos de CPU, essa alocação de 99% só é vista no boot inicial do sistema e em momentos de transição dos modos de operação. Em relação ao espaço do código, houve um aumento de 14% algo relativamente pequeno. Isso mostra que a Camada de Serviço do Sistema possui um *footprint* pequeno e que caso haja a necessidade de adicionar mais funcionalidades no sistema, há 86% (215Kb) de memória FLASH livre.

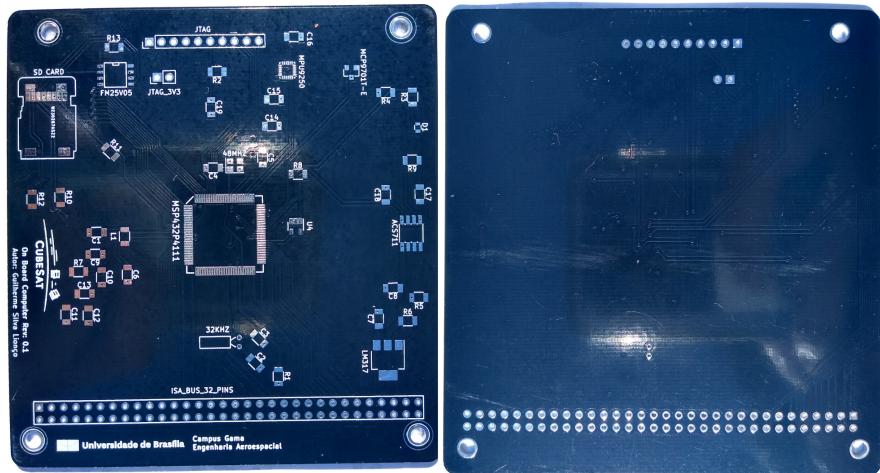
**Figura 46** – Valores de FLASH e RAM alocados para o *software* embarcado.



## 5.6 Printed Circuit Board

A partir do layout realizado no KiCad, gerou-se os arquivos .gerbers que contem todas as informações para a fabricação da PCB. Utilizou-se a fabricante PCBWay<sup>2</sup> para realizar a fabricação da PCB. O resultado final é mostrado na figura 47.

**Figura 47** – Vista superior e inferior da PCB.



A fabricação da PCB representa a etapa final que o projeto do *hardware* chegou. Os passos seguintes seriam a compra e soldagem dos componentes e, por fim, o teste do OBC. A seção 6.2 mostrará as instruções para os trabalhos futuros do projeto.

<sup>2</sup> Para mais informações, consulte: <<https://www.pcbway.com/orderonline.aspx>>



# 6 Conclusão e Recomendações

## 6.1 Conclusão

Este trabalho apresentou o projeto do *On Board Computer* para a futura missão entre a Universidade de Brasília e o Instituto de Aviação Varsóvia. Como foi mencionado durante o texto, a negociação do projeto estava em andamento durante todo o projeto de pesquisa, o que impossibilitou a coleta de requisitos e o custeio dos componentes eletrônicos. Tais pontos não desmerecem o projeto, pois houve um acúmulo de conhecimento enorme durante a pesquisa.

Inicialmente foi realizado uma conceituação básica do tema. Apresentou-se a categorização dos pequenos satélites e seus principais subsistemas, com foco nos nanosatélites. Mostrou-se a importância do OBC em CubeSats, bem como a comparação entre alguns OBCs utilizados em missões. Para fechar a parte introdutória, expôs-se o ambiente espacial e sua influência nos equipamentos eletrônicos. Esses tópicos serviram de base para o entendimento dos fatores a serem analisados durante o processo de desenvolvimento do OBC.

Logo em seguida, foi mostrado o procedimento utilizado para contornar a ausência de requisitos do OBC, procedimento que ajudou na limitação do escopo do projeto. Mostrou-se o critério utilizado para escolha do microcontrolador, componente mais importante do OBC. A partir desta primeira seleção, escolheu-se os outros componentes que o auxiliariam no controle do OBC, como sensor inercial, corrente, temperatura, etc.

Também foi visto alguns sistemas para diminuir os riscos dos efeitos da radiação sobre o microcontrolador, como o uso de watchdog em vários níveis. Por fim, para a parte de hardware foi realizado o layout do circuito impresso no KiCad.

Para o desenvolvimento do software utilizou-se a mesma alternativa para contornar os escassez de requisitos. Foi mostrado a arquitetura em camadas, proposta para ajudar na abstração dos componentes e interfaces. O pacote DriverLib foi utilizado como interface HAL, uma vez que suas APIs abstraem toda a tratativa dos registradores do microcontrolador. Para a camada acima, foi escolhido o FreeRTOS como sistema operacional do OBC, sendo uma ótima escolha para sistemas críticos de tempo real onde o poder de processamento é limitado. Também foi desenvolvido a Camada de Serviço do Sistema, utilizando os conceitos de *Unified Modeling Language* para facilitar a manutenção dessa camada em futuras melhorias.

Durante os testes com a launchpad, foi possível testar a coleta e armazenamento de dados, a solução antitravamento e consumo. A utilização da ferramenta Tracealyzer

facilitou a visualização dos modos de operação do sistema operacional. Tal ferramenta ajudou na validação dos modos de operação do *software* embarcado. Foi visto que o consumo do microcontrolador atende aos requisitos de baixo consumo de um CubeSat.

Alguns pontos não foram abordados, ou desenvolvidos, durante a pesquisa devido algumas dificuldades que o aluno passou durante o desenvolvimento do OBC. Entretanto, tais pontos não desmerecem o trabalho, visto que foi possível desenvolver grande parte do software embarcado e o desenvolvimento por completo do layout da PCB.

Algo importante a ser mencionado é o benefício que a utilização da plataforma GitHub trouxe ao projeto, todo o código ficou versionado no repositório do aluno. Ao todo houveram trinta e dois commits, ou seja, trinta e duas versões do código. Caso haja uma modificação muito drástica no código e este não funcione perfeitamente, é possível voltar para uma versão mais estável. Outro ponto importante é que o GitHub serviu como divulgação da pesquisa. Até o presente momento, 135 visualizações e 16 clones no repositório do projeto.

## 6.2 Recomendações

Os trabalhos futuros são de extrema importância para a melhoria do OBC e êxito da futura missão. Como este foi um dos primeiros trabalhos pesquisas realizado na missão, vários empecilhos foram desconsiderados durante a fase inicial da pesquisa. Tais pontos devem ser considerados nos próximos trabalhos.

A compra e soldagem dos componentes é um ponto extrema prioridade, pois não foi possível testar a placa desenvolvida no projeto. Alguns componentes só poderão ser testados ao serem soldados na PCB, como é o caso do watchdog externo e da memória FRAM.

Outro ponto que deve ser aprofundado é a utilização do clock de 32KHz como fonte de sincronismo do *SysTick*, durante o modo de hibernação do satélite. Foi visto que a utilização de um único clock, tanto para o modo de alto desempenho quanto para o modo de hibernação, não torna o OBC robusto em cenários de baixa bateria. Fazendo uma estimativa, o MSP432 aumentaria a vida útil da bateria, em modo de hibernação, em mais de duas mil vezes, consumindo 11uW caso utilizasse o clock de 32KHz<sup>1</sup>.

O uso de vários níveis de watchdog não é necessário para diminuição do risco dos efeitos da radiação sobre o OBC. A utilização de componentes COTS diminuem a confiabilidade do sistema e outras formas de proteção devem ser analisadas. Recomenda-se que haja pesquisas em *shields* metálicos para proteção de todo o sistema embarcado, mas levando em consideração a massa deste para não onerar o lançamento do satélite,

---

<sup>1</sup> Para mais detalhes, consultar Apêndice C

visto que os lançadores cobram em torno de USD 20/kg<sup>2</sup>.

Foi constatado que o escopo do projeto foi subestimado e o projeto tem uma proporção bem maior do que foi imaginado. O desenvolvimento do OBC, tanto para o *hardware* quanto para o *software* utilizando apenas um aluno se torna inviável. O uso da metodologia co-design é ideal para engenheiros que possuem grande experiência no desenvolvimento de soluções embarcadas, mas a nível de graduação, ela se torna muito difícil pois há inúmeros pontos que o aluno pode travar durante o projeto. Sugere-se que o projeto seja dividido para no mínimo dois alunos, um para o *hardware* e outro para o *software*.

Por fim, poderia ser abordada a parceria com o grupo de pesquisa LAICAnSat. Essa parceria seria ideal para o desenvolvimento de outros subsistemas da missão e teste do OBC, como *payload* de teste, em lançamentos com balão estratosférico. Os testes seriam possível pois a interface de comunicação do sistema embarcado do LAICAnSat também utilizam o barramento ISA BUS e o padrão PC104.

É encorajado o prosseguimento desse projeto e a abertura de demais temas para projeto de pesquisa e trabalho de graduação. O seguimento aeroespacial é um dos mais que possibilitam a interdisciplinaridade entre os cursos.

---

<sup>2</sup> Para mais detalhes, consultar a página 18: <<http://www.scielo.br/pdf/jatm/v9n3/2175-9146-jatm-09-03-0269.pdf>>



## 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](http://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](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](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](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](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](https://goo.gl/rALuMR)>. Acesso em: 03 jun. 2018.

Cohen R., Wang T. **.Overview of Embedded Application Development for Intel Architecture.** In: Android Application Development for the Intel® Platform. 2014. Apress, Berkeley, CA. Disponível em: <<https://link.springer.com/content/pdf/10.1007%2F978-1-4842-0100-8.pdf>>. Acesso em: 06 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](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](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](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](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.

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](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](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)- Universidade de São Paulo, São Paulo, 2010.

ria de Software)- University of York, Departamento de Ciência da Computação, Estados Unidos, 2010. Disponível em: <[goo.gl/R397j8](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](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](https://www.qb50.eu/index.php/tech-docs/category/QB50_system_requirements_issue_606e0.pdf?download=58:qb50-docs)>. Acesso em: 06 jun. 2018.

MICROCHIP. Low-Power Linear Active Thermistor ICs. [S. l.], 2016. Disponível em: <<http://ww1.microchip.com/downloads/en/devicedoc/20001942g.pdf>>. Acesso em: 30 set. 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](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](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\\_CSX\\_FM430\\_710-00252-C.pdf](http://www.cubesatkit.com/docs/datasheet/DS_CSX_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](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 18 – Análise de OBCs em Missões Passadas**

Missão				Microcontrolador												
Informações Gerais				Especificações Gerais				Conversores		Interface e Portas		Informações Adicionais				
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	Demonstrador 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 AT712D 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 Observação da Terra	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	Keil RTX166 PIC 16F877	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			-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
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 19. Nessa tabela, a divisão dos requisitos ocorre da seguinte maneira: [1] são os da própria missão, [2] QB50 e [3] Aalto-1.

**Tabela 19** – 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 os 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 20** – Modos de Operação e Consumo do MSP432P4111

MODOS DE OPERAÇÃO	DESCRÍÇÃO	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 21** – 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
<b>TOTAL</b>	360	bits
	45	bytes

**Tabela 22** – 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 24** – 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
<b>TOTAL</b>	120	bits
	15	bytes

**Tabela 25** – 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	
<b>TOTAL</b>	32	bits
	4	bytes

**Tabela 26 – 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 28** – 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
<b>TOTAL</b>	96	bits
	12	bytes

**Tabela 29** – 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
<b>TOTAL</b>	48	bits
	6	bytes

**Tabela 30** – 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
<b>TOTAL</b>	32	bits
	4	bytes

**Tabela 31** – PAYLOAD - Image Sensor

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

**Tabela 32** – Quantidade total de dados.

TELEMETRIA		PAYLOAD		
210	bytes	a cada 2 s	115208	bytes a cada 5 min
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

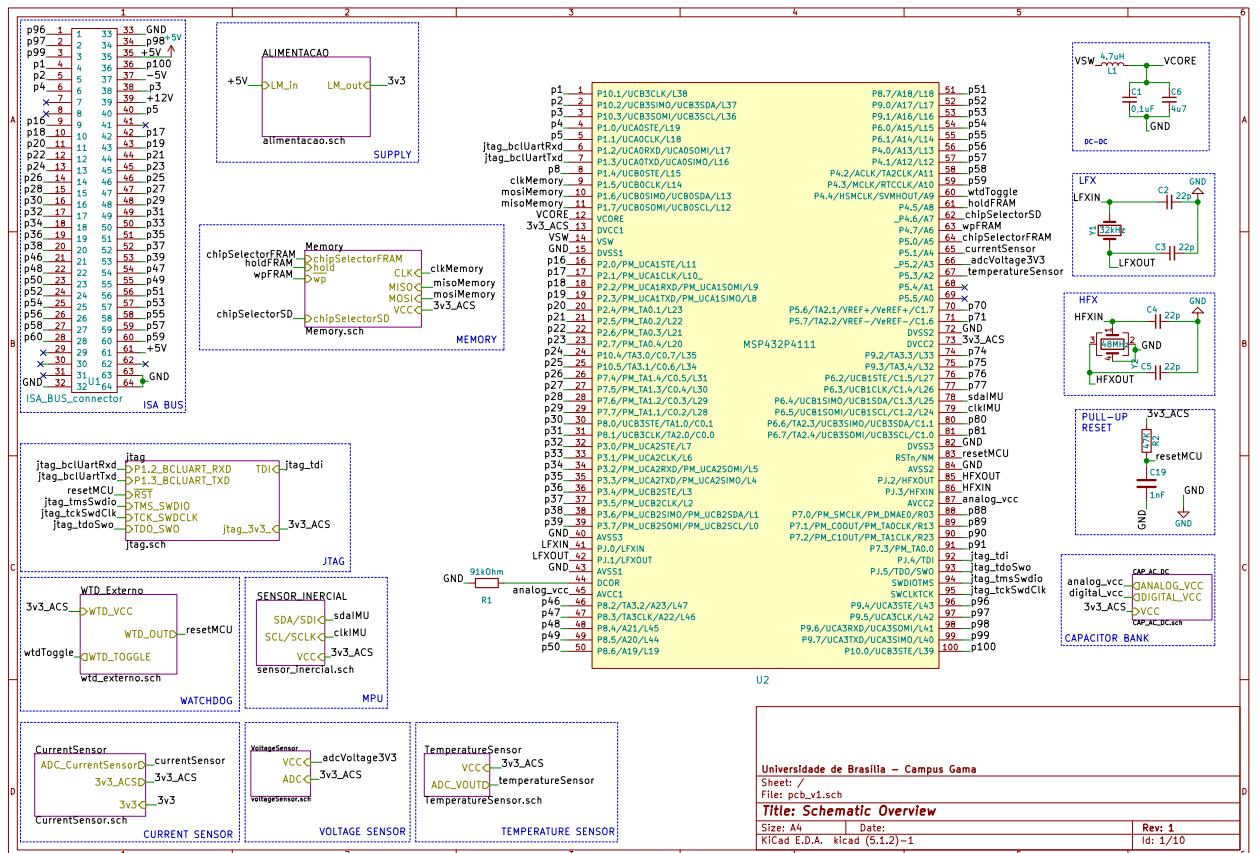
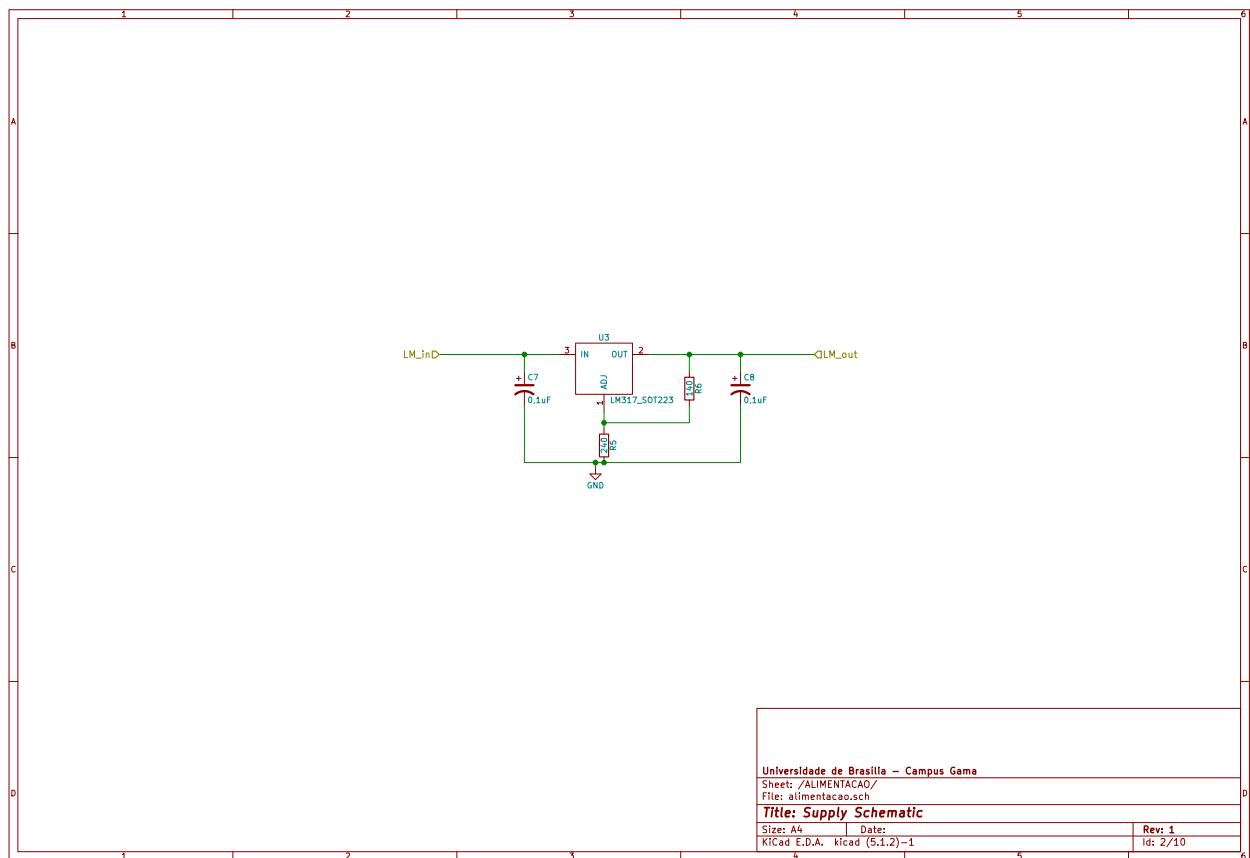
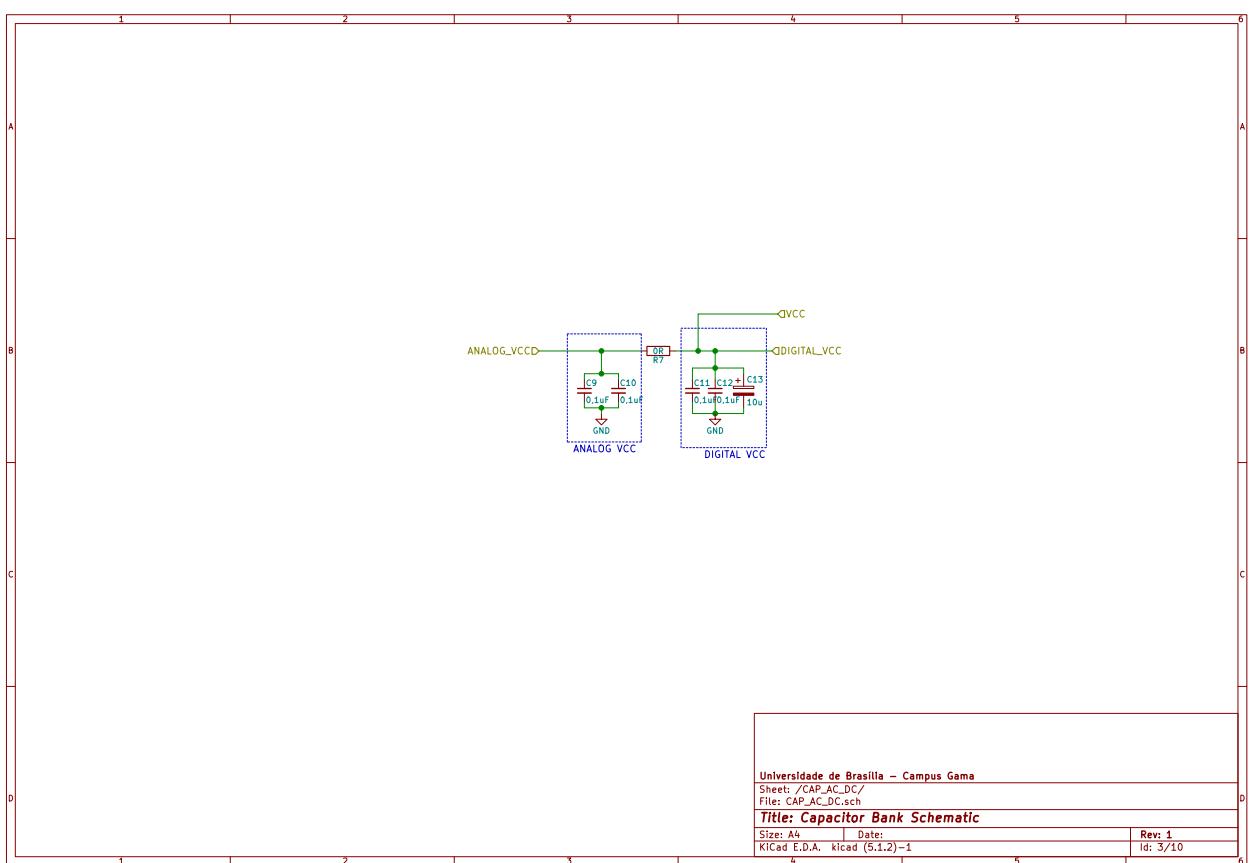


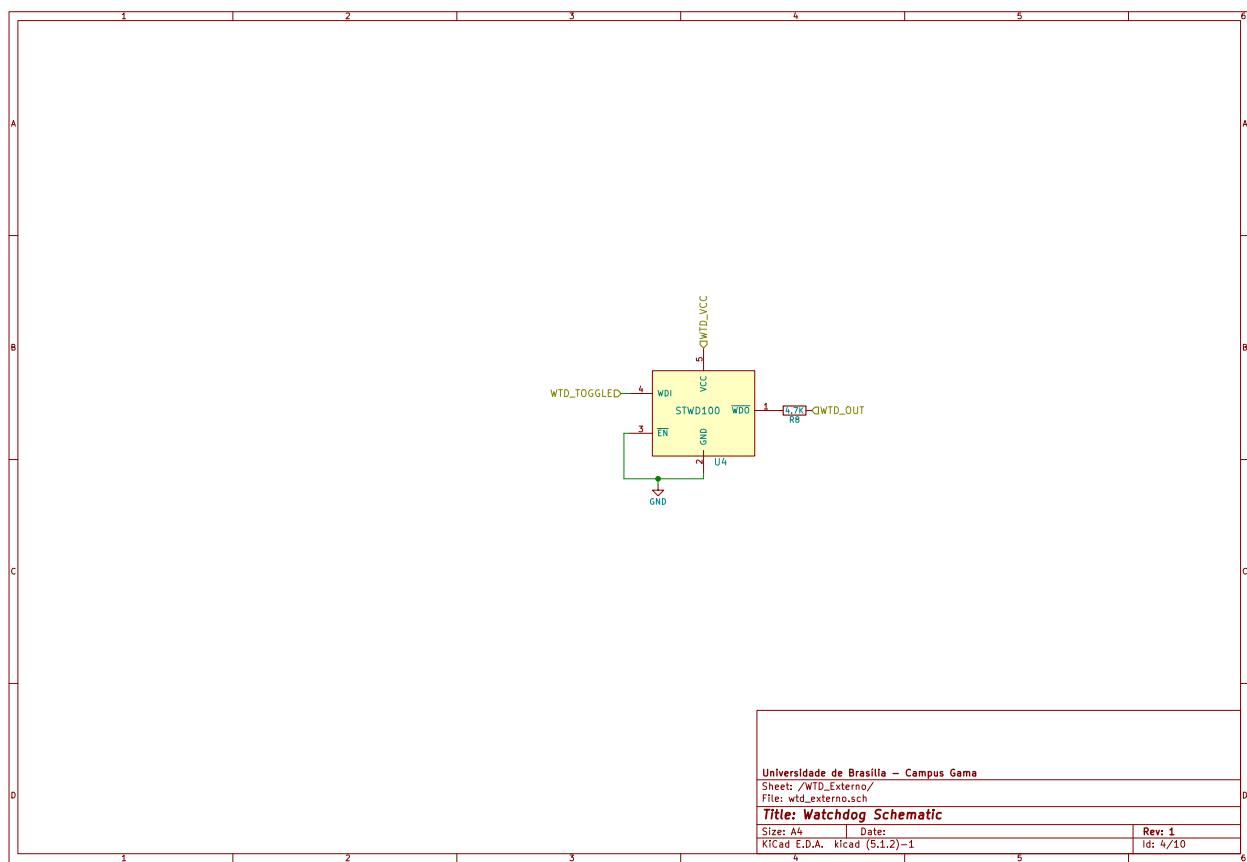
Figura 48 – Esquemático Eletrônico do OBC.



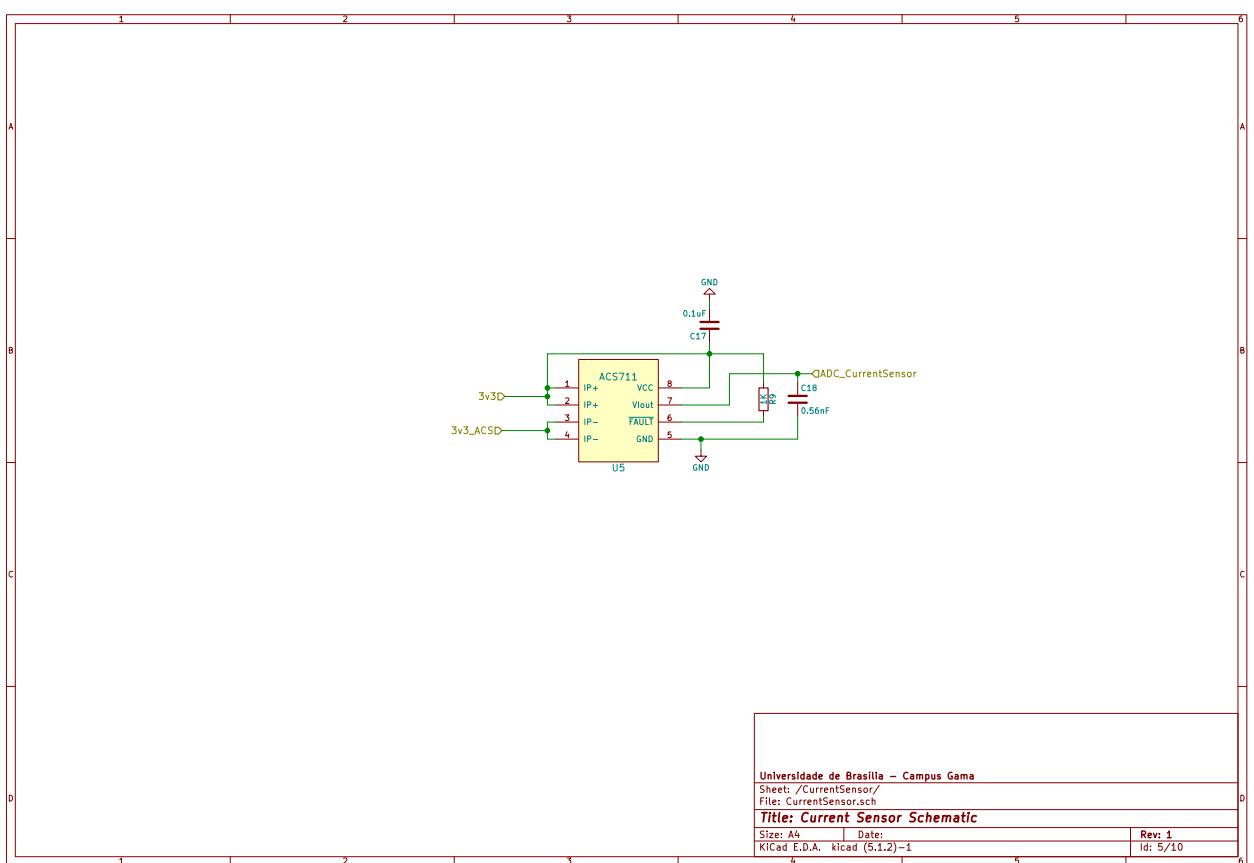
**Figura 49** – Esquemático Regulador de Tensão.



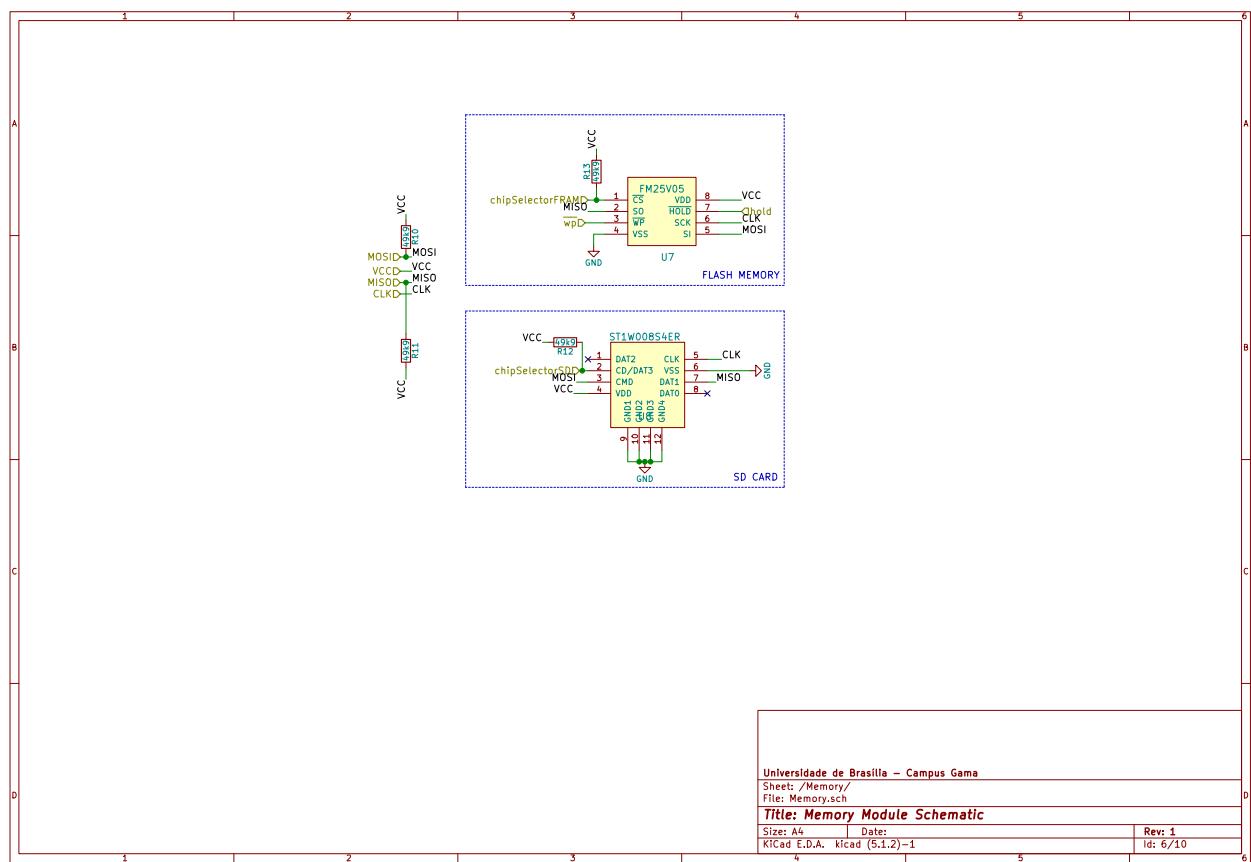
**Figura 50** – Esquemático Banco de Capacitores.



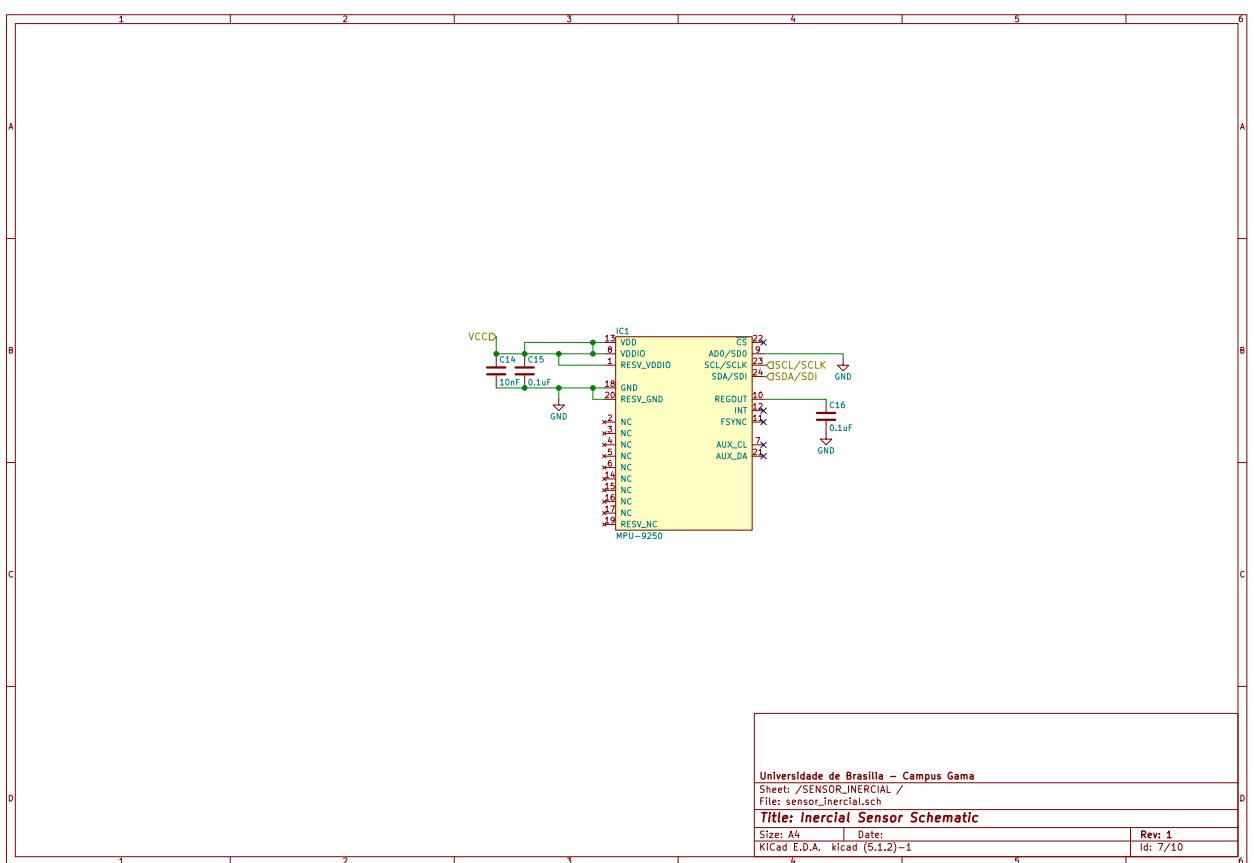
**Figura 51** – Esquemático Watchdog Externo.



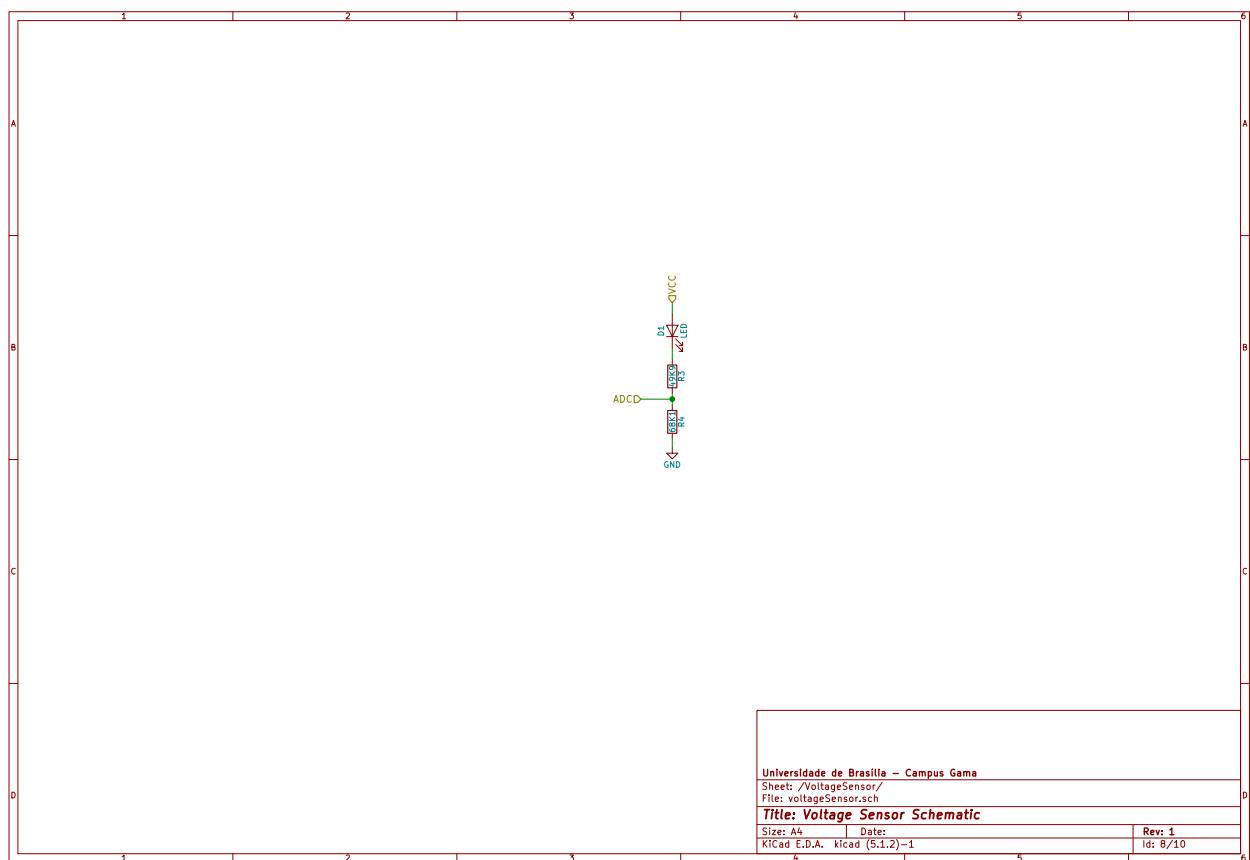
**Figura 52** – Esquemático Sensor de Corrent.



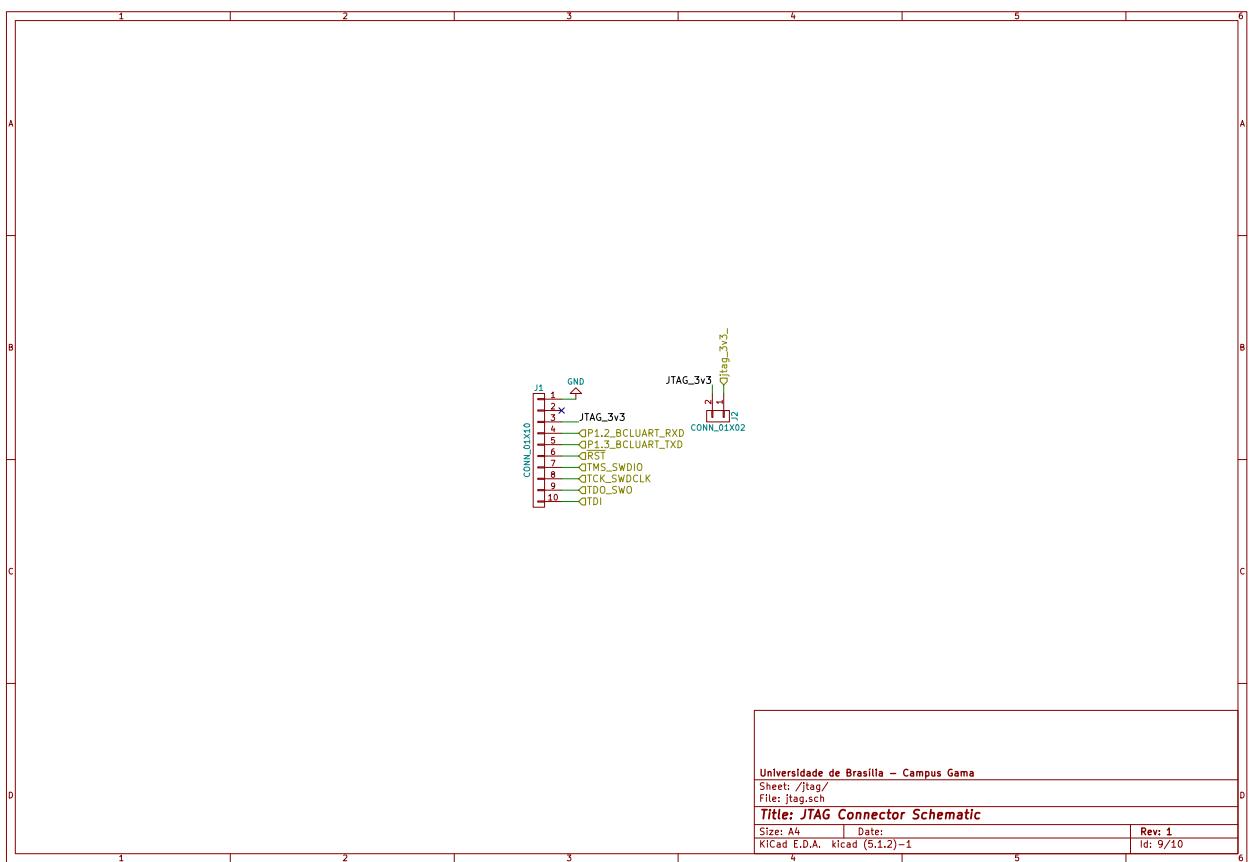
**Figura 53** – Esquemático Armazenamento de Dados.



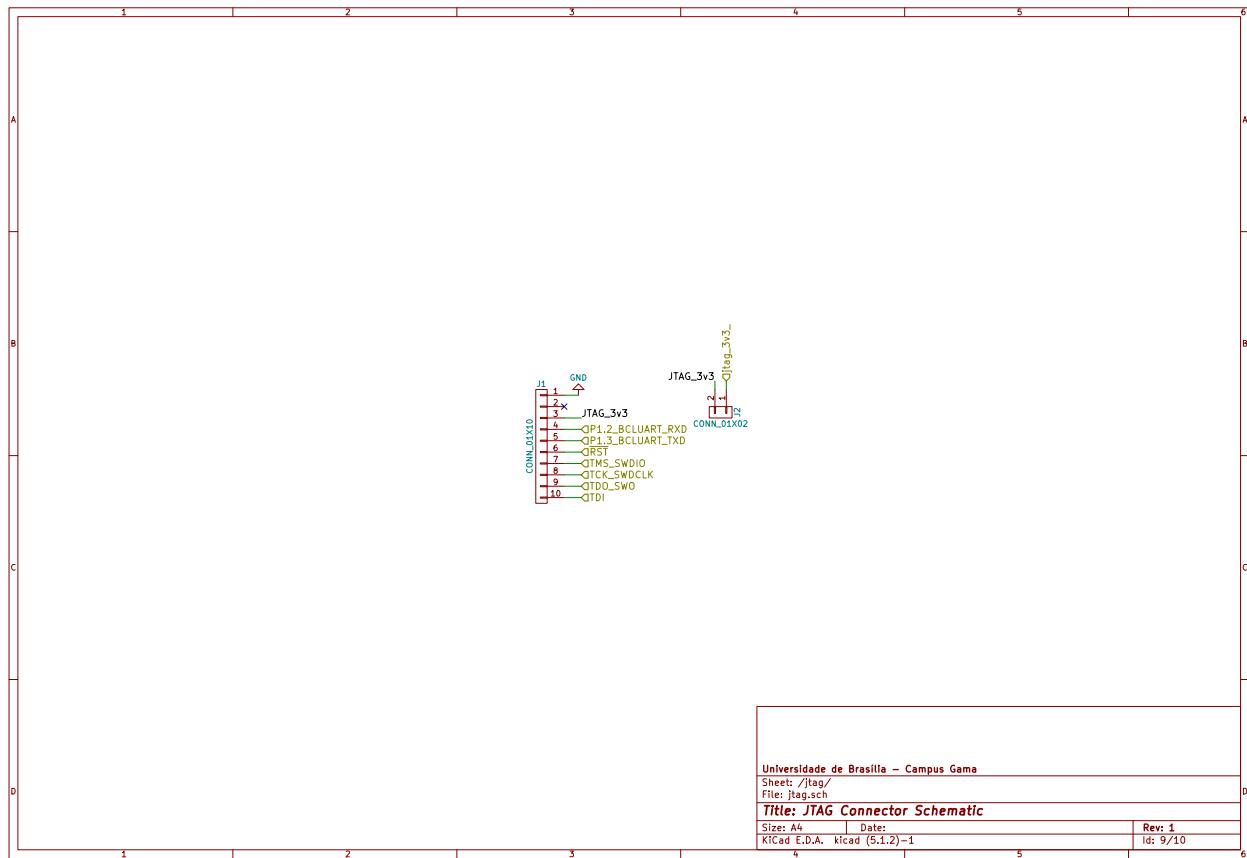
**Figura 54 – Esquemático Sensor Inercial.**



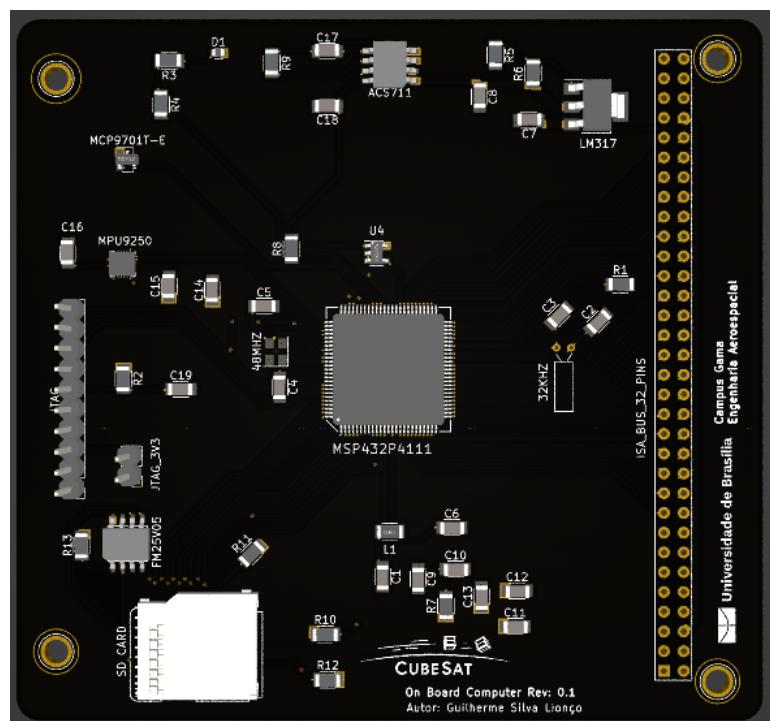
**Figura 55** – Esquemático Medidor de Tensão.



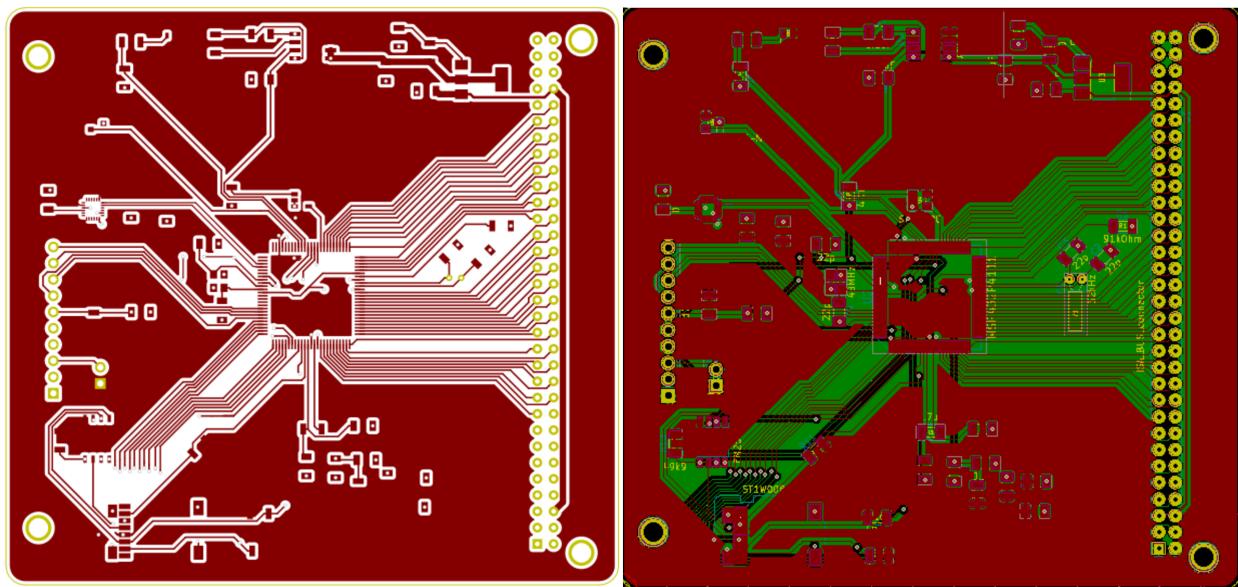
**Figura 56** – Esquemático JTAG.



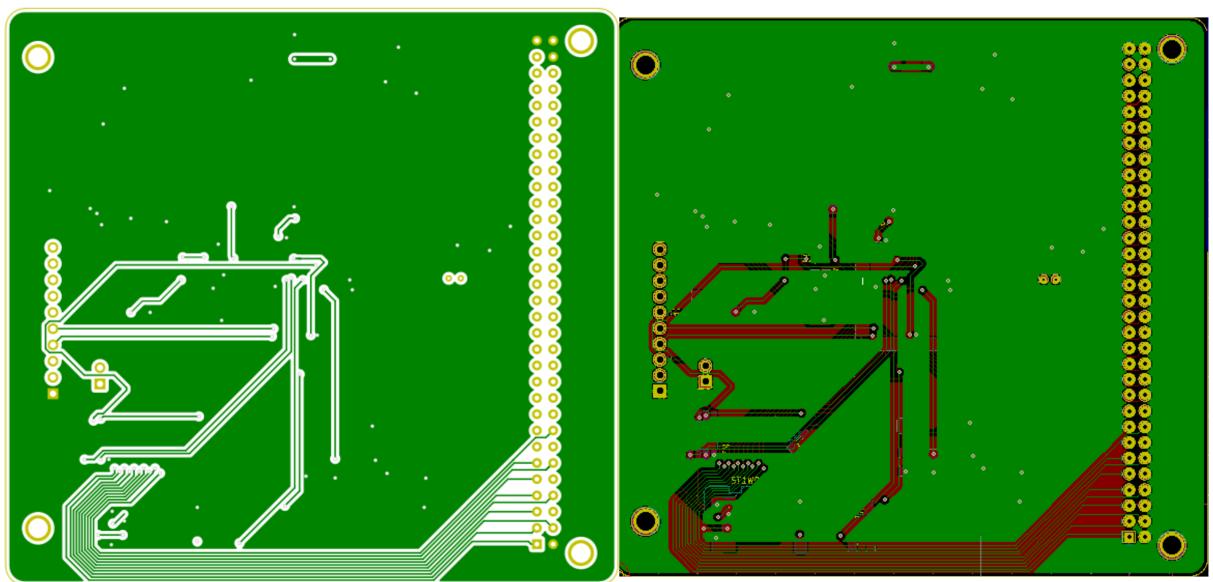
**Figura 57** – Esquemático Sensor de Temperatura.



**Figura 58** – PCB - Vista Isométrica.



**Figura 59 – PCB - Plano de Cobre Superior [3V3].**



**Figura 60 – PCB - Plano de Cobre Inferior [GND].**



# APÊNDICE F – Bill of Material

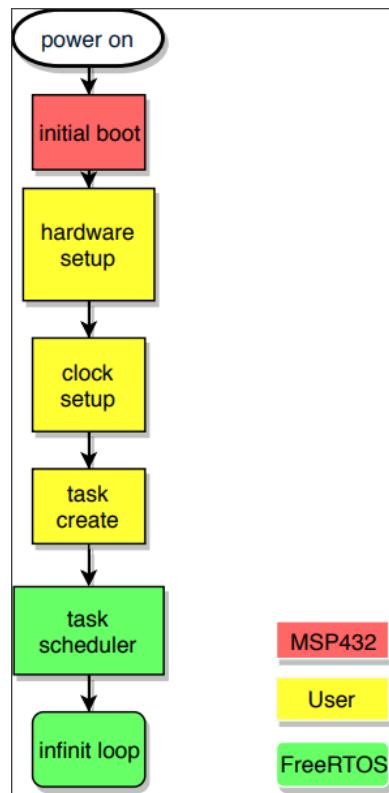
Nº	Quant.	Part Number	Manufacturer Part Number	Description	Available	Unit Price	Total Price
1	2	478-2368-1-ND	TAJA104K035RNJ	CAP TANT 0.1UF 10% 35V 1206	2	0,49	0,98
2	2	490-10390-1-ND	GRM033C80J104KE15D	CAP CER 0.1UF 6.3V X6S 0201	2	0,1	0,20
3	6	311-1179-1-ND	CC1206KRX7R9BB104	CAP CER 0.1UF 50V X7R 1206	6	0,24	1,44
4	1	478-1653-1-ND	TAJA106K006RNJ	CAP TANT 10UF 10% 6.3V 1206	1	0,34	0,34
5	1	490-7194-1-ND	GRM033R61E103KA12D	CAP CER 10000PF 25V X5R 0201	1	0,1	0,10
6	1	RMCF1206ZT0R00CT-ND	RMCF1206ZT0R00	RES 0 OHM JUMPER 1/4W 1206	1	0,1	0,10
7	1	RMCF1206FT140RCT-ND	RMCF1206FT140R	RES 140 OHM 1% 1/4W 1206	1	0,1	0,10
8	1	RNCP1206FTD1K00CT-ND	RNCP1206FTD1K00	RES 1K OHM 1% 1/2W 1206	1	0,1	0,10
9	1	P240BCCT-ND	ERA-8AEB241V	RES SMD 240 OHM 0.1% 1/4W 1206	1	0,66	0,66
10	1	A130181CT-ND	CRGCQ1206J4K7	CRGCQ 1206 4K7 5%	1	0,1	0,10
11	1	RMCF1206JT47K0CT-ND	RMCF1206JT47K0	RES 47K OHM 5% 1/4W 1206	1	0,1	0,10
12	5	P49.9KBCCT-ND	ERA-8AEB4992V	RES SMD 49.9K OHM 0.1% 1/4W 1206	5	0,66	3,30
13	1	RNCP1206FTD68K1CT-ND	RNCP1206FTD68K1	RES 68.1K OHM 1% 1/2W 1206	1	0,1	0,10
14	1	311-91.0KFRCT-ND	RC1206FR-0791KL	RES SMD 91K OHM 1% 1/4W 1206	1	0,1	0,10
15	1	399-17482-1-ND	C1206C511J5GAC7800	CAP CER 510PF 50V NPO 1206	1	0,45	0,45
16	1	399-1222-1-ND	C1206C102K1RACTU	CAP CER 1000PF 100V X7R 1206	1	0,32	0,32
17	6	490-11621-1-ND	GRM31A5C2J220JW01D	CAP CER 22PF 630V COG/NPO 1206	6	0,47	2,82
18	1	399-1262-1-ND	C1206C475Z4VACTU	CAP CER 4.7UF 16V Y5V 1206	1	0,27	0,27
19	1	SML-D12U1WT86CT-ND	SML-D12U1WT86	LED RED DIFFUSED 1608 SMD	1	0,22	0,22
20	1	SER3310-ND	C-2 32.0000K-P:PBFREE	CRYSTAL 32.0000KHZ 11PF T/H	1	0,94	0,94
21	1	490-6706-1-ND	LQM31PN4R7M00L	FIXED IND 4.7UH 700MA 300 MOHM	1	0,4	0,40
22	1	535-11305-1-ND	ABM3C-48.000MHZ-D4Y-T	CRYSTAL 48.0000MHZ 18PF SMD	1	0,86	0,86
23	1	620-1370-1-ND	ACS711ELCTR-12AB-T	SENSOR CURRENT HALL 12.5A AC/DC	1	2,89	2,89
24	1	428-3213-ND	FM25V05-G	IC FRAM 512K SPI 40MHZ 8SOIC	1	13,89	13,89
25	1	296-12602-1-ND	LM317DCYR	IC REG LIN POS ADJ 1.5A SOT223-4	1	0,8	0,80
26	1	MCP9701T-E/TTCT-ND	MCP9701T-E/TT	SENSOR ANALOG -10C-125C SOT23-3	1	0,26	0,26
27	1	670-1528-1-ND	ST1W008S4ER1500	CONN MICRO SD CARD HINGED TYPE	1	1,99	1,99
28	1	497-10058-1-ND	STWD100NYWY3F	IC WATCHDOG TIME CIRCUIT SOT23-5	1	0,85	0,85
29	1	296-50599-ND	MSP432P4111IPZ	IC MCU 32BIT 2MB FLASH 100LQFP	1	13,35	13,35
30	1	1428-1011-1-ND	MPU-6500	IMU ACCEL/GYRO 3-AXIS I2C 24QFN	1	8,31	8,31

Figura 61 – Bill of Material do OBC

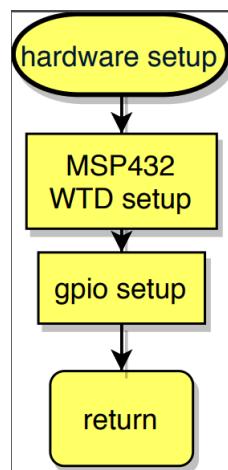


# APÊNDICE G – Fluxogramas da Camada de Serviço

Este apêndice contem os fluxogramas da Camada de Serviço do Software Embocado.



**Figura 62** – Fluxograma de Inicialização.



**Figura 63** – Fluxograma de Inicialização do Hardware.

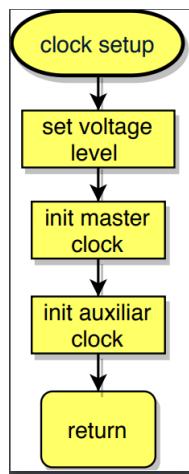


Figura 64 – Fluxograma de Inicialização do Clock.

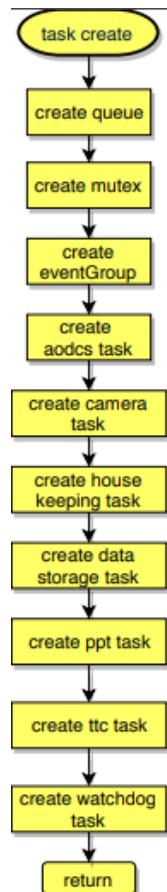
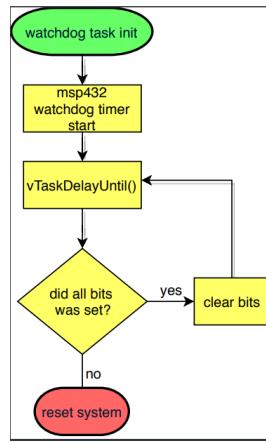
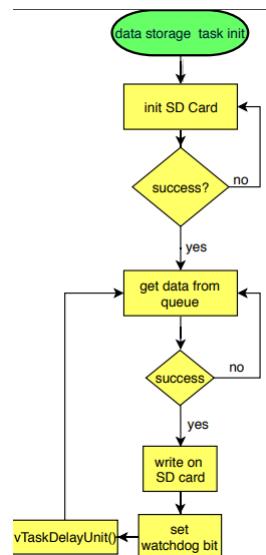


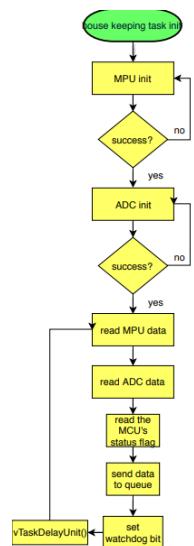
Figura 65 – Fluxograma de criação das Tasks.



**Figura 66** – Fluxograma do *Watchdog Task*.



**Figura 67** – Fluxograma do *DataStorage Task*.



**Figura 68** – Fluxograma do *HouseKeeping Task*.