UnB On-Board Computer Prototype for CubeSats

G. Silva Lionço(1), G. Santilli (2), L. Aguayo (3).

(1) Universidade de Brasilia
Faculdade Gama Brasília, Brazil
Phone: +55 61 9 9227-8739, Mail: guilherme.lionzo@gmail.com
(2) Universidade de Brasilia
Faculdade Gama Brasília, Brazil
Phone: +55 61 9 8355-0174, Mail: santilli@aerospace.unb.br
(3) Universidade de Brasília
Faculdade Gama Brasília, Brazil
Phone: +55 61 phone, Mail: aguayo@unb.br

Abstract: Researchers from the University of Brasília (UnB) 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. It was concluded that the use of the TI MSP432 is a great choice for low-power and intermediate performance scenarios. The use of FreeRTOS as a real-time operating system for low memory systems, as well as the use of watchdog utilization at software level has been ratified.

1. INTRODUÇÃO

Imerso em um processo global de capacitação e melhoria do conhecimento que visa alcançar a autonomia e independência na implementação de pequenas missões por satélite, pesquisadores da Universidade de Brasília (UnB) estão estudando a viabilidade de uma missão CubeSat 3U. Essa futura 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.

Alguns estudos já estão sendo realizados, com o intuito de oferecer soluções para essa futura missão. Esse artigo visa retratar as lições aprendidas na prototipagem do UNB OBC. Nas seções a seguir serão mostradas

2. REQUISITOS DO OBC

Os requisitos levantados na fase inicial do projeto do OBC são mostrados abaixo.

- Controlar uma Câmera CMOS (*Complementary Metal-Oxide-Semiconductor*);
- Controlar um PPT (do inglês Pulsed Plasma Thruster);
- Controlar um Sensor Inercial;
- Garantir um subsistema com um alto nível de confiança, mesmo não utilizando dispositivos resistentes à radiação;

- Armazenar dados em um memoria não volátil;
- Possuir um sistema antitravamento;
- Alterar modos de operação de acordo com o nível de bateria;

3. DESENVOLVIMENTO DO OBC

Para o desenvolvimento do OBC, utilizou-se a metodologia *Co-Design*, o que significa que o desenvolvimento do hardware e do software aconteceram simultaneamente. Nesta seção, serão mostrados os componentes escolhidos para a arquitetura do hardware e a solução de software. A Figura 1 mostra a arquitetura do OBC, bem como as interfaces para a comunicação com os outros subsistemas.

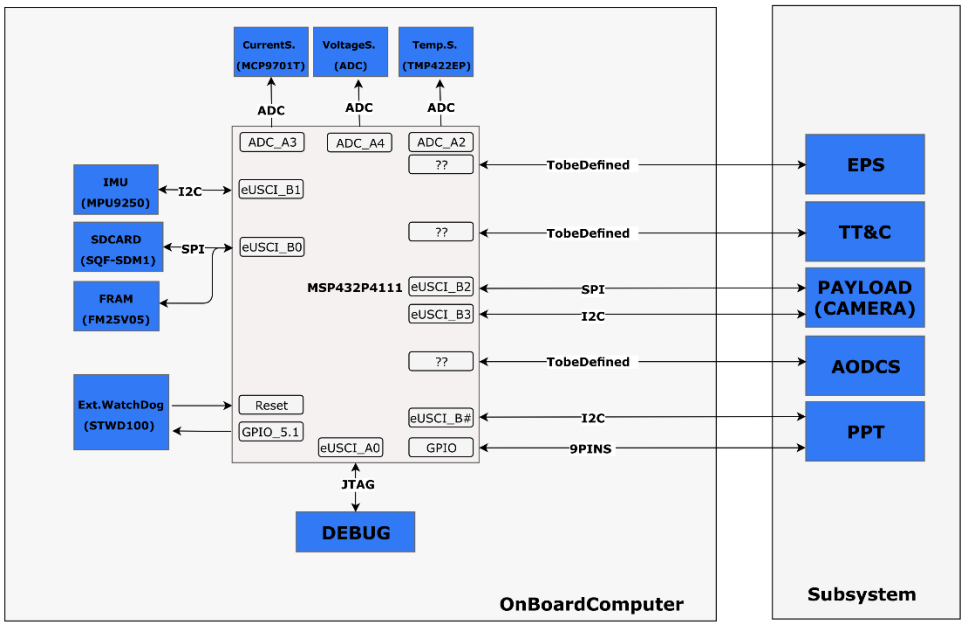

Figura 1 - Arquitetura do OBC.

3.1. Arquitetura do Hardware

O desenvolvimento do hardware começou primeiramente pela seleção do microcontrolador, pois ele é o componente que delimitará a performance do OBC. O critério de seleção levou em conta: Baixo Consumo; ADC; GPIO; Interface Serial; PWM; Clock. A partir desses critérios, procurou-se por microcontroladores das fabricantes mais usuais, como a *Microchip* e *Texas Instruments*, e optou-se pela seleção de microcontroladores que possuíssem uma placa de desenvolvimento, para viabilizar o tempo de implementação.

O microcontrolador selecionado foi o MSP432P4111 devido sua performance intermediária e baixo consumo. Na seção a seguir há uma breve descrição do microcontrolador.

3.1.1. 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 baixo consumo, eficiência, boa capacidade de processamento intermediaria, baixo custo e versatilidade. Tal microcontrolador, conhecido por ter um baixo consumo de operação, possui 18 modos de operação e em média consome 520uW/MHz. Ele conta com 4 timers de 16-bits, 24 ADC de 14-bit,8 interfaces de comunicação serial, bloco de real-time clock (RTC) e mais de 84 pinos de I/O. O

MSP432P4111 possui 2048 KB Memória Flash principal; 32KB Memória Flash de informação (área para Bootloader, TVL e Flash MailBox); 256KB SRAM, incluindo 8KB de memória de backup. [1]

3.1.2. Memory Unit

A unidade de armazenamento é responsável por armazenar os dados provenientes do OBC, possuindo duas subunidades: 1 - armazenamento da telemetria e *payload*; 2 - backup do software embarcado. Para realizar a estimativa de dados gerados em um dia, utilizou-se os parâmetros de armazenagem da missão *SWISSCube* [2].

A estimativa de dados gerados em um dia foi de 176MB, 10MB provenientes da telemetria e 166MB da Payload (imagens). Como não se sabe a quantidade de estações terrenas disponíveis para descarregar os dados, escolheu-se o tamanho máximo que o microcontrolador poderia suportar, que no caso é de 4GB, dando 22 dias de armazenamento.

Já para o armazenamento do backup do software embarcado, escolheu-se uma memória FRAM, devido sua resistência à efeitos TID e SEU [3]. O tamanho definido foi o dobro da memória SRAM do microcontrolador, resultando em 512KB.

3.1.3. Periféricos

Os periféricos são componentes que, assim como a unidade de armazenamento, tem o objetivo de auxiliar o microprocessador a cumprir os requisitos do OBC. Essa unidade é composta basicamente de quatro componentes: sensor de corrente, sensor de inercial, sensor de temperatura, Watchdog externo (contador). O sensor de corrente escolhido foi o ACS70331, sensor que também serve para proteção de *overcurrent* [4]. O sensor inercial MPU9250 é destinado à aferição da aceleração, campo magnético e rotação nos três eixos acelerômetro, magnetômetro nos 3 eixos [5]. O sensor de temperatura MCP9701T é de fácil usabilidade e baixo consumo, sendo capaz de medir temperaturas com uma precisão de 2°C em uma faixa de temperatura de -40°C a 125ºC [6]. Por fim, o contador STWD100 tem o objetivo de trazer maior robustez ao OBC, em casos de evento SEE, servindo como Watchdog externo [7].

3.1.4. Interfaces

O OBC possui duas interfaces principais, a primeira é destinada ao “debug” do microcontrolador e a segunda é destinada à comunicação com outros subsistemas do satélite. A interface de debug é destinada à fase de projeto e testes do OBC, e é composta pelos pinos do MSP432 de JTAG (*Joint Test Action Group*) e SWD (*Serial Wire Debug*) [8]. A segunda interface é reservada para a comunicação com os subsistemas do satélite, sendo feita por meio do barramento ISA de 16bit, comumente utilizado em placas do padrão PC104 [9].

3.2. Arquitetura do Software

Para facilitar o desenvolvimento e a abstração de algumas interfaces do software embarcado, utilizou-se a arquitetura em camadas. Isso faz com que o usuário não precise ter noção de partes muito específicas do sistema, facilitando a usabilidade e manutenção do *software.* Abaixo há a arquitetura de software proposta para o OBC.

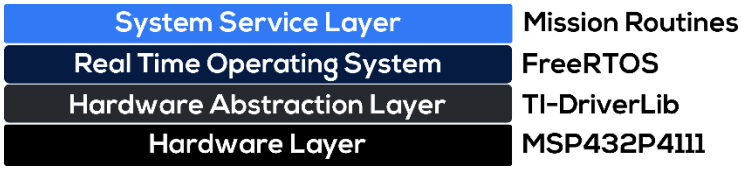

Figura 2 – Arquitetura de abstração em Camadas.

Camada de Abstração de Hardware
A camada de HAL é desempenhada pelo pacote *Driver Library* (*DriverLib*), 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. [10]

Camada do Sistema Operacional
O RTOS escolhido para o software embarcado foi o *FreeRTOS* devido sua vasta utilização em missões CubeSat. Esse kernel, desenvolvido e mantido pela Real Time Engineers Ltd, é distribuído gratuitamente sobre a licença General Public License (GPL). No contexto do FreeRTOS, cada tarefa em execução é chamada de 'task'. No contexto do projeto, o uso das tasks é fundamental para criar um certo nível de abstração e garantir o requisito de Hard-RTOS. [11]

Camada de Serviço do Sistema
A Camada de Serviço do Sistema (CSS) é a camada onde há a implementação das rotinas destinas à missão. No desenvolvimento dessa camada, utilizou-se a padronização *FuncionalC,* extensão da UML que permite a modelagem de sistemas baseados na linguagem C [12]. Foram elaborados a Máquina de Estados e o Diagrama de Arquivos.

A máquina de estados contem quatro modos de operação: **Nominal Mode**, **Safe Mode**, **Battery Low Level Mode** e **Hibernate Mode**, conforme mostrado na Figura 3 abaixo. Vale a pena ressaltar que não foi levado em consideração os estágios de pré-lançamento e *deployment*.

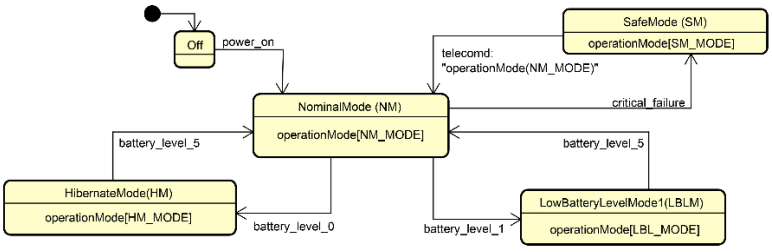

Figura 3 - Máquina de Estados da Camada de Serviço.

O Diagrama de Arquivos 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 referentes aos subsistemas do CubeSat.

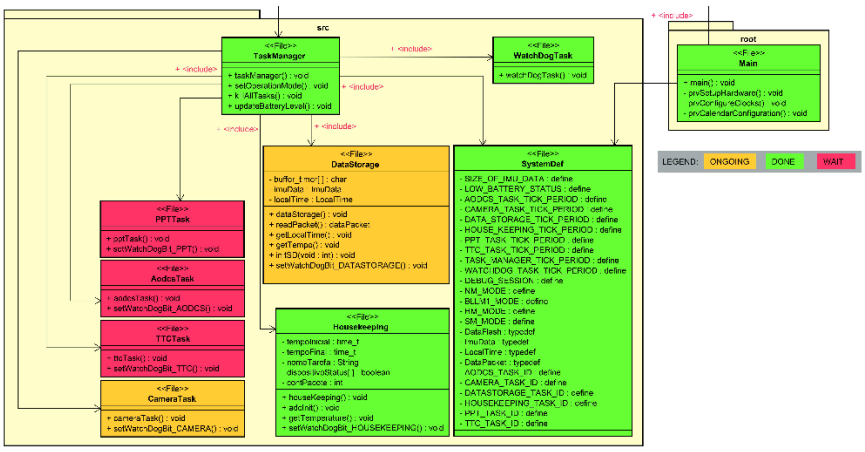

Figura 4 - Diagrama de Arquivos do Sistema.

4. RESULTADOS

Como o hardware do OBC estava em fase de construção, o software embarcado foi simulado na *LaunchPad* do MSP432. Utilizou-se um *photoresistor* para simular o nível de bateria do EPS do CubeSat. Essa abordagem condiz, de certa forma, com a realidade pois a incidência de luz interfere na quantidade de energia armazenada no *EPS*. A Figura 5 mostra a conexão dos componentes na *LaunchPad*. Nesta foto há os módulos COTS utilizados (SD Card, MPU9255 e *photoresistor*) e a câmera (*Paylod*).

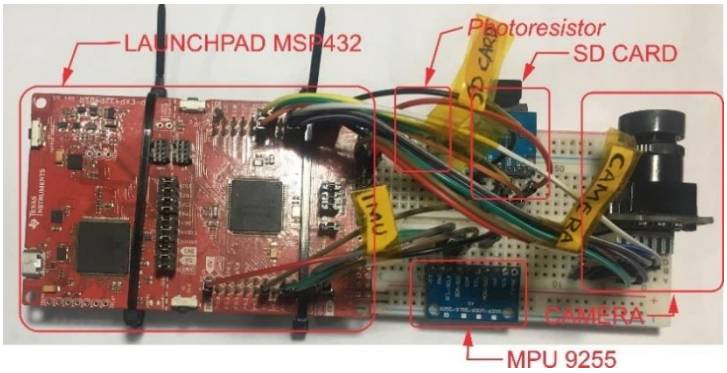

Figura 5 – Protoboard com os componentes COTS e a *LaunchPad*.

4.1. 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 *ASCII*, para facilitar o debug. A Figura 6 mostra o arquivo de telemetria salvado no SD Card.

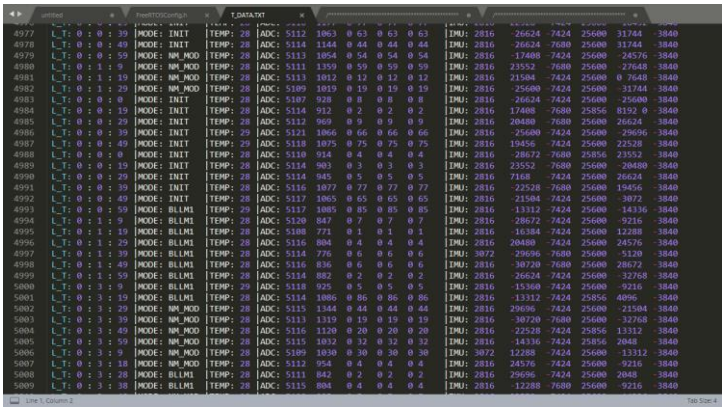

Figura 6 – Dados da telemetria armazenados no cartão de memória.

Em relação as fotos, [TESTES COM A CAMERA]

4.2. Consumo

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 consumida pelos módulos/sensores alimentados pela *Launchpad*. Houveram três baterias de testes de cinco minutos, um para cada estado.

Comparando o estado 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 7 mostra a comparação em forma gráfica, hibernação em azul e nominal em amarelo.

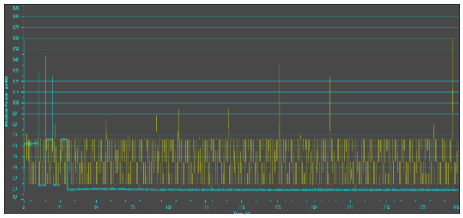

Figura 7 – Comparação entre os Modos de hibernação (azul) e Pouca Bateria (amarelo).

Não foi possível concluir a mesma melhora comparando o modo de baixo consumo com o modo de hibernação. Houve apenas uma economia de 9%, 0.6 dias. Esse fato ocorre porque 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 8 mostra a comparação em forma gráfica, hibernação em azul e nominal em amarelo.

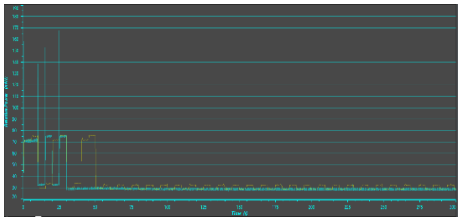

Figura 8 – Comparação entre os Modos de hibernação (azul) e Pouca Bateria (amarelo).

4.3. Modos de Operação

Para testar a máquina de estados do OBC, foi utilizado o *software* Tracealyzer, da empresa *Percepio* [13]. No decorrer dos testes foi possível observar vários fenômenos interessantes. Na inicialização do sistema 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(***WTDTask*** e ***TaskManager***) e manipulação de dados (***HouseKeeeping*** e ***DataManager***) foram executadas.

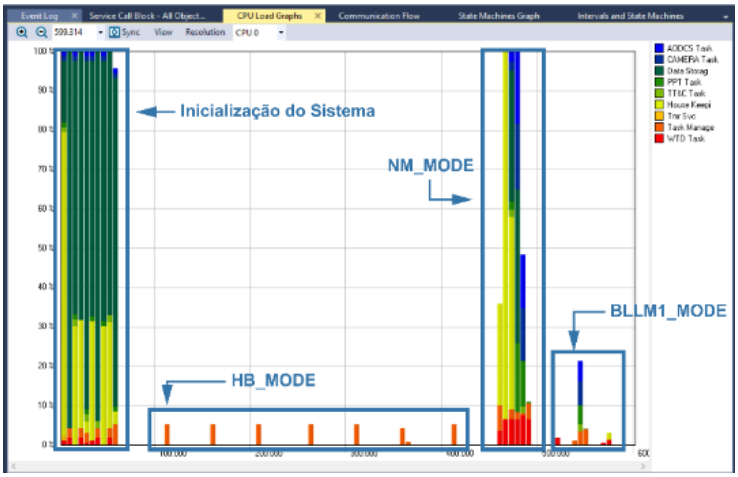

Figura 9 – Snapshoot no Tracealyzer do sistema executando em todos os modos.

4.4. Sistema Antitravamento

Para testar o Watchdog a nível de software, simulou-se um travamento no ***TT&C Task***, fazendo com que um bit não fosse setado no handler do ***WatchDogTask***. Como era de se esperar, a ***WatchDogTask*** conferiu os bits e deletou a ***TT&C Task***, conforme mostrado nas Figuras 10 e 44. Antes de ser delatada, a *task* do TT&C possuía a label **TT&C Task (2)** [azul] e, após ser reiniciada, mudou para **TT&C Task (3)** [verde].

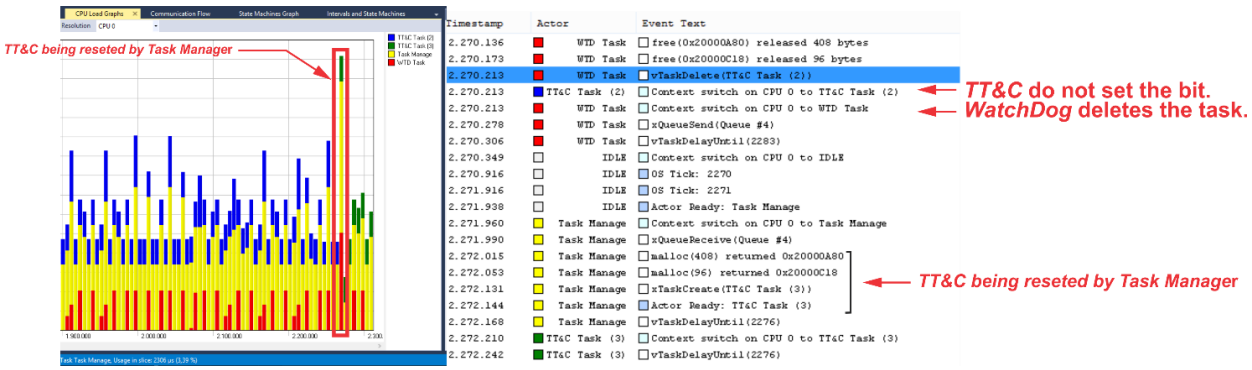

Figura 10 – Simulação do travamento do ***TT&C Task***.

4.5. Printed Circuit Board

A partir do layout realizado no KiCad, gerou-se os arquivos .gerbers que contêm todas as informações para a fabricação da PCB. Utilizou-se a fabricante PCBWay para realizar a fabricação da PCB. O resultado final é mostrado na Figura 11.

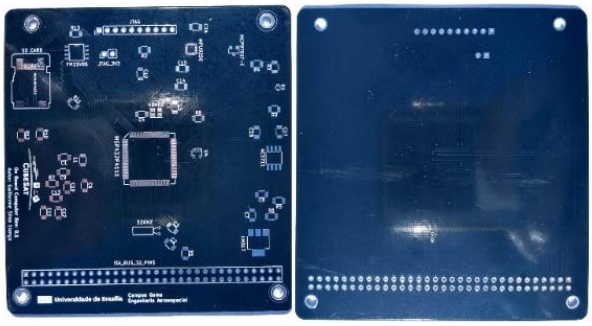

Figura 11 – Vista superior e inferior da PCB.

A fabricação da PCB representa a atual etapa em que o projeto se encontra. Os próximos passos, no desenvolvimento do OBC, serão mostrados na seção 7.

5. CONCLUSÃO

Este trabalho apresentou os principais avanços obtidos, até o presente momento, na prototipagem do UNB *On Board Computer,* para a futura missão CubeSat da Universidade de Brasília. O desenvolvimento foi dividido em duas partes: Hardware, onde foi selecionado o microcontrolador e os periféricos; Software, onde foi definido a arquitetura e componentes para cada camada. De acordo com os resultados obtidos, chegou-se à conclusão o MSP432P4111 é uma ótima opção para cenários de desempenho intermediário e de baixa potência. O uso do FreeRTOS como um sistema operacional em tempo real é uma ótima opção para microcontroladores com pouca memória RAM. Também foi visto que o watchdog a nível de software funcionou como forma de redundância em casos de

travamento parcial do código. Além disso, o software preliminar já consegue trocar de estado de acordo com algum input, por exemplo a luminosidade.

6. TRABALHOS FUTUROS

Alguns requisitos estabelecidos no projeto ainda não foram atendidos devido à complexidade do projeto e ao tempo limitado disponível. Todos esses pontos serão retomados, desenvolvidos e aprofundados durante os próximos meses para chegar à conclusão deste protótipo de computador de bordo da UnB para o CubeSats. Um ponto de extrema prioridade, a ser retomado, é a compra e soldagem dos componentes, pois não foi possível testar a placa desenvolvida no projeto. 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. O uso de vários níveis de *watchdog* não é suficiente 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.

7. REFERENCIAS

[1] TEXAS INSTRUMENTS, MSP432P411x, MSP432P401x SimpleLink TM Mixed Signal Microcontrollers. p.214. (2018)

[2] F. George, SwissCube HouseKeeping Parameters. p.1. Space Center EPFL, Switzerland (2009)

[3] C. Frost and E. Agasid, Small Spacecraft Technology State of the Art. p.97. NASA Ames Research Center, California (2015)

[4] ALLEGRO, High Sensitivity, 1 MHz, GMR-Based Current Sensor IC in SpaceSaving, Low Resistance QFN and SOIC-8 Packages. p.1. Manchester, New Hampshire (2018)

[5] INVENSENSE, MPU-9250 Product Specification Revision 1.1. p.28. San Jose, California (2018)

[6] MICROCHIP, Low-Power Linear Active Thermistor ICs. p.1. Chandler, Arizona (2016)

[7] C. Frost and E. Agasid, Small Spacecraft Technology State of the Art. p.95. NASA Ames Research Center, California (2015)

[8] TEXAS INSTRUMENTS, MSP432P4111 SimpleLink™ microcontroller LaunchPad™ development kit user's guide (Rev. B). p.11. (2019)

[9] PC/104 Embedded Consortium. PC/104 Embedded Consortium. 2nd. p.25. (2008)

[10] TEXAS INSTRUMENTS, USER'S GUIDE MSP432® Peripheral Driver Library. p.14. (2015)

[11] R. Barry, Mastering the FreeRTO STM Real Time Kernel: A Hands-On Tutorial Guide. p.17. Real Time Engineers Ltd., USA (2016)

[12] B. P. Douglass, UML for the C programming language. p.12. IBM Corporation, USA (2009)

[13] P. AB, Tracealyzer for FreeRTOS. p.1. Percepio AB, USA (2019)