MINISTÉRIO DA CIÊNCIA, TECNOLOGIA, INOVAÇÕES E COMUNICAÇÕES
INSTITUTO NACIONAL DE PESQUISAS ESPACIAIS

sid.inpe.br/mtc-m21c/2018/04.03.18.23-TDI

COMPUTAÇÃO DE BORDO PARA CARGAS ÚTEIS CIENTÍFICAS EM NANO SATÉLITES

Lázaro Aparecido Pires de Camargo

Dissertação de Mestrado do Curso de Pós-Graduação em Engenharia e Tecnologia Espaciais/Engenharia e Gerenciamento de Sistemas Espaciais, orientada pelo Dr. Walter Abrahão dos Santos, aprovada em 25 de maio de 2018.

URL do documento original:
<http://urlib.net/8JMKD3MGP3W34R/3QRDDK5>

INPE
São José dos Campos
2018

PUBLICADO POR:

Instituto Nacional de Pesquisas Espaciais - INPE
Gabinete do Diretor (GBDIR)
Serviço de Informação e Documentação (SESID)
CEP 12.227-010
São José dos Campos - SP - Brasil
Tel.:(012) 3208-6923/7348
E-mail: pubtc@inpe.br

MINISTÉRIO DA CIÊNCIA, TECNOLOGIA, INOVAÇÕES E COMUNICAÇÕES
INSTITUTO NACIONAL DE PESQUISAS ESPACIAIS

sid.inpe.br/mtc-m21c/2018/04.03.18.23-TDI

COMPUTAÇÃO DE BORDO PARA CARGAS ÚTEIS CIENTÍFICAS EM NANO SATÉLITES

Lázaro Aparecido Pires de Camargo

Dissertação de Mestrado do Curso de Pós-Graduação em Engenharia e Tecnologia Espaciais/Engenharia e Gerenciamento de Sistemas Espaciais, orientada pelo Dr. Walter Abrahão dos Santos, aprovada em 25 de maio de 2018.

URL do documento original:
<http://urlib.net/8JMKD3MGP3W34R/3QRDDK5>

INPE
São José dos Campos
2018

Dados Internacionais de Catalogação na Publicação (CIP)

Camargo, Lázaro Aparecido Pires de.

Ca14c Computação de bordo para cargas úteis científicas em nano satélites / Lázaro Aparecido Pires de Camargo. – São José dos Campos : INPE, 2018.

xxii + 83 p. ; (sid.inpe.br/mtc-m21c/2018/04.03.18.23-TDI)

Dissertação (Mestrado em Engenharia e Tecnologia Espaciais/Engenharia e Gerenciamento de Sistemas Espaciais) – Instituto Nacional de Pesquisas Espaciais, São José dos Campos, 2018.

Orientador : Dr. Walter Abrahão dos Santos.

1. Computação de bordo. 2. Nanossatélites. 3. Sistemas embarcados. 4. Ciência espacial. 5. Cargas úteis. I.Título.

CDU 629.7.05:629.78

Aluno (a): *Lazaro Aparecido Pires de Camargo*

Título: "COMPUTAÇÃO DE BORDO PARA CARGAS ÚTEIS CIENTÍFICAS EM NANO SATÉLITES".

Aprovado (a) pela Banca Examinadora em cumprimento ao requisito exigido para obtenção do Título de ***Mestre*** em

Engenharia e Tecnologia Espaciais/Eng. Gerenc. de Sistemas Espaciais

Dra. Maria de Fátima Mattiello-Francisco

Presidente / INPE / São José dos Campos - SP

( ) Participação por Video - Conferência

Dr. Walter Abrahão dos Santos

Orientador(a) / INPE / São José dos Campos - SP

( ) Participação por Video - Conferência

Dr. Eurico Rodrigues de Paula

Membro da Banca / INPE / SJCampos - SP

( ) Participação por Video - Conferência

Dr. Douglas Soares dos Santos

Convidado(a) / ITA / São José dos Campos - SP

( ) Participação por Video - Conferência

Este trabalho foi aprovado por:

( ) maioria simples

(X) unanimidade

São José dos Campos, 25 de maio de 2018

“Labor improbus ominia vincit”.

Às minhas queridas, Maura e Taykeline.

Ao meu pai e minha mãe.

AGRADECIMENTOS

Ao meu orientador Prof. Dr. Walter Abrahão dos Santos pela orientação, incentivo, motivação e por ter acreditado neste trabalho.

Ao meu supervisor na CEA, Dr. Eurico Rodrigues de Paula.

Aos colegas de trabalho na CEA, Eng. Sinval Domingos e Andrés Paredes Horna.

Aos colegas de projeto Nanosat-BR, Dra. Maria de Fátima Mattiello Francisco, Dr. Otavio Cupertino Durão, Marcelo Essado e Wendell Silva.

A Dra. Patrícia Mara S. Negretti pelas sugestões e revisão deste trabalho.

A Simone Del Ducca, André Luis Dias Fernandes e a equipe da Biblioteca do INPE pelas revisões e apoio.

Aos membros da banca Dr. Eurico Rodrigues de Paula e Dr. Douglas Soares dos Santos pelas sugestões.

À minha família: minhas queridas Maura e Taykeline e meus pais.

RESUMO

Os nanossatélites vêm sendo empregados para embarcar várias missões originalmente idealizadas para satélites de maior porte devido a avanços em computação e nanotecnologia. Neste contexto, o INPE tem interesse em migrar alguns experimentos científicos que foram originalmente planejados para microssatélites em plataformas de nanossatélites permitindo medidas *in-situ*. Este trabalho busca prover uma solução de computação de bordo que apoie cargas úteis prospectivas da CEA (Coordenadoria de Ciências Espaciais). Para tanto, é proposto uma metodologia de desenvolvimento integrado de hardware e software tipicamente aplicada à sistemas embarcados com diretivas da Engenharia de Sistemas e TDD (*Test-Driven Development*). Um esquema de modelos incrementais foi adotado partindo da emulação, prototipação e *porting* para um processador de vôo e sua comunicação com uma sonda de Langmuir, usada como carga útil em um estudo de caso. Um protocolo de bordo baseado em I2C foi definido e exercitado. As bibliotecas de software, baseada no *framework* Mbed, foi escrita e testada para que o computador de bordo de subsistema de carga útil (OBC-P) possa gerenciar cargas uteis cientificas. Instâncias do sistema foram geradas com aplicações para o NanoSatBr2 bem como para um prospectivo nanossatélite educacional-científico denominado Alpha-CTEE da PG-ETE.

Palavras-chave: Computação de Bordo. Nanossatélites. Sistemas Embarcados. Ciência Espacial, Cargas Úteis.

ON BOARD SOFTWARE FOR SCIENTIFIC PAYLOADS ON NANOSATELLITES

ABSTRACT

Nanosatellites are being used to carry out missions originally designed for large satellites due to advances in computing and nanotechnology. In this context, INPE is interested in migrating some scientific experiments that were originally planned for microsatellites into platforms based on nanossatélites and allowing *in-situ* measurements. This work aims at providing a on-board computing solution to support prospective payloads from CEA-INPE (Coordination of Space Sciences). Therefore, an integrated hardware and software development methodology is proposed, typically applied to embedded systems, using directives from Systems Engineering and TDD (Test-Driven Development). A scheme employing incremental models was adopted starting from emulation, prototyping and porting for a final flight processor and its communication with a Langmuir probe, used as payload in a case study. An I2C-based on-board protocol was defined and exercised. The software libraries, based on the Mbed framework, have been written and tested so that the on-board computer (OBC-P) can manage useful scientific loads. Instances of this system were generated with applications for the NanoSatBr2 nanosatellite as well as to a prospective educational-scientific nanosatellite named Alpha-CTEE from PG-ETE.

Keywords: On-Board Computer. Nanosatellites. Embedded Systems. Space Science. Payload.

LISTA DE FIGURAS

Pág.

LISTA DE TABELAS

Pág.

LISTA DE SIGLAS E ABREVIATURAS

API	*Application Programming Interface*
CAN	Controller Area Network
CEA	Coordenação de Ciências Espaciais
COTS	Commercial off-the-shelf
CTEE	Capacitação em Tecnologia e Engenharia Espacial
I2C	Inter-Integraded Circuit
ICD	ICD (Interface Control Document)
INPE	Instituto Nacional de Pesquisas Espaciais
iOBC-P	ISIS On Board Computer
ISIS	Innovative Solutions in Space
JTAG	Joint Test Action Group
OBC-P	On Board Computer
RTOS	Real time operation system
RTX	Real time operation sytem da Keil
SLP	Sonda de Langmuir

SUMÁRIO

Pág.

1 INTRODUÇÃO

Este documento apresenta a dissertação de mestrado com o título "Computação de Bordo para Cargas Úteis Científicas em Nanossatélites", e tendo como estudo de caso, o desenvolvimento de uma plataforma mínima de hardware e software para apoiar cargas úteis de experimentos da coordenação de Ciências Espaciais e Atmosféricas (CEA) do INPE.

1.1. **Motivação**

Nanossatélites vêm sendo empregados para embarcar várias missões originalmente idealizadas para satélites de maior porte devido a avanços em computação e nanotecnologia. Neste contexto, este trabalho espera contribuir com o tema uma vez que o INPE tem interesse em migrar alguns experimentos científicos, originalmente planejados para microssatélites, para plataformas em nanossatélites, como previsto no Plano Diretor do INPE 2016-2019 (INPE, 2016).

Os diversos grupos de pesquisa do CEA no INPE realizam sensoriamento de diversas camadas da atmosfera com instrumentos em solo e também embarcados em foguetes e balões. Medidas *in-situ* no espaço apenas são possíveis mediante satélites artificiais e podem-se estender as medidas para regiões da atmosfera que não são acessíveis via solo. Adicionalmente, isto permite também a verificação e teste de novos conceitos para posterior utilização em satélites de maior porte.

1.2. **Definição do problema**

O uso de nanossatélites, como o proposto aqui, permite o sensoriamento *in-situ*, entretanto necessita-se que as prospectivas cargas úteis da CEA sigam um padrão mínimo de interface para facilitar sua utilização de nanossatélites. Adicionalmente, algumas medidas coletadas no espaço exigem um pré-

processamento a bordo antes que possam ser descarregadas para solo, logo a provisão de uma infraestrutura computacional também é premente.

1.3. **Objetivos da dissertação**

Neste sentido, este trabalho deve definir uma estrutura de computação de bordo que atenda e defina um mínimo padrão de processamento interno e bem como as interfaces físicas para cargas úteis científicas da CEA.

1.4. **Solução proposta**

Esta dissertação propõe o desenvolvimento integrado de HW/SW (Hardware/Software) de uma solução de computação de bordo visando apoiar cargas úteis científicas embarcadas em um nanossatélite e utilizando experimentos da CEA como estudo de caso.

O desenvolvimento integrado de HW/SW de computação de bordo demandará uma adaptação não só de HW e SW bem como aspectos de Engenharia de Sistemas Espaciais. Uma abordagem efetiva para validação, verificação e testes será utilizada mediante TDD (*Test-Driven Development*).

Sinteticamente, o trabalho adotará uma estratégia de desenvolvimento por modelos incrementais começando com sistemas emulados, prototipação em HW/SW e finalmente *porting* para um processador *target* de vôo. Um estudo de caso com um OBC-P (On board computer for payload) e comunicação com protocolo de bordo utilizando o barramento I2C com uma carga útil Sonda de Langmuir, servirão para fins de verificação, validação e testes da solução proposta.

1.5. **Organização do trabalho**

Este documento está estruturado sumariamente da seguinte forma. O capítulo 1 consta dessa apresentação de motivação e os objetivos do trabalho. O capítulo 2 discute a fundamentação teórica. O capítulo 3 expõe a metodologia a

ser utilizada para o seu desenvolvimento. O capítulo 4 mostra o desenvolvimento e como é instanciado para uma solução e seu estudo de caso. O capítulo 5 apresenta os testes e resultados do sistema obtido. O capítulo 6 conclui este trabalho com suas considerações finais, produções feitas, limitações encontradas e sugestões de futuros trabalhos.

2 FUNDAMENTAÇÃO TEÓRICA

Nesta seção são apresentados tópicos relativos a fundamentos necessários ao entendimento e contextualização deste trabalho e seu desenvolvimento.

2.1. **Engenharia de sistemas espaciais**

O desenvolvimento de sistemas complexos como satélites deve ser realizado utilizando-se a Engenharia de Sistemas. De acordo com o *International Council on Systems Engineering* (INCOSE), Engenharia de Sistemas é a disciplina de engenharia responsável por criar e executar um processo interdisciplinar, assegurando que as necessidades dos clientes e stakeholders serão atendidas em um modo eficiente, confiável e de alta qualidade em tempo e custo, em todo o seu ciclo de vida. Em Wertz et al. (2011), é apresentada uma abordagem de engenharia de sistemas especialmente desenvolvida para aplicação no setor espacial por ele denominada Engenharia de Missão Espacial. Esta será adotada no desenvolvimento do presente trabalho.

A Engenharia de Missão Espacial é o refinamento de requisitos e definição de parâmetros de missão para cumprir os objetivos gerais de uma missão espacial em tempo hábil, a um custo e risco mínimos.

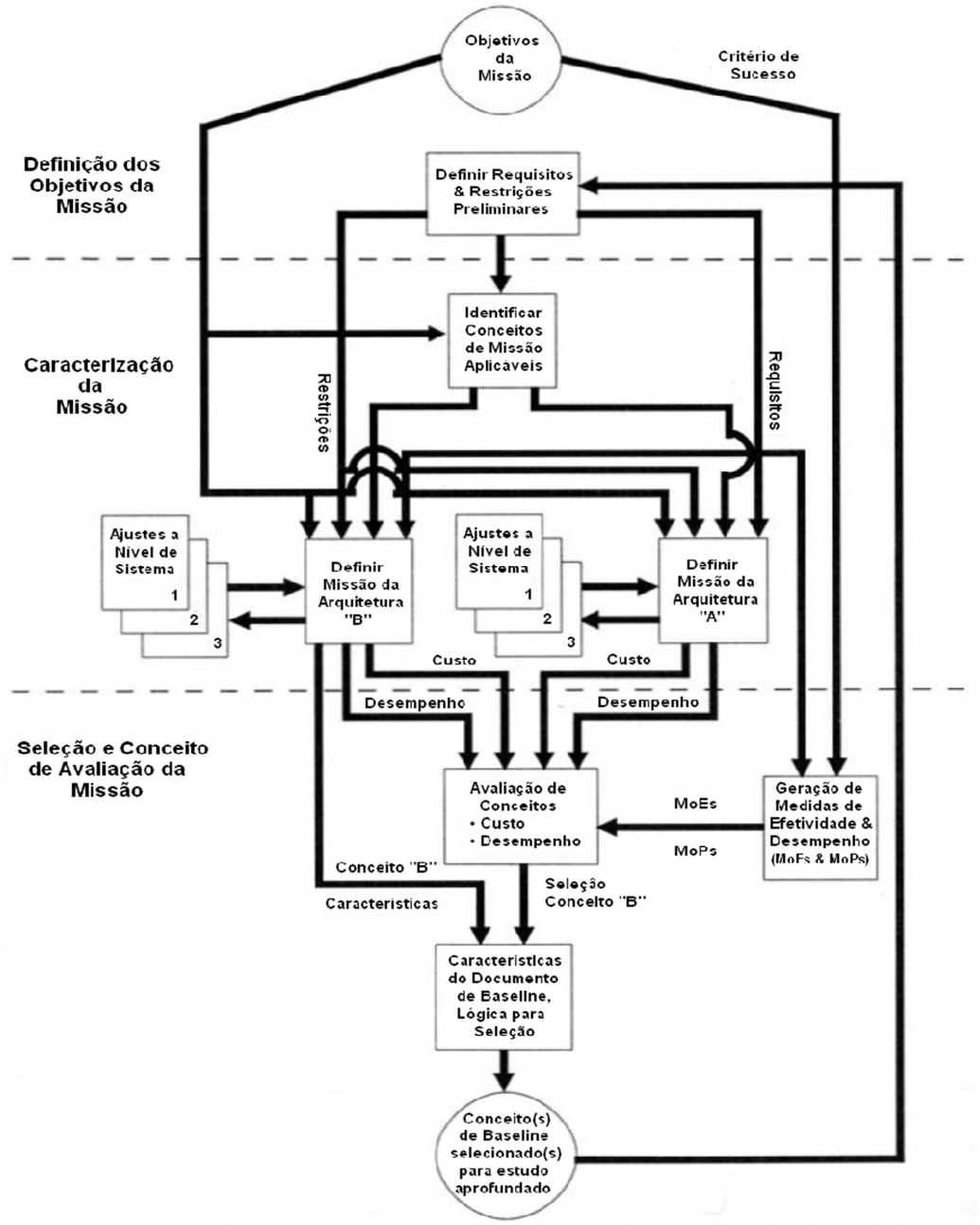
O processo de engenharia de missão espacial é sintetizado na Figura 1, adaptada de Wertz et al. (2011). Adicionalmente, define-se uma sequência de quatorze passos, constantes da Tabela 1, que são resumidamente apresentados a seguir.

Passo 1: Definir quais são os objetivos qualitativos e o seu porquê. Deve-se revisitá-los várias vezes para verificar se estão sendo atingidos e pode-se fazer isto melhor e mais barato.

Passo 2: Identificar os atores principais, os “Stakeholders”, e as comunidades espaciais de que fazem parte. Os usuários finais e os financiadores tipicamente

não são os mesmos e é importante que todas as partes envolvidas tenham seus objetivos atingidos.

Figura 1.1 - Fluxo de trabalho para exploração de conceitos de missões espaciais.



Fonte: adaptada de Wertz et al. (2011).

Tabela 2.1 - O Processo SMAD (Space Mission Analysis and Design)

Steps
Define Objectives and Constraints
1. Define the Broad (Qualitative) Objectives and Constraints
2. Define the Principal players
3. Define the Program Timescale
4. Estimate the Quantitative Needs, requirements and Constraints
Define alternative Mission Concepts or Designs
5. Define Alternative Mission Architecture
6. Define Alternative Mission Concepts
7. Define the Likely System Drivers and Key requirements
Evaluate the Alternative Mission Concepts
8. Conduct Desempenho Assessments and System Trades
9. Evaluate Mission Utility
10. Define the Baseline Mission Concept and Architecture
11. Revise the Quantitative Requirements and Constraints
12. Iterate and Explore Other Alternatives
Define and Allocate System Requirements
13. Define System requirements
14. Allocate the Requirements to System elements

Fonte: Adaptada de Wertz et al. (2011).

Passo 3: Definir o cronograma em que o programa precisa ser executado para ser útil.

Passo 4: Quantificar quão bem se deseja atingir os amplos objetivos, dadas as necessidades, a tecnologia aplicável, quem são os usuários e as restrições de

custo e agenda. Tais requisitos quantitativos devem ser objeto de negociação à medida que se avança.

Passo 5: Definir as combinações alternativas dos elementos de missão ou da arquitetura da missão espacial para atender aos objetivos da missão e seus requisitos. Isto é feito analisando cada elemento individual da missão como a carga útil, ou a órbita do satélite e considerando as principais alternativas para cada um deles que melhor atenderia os objetivos da missão.

Passo 6: Depois de identificar todas as partes de uma solução de missão, é necessário entender como elas irão trabalhar em conjunto para atender as necessidades dos usuários finais. Inicia-se este processo pelo desenvolvimento de conceitos de missão alternativos. Um “Conceito de Missão” ou “Conceito de Operação” é uma declaração ampla de como a missão irá funcionar na prática. Tipicamente os “Conceitos de Missão” mais efetivos são aqueles que são totalmente transparentes para o usuário final.

Passo 7: Identificar os prováveis “System Drivers” e requisitos chaves. Em qualquer sistema real muitas coisas influenciam os custos globais, desempenho ou o projeto detalhado dos componentes. Entretanto elas são influenciadas principalmente por um pequeno número de parâmetros ou componentes-chaves chamados “drivers”. Devem-se identificar os principais drivers de desempenho e custo para cada alternativa de conceito de missão. Similarmente, há requisitos críticos que são aqueles que têm o maior impacto em custo, risco e cronograma. Para muitas missões espaciais “*system drivers*” incluem: número de satélites, altitude, potência, tamanho e peso dos instrumentos; requisitos críticos incluem: vida útil, disponibilidade, cobertura, resolução, potência de sinal. Por identificar explicitamente os “*system drivers*” e os requisitos críticos podem-se concentrar esforços nos parâmetros que tem o maior impacto no projeto e, portanto, no custo e desempenho da missão espacial. Isto melhora as chances de obter o melhor projeto possível com o orçamento disponível. Os “*systems drivers*” e requisitos críticos serão provavelmente diferentes para diferentes conceitos de missão.

Passo 8: Conduzir avaliações de desempenho e "*system trades*", é o passo mais trabalhoso em um projeto de missão porque ele define em detalhe o que o sistema é e faz. Aqui é determinado o balanço de potência, peso e apontamento e é decidido o que processar no espaço e em terra. Caracterizar a missão, e conduzir as negociações entre alternativas, é sempre o passo mais custoso porque requer a experiência de muitas pessoas.

Passo 9: Os passos anteriores formam a base da "Análise de utilidade da missão", neste passo deve-se quantificar quão bem os objetivos gerais e as necessidades dos usuários finais estão sendo atingidos, em função do custo e das escolhas chaves do projeto do sistema. Somente o usuário final ou o desenvolvedor do sistema podem em última instância definir quão boas são estas medidas de desempenho críticas.

Passo 10: Tendo avaliado projetos alternativos e realizado uma avaliação preliminar da utilidade da missão, deve-se selecionar um ou mais *baseline* de projetos de sistema. Um *baseline* de projeto é uma definição consistente e única do sistema contendo um conjunto particular de valores para todos os parâmetros do sistema, os quais são consistentes um com os outros. Conforme o projeto do sistema matura, o baseline se torna mais firme e eventualmente se torna o projeto do sistema.

Passo 11: Tendo determinado os requisitos críticos e seu impacto no projeto da missão e utilidade da missão, neste momento devem-se revisar os requisitos do sistema e as restrições, consistente com o que foi aprendido. É preciso encontrar os requisitos que reflitam o que se quer atingir (objetivos da missão) e o que pode ser atingido com as ferramentas, técnicas, e o hardware disponível.

Passo 12: Por fim, deve-se retornar e explorar alternativas e, eventualmente, repetir o projeto se necessário. O resultado que se procura é um equilíbrio entre o que é desejável e o que foi possível obter em tempo, dentro do orçamento e com um nível aceitável de risco. Quando se inicia, não se sabe como será. A razão para se explorar alternativas e procurar opções é devido à

incerteza existente no inicio do projeto. Perguntas típicas desta fase são: Há outra abordagem para atingir melhor os objetivos ou a um custo mais baixo? Ou se alternar um pouco o foco dos objetivos pode-se obtê-los a um custo mais baixo ou com menos risco?

Passo 13: Os primeiros 12 passos são o que é chamado exploração de conceito olhando para requisitos alternativos, restrições e sistemas alternativos. Pelo fato de construtores de sistemas espaciais tradicionalmente trabalharem a partir de requisitos específicos, deve-se traduzir agora os “objetivos melhor definidos”, restrições e requisitos, em requisitos de sistema bem definidos. Normalmente, a engenharia de sistema tradicional começa por aqui.

Passo 14: Neste ponto, são alocados os requisitos numéricos para os componentes da missão espacial do mesmo modo que um balanço de engenharia de sistemas, por exemplo, define peso e potência para os componentes do satélite. Isto irá se tornar uma lista detalhada de requisitos e, em última análise, uma missão espacial em órbita.

2.2. **Nanossatélites e Cubesats**

O termo nanossatélite é utilizado genericamente para designar satélites com massa compreendida entre 1 a 10 kg. Nesta categoria de satélites, os cubesats se tornaram hegemônicos embora existam outras plataformas numa categoria abaixo como os picossatélites, como por exemplo, tubesats e pocketQub. Cubesats vem sendo desenvolvidos com diversas inovações na Morehead State University, EUA, por exemplo.

O conceito do cubesat, caso particular de um nanossatélite, foi desenvolvido pelo Professor Robert Twiggs então no Departamento de Aeronáutica e Astronáutica da Universidade de Stanford e pelo professor Jordi Puig-Suari então no Departamento Aeroespacial da Calpoly State University, no fim de 1999. O conceito de cubesat foi originado do projeto OPAL (*Orbit Picosat Automated Launcher*), um microssatélite de 23 kg desenvolvido por estudantes

da Universidade de Stanford e pela Aeroespace Corporation para demonstrar a validade e funcionalidade dos picossatélites e o conceito de lançar um picossatélite e de um satélite maior em órbita. A missão OPAL representou um importante marco na evolução dos picossatélites por duas razões: (1) provou a viabilidade do conceito de picossatélites e (2) introduziu e iniciou o desenvolvimento de um inovador sistema *deployment* de lançamento. Desta experiência, derivou o conceito de cubesat bem como do *deployer* P-Pod.

Basicamente, o cubesat é um satélite na forma de um cubo de 10 cm de aresta que originalmente teria massa de até 1 kg, no padrão atual até 1,33 kg, o que se convencionou chamar de 1U, podendo ser escalado para 2U e 3U. Outro requisito é tenham em qualquer dos casos o centro de gravidade CG, aproximadamente no centro da estrutura.

A padronização dimensional e da interface com o lançador, o P-POD, que adicionalmente protege a carga útil principal do lançador de qualquer problema que venha a ocorrer com o Cubesat. O P-POD, que funciona como *deployer* e *container* ao mesmo tempo, permitiu um enorme aumento das oportunidades e o barateamento do custo de lançamento para satélites universitários, o que sempre foi a principal limitação destes programas.

A primeira geração de cubesats tinha missões relacionadas ao treinamento de estudantes. O professor Twiggs descreve com suas palavras o efeito destes satélites em seus cursos e projetos como:

> "*The excitement and challenge of having opportunity for the new students to do "real" things in space exploration cannot be achieved by lectures in the classroom and represents education at its best*".

Atualmente, cubesats são amplamente utilizados pela indústria espacial e por departamentos de defesa. A Tabela 2.1 é uma amostra representativa de missões Cubesats que foram lançados entre 2003 e 2011 apenas.

Tabela 2.2 - Missões cubesats que foram lançados entre 2003 e 2011.

Nome	Tipo	Organização	Missão	Lançamento	Foguete
Quakesat	3U	Stanford University / Quake Find USA	Earthquake detection	6/30/ 2003	Rokot/ Briz-KM
CUTE1 (Oscar 55)	1U	Tokio Institute of Technology, Japan	Separation system demonstration, Technology testing & amateur radio	6/30/2003	Rokot/ Briz-KM
CP-3	1U	California Polytechnic University, USA	Educational, Technology testing & amateur radio	4/17/2007	Dneper-1
MAST	1U	Tethers Unlimited, USA	Tethers experiments operated from 4/20/ 2007 to 5/30/2007	4/17/2007	Dneper-1
COMPASS-1	1U	FH Aachen – University of Applied Science Germany	Demonstration of commercial off-the-shelf components and taking photos	4/28/2008	PSLV-CA
CanX-2	3U	University of Toronto Canada	Technology demonstrator for formation flying	4/28/2008	PSLV-C9
k-Sat	1U	Kagoshima University	Technology demonstrator and Earth remote Sensing of Water Vapor	5/20/2010	H-II 202
RAX	3U	University of Michigan and SRI	Bi-static RADAR measurement of plasma formation in Earth's ionosfere	11/20/2010	Minotaur

Fonte: Adaptada de Tikami (2016)

Cubesats provaram ser uma maneira efetiva de colocar cargas úteis em órbita de forma rápida e barata. Muitos experimentos de ciência de alto nível vêm sendo realizados usando cubesat como, por exemplo, trabalhos em Astrobiologia realizados na NASA Ames bem como o projeto do NSF RAX da Universidade de Michigan.

A Tabela 2.2 mostra a gama de lançadores usados no lançamento de cubesats, de 2003, data do lançamento do primeiro satélite desta classe, até 2010.

Tabela 2.3 - Lançadores usados no lançamento de cubesats de 2003 até 2010.

Lançadores	**País de Origem**	**Ano 1o. Lançamento**
Rokot-KM	Russia	2003
Kosmos—3M	Russia	2005
Dneper-1	Russia	2006
PSLV930	India	2008
PSLV-CA	India	2009
H-2A-202	Japan	2010
M-5(2)	US	2008
Minotaur-1	US	2006
Minotaur-4 HAPS	US	2010
Falcon-1	US	2008
Falcon-9	US	2010

Fonte: Adaptada de Tikami (2016)

2.3. **Mbed RTOS**

O projeto e o desenvolvimento de software para cubesats, seguem os mesmos preceitos para projetos em sistemas embarcados, que são compostos por diversas tarefas executadas para atender os requisitos da missão. Tais tarefas podem exigir restrições de tempo, além de interagirem umas com as outras. E o uso de arquiteturas simples como interrupções ou máquina de estados,

tornam o código mais complexo e de difícil manutenção. Sistemas operacionais de tempo real (RTOS), podem ser uma alternativa para atender restrições temporais e tornar o desenvolvimento do código mais organizado e flexível. (EMBARCADOS, 2017).

Um RTOS é um sistem operacional destinado a gerenciar múltiplas tarefas com períodos de execução pré-definidos. Este gerenciamento é feito pelo escalonador (**scheduler**) utilizando critérios como tempo máximo de execução, prioridade, criticidade de um evento e sequência de execução.

As tarefas ou tasks, em um RTOS são basicamente subprogramas que possuem seu espaço próprio de contexto e variáveis, porém compartilhando o mesmo mapa de memória físico de um processador.

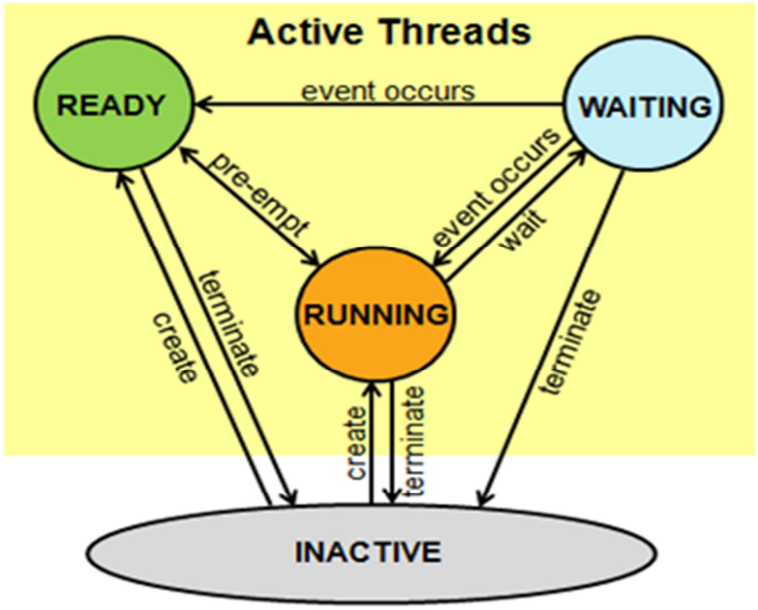
E como opção pode-se utilizar o Mbed RTOS, que é um sistema operacional de tempo real que será utilizado na plataforma de computação de bordo em cubesats. O MBed utiliza o kernel de tempo real da Keil, o RTX, o qual é um sistema operacional de tempo real que faz parte do pacote Keil uVision.

A equipe da Mbed adicionou o porte para o RTX e estendeu suas chamadas em formas de classes, permitindo seu uso como objetos em C++. É possível se usar o sistema operacional Mbed utilizando um compilador online ou mesmo off-line. O uso off-line é possível mediante emprego do utilitário CLI, que é uma ferramenta de linha de comando que permite criar, compilar e exportar um projeto em Mbed, para diversas IDEs, como a KEIL ou o GNU-ARM.

O Mbed RTOS utiliza o "*System tick timer*" (SysTick) para temporização e alocação de tarefas. Este RTOS possui APIs para gerenciar a criação e finalização de *threads*:

Uma *thread* é mapeada a uma classe que permite definir, criar e controlar tarefas em paralelo. Uma *thread* pode ter os seguintes estados: RUNNING, READY, WAITING, INACTIVE, cuja a transição entre os mesmos é ilustrada na Figura 2.1 para uma classe *Thread*.

Figura 2.1 - Comportamento de uma classe *Thread* no Mbed RTOS.



Fonte: Adaptado de Mbed (2018).

2.4. **O computador de bordo iOBC - ISIS**

Para a computação de bordo em nanossatélites, este trabalho se pautou por algumas soluções baseadas em sistemas COTS disponíveis. Dentre eles, se destaca o iOBC que é um computador de bordo (OBC – *On-Board Computer*) da empresa holandesa ISIS e é compatível com o padrão cubesat para uso em nanossatélites. Este OBC utiliza o microcontrolador AT91SAM9G20 como base para seu processamento.

Algumas características da iOBC são destacadas a seguir na Tabela 2.3 como processador, memória disponível para dados e para o firmware, contadores de tempo real, interfaces, sensores, massa, consumo. Essas informações são tipicamente utilizadas em documentos de interface ICD (Interface Control Document) e são essenciais para se estabelecer um balanceamento de recursos adequado aos objetivos da missão que se busca atender.

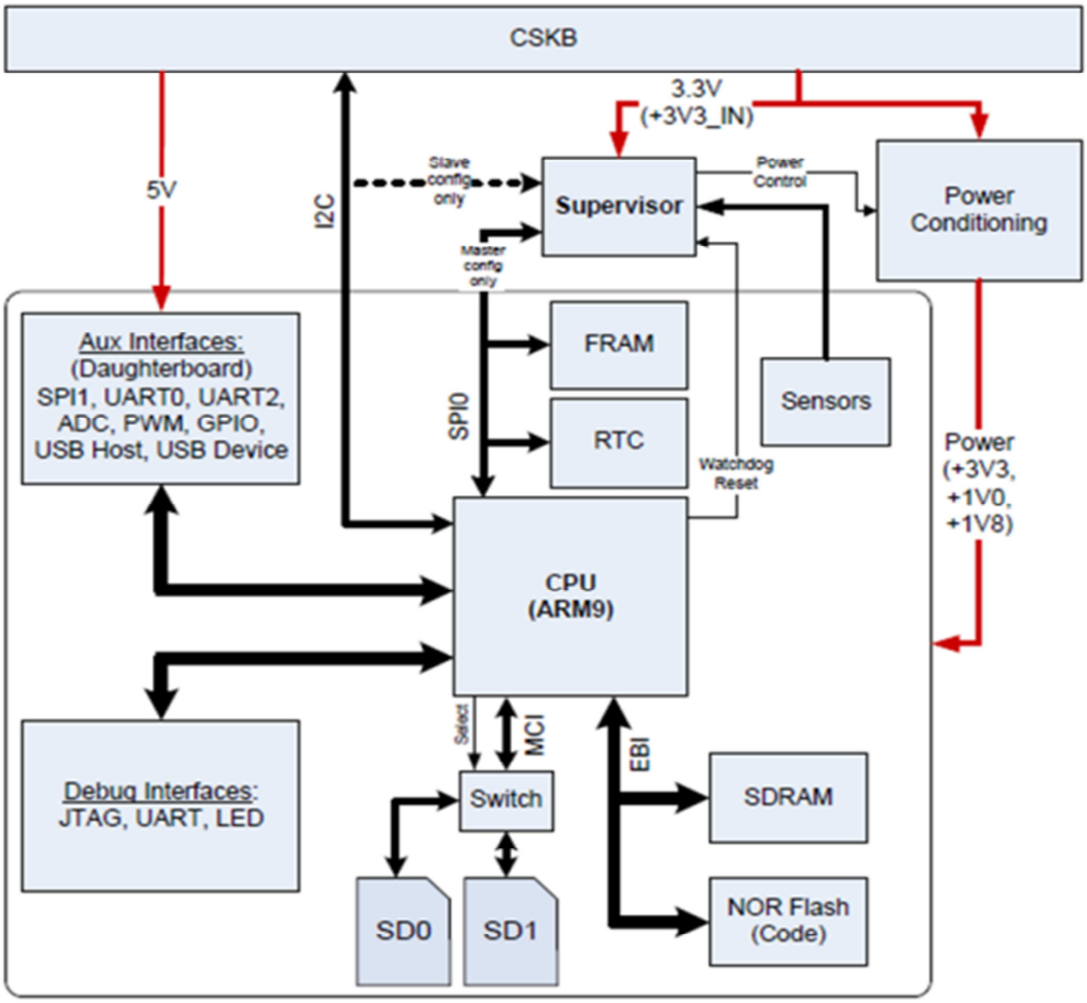
Tabela 2.4 – Principais características de interface e uso do iOBC

Processador	400MHz, ARM9 32-bit (AT91SAM9G20)
RAM	32MB SDRAM
Mémoria não volátil	2 x 2GB SD-Cards com FAT32 256kB FRAM
Mémoria para código	1MB NOR-Flash
RTC	2 RTC redundantes
Sensores on-board	Temperatura, corrente e voltage, com limite de sobrecorrente
Interfaces	1 x I2C 1 x SPP 2 UARTs 1 ADC: 8 canais – 8 ou 10 bits PWM: 6 canais GPIO: 27 pinos USB: 1 host, 1 device 1 Interface para sensor imagem CMOS
Programação	JTAG e UART (console debug) 4 leds
Potência media	380mV em 3,3V
Faixa de temperature qualificada	-25oC a +65oC
Massa	106g

Fonte: Adaptado de ISIS (2013).

A arquitetura interna do microcontrolador é apresentada na Figura 2.7 onde se destaca o diagrama de blocos da iOBC, maiores detalhes estão disponíveis em ISIS (2013).

Figura 2.7 – Diagrama de blocos da iOBC-P



Fonte: ISIS (2013).

2.5. **Sonda de Langmuir (LP)**

A Divisão de Aeronomia, que faz parte da Coordenação de Ciências Espaciais no INPE, realiza pesquisas da Ionosfera desde 1963. O estudo da ionosfera com foguetes de sondagem vem sendo realizado em colaboração com o IAE/CTA, desde 1984, com cargas úteis científicas desenvolvidas no INPE,

para realizarem medidas “in loco” dos parâmetros ionosféricos. Mais de dez foguetes brasileiros da série SONDA e foguetes estrangeiros, e dois satélites científicos e um tubesat foram lançados com experimentos desenvolvidos no INPE. Os lançamentos dos foguetes foram realizados nos centros de lançamentos brasileiros em Natal (Rio Grande do Norte), e em Alcântara (Maranhão). Os Experimentos desenvolvidos são (DAE, 2017):

a) Sonda Capacitiva em Alta Frequência (HFC)

b) Sonda de Langmuir (LP)

c) Sonda dupla de Campo Elétrico (EFP)

Umas das cargas úteis que será utilizada para estudo de caso a frente será a sonda de Langmuir. A área de Ciências Espaciais e Atmosféricas - CEA do INPE é constituída pela Aeronomia, Astrofísica e Geofísica Espacial e sua missão é gerar conhecimentos científicos, formar e treinar pessoal especializado, desenvolver tecnologia e assessorar órgãos governamentais e empresas privadas em assuntos relativos às ciências e tecnologias espaciais e atmosféricas (INPE, 2017).

Mais especificamente, a CEA objetiva a realização de pesquisas básicas, pesquisas aplicadas e desenvolvimento de instrumentação científica para uso em solo, plataformas suborbitais (balões estratosféricos), foguetes de sondagem e satélites científicos, com a finalidade de entender os fenômenos físicos e químicos que ocorrem na atmosfera e no espaço, de interesse para o país. Dentre diversas cargas úteis da CEA, a LP será utilizada neste trabalho e apresentada brevemente a seguir.

Numa sonda de Langmuir (TIKAMI, 2016), a corrente entre o plasma e um sensor metálico é medida em função do potencial aplicado ao sensor. A LP é utilizada para medir a densidade, a temperatura e as irregularidades dos elétrons na Ionosfera. Uma varredura de voltagem, aplicada no sensor, faz com que o mesmo colete corrente (da ordem de nanoampere) variável em função

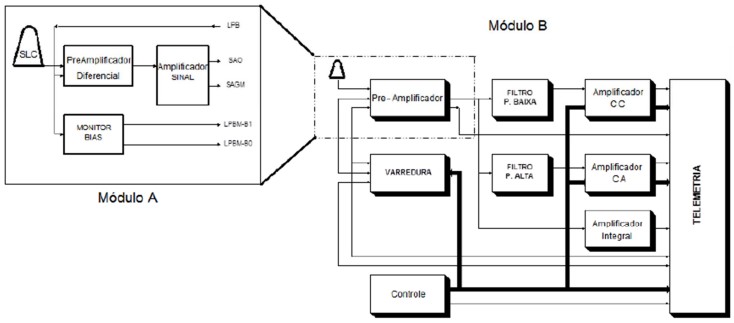
da voltagem, cuja intensidade é proporcional à densidade eletrônica e cujo gradiente com voltagem depende da temperatura eletrônica.

Os sensores são operados em dois modos - modo de elétrons e modo de íons - para medir a densidade eletrônica e iônica. A variação dinâmica das densidades cobre uma faixa de quatro ordens de magnitude, de 5 x $10^{8}m^{-3}$ a 5 x $10^{12}m^{-3}$. Tal faixa pode ser coberta através do uso de amplificadores com controle automático de ganho. A saída destes amplificadores e as informações de ganho são transmitidas às estações terrestres.

A sonda de Langmuir é usada para medir a concentração de elétrons ou íons no plasma ionizado através da determinação da corrente entre o plasma e o sensor. A corrente de elétrons e íons recolhida pelo sensor depende da forma geométrica do sensor, do potencial do sensor, das características físicas do plasma ambiental e da velocidade do sensor. O sensor pode ser mantido no potencial do plasma ambiental ou num potencial negativo para recolher corrente de íons positivos ou num potencial positivo para recolher corrente nos elétrons.

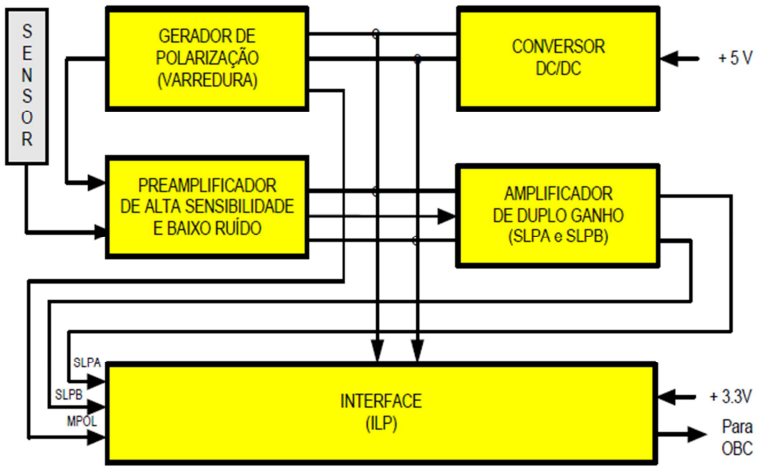
A figura 2.8 mostra o diagrama de blocos de uma sonda de Langmuir utilizada em foguetes e a figura 2.9 mostra a adaptação para um cubesat.

Figura 2.8 – Diagrama de blocos da sonda de Langmuir



Fonte: Odriozola (2017).

Figura 2.9 – Diagrama de blocos da sonda de Langmuir adaptada para Cubesats



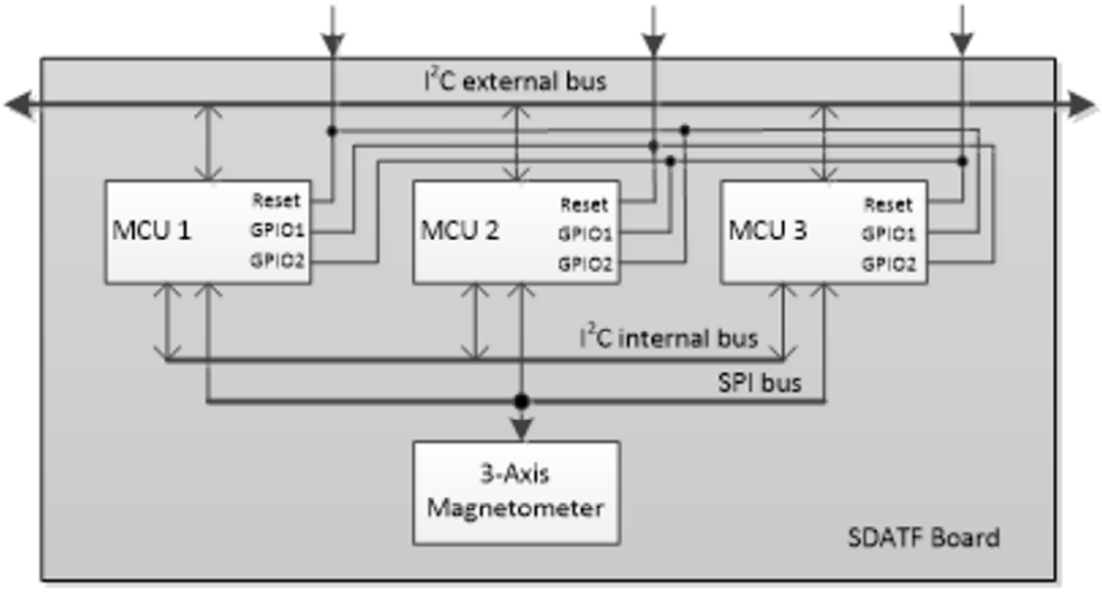
Fonte: Horda (2016).

2.6. **Sistema de Determinação de Atitude Tolerante a Falhas (SDATF)**

Outra carga útil científica de interesse é o Sistema de Determinação de Atitude Tolerante a Falhas (SDATF) desenvolvida pela UFMG-UFABC-Ubiqua, com a função de calcular a atitude do cubesat NanosatC-BR2. O sistema executa suas funções a partir de medições do magnetômetro triaxial XEN-1210, instalado na própria placa da SDATF. Para realizar o cálculo da atitude do cubesat, o SDATF deve receber, via barramento I2C, os seguintes dados do OBC-P: (a) Dados orbitais (TLE); (b) Valores dos sensores de sol; (c) Hora (UTC) destas medidas e (d) Comandos de reset e calibração do magnetômetro;

O SDATF é composto de três microcontroladores STM32303CCT6 operando de forma redundante conforme mostra seu diagrama de blocos na Figura 2.8. O mesmo firmware é gravado nos três microcontroladores. Este firmware fornece uma configuração diferenciada para cada microcontrolador de acordo com o modo de operação do SDATF.

Figura 2.8 – Diagrama de blocos da iOBC-P



Fonte: Duarte (2016).

Durante a operação normal do NanosatC-BR2, o SDATF tem um dos seus microcontroladores trabalhando em modo *Master*, outro em modo *Partner* e um terceiro em modo *Sleeper*.

a) Modo *Master* - Recebe do OBC-P um pacote de dados via barramento I2C externo, chamado pacote PCK_TASK. É o microcontrolador (MCU) responsável por enviar para o OBC-P, a pedido do último, outro pacote de dados através do barramento I2C externo;
b) Modo *Partner* - Recebe do OBC-P um pacote de dados via barramento I2C externo, chamado pacote PCK_TASK. É o microcontrolador (MCU) responsável por receber os dados processados do magnetômetro fornecido pelo MCU configurado em Modo Master via barramento I2C interno.
c) Modo *Sleeper*: Recebe do OBC-P um pacote de dados via barramento I2C externo, chamado pacote PCK_TASK. É o microcontrolador (MCU) responsável por receber os dados processados do magnetômetro

fornecido pelo MCU configurado em Modo Master via barramento I2C interno.

Como a carga útil SDATF utiliza um magnetômetro como elemento sensor, é algo de grande relevância para o CEA, pois para medir a intensidade e direção de campo magnético da Terra, são utilizados magnetômetros, para fins de estudo e predição do clima espacial (CLIMA ESPACIAL, 2014).

2.7. **Missões anteriores em computação a bordo de Nanosats**

Um levantamento de trabalho com escopo semelhante a este foi realizado e a Tabela 2.4 mostra alguns exemplos de cubesats científicos, indicando o ano de *deploy*, o processador utilizado e qual a solução utilizada para o software de bordo, tipicamente sistema operacional.

Tabela 2.5 – Principais características de interface e uso do iOBC-P

Satélite	**Tipo de Missão**	**Ano**	**Processador**	**Solução Adotada**
QuakeSat	Científica	2003	ZFx86	Linux
SwissCube	Científica	2009	ARM7TDMI	FreeRTOS
AAUSAT2	Cientifica	2008	AT91SAM7A1	eCos
ITUpSAT1	Cientifica	2009	MSP430F1611	Salvo 4
Cute-1	Prova de Conceito	2003	Hitachi NPD-20JWL (PDA)	Windows CE.NET
Este Trabalho	**Prova de Conceito**	**2018**	**STM32**	**MBED**

Fonte: Produção do autor.

O QuakeSat é um nanosatélite científico que utilizava o processador ZFx86 e o sistema operacional Linux. feito com uma colaboração do SSDL (*Space Systems Development Laboratory*) da Universidade de Stanford, Stanford, CA, e da equipe QuakeFinder de Palo Alto, CA e cujo objetivo é a detecção de assinaturas de terremotos (QuakeSat, 2018).

O SwissCube-1, (SwissCube, 2018) foi um cubesat científico de 1U suíço operado pela *Ecole Polytechnique Fédérale de Lausanne* (EPFL) para realizar pesquisas sobre o *nightglow* (fenômeno de brilho noturno na atmosfera terrestre) e também para desenvolver tecnologia para futuras naves espaciais. A computação de bordo utilizava o processador ARM7TDMI e o sistema operacional de tempo real FreeRTOS.

A Universidade de Aalborg, Aalborg, Dinamarca desenvolveu cubesat AAUSat-2 (AAUSat-2, 2018), uma *joint venture* de cinco institutos universitários. O cubesat utilizava o procesador AT91SAM7A1 e embarcava o sistema operacional eCos para gerenciar dois experimentos científicos, um sistema ADCS e um detector de raios gama.

O ITUpSat-1 (*Istanbul Technical University PicoSatellite-1*) foi projetado por um estudante (ITUpSat-1, 2018) de Istambul, Turquia dentro dos padrões CubeSat e sua computação realizada pelo MSP430F1611 e RTOS Salvo da Pumpkin. Além dos objetivos educacionais, a missão era também capturar imagens da carga útil do CMOS e estudar o comportamento do sistema de estabilização passiva do CubeSat.

O cubesat CUTE-I (*Cubical Tóquio Tech Engineering Satellite-I*) é um projeto cooperativo do LSS (*Laboratory for Space Systems*) e o SRTL (*Space Robotics and Teleoperations Laboratory*) (CUTE-I, 2018), ambos de Tóquio. Sua computação ela implementada pelo processador Hitachi NPD-20JWL e executava um Windows CE.NET. O projeto educacional de baixo custo usava componentes COTS e uma missão tecnológica.

3 METODOLOGIA ADOTADA PARA COMPUTAÇÃO DE BORDO

Neste capítulo é exposta a metodologia a ser aplicada para fornecimento de recursos de computação de bordo para algumas cargas científicas bem como algumas decisões de projeto tomadas a partir da metodologia.

Sumariamente são apresentados: (1) requisitos gerais para OBC-P; (2) aspectos de escolha do microcontrolador; (3) escolha do ambiente IDE; (4) escolha de um sistema operacional de tempo real e suas primitivas; (5) desenvolvimento orientado a testes (TDD); (6) mapeamento de TDD para o ambiente Mbed e (7) aspectos de engenharia de sistemas.

3.1. **Requisitos gerais para OBC-P em aplicações espaciais**

Computadores de bordo para aplicações espaciais possuem alguns requisitos específicos (EICKHOFF, 2013), comparados com sistemas embarcados em Terra:

a) O OBC-P em um veículo espacial, precisa ter desempenho numérico para o propósito da missão, como processamento de dados, controle, gerenciamento de dados da carga útil e outras funções;

b) O OBC-P precisa ser mecanicamente robusto para suportar a fase de lançamento;

c) Em órbita, o OBC-P tem que operar em condições eletromagnéticas adversas (cinturão de van Allen);

d) O OBC-P deve suportar mudanças térmicas bruscas;

e) O OBC-P deve suportar doses de radiação de partículas de alta energia;

f) O OBC-P deve ser capaz de suportar substâncias químicas agressivas, por exemplo, em órbitas baixas, a presença de oxigênio atômico;

g) O consumo do OBC-P deve ser limitado por restrições devido ao sistema de geração de energia;

h) O OBC-P precisa ter critérios definidos contra falhas e redundância.

Alguns requisitos não funcionais (como restrições nas funções oferecidas pelo sistema) (FREITAS, 2007) para uma OBC-P, comparadas a sistemas embarcados, são:

- a)Tempo, Temporização, Perído: O sistema deve ser capaz de ler os dados relevantes ao controle, em uma taxa adequada;
- b)Distribuição dos dados: O sistema deve ser capaz de fornecer os dados que a OBC solicitar;
- c)Consumo de Energia: O sistema deve operar no envelope de energia, adequado para a missão;
- d)Desempenho, Tempo de Resposta: O sitema deve ser capaz de responder a comandos da OBC em um intervalo de tempo adequado;

3.2. **Escolha de microcontroladores - STM32**

A escolha de um microcontrolador deve satisfazer as características técnicas do projeto, neste sentido é muito importante considerar os seguintes pontos fundamentais: (1) Arquitetura; (2) Consumo de potência, caso o projeto demande baixo consumo, ou seja, móvel o que neste caso de nanossatélites se enquadra em ambos os critérios; (3) Periféricos; (4) Velocidade e capacidade de processamento; (5) Tamanho e encapsulamento e (6) Escalabilidade.

Critérios de escolha destes componentes de forma mais abrangente envolvem 12 especificações citadas em (Teel, 2018) que um desenvolvedor deve levar em conta na escolha onde uma plataforma de suporte o desenvolvimento em solo é relevante bem como seu uso no ambiente destino.

Diante das considerações citadas, optou-se pelo STM32 que é uma família de microcontroladores de 32 bits da STMicroelectronics. São agrupados em séries seguindo os núcleos do processador ARM 32-bit, como Cortex-M7F, Cortex-M4F, Cortex-M3, Cortex-M0+, ou Cortex-M0 (STMICROELECTRONICS, 2018).

Internamente, cada microcontrolador contém um núcleo, memória RAM, memória flash, interface para depuração e gravação, e vários periféricos.

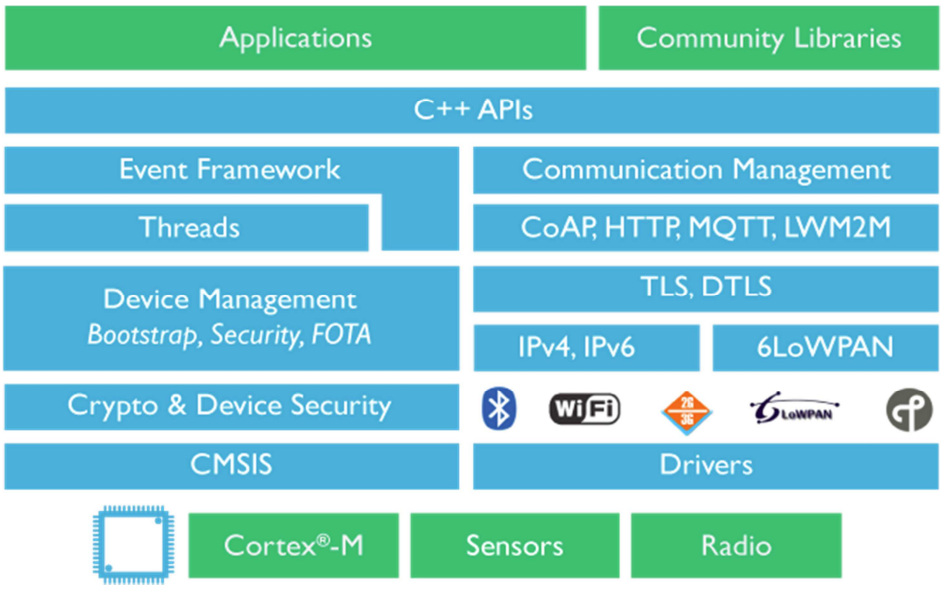
3.3. **Escolha de uma plataforma de desenvolvimento - ARM Mbed**

Uma vez definido qual microcontrolador utilizar se depara com a escolha de uma ferramenta que possa prover um bom ambiente de desenvolvimento de forma a maximizar chances de sucesso ao projeto. Atualmente, com o crescimento do mercado deste tipo de ferramentas nota-se uma demanda grande por ferramentas para construir aplicativos de forma rápida e econômica.

Há uma grande variabilidade destes ambientes no mercado e ao que eles ofertam. Deve-se estar atento às possibilidades que prometem soluções rápidas sem a sofisticação que os usuários finais exigem. Neste sentido, são destacados os três pontos não se esgotando a lista (CODESCHOOOL, 2014): (1) *Tooling* – Algumas IDEs também incluem ferramentas específicas a plataformas; (2) Sugestão de Código - outra característica comum entre diferentes IDEs é a sugestão de código inteligente; (3) Depuração - ajudam o desenvolvedor a encontrar bugs em seu código enquanto se está em execução, para que se possa removê-los antes de liberar um aplicativo.

Com o microcontrolador STM32, escolhe-se a IDE ARM MBED por suas características adequadas ao escopo deste projeto de computação embarcada e em destaque na Figura 3.1 (Mbed, 2017).

Figura 3.1 – Ambiente ARM Mbed.



Fonte: Adaptada de Mbed (2017).

O ambiente compreende toda uma plataforma de desenvolvimento para ARM (Cortex-M0 à M4), fornece um sistema operacional (RTOS), serviços de nuvem e ferramentas de desenvolvimento.

Adicionalmente, aplicações para a plataforma mbed podem ser desenvolvidas utilizando a IDE online mbed, que é um editor e compilador gratuito. Alternativamente, é possível exportar para outros ambientes de desenvolvimento também populares como o Kei uVision, IAR Embedded Workbench e o Eclipse como o GCC ARM Embedded tools.

A utilização do ambiente Mbed permite o **porting** de um programa feito para um determinado microcontrolador, para outros “targets”, ou seja, um programa desenvolvido para uma plataforma, pode ser compilado e gravado em outra plataforma de hardware, com poucas ou nenhuma modificação. Isto é algo muito importante, levando em conta que são utilizados microcontroladores de fabricantes diferentes em subsistemas utilizados pela comunidade de Cubesats. É possível utilizar 138 plataformas de hardware diferentes com o Mbed.

3.4. **Seleção de RTOS - Mbed RTOS**

A escolha de um RTOS adequado às necessidades do projeto é uma decisão muito importante. Um RTOS consiste em um kernel, parte central do projeto embarcado, e mais um *middleware* necessário para executar certas funções de maneira eficiente e confiável.

A escolha implica em se responder algumas questões como: (1) Robustez: o RTOS contribui para a robustez do dispositivo ou compromete-o? É propenso a erro do usuário? (2) Desempenho: O RTOS pode facilitar o desenvolvimento do código do aplicativo? O código executa dentro dos parâmetros requeridos? (3) Confiabilidade: o RTOS afeta a confiabilidade do dispositivo? e (4) Funcionalidade: O RTOS possui os recursos necessários para realizar o trabalho?

A escolha também implica em levantamentos de necessidades referente ao alvo final do sistema embarcado em aspectos como (Ralph Moore, 2018): (1) Uso de ROM; (2) Desempenho; (3) Recursos de depuração e gerenciamento de erros; (4) Solução balanceada ao problema tridimensional envolvendo tamanho, desempenho e recursos.

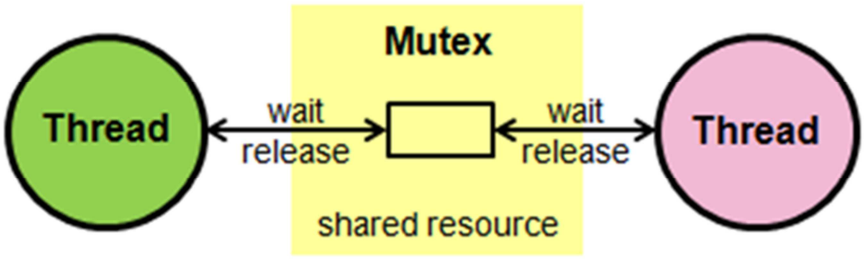
Levando em considerações estes diversos aspectos e o próprio hardware onde o sistema ficará residente bem como primitivas de coordenação e sincronização de processos disponíveis, neste trabalho optou-se pelo Mbed RTOS. A base do Mbed RTOS é a CMSIS-RTOS, que é uma API para sistemas operacionais de tempo real. Ela provê uma interface de programação padronizada que permite templates, middleware, bibliotecas e outros componentes que podem trabalhar em sistemas suportados por RTOS.

Em particular, a implementação de primitivas de coordenação entre threads é provida pelo MBed mediante diversas estruturas nativas do ambiente e que serão apresentadas brevemente a seguir. A função **`main()`** é uma *thread* especial, que inicia sua execução na inicialização do sistema e tem prioridade

inicial definida por uma variável, **`osPriorityNormal`**. Esta função é a primeira *thread* que o RTOS agenda.

Seções críticas são lidadas mediante uso de estruturas tipo **`Mutex`**, que é classe usada para sincronizar a execução de *threads* mostrada na Figura 3.2.

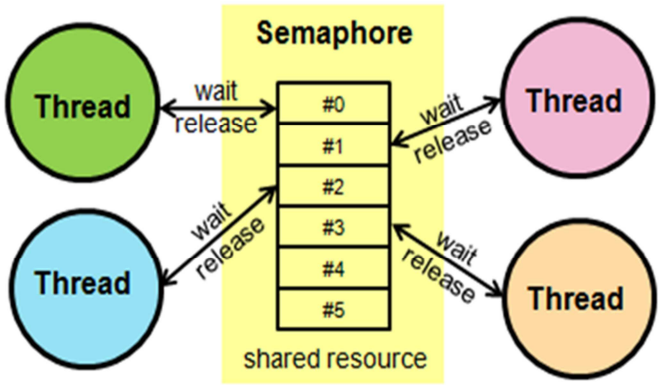
Figura 3.2 – Controle de seções crítivas com uso de **`Mutex`** no Mbed RTOS



Fonte: Adaptado de Mbed (2018).

Também disponíveis são estruturas semáforos mediante uso da classe **`Semaphore`**, mostrado na Figura 3.3, que gerencia o acesso de threads ao conjunto de recursos compartilhados.

Figura 3.3 – **`Semaphore`** no Mbed RTOS

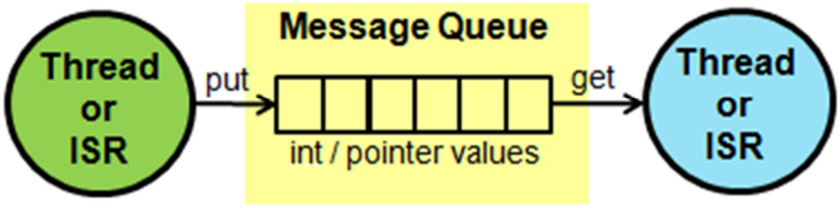


Fonte: Adaptada de Mbed (2018).

A sinalização entre processos e threads utiliza a estrutura **`Signals`**, onde cada *thread* pode esperar por sinais e ser notificada por eventos.

Também disponível é uma estrutura de dados em fila denominada **`Queue`** que é provida via uma classe que permite enfileirar ponteiros de dados de de *threads* produtores para *threads* consumidores como ilustrado na Figura 3.4 com uma **`Queue`** do Mbed RTOS.

Figura 3.4 – Estutura de dados Queue no Mbed RTOS

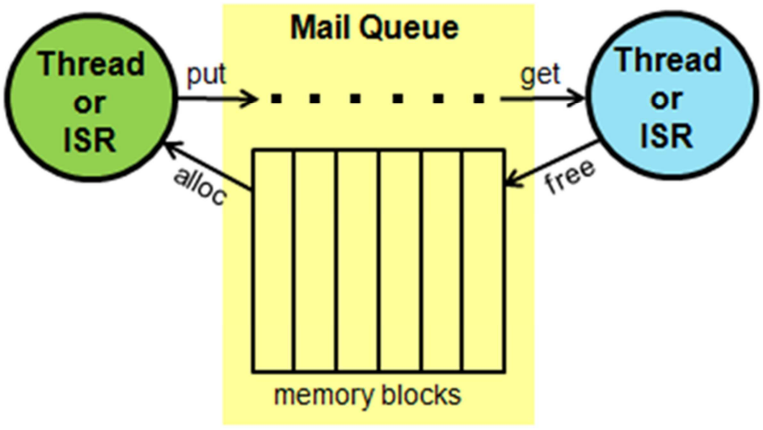


Fonte: Adaptada de Mbed (2018).

Para se definir e gerenciar recursos com memória de tamanho fixo, o Mbed provê uma classe chamada **`MemoryPool`**.

Uma estrutura para intercomunicação entre processos no MBed é a classe **`Mail`**, mostrada na Figura 3.5, que provê uma fila combinada com recursos de memória para alocar mensagens.

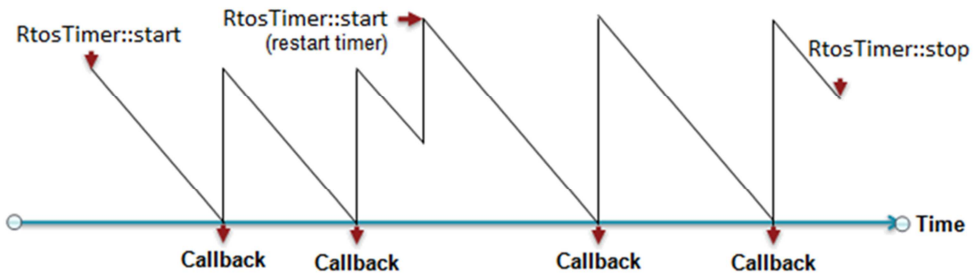
Figura 3.5 – O mecanismo de intercomunicação entre processos **`Mail`** no Mbed RTOS



Fonte: Adaptada de Mbed (2018).

Toda a temporização de sistema é provida utilizando-se a classe **`RtosTimer`** que controlar funções temporizadas e cujo comportamento é mostrado na Figura 3.6.

Figura 3.6 – Temporização de sistema provida por **`RtosTimer`** no Mbed RTOS



Fonte: Adaptada de Mbed (2018).

Finalmente o gerenciamento de eventos é realizado empregando-se duas classes especializadas. A classe **`EventFlags`** provê uma forma de notificar *threads* sobre condições ou eventos. Por outro lado, a classe **`Event`** provê uma abstração em estrutura de fila que armazena eventos, os extrai para executá-los posteriormente.

3.5. **Desenvolvimento Orientado a Testes (TDD)**

Neste trabalho foi adotada a estratégia de desenvolvimento orientado a testes (TDD) por este auxiliar o programador a elaborar um código com cada vez mais qualidade criando objetos concisos e com menos dependências.

TDD é uma técnica de desenvolvimento de software em que o desenvolvedor deve começar a implementação pelo teste e, deve, durante todo o desenvolvimento, manter o código simples e com qualidade. O primeiro passo é escrever um teste que falha. Em seguida, é modificado o código para que passe no teste, e depois é feita a refatoração para melhor o código escrito.

baseada na criação de um teste automatizado que falhe antes mesmo da implementação de um novo código (ANICHE, 2012).

O ciclo de um desenvolvimento utilizando TDD é feito da seguinte maneira:

a) Inicialmente implementa-se um teste para a funcionalidade em questão. Sabe-se que o teste não vai passar, pois ainda não foi implementado o código necessário;

b) Depois de executar o teste, e verificar a falha, escreve-se o código da funcionalidade de forma simples, para apenas atender, momentaneamente, o teste quebrado.

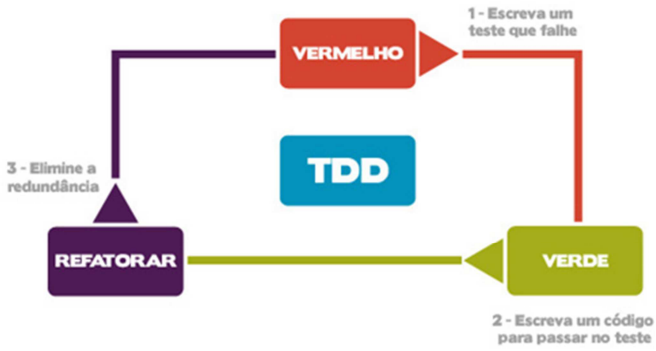
c) Finalizada essa implementação, deve-se novamente executar o teste que estava falhando e o esperado é que rode com sucesso. Caso o teste não passe deve-se rever o código implementado até que ele seja executado corretamente.

d) Feito isso, deve-se verificar o código para refatorar o mesmo, de forma a aplicar padrões de projeto.

e) A cada refatoração realizada, deve-se executar novamente o teste e verificar se ainda está passando pelo teste. Se tudo estiver funcionando, reinicia-se o processo com a próxima funcionalidade do sistema.

Analisando uma nova funcionalidade em específico, aplica-se o TDD até que seu desenvolvimento se conclua. Esse conjunto de passos, mostrado na Figura 3.7, é muito difundido entre os desenvolvedores que trabalham com TDD pelo nome de "*red-green-refactor*".

Figura 3.7 – Diagrama de blocos da iOBC-P



Fonte: Artigo TDD (2017).

Dentre as vantagens desse tipo de desenvolvimento pode-se destacar:

a) Apesar de o processo parecer mais demorado, pois se deve escrever teste e depois código, essa prática se converte em menor tempo de depuração e correção de defeitos, além de naturalmente aumentar a qualidade do código escrito.

b) Com a utilização do TDD, o código ficará mais fácil de ser testado. Como consequência disso, é bem provável que o código fique mais claro, dispensando documentações desnecessárias.

c) Utilizando TDD tem-se a garantia que todas as funcionalidades do sistema estarão cobertas por testes. Sendo assim, consegue-se garantir que alterações futuras no código não afetarão implementações de funcionalidades já concluídas, pois o teste previamente implementado irá falhar.

3.6. **Mapeamento de TDD para o ambiente Mbed**

O ambiente de programação Mbed possui ferramentas para se implementar TDD, quais sejam: *Greentea*, htrun e mbed-ls.

O Greentea (*Generic Regression Environment for Test Automation*) é uma ferramenta para automatizar testes do ambiente Mbed. Ele automatiza o processo de gravação placas Mbed, realizar testes e acumular resultados.

Greentea utiliza uma conexão serial duplex entre o host e o dispositivo Mbed. Os testes rodam no dispositivo sob supervisão dos scripts de teste do host, providos pelo Greentea sem qualquer interação do usuário. Os scripts geram uma série de chamadas de **`print`** e **`sscanf`**, que são mostradas de forma legível.

Para cumprir suas funcionalidades, o Greentea utiliza duas ferramentas, que são distribuídas em scripts Python:

a) mbed-htrun: é o mecanismo do Greentea, que monitora a execução e controla o fluxo de testes;

b) Mbed-ls: detecta e lista os dispositivos Mbed bem como seus parâmetros de configuração;

Para se utilizar o framework Greentea deve-se seguir este procedimento:

a) Escrever os testes em C ou C++. Estes testes precisam incluir procedimentos de teste e definir como serão os resutados dos testes;

b) Se necessário, pode-se escrever testes para o host em Python;

c) Invocar o Greentea e apontar para onde os testes estão;

d) Greentea utiliza a ferramenta mbed-ls para descobrir as placas conectadas no host;

e) Para cada placa identificada pelo mbed-ls, um script mbed-htrun reseta os dispositivos;

f) O teste é executado automaticamente, depois que o dispositivo é resetado. Um script mbed-htrun gera um relatório de teste (com arquivo de texto ou no JSON);

g) Greentea mostra os resultados dos testes de forma legível.

Adicionalmente às ferramentas apresentadas, também é possível se utilizar outras ferramentas de texto, como a CppUTest.

3.7. **Aspectos de engenharia de sistemas**

Além dos aspectos referentes a infraestrutura computacional propriamente dita, é necessário se considerar aspectos mais abrangentes relativos ao sistema em si.

O desenvolvimento integrado de HW/SW de computação de bordo para apoio a cargas úteis em plataformas demanda uma adaptação não só de HW e SW, mas também de aspectos de Engenharia de Sistemas Espaciais. Alguns desafios são listados abaixo:

a) Analisar Custo-Benefício e viabilidade de embarque de cargas úteis da CEA;

b) Compactação física com componentes eletrônicos reduzidos;

c) Emprego de outras soluções de rede de dados a bordo, e. g. CAN Bus (*Controller Area Network*);

d) Obedecer a envelopes de Engenharia de Sistemas, tipicamente massa, potência, interferência/compatibilidade eletromagnética EMI/EMC (Interferência Eletromagnética - Electromagnetic Interference, Compatibilidade Eletromagnética - Electromagnetic Compatibility), radiação, entre outros;

e) Manter a funcionalidade computacional do sistema original em seu equivalente reduzido de forma a continuar lendo dados das cargas úteis e enviando-os ao Computador de Bordo;

f) Investigar eventual processamento a bordo para diminuir a largura de banda dos dados a serem enviados para solo pelo Computador de Bordo do Nanossatélite e/ou algoritmos de compactação;

g) Explorar o emprego de processadores disponíveis e seu desenvolvimento;

h) Definição de uma arquitetura final mínima para cargas úteis do CEA, para tanto diversos modelos devem ser gradualmente trabalhados.

A aplicação destas considerações resulta no trabalho descrito no Capítulo 4 a seguir no qual foram usados os requisitos mínimos para que atendam as cargas úteis da CEA. Isto implicará na definição de três modelos de trabalhos da Interface OBC-P – Carga Útil, a saber: (1) Modelo Emulado; (2) Modelo com Prototipação e (3) Modelo no Target Final.

4 DESENVOLVIMENTO DA COMPUTAÇÃO DE BORDO

Neste capítulo é apresentado o desenvolvimento da computação de bordo mediante as premissas estabelecidas no Capítulo 3 que estabeleceu, em alto nível, a metodologia adotada e aqui executada visando cargas úteis da CEA-INPE. Este desenvolvimento foi feito utilizando três etapas de modelos da Interface OBC-P – Carga Útil visando emular, prototipar e finalmente, executar no processador *target* para vôo.

Após a emulação e suas verificações, é feita a prototipação de um sistema de computação para suporte a uma Sonda de Langmuir SLP (*Sonda de Langmuir Probe*) e, em seguida, reposicionado para ser prospectivamente embarcado em um nanossatélite. Atualmente na Pós-Graduação do PG-ETE do INPE o nanossatélite Alpha-CTEE está em fase de projeto conceitual e poderá embarcar este experimento.

4.1. **Emulação Mbed da comunicação entre OBC-P e Cargas Úteis**

Visando iniciar o trabalho prático e provar o conceito por trás desta dissertação, esta seção descreve um pequeno sistema de emulação da comunicação I2C entre um OBC-P e uma carga útil. Esta emulação foi realizado no LabSim – Laboratório de Simulação do INPE no âmbito do programa NanosatC-BR para desenvolvimento de cubesat. Em especial, visa o desenvolvimento final de rotinas para operação do satélite NanosatC-BR2.

O NanosatC-Br2 consiste em um nanossatélite concebido pelo Centro Regional Sul (CRS) do Instituto Nacional de Pesquisas Espaciais (INPE) em parceria com a Universidade Federal de Santa Maria - UFSM, cuja missão é colocar em órbita cargas científicas e tecnológicas, para estudo e validação. Esta arquitetura é uma evolução do NanosatC-Br1, primeiro nanossatélite brasileiro lançado.

A abordagem pragmática adotada emula um sistema de comunicação entre um OBC-P e uma carga útil, Sonda de Langmuir (SLP), utilizando o padrão I2C. Apesar de simples, este recurso é muito utilizado para validar conceitos de interface entre os dispositivos em algumas aplicações.

O resultado do trabalho é um conjunto de testes e simulações no que tange ao princípio de funcionamento do I2C, sua transmissão de dados e possíveis limitações da arquitetura de comunicação.

4.1.1. Planejamento da emulação

Para a consecução desta etapa do desenvolvimento, um planejamento passo-a-passo foi realizado buscando o cumprimento das atividades previstas para pesquisa, análise, execução e documentação do realizado, listadas a seguir:

a) Estudar o microcontrolador STM32F103C8T6, e da placa de desenvolvimento;

b) Montar a primeira placa STM32F103C8T6 no *protoboard*;

c) Realizar um estudo do ambiente e linguagem de programação MBED para microcontroladores ARM;

d) Testar o ambiente de desenvolvimento (hardware e software), acionando o LED existente na placa STM32, e comunicação serial com o PC;

e) Montar a segunda placa STM32F103C8T6, e repetir o item (d);

f) Realizar um estudo do barramento I2C, e dos comandos da carga útil escolhida para o trabalho (ISLP).

Os materiais e softwares de apoio utilizados nos testes e simulações são detalhados a seguir:

- Hardware
    - 02 placas de desenvolvimento STM32F103C8T6

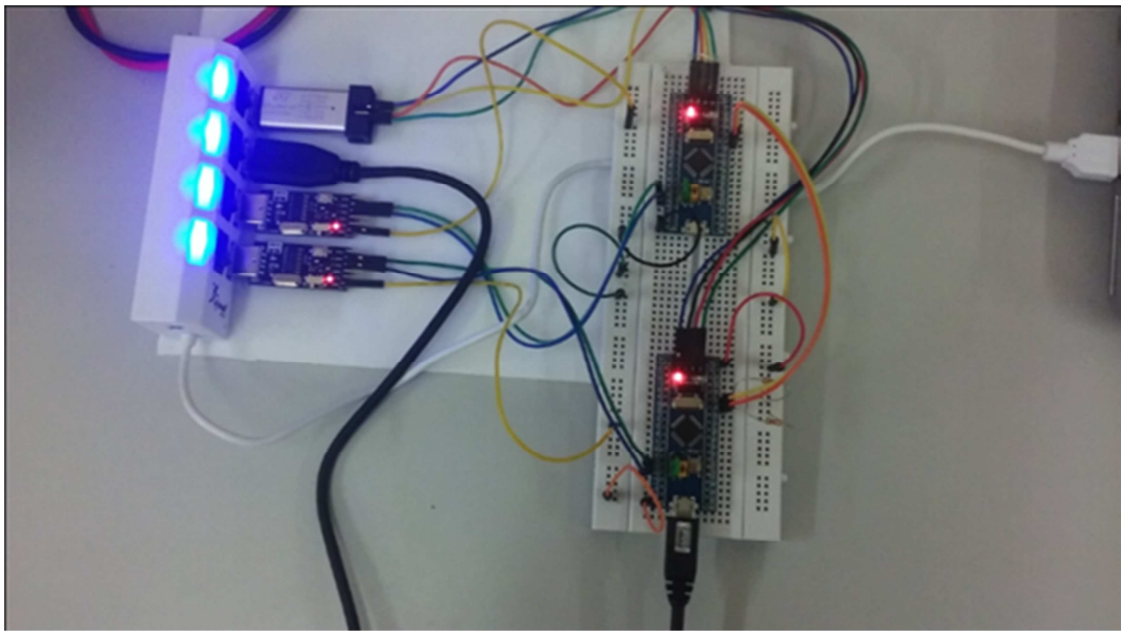
- 01 programador *in-circuit debugger* ST-LINK V2
- 01 interface serial USB-TTL (3V3)
- 01 hub USB – 3 portas
- *Protoboard* e *jumpers*
- Microcomputador com Win7 e conexão com internet

- Software:
  - Ambiente MBED de programação (online)
  - Terminal serial (Putty ou Real Term)

4.1.2. Simulação do computador de bordo e sonda de Langmuir

A simulação do computador de bordo e da sonda de Langmuir foi feita com a utilização de dois microcontroladores STM32F103C8T6, mostrados na Figura 4.1, que possuem diversas configurações de comunicação como: Serial, I2C e CAN. Este microcontrolador dispõe de uma saída USB com a qual se pode configurá-los.

Figura 4.1. Emulação Mbed da comunicação I2C entre um OBC-P e uma SLP.



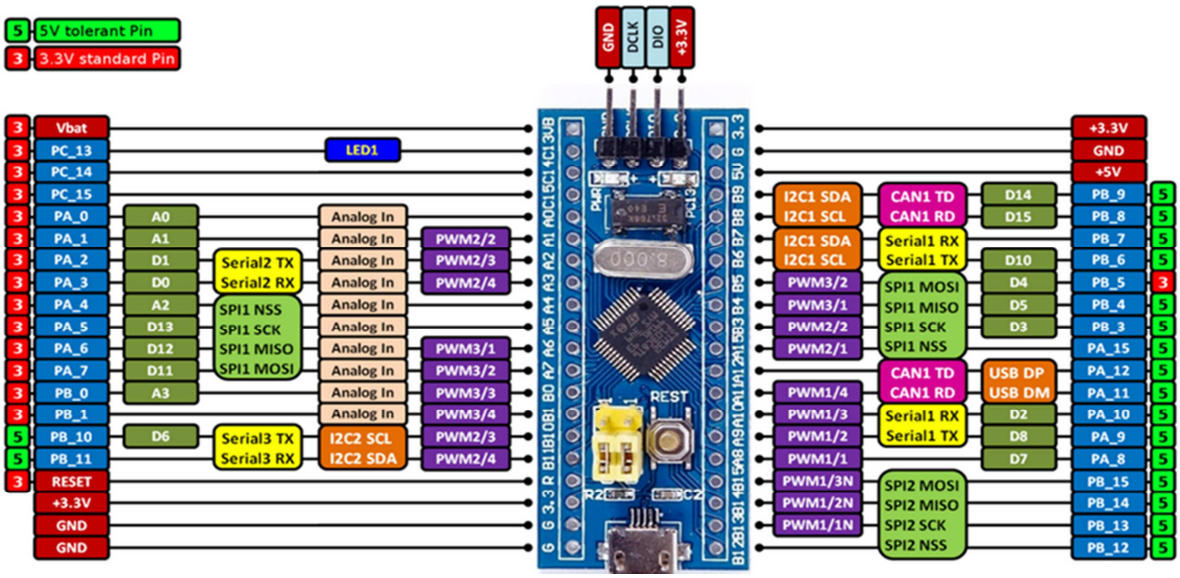
Fonte: Produção do autor.

Foi simulada a comunicação entre o computador de bordo e a sonda de Langmuir ao se enviar um horário para inicializar as medidas de dados, e posterior armazenagem de forma a poder modificá-los ao decorrer das medições.

Dois meios de comunicação foram escolhidos, *a priori* seriais e *posteriori* I2C, os quais possibilitam estabelecer uma troca de informações entre os dispositivos e observar o terminal do computador simultaneamente para fins de *cross-checks*.

Para configuração da taxa de transferência de dados tanto *upload* (Rx) quanto *download* (Tx) na interface serial e a determinação do **`Serial Data SDA`** e **`Serial Clock SCL`** no padrão I2C, seguiu-se pinagem descrita na Figura 4.2 abaixo. Este dispositivo apresentou-se muito versátil e de rápida familiaridade durante a condução do trabalho.

Figura 4.2. Configuração de sinais da interface serial do STM32F103C8T6.



Fonte: Adaptada de Hudak (2017).

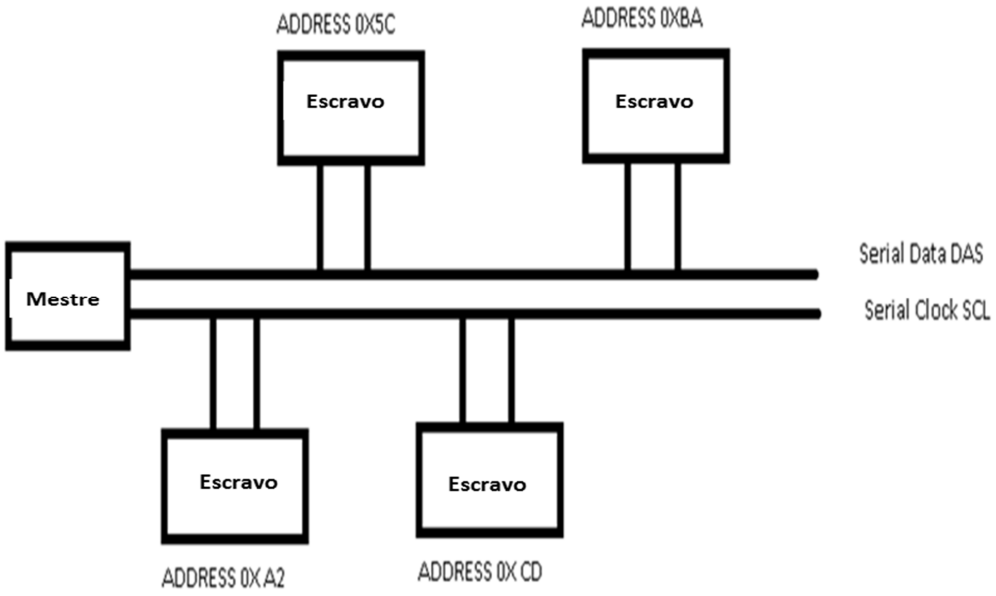
Para a simulação de dados do experimento Sonda de Langmuir, utilizou-se duas implementações, para enviar dados para a ISLP: utilizando um divisor de

tensão para fornecer dados pré-definidos de tensões e um gerador de sinais, para simular os quatro canais analógicos com taxa de aquisição de 16 Hz (SLPA, SLPB, SLPC e SLPP), e o canal SLPI com taxa de aquisição de 512 Hz.

4.1.3. Emulação do protocolo de comunicação OBC-P - ISLP

O barramento I2C é um padrão de comunicação que possibilita a troca de informações entre componentes padronizados com eficiência e facilidade aceitáveis. Os dispositivos ligados aos sinais DAS e SCL possuem endereços fixos e são configurados para transmitir ou receber informações, assumindo assim o papel de Mestre ou Escravo. A Figura 4.3 apresenta o diagrama estrutural do padrão I2C com um esquema de interconexões e endereçamento.

Figura 4.3. Diagrama do padrão I2C.



Fonte: Produção do autor.

No padrão I2C, o protocolo de comunicação só se inicia quando o mestre envia um dado de tamanho de 1 byte no barramento (Serial Data SDA) com endereço de um escravo. Caso o dispositivo destino não possa operar com o

dado, este poderá alterar a linha do Clock (SCL), colocando-a no nível baixo, fazendo com que o mestre entre em um estado de espera.

Após o processo de escrita ou leitura do byte do barramento, o dispositivo receptor gera automaticamente um sinal de reconhecimento (*acknowledge*).

Dentro do byte transmitido, é reservada uma região para o endereçamento de 7 bits para especificar o dispositivo escravo a ser acessado, seguido por um bit indicador de escrita ou leitura.

Para definir o início da leitura de dados, é necessário que o computador de bordo (mestre) envie ao controlador da sonda (escravo) um tag de tempo que sinaliza ao mesmo o *start* da aquisição de pacotes. O *clock* interno do dispositivo escravo passa por ajustes periódicos para sincronização a cada solicitação de dados.

Ao término da sincronização, são enviados os dados coletados para um buffer circular. Quando o computador de bordo lhe envia o número de blocos desejados para leitura, o dispositivo escravo fornece estes dados bem como seu *tag* de tempo atualizado.

Após definidas a configuração do microcontrolador e estrutura do protocolo de comunicação, utilizou-se uma interface de programação em linguagem C para simular a comunicação entre o computador de bordo e a sonda.

Para tanto foi concebido uma biblioteca para a ISLP no ambiente Mbed listada na Tabela 4.1 e denominada **`Slp`**.

Tabela 4.1 – Métodos da classe **SLP**

Função	Descrição
`Slp::Slp(PinName sda, PinName scl)`	Define pinos do canal I2C
`void Slp::configura(int endereco,int num_blocos, int indice_bloco)`	Define endereco, numero de blocos e o indice do bloco que será lido
`int Slp::disponivel()`	Verifica se a ISLP está disponível. Se náo esta realizando o processamento dos blocos
`char Slp::escreve_hora()`	Envia a hora para a ISLP
`char Slp::configura_bloco(int numero_blocos)`	Seleiciona quantos blocos serão lidos
`char Slp::configura_indice_bloco(int indice)`	Seleciona o índice do bloco que será lido
`char * Slp::inicia_tx()`	Inicial leitura dos dados
`char Slp::armazena_memoria()`	Armazena dados no cartão micro SD do computador de bordo.

Fonte: Produção do autor.

A biblioteca para a ISLP no Mbed foi estruturada em dois arquivos, **`Slp.h`** e **`Slp.cpp`**, para a publicação e estão disponíveis nos apêndices A e B.

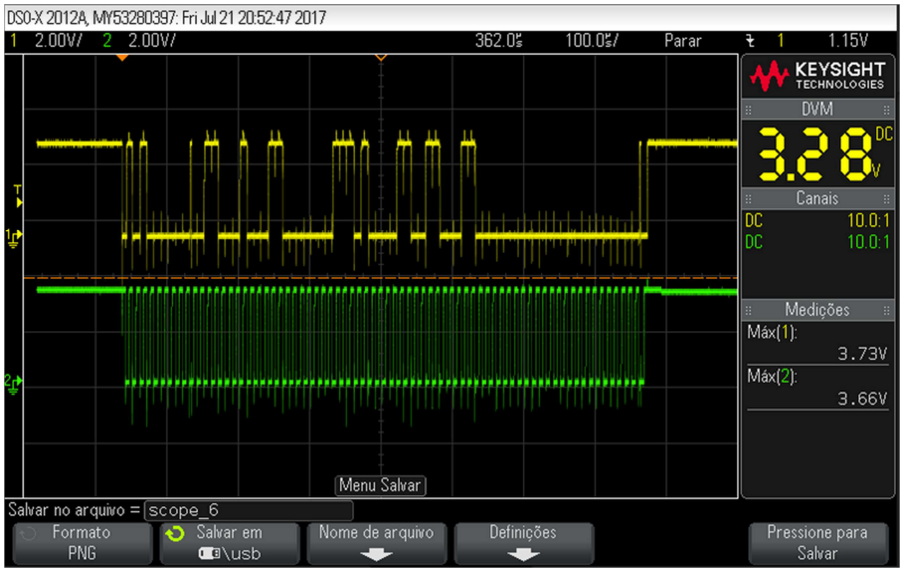
4.1.4. Testes e resultados obtidos na emulação

Foram realizados alguns testes e simulações com o intuito de verificar o funcionamento e cumprimento de cada etapa de atividades planejada. Em um primeiro momento foi realizada a comunicação com o terminal do computador e

posteriormente foi simulada uma comunicação mestre-escravo entre dois microcontroladores STM32F103C8T6 com troca de dados sinalizada por LED vermelho piscando como mostrado anteriormente na Figura 4.1.

Ao decorrer da simulação foi verificada a compatibilidade do hardware com o software e possíveis limitações como mostra a Figura 4.4. A comunicação I2C foi validada com o auxílio da interface computacional (terminal) e um osciloscópio o qual possibilitou susceptíveis medições e análises dos dados transmitidos pelo barramento.

Figura 4.4. Verificação e validação de sinais do nível físico do protocolo



Fonte: Produção do autor.

A emulação do sistema de comunicação entre a OBC-P e a carga útil é muito utilizada na verificação e validação do interfaceamento para aplicações de satélites. Este cuidado antes de se implementar o sistema final é importante para garantir que critérios e restrições sejam respeitadas, de forma a evitar possíveis falhas. Uma vez realizado esta etapa, passa-se a etapa de prototipação e testes da interface com mais detalhes.

4.2. **Prototipação de computação de bordo para sonda Langmuir**

Esta fase de prototipação foi necessária, pois tanto a carga útil final quando também o processador final não estava disponível. Os conceitos trabalhados serão adaptados quando da versão final do processar alvo da missão nanossatélite.

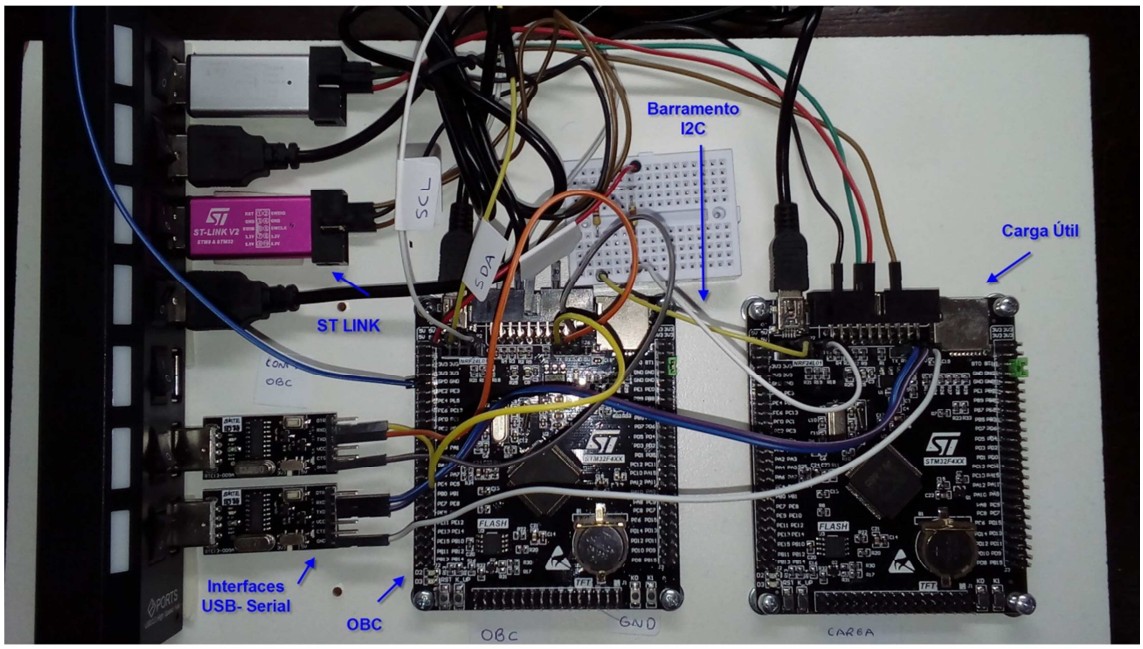
A prototipação consiste na definição de hardware mínimo e de um conjunto de aplicativos / protocolo de comunicação entre o OBC-P e a carga útil propriamente dita.

4.2.1. Sistema mínimo de hardware

O sistema mínimo de hardware consiste basicamente de um conjunto de componentes eletrônicos comerciais e placas de forma a se iniciar o desenvolvimento do projeto e sua prototipação.

Este sistema mínimo foi montado, ilustrado na Figura 4.5, para servir como *baseline* e é constituído de duas placas de desenvolvimento com o microcontrolador STM32F407VET de baixo custo. Uma placa emulando o computador de bordo OBC-P e a outra emulando uma carga útil Sonda de Langmuir.

Figura 4.5 – Sistema mínimo de hardware para computação de bordo



Fonte: Produção do autor.

Um barramento I2C foi utilizado para que o OBC-P se comunicasse com a carga útil bem como foi desenvolvido um sistema de testes da comunicação entre ambos. Uma das placas simula o OBC-P com seu respectivo SD Card, para armazenar os dados da carga útil, A outra placa emula a Sonda de Langmuir.

Para a gravação dos microcontroladores e depuração, foram utilizadas interfaces ST LINK (JTAG) e para envio de mensagens de comunicação serial, conversores USB-Serial. Com este sistema é possível se emular a comunicação via interface I2C e também CAN.

4.2.2. Biblioteca para a carga útil sonda SLP

Para se elaborar a biblioteca de software para a leitura de dados da carga útil SLP pelo computador de bordo, foi utilizado como referência o documento “Interface ILP/BR2 – Interface para o Experimento Sonda de Langmuir para Cubesat BR2” (HORNA, 2016).

A interface da Sonda de Langmuir (ISLP) adquire cinco sinais analógicos: SLPA, SLPB, SLPC, SLPP e SLPI. Os 4 primeiros sinais que possuem uma taxa de amostragem de 16 Hz e o último sinal SLPI é amostrada numa taxa mais alta, 512 Hz, conforme mostra a Tabela 4.2. Com relação ao pré-processamento destes dados, apenas o sinal SLPI demandará FFT (*Fast-Fourier Transform*) como forma de se compactar dados, uma vez que apenas o conteúdo espectral do sinal é enviado.

Tabela 4.2 – Sinais analógicos do experimento SLP

Sinal	Canal	Nome	Tipo (D/A)	Varredura Hz	FFT
SLPA	A	Sinal de Alto Ganho	Analog	16	No
SLPB	B	Sinal de Baixo Ganho	Analog	16	No
SLPC	C	Sinal Compactado	Analog	16	No
SLPP	P	Sinal Polarização	Analog	16	No
SLPI	I	Sinal Integral	Analog	512	Sim

Fonte: Produção do autor.

Os dados lidos do experimento são gravados num bloco de envio de 1 segundo de duração de aquisição destes dados. Alternativamente, pode-se transferir essa informação também via pacotes de dados retirados da memória RAM do microcontrolador da ISLP e posteriormente transmitidos ao computador de bordo quando ele próprio solicitar.

Um pequeno protocolo de transferências de dados foi utilizado e estruturado como descrito a seguir. Cada bloco de envio de 1 segundo de duração é precedido de um header de 18 bytes e finalizado com 2 bytes de *checksum*, tendo o bloco total, com a seguinte formatação de dados:

a) 18 bytes para o header
b) 16 bytes para cada canal de 16 Hz (sinais LPA, LPB, LPC e LPF), totalizando 64 bytes para os 4 canais

c) 16 bytes para o canal de 512 Hz (sinal LPI compactado por FFT)

d) 2 bytes para o *checksum* do bloco

O metadados dessa transferência de dados segue o formato mostrado na Tabela 4.3.

Tabela 4.3 – Formato de bloco de envio de 1 segundo

Informação	**No.Bytes**	**Descrição**
Header	18	Header do bloco
Bloco Canal A	16	Sinal SLPA de 16Hz
Bloco Canal B	16	Sinal SLPB de 16Hz
Bloco Canal C	16	Sinal SLPC de 16Hz
Bloco Canal P	16	Sinal SLPP de 16Hz
Bloco Canal I ou FFT	16	Sinal SLPI de 512Hz compactado por FFT
Checksum	2	Checksum do bloco
Total	100	Tamanho do bloco

Fonte: Produção do autor.

O software da ISLP armazena até 300 blocos de 1 segundo de duração de dados adquiridos, ou seja, 300 segundos equivalentes a 5 minutos de dados.

Este dados passam por uma gravação realizada de forma cíclica adotando um esquema de *Round-Robin* usando um buffer circular. Uma vez gravado o último bloco (300/300), sobrescreve-se desde o começo (1/300).

A ISLP se comunica com o computador de bordo através do barramento I2C, no modo *Slave*, onde a este periférico é designado o endereço 0x3C.

Os comandos que compõe o vocabulário de comunicação e que devem ser enviados pelo *Master* (computador de bordo) ao ISLP são definidos na Tabela 4.4.

Tabela 4.4 – Comandos enviados pelo computador de bordo para a ISLP

Comando	Byte	Descrição
SlaveReady	0xF0	Perguntar se Slave está pronto para receber comandos
SetDateTime	0xF1	Envia data e hora para o timestamp do header de cada bloco e indica à interface que deve começar a preencher o buffer de 30K
SetBlockTx	0xF2	No. de blocos de 100 bytes a enviar
SetBloIndex	0xF3	Seta índice do bloco do buffer a ser enviado
StartTx	0xF4	Começar a transmissão

Fonte: Produção do autor.

Por sua vez, a semântica de retorno dos periféricos, ou seja, as respostas que compõe o vocabulário de comunicação e que devem ser enviados pelo *Slave* (ISLP) ao OBC-P são definidas na Tabela 4.5.

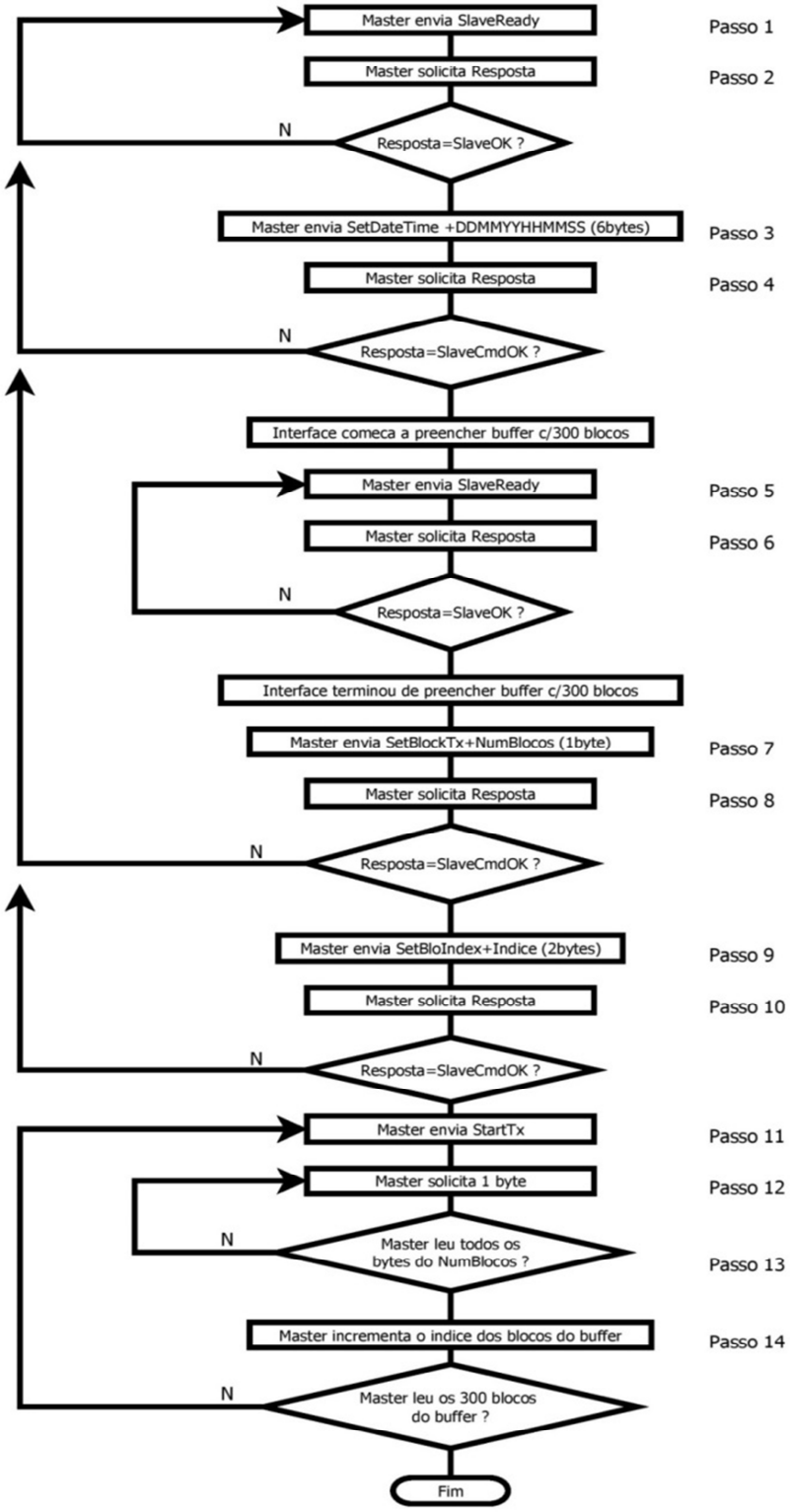
Tabela 4.5 – Mensagens enviadas pela ISLP ao computador de bordo

Comando	Byte	Descrição
SlaveOk	0xFA	Slave disponível para receber comandos
SlaveBusy	0xFB	Slave ocupado preenchendo buffer de dados
SlaveCmdOk	0xFC	Slave recebeu comando válido
SlaveCmdUknw	0xFD	Slave recebeu comando invalido ou desconhecido

Fonte: Produção do autor.

O fluxograma da Figura 4.6 mostra um modelo de sessão de diálogo entre o computador de bordo com a ISLP.

Figura 4.6 Fluxograma do programa que simula interação OBC-P e ISLP



Fonte: Adaptada de Horda et al. (2016).

4.3. **Computador de bordo para o prospectivo nanossatélite Alpha-CTEE**

O programa CTEE (Capacitação Tecnológica em Engenharia Espacial) é uma iniciativa da Pós-Graduação em Engenharia e Tecnologia Espacial PG-ETE em aprendizado baseado em problemas (PBL – Problem-Based Learning) [HMELO-SILVER, 2004] e desenvolvida pelos seus alunos e professores. CTEE pretende realizar uma missão constituída de três nanossatélites baseados na plataforma cubesat de desenvolvimento incremental e denominados de Alpha, Beta e Gama [CTEE, 2017].

Este programa visa internalizar conceitos teóricos em aspectos práticos na construção de nanossatélites que atendam objetivos de interesse ao INPE. Um desses aspectos é o da computação de bordo, motivo pelo qual este trabalho o aborda.

Neste escopo, o trabalho apresenta uma proposta de plataforma de prospectivo computador de bordo em hardware e software para o nanossatélite baseado na plataforma cubesat denominado Alpha do programa CTEE e seguindo diretrizes listadas em [WERTZ, 2011].

4.3.1. Seleção de uma plataforma computacional

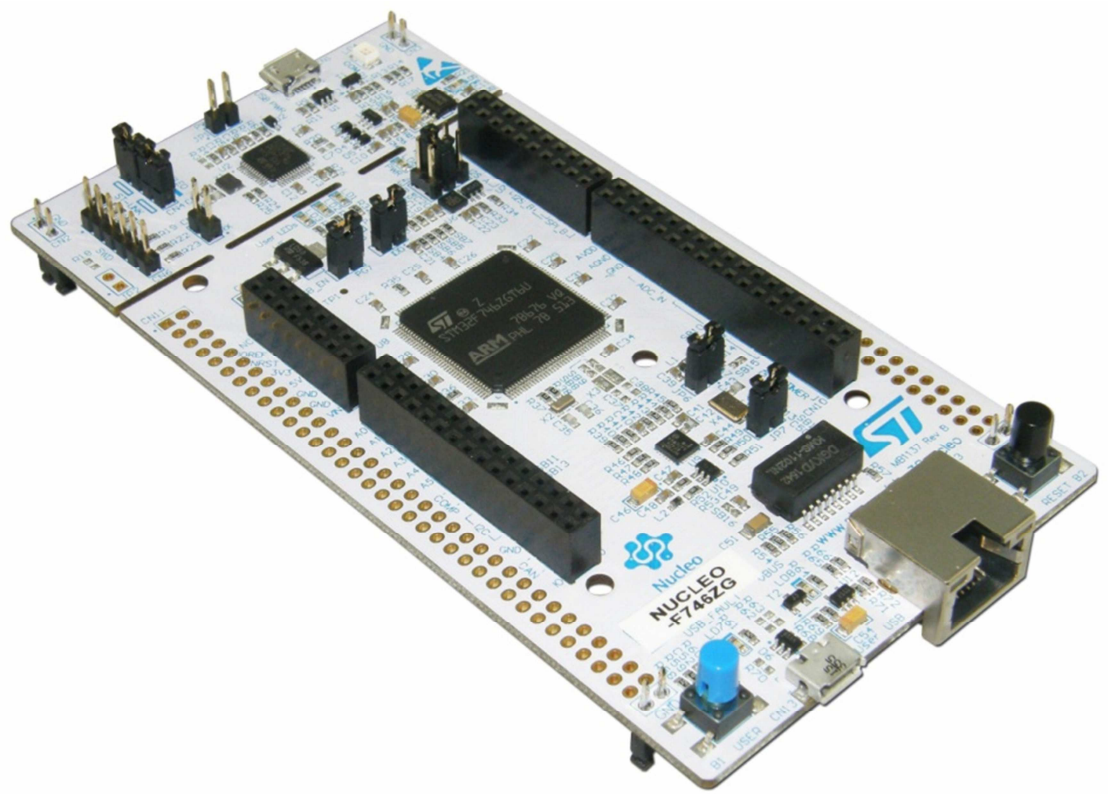
Após diversas reuniões de análise iniciais com *stakeholders* da primeira missão do cubesat Alpha, alguns requisitos foram identificados que culminaram com a seleção da plataforma STM32 da STMicroelectronics [STMicroelectronics, 2017].

Esta plataforma foi adotada como base de hardware para o OBC-P por apresentar histórico de vôo, baixo custo e um conjunto de ferramentas de desenvolvimento de software gratuitas. O seguinte roteiro de ações foi realizado para o desenvolvimento de hardware e software:

a) Iniciar o desenvolvimento como base na placa Núcleo F746ZG, mostrada na Figura 4.7. Essa placa utiliza o microcontrolador STM32F746ZG (STMicroelectronics, 2017).

b) Desenvolver uma implementação em software para testes utilizando o ambiente Mbed [MBED, 2017]. Para a concretização de alguns modos de operação e comunicação com cargas úteis, foi concebido o diagrama de estados do OBC-P embarcado no cubesat Alpha, discutido a seguir.

Figura 4.7. Placa núcleo F746ZG com processador STM32F746ZG.



Fonte: Adaptada de STM32 (2017).

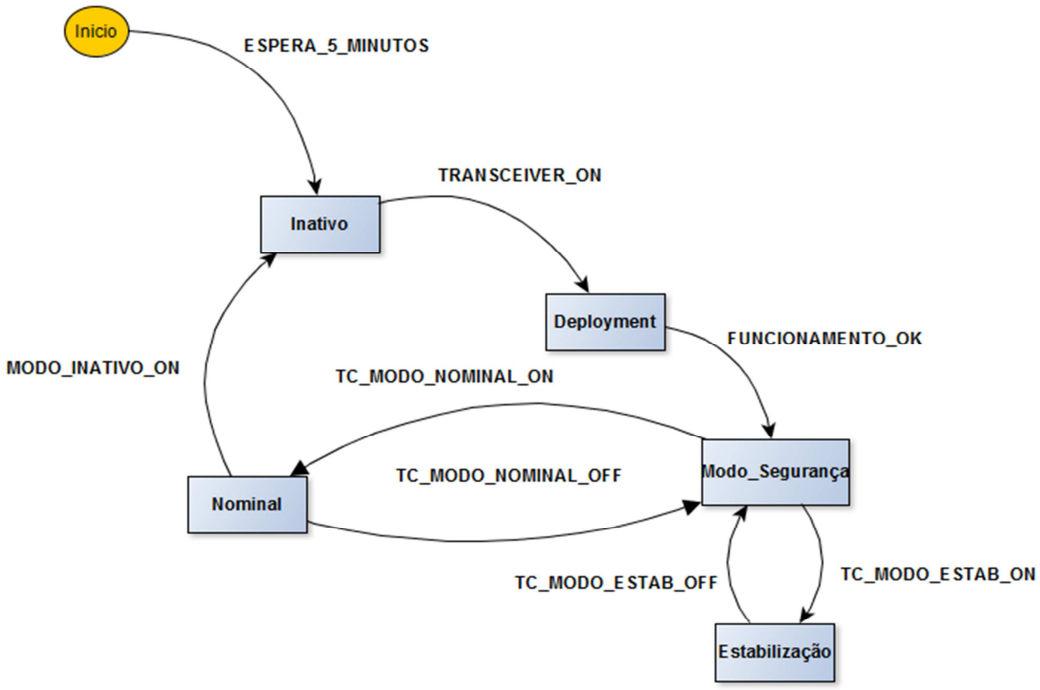
Portanto a arquitetura mínima escolhida para este OBC-P do cubesat Alpha possui a seguinte configuração:

- Hardware: OBC-P com o microcontrolador STM32F746ZG
- Software de bordo: desenvolvimento em mbed (com RTOS)

4.3.2. Máquina de estados macro para o software de bordo

Após diversas reuniões de análise iniciais foi concebida uma máquina de estados que descreve o comportamento macro do OBC-P para uma missão genérica do nanossatélite Alpha, conforme mostrada na Figura 4.8. Basicamente, o software de teste do OBC-P implementa a máquina de estado que descreve o seu comportamento desejado.

Figura 4.8. Diagrama de estados macro do software de teste da OBC-P



Fonte: Produção do autor.

Sinteticamente, a máquina de estados do software da OBC-P possui o seguinte funcionamento:

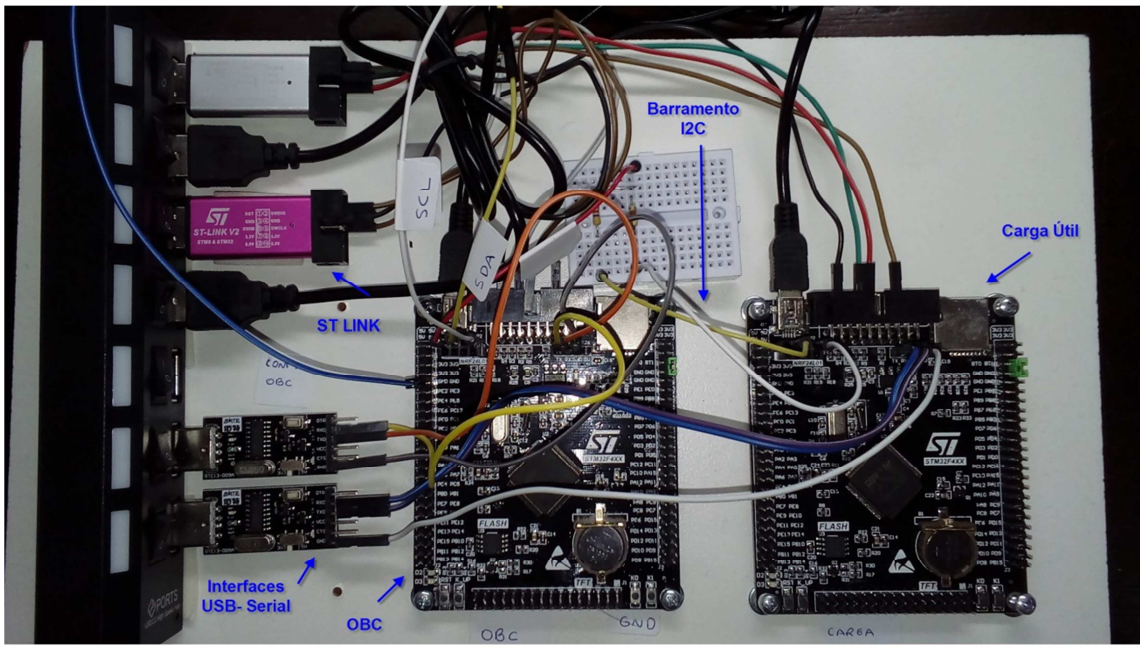
- Após a ejeção do nanossatélite, todos os subsistemas estarão desligados. Após 5 minutos, o software entrará no modo inativo no qual apenas é iniciado o sinal de *beacon*.
- São realizados testes para verificar a integridade da OBC-P e do sistema de armazenamento de dados da OBC-P.

- Caso esteja OK, o software entra no modo de *Deployment* em que é realizada uma verificação do módulo de radio habilitando-o para recepção.
- Após estes testes, o software entra em modo de Segurança, aguardando novos telecomandos.
- O OBC-P aguarda o recebimento de um telecomando para entrar em modo Nominal.
- Em operação Nominal, as cargas úteis começam a operar segundo o plano de vôo executando tarefas agendadas para sua missão.
- O modo de estabilização de atitude será ativado, caso receba um telecomando para habilitar o funcionamento de um determinado módulo de interesse da missão, por exemplo, o módulo SDATF que realiza experimentos de determinação de atitude.

A vantagem de desenvolvimento no ambiente Mbed é explicitada na Figura 4.9, pois acaso haja um novo nanossatélite que faça uso de outro processador *target*, é possível o reuso do código implementado. O processo de *deploy* para outros *targets* se desdobra nos seguintes passos básicos:

- Passo 1 – O código em C++ é construído usando o framework do Mbed;
- Passo 2 – Dentro do ambiente Mbed define-se uma *target*;
- Passo 3 – Um processo de compilação para o *target* é iniciado
- Passo 4 – O código objeto sofre *deploy* para a plataforma final.

Figura 4.9. *Porting* Mbed para diferentes plataformas



Fonte: Produção do autor.

Um maior detalhamento da máquina de estados da Figura 4.8, anteriormente apresentada, definirá o comportamento da arquitetura de software do OBC-P e seus periféricos. Esta máquina de estados mais complexa inclui uma série de estados aninhados que basicamente descrevem o software de tratamento dos diversos subsistemas como os de potência, AOCS (controle a atitude e apontamento), TTC (telemetria e telecomando) e particularidades de cargas úteis do CEA.

Devido ao escopo deste trabalho e restrições de tempo, esse maior desenvolvimento será sugerido como trabalhos futuros.

5 TESTES DA INTERFACE OBC-P – CARGAS ÚTEIS

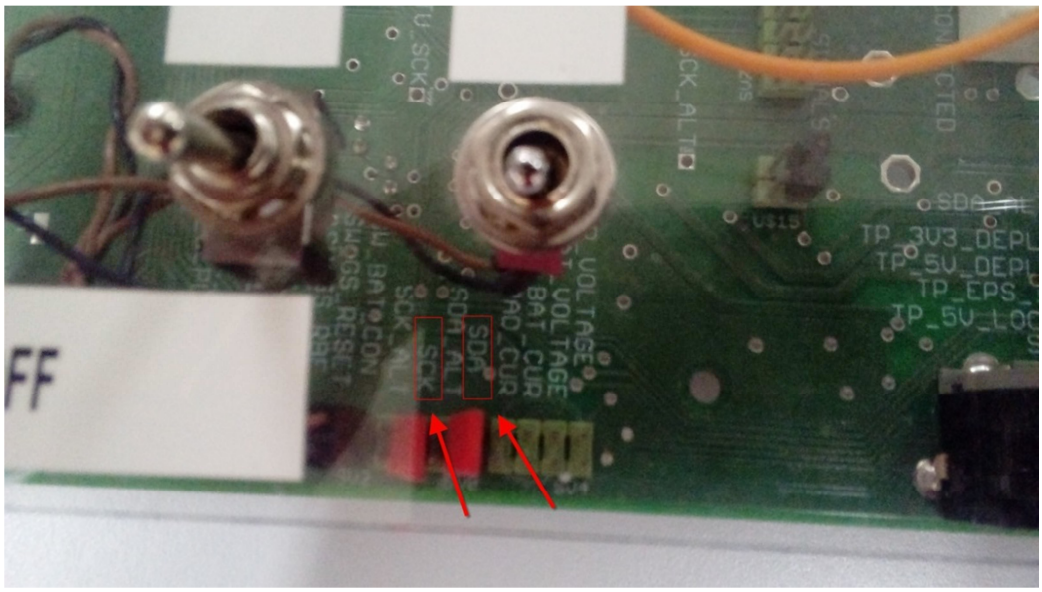
Foram realizados alguns testes preliminares para verificar as funcionalidades da interface entre o computador de bordo e as cargas úteis ISLP e SDATF. Nestes testes foram utilizados o iOBC-P do cubesat NanoBr2 e, alternativamente, o OBC-P usado no projeto ITASAT.

5.1. **Setup de teste da ISLP**

Um setup de teste foi realizado para validar a implementação do protocolo de comunicação I2C entre o iOBC-P integrado à estrutura do cubesat NanoBr2 e a sua respectiva carga útil, ISLP.

É necessária uma configuração prévia de *jumpers* para realização segura deste teste conforme ilustrado na Figura 5.1. Neste teste, o equipamento de suporte elétrico em solo (EGSE – *Electrical Ground Support Equipment*) da ISIS requer que resistores de *pull-up* sejam desconectados para evitar uma sobrecorrente nas linhas I2C do barramento.

Figura 5.1 – *Setup* de *jumpers* para teste I2C no EGSE – NanoBr2.



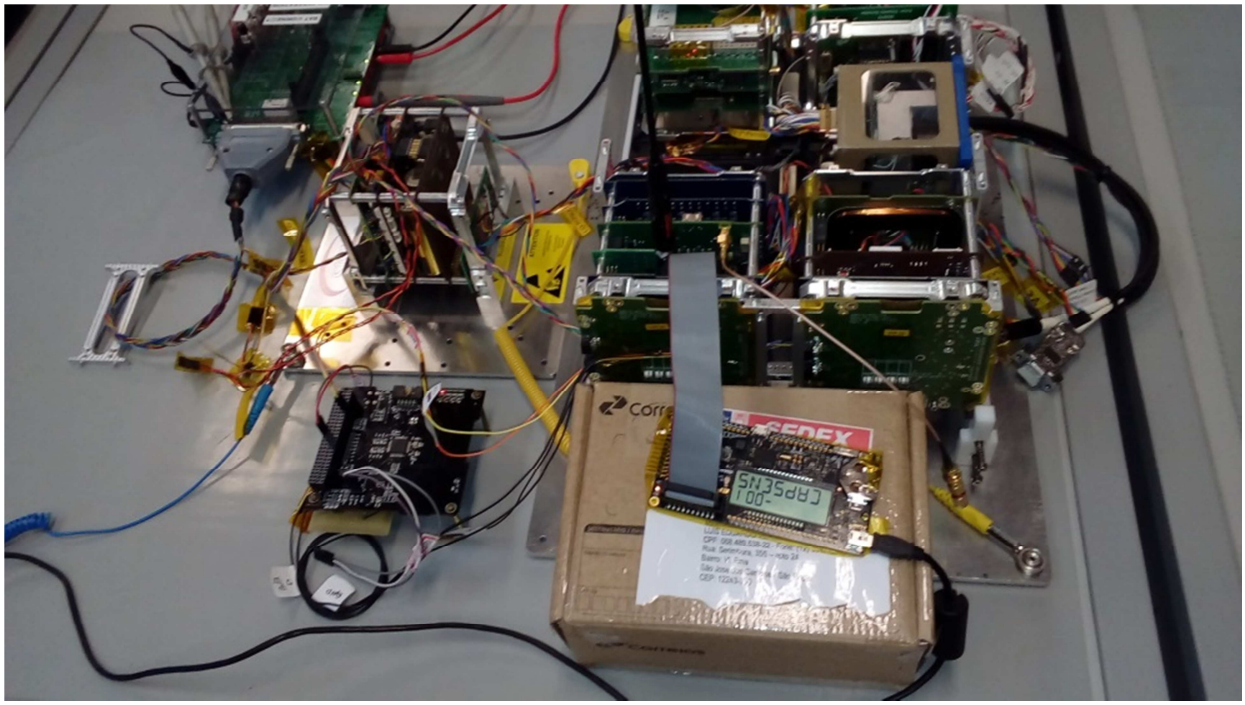
Fonte: Produção do autor.

Uma série de dificuldades surgiram na integração da ISLP com o iOBC-P. Devido a isto, alternativamente levou-se a carga útil para as instalações do projeto ITASAT do Instituto Tecnológico de Aeronáutica – ITA para consecução dos testes no barramento I2C, o qual é descrito a seguir.

5.2. **Teste da ISLP realizado no ITASAT**

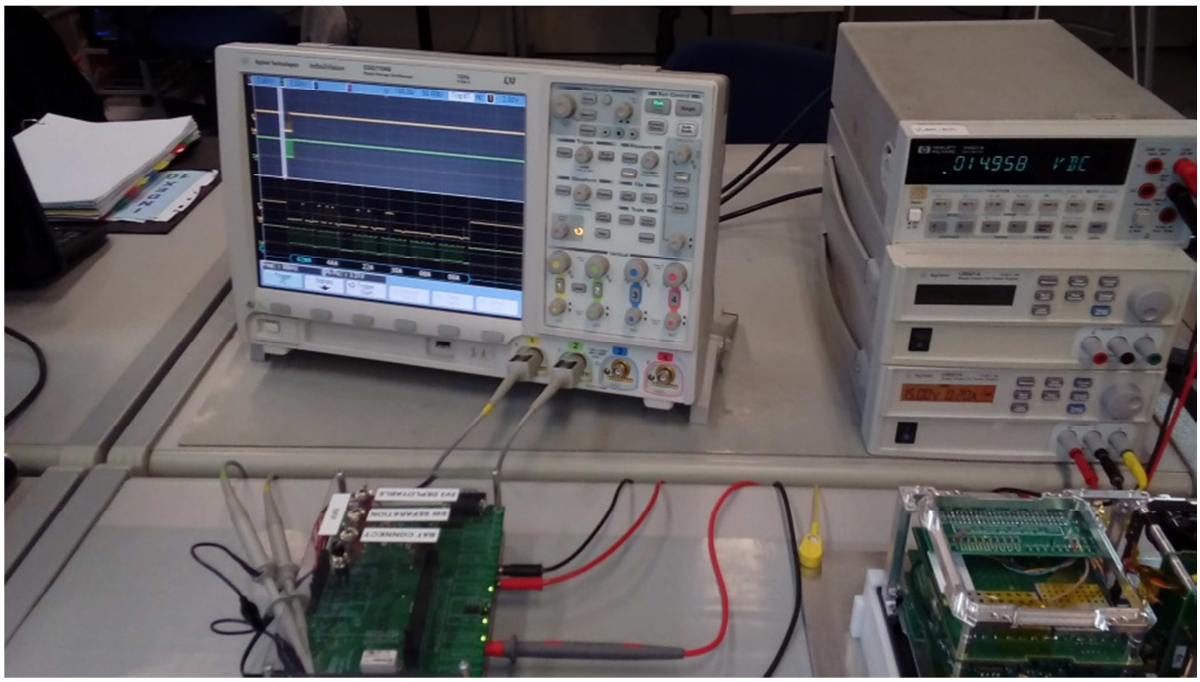
Foi realizado um teste com a ISLP utilizando o computador de bordo do ITASAT. Este OBC-P utiliza o ambiente *Simplicity Studio da Silicon Labs* que faz uso do processor da Silicon Labs. O setup para realização destes testes é mostrado nas Figuras 5.1 e 5.2.

Figura 5.1 – Teste da ISLP com a OBC-P do ITASAT.



Fonte: Produção do autor.

Figura 5.2 – Testes de comunicação entre a ISLP e o ITASAT

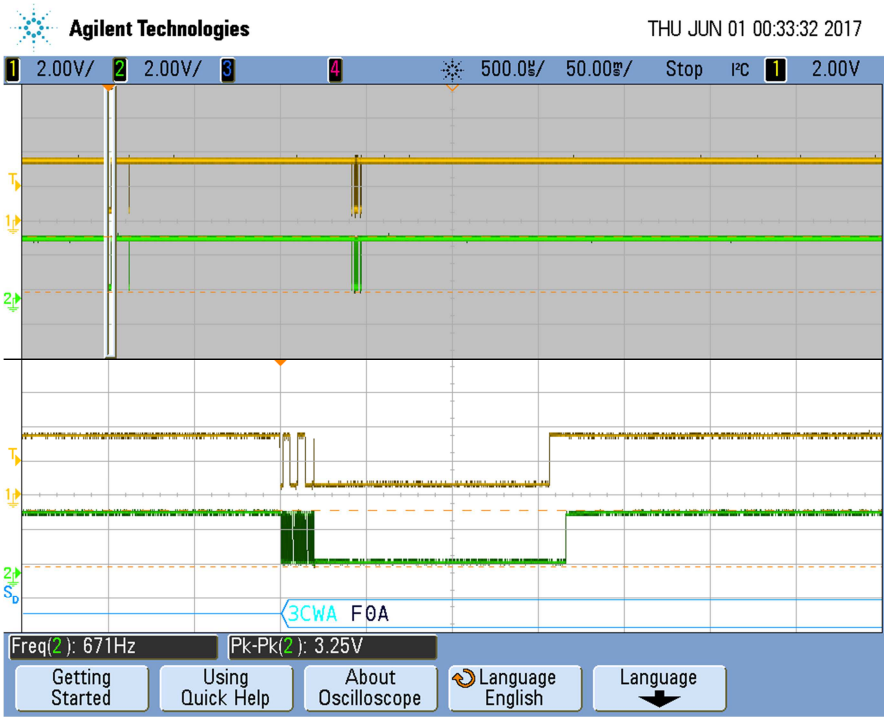


Fonte: Produção do autor.

Foi feita a integração de hardware entre o ITASAT e a ISLP. Todos os subsistemas do ITASAT funcionaram normalmente com a ISLP ligada à plataforma. Não houve variação no nível de sinal I2C (SDA e SCL) e tampouco a plataforma apresentou comportamento anômalo com a ISLP ligada no barramento I2C.

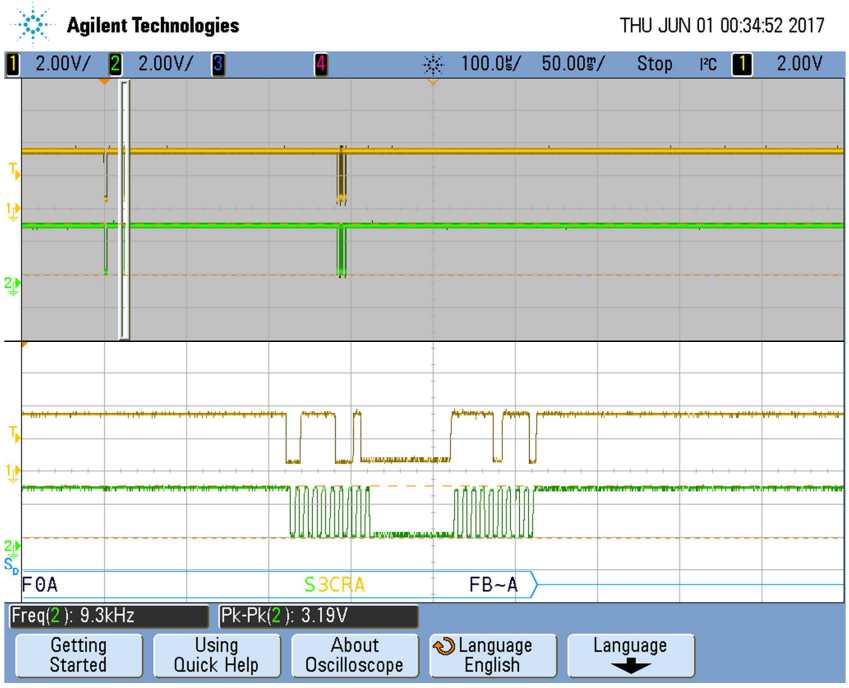
Para melhor caracterização e verificação de status da camada física do protocolo I2C, foram adquiridas as imagens do osciloscópio contendo as linhas do barramento I2C mostradas nas Figuras 5.3 e 5.4 para testes de escrita e leitura respectivamente.

Figura 5.3 – Teste de escrita de sinais via I2C no ISLP com OBC-P do ITASAT



Fonte: Produção do autor.

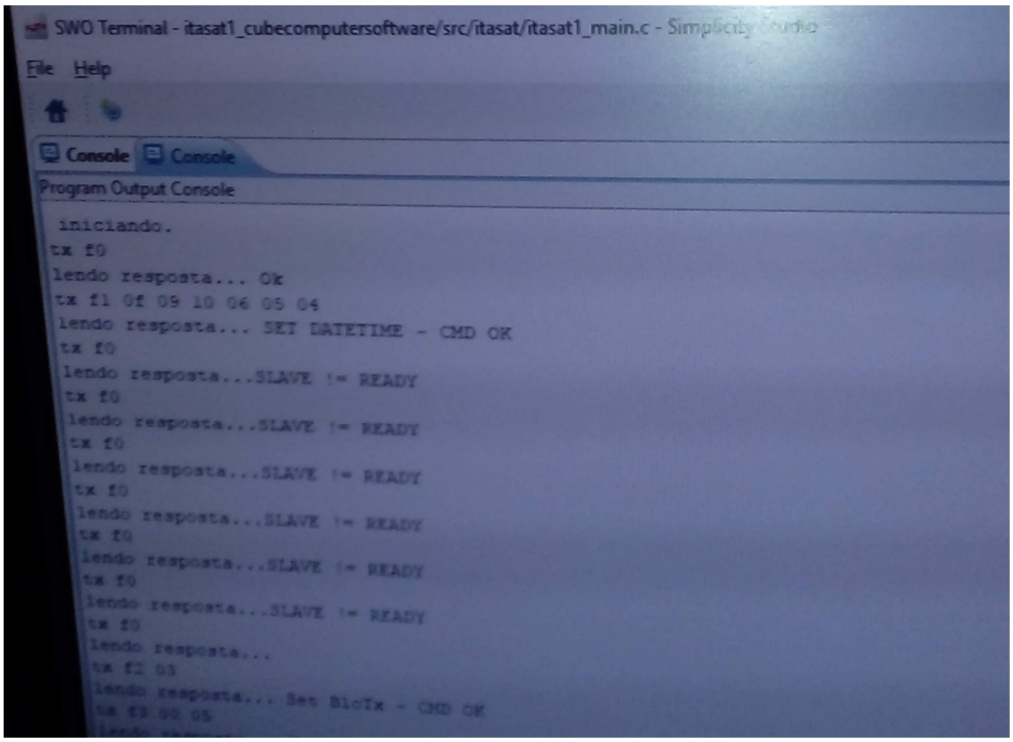
Figura 5.4 – Teste de leitura de sinais via I2C do ISLP para OBC-P do ITASAT



Fonte: Produção do autor.

A Figura 5.5 mostra as mensagens de comunicação, mediante um terminal serial, entre o OBC-P do ITASAT e uma carga útil ISLP.

Figura 5.5 – Teste de comunicação serial entre ISLP e o OBC-P do ITASAT

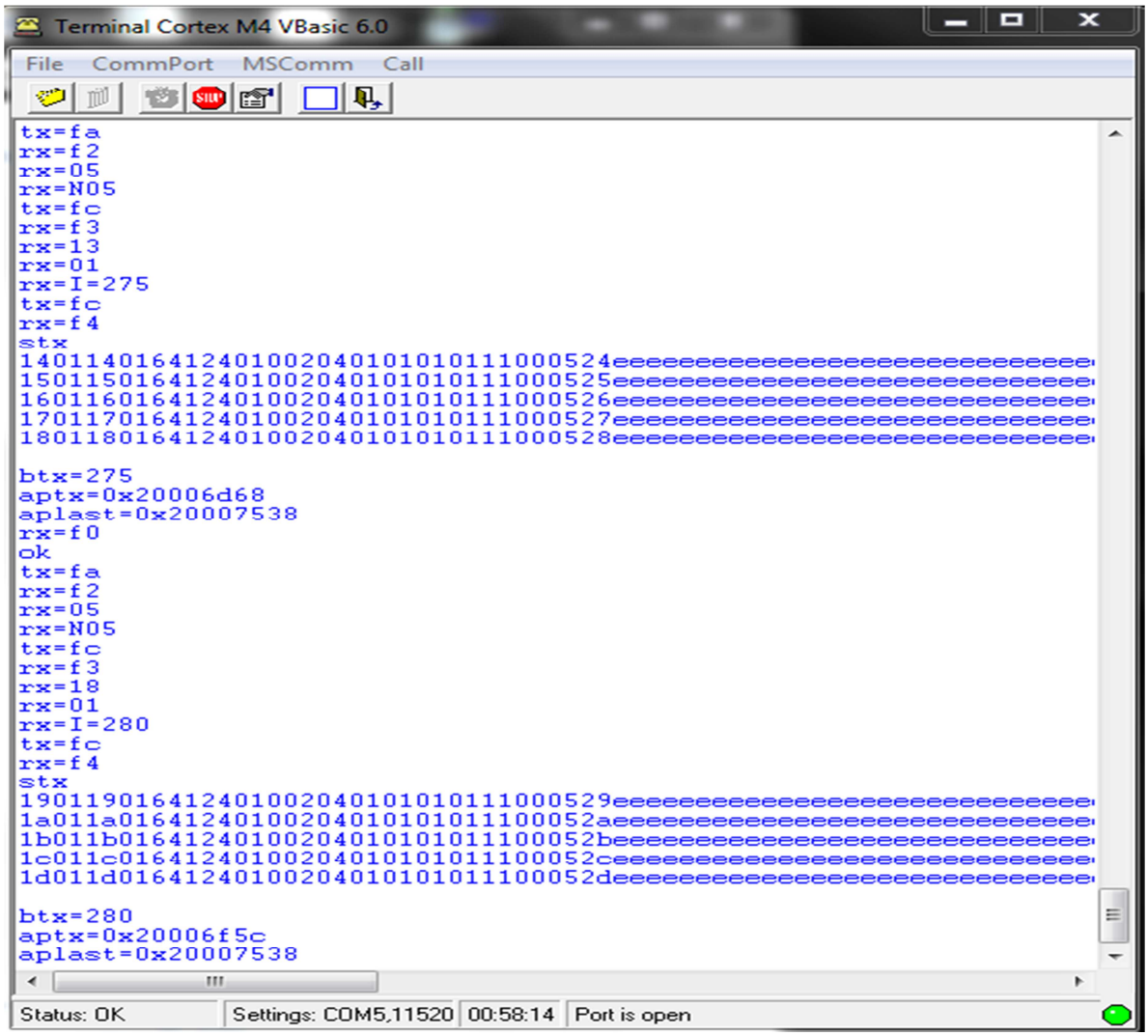


Fonte: Produção do autor.

5.3. **Teste da ISLP realizados com a placa STM32**

Foram realizados testes com a ISL com a placa STM32F407VET utilizando a biblioteca desenvolvida no ambiente Mbed. A Figura 5.6 mostra as respostas obtidas pela OBC-P (STM32). Este teste mostrou que a biblioteca consegue comunicar-se com a ISLP, enviar comandos e receber dados da mesma. A Figura 5.6 mostra os comandos enviados para a ISLP e os dados recebidos, utilizando o terminal serial para debug.

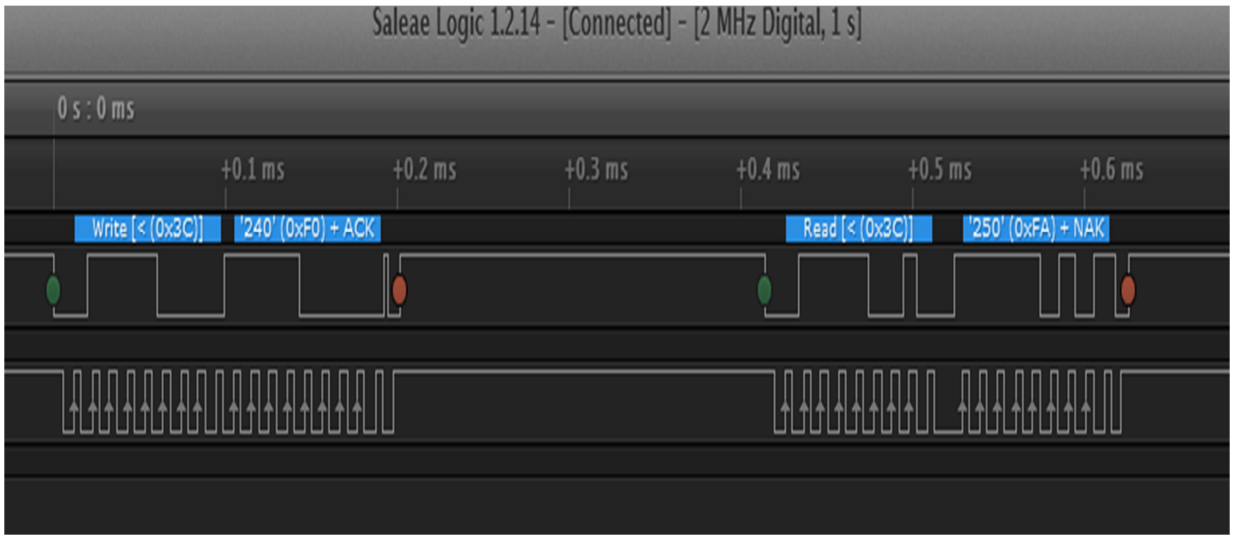
Figura 5.6 – Respostas recebidas da ISLP (*slave*) pelo OBC-P STM32 (*master)*



Fonte: Produção do autor.

A Figura 5.7 mostra as formas de onda da troca de dados entre a OBC-P e a ISLP, utilizando um analisador lógico.

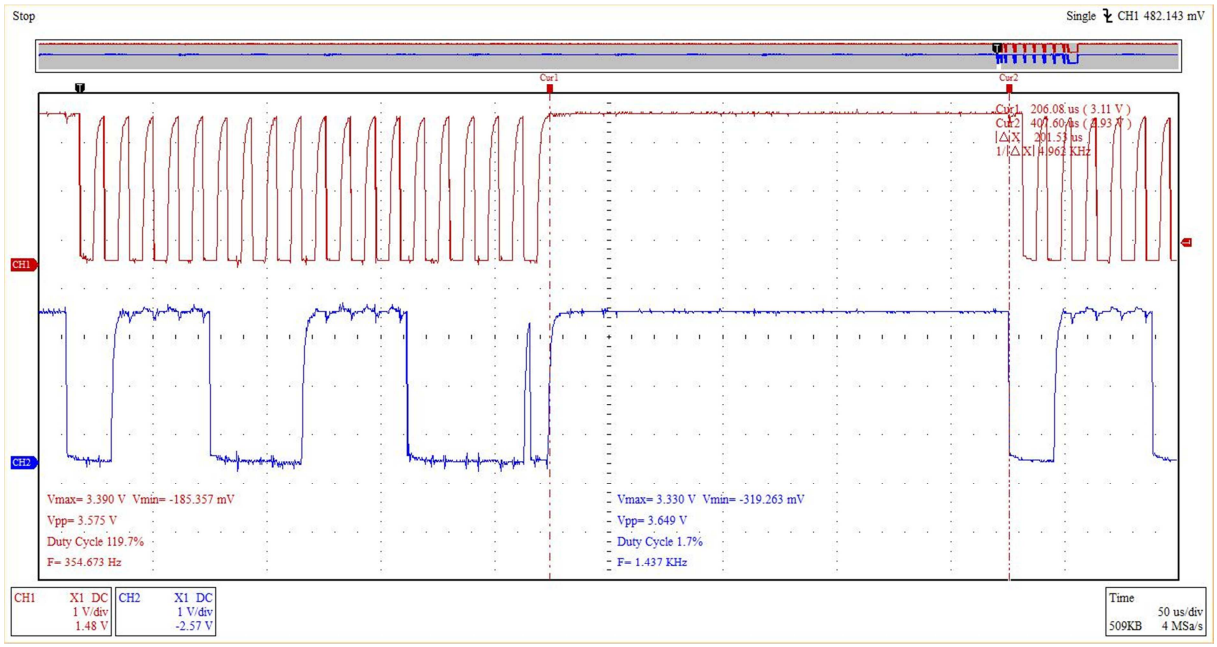
Figura 5.7 – Comunicação entre Master(STM32) e a ISLP – Analisador Lógico



Fonte: Produção do autor.

A troca de dados para verificar a integridade dos sinais SDA e SCL do barramento I2C é mostrada na Figura 5.9 utilizando um osciloscópio.

Figura 5.8 – Comunicação entre Master(STM32) e a ISLP.

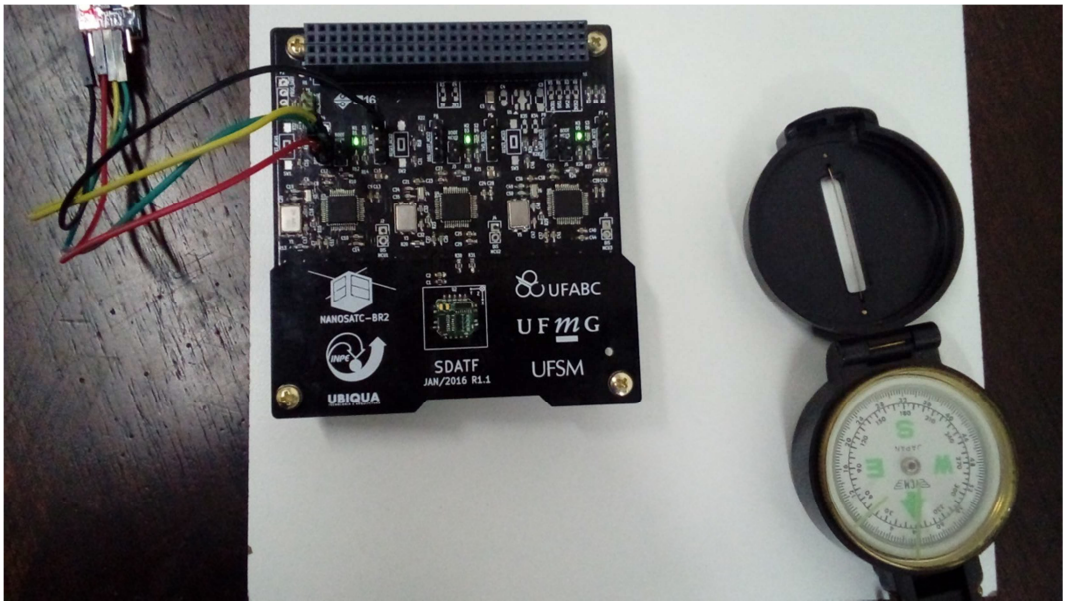


Fonte: Produção do autor.

5.4. **Testes com SDATF**

Esta série de testes é muito semelhante aos testes realizados com a carga útil SLP e são importantes para verificar detalhes de interoperabilidade entre diferentes tipos de carga útil. A Figura 5.9 mostra o setup de teste utilizado para conectar serialmente a carga útil SDATF ao OBC-P STM32.

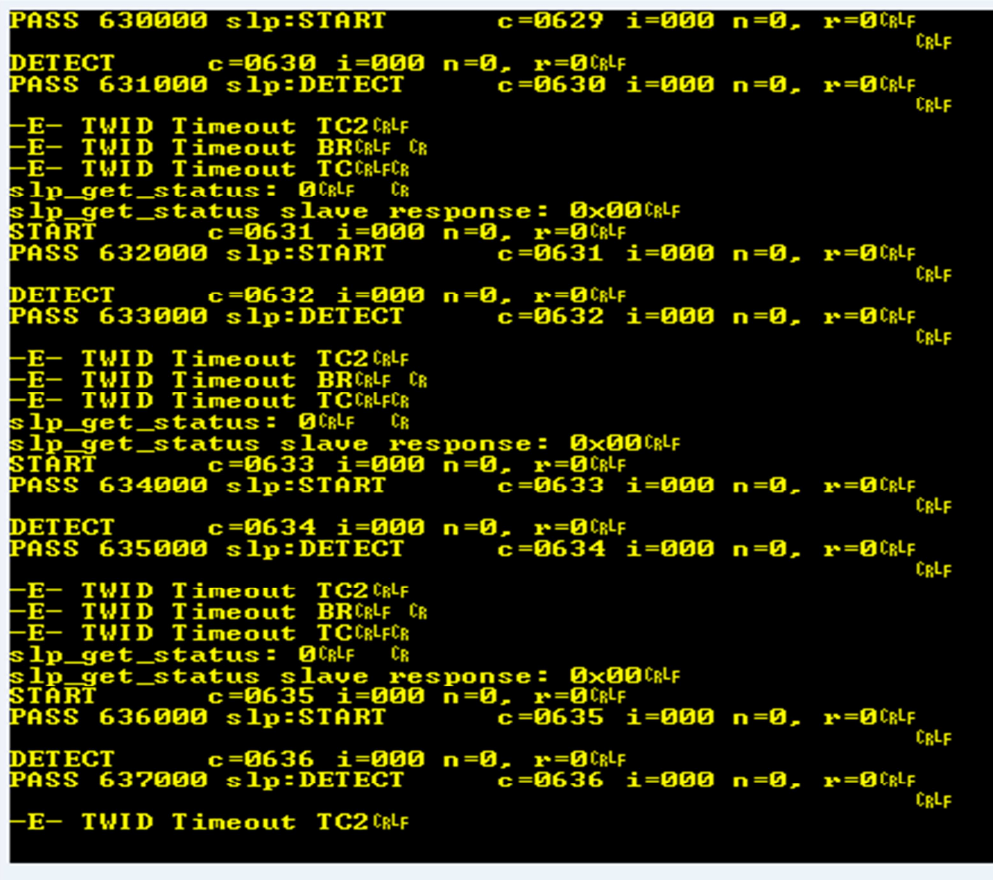
Figura 5.9 – Setup de teste de comunicação entre SDATF e o OBC-P STM32



Fonte: Produção do autor.

Basicamente foram realizados testes com a placa SDATF utilizando a placa STM32 como computador de bordo. O primeiro teste, mostrado na Figura 5.10, consiste em requisitar informações de status da SDATF via I2C e receber da mesma, a resposta via interface serial simples RS-232 somente para *debugging*.

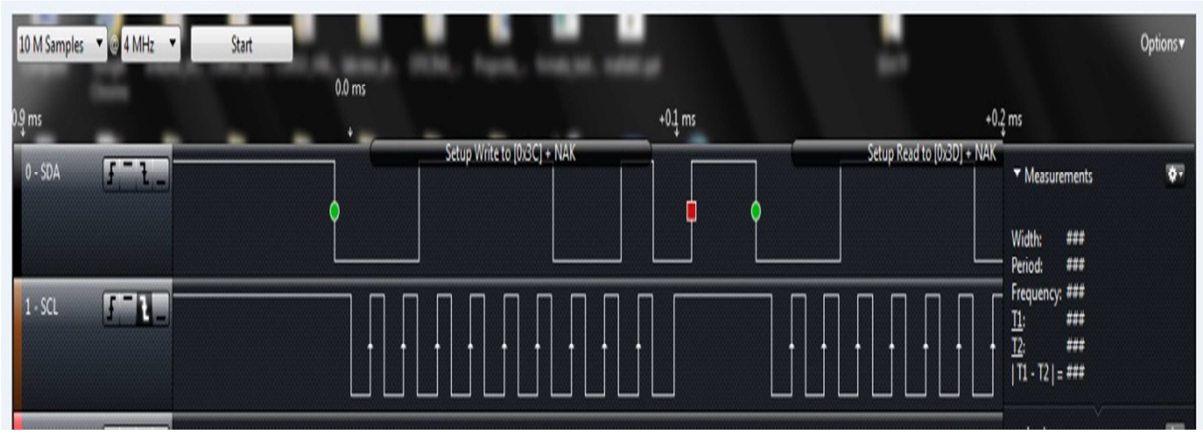
Figura 5.10 – Teste de comunicação serial entre SDATF e o OBC-P STM32



Fonte: Produção do autor.

O teste seguinte, ilustrado na Figura 5.11, verifica a integridade do nível físico do protocolo I2C mediante aquisição da forma de onda de sinais (**`Start Bit`**, **`Stop Bit`**, etc) entre o OBC-P e a SDATF utilizando um analisador lógico.

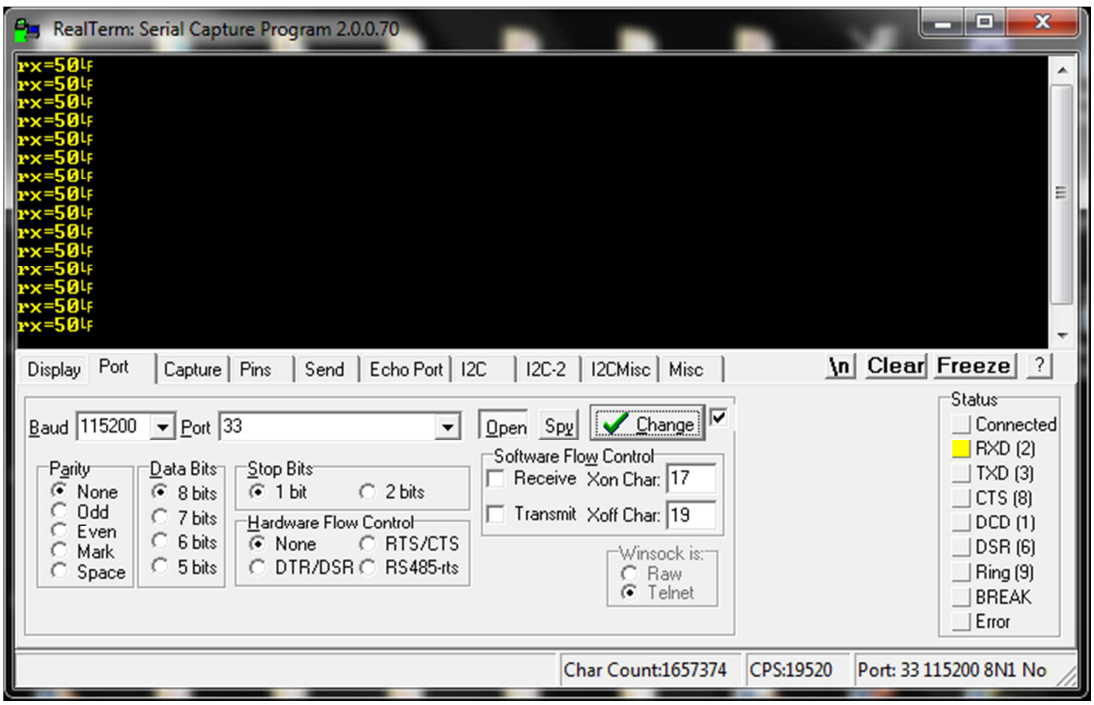
Figura 5.11 – Comunicação entre Master (STM32) e a SDATF



Fonte: Produção do autor.

O terceiro teste, mostrado na Figura 5.12, consiste em se verificar a comunicação I2C mediante transmissão de comandos de *tasks* da SDATF para sua execução. Neste caso, uma série de comandos hexadecimais 0x50, solicita o cálculo de atitude utilizando os magnetômetros internos do SDATF.

Figura 5.12 – Comunicação entre Master (STM32) e a SDATF

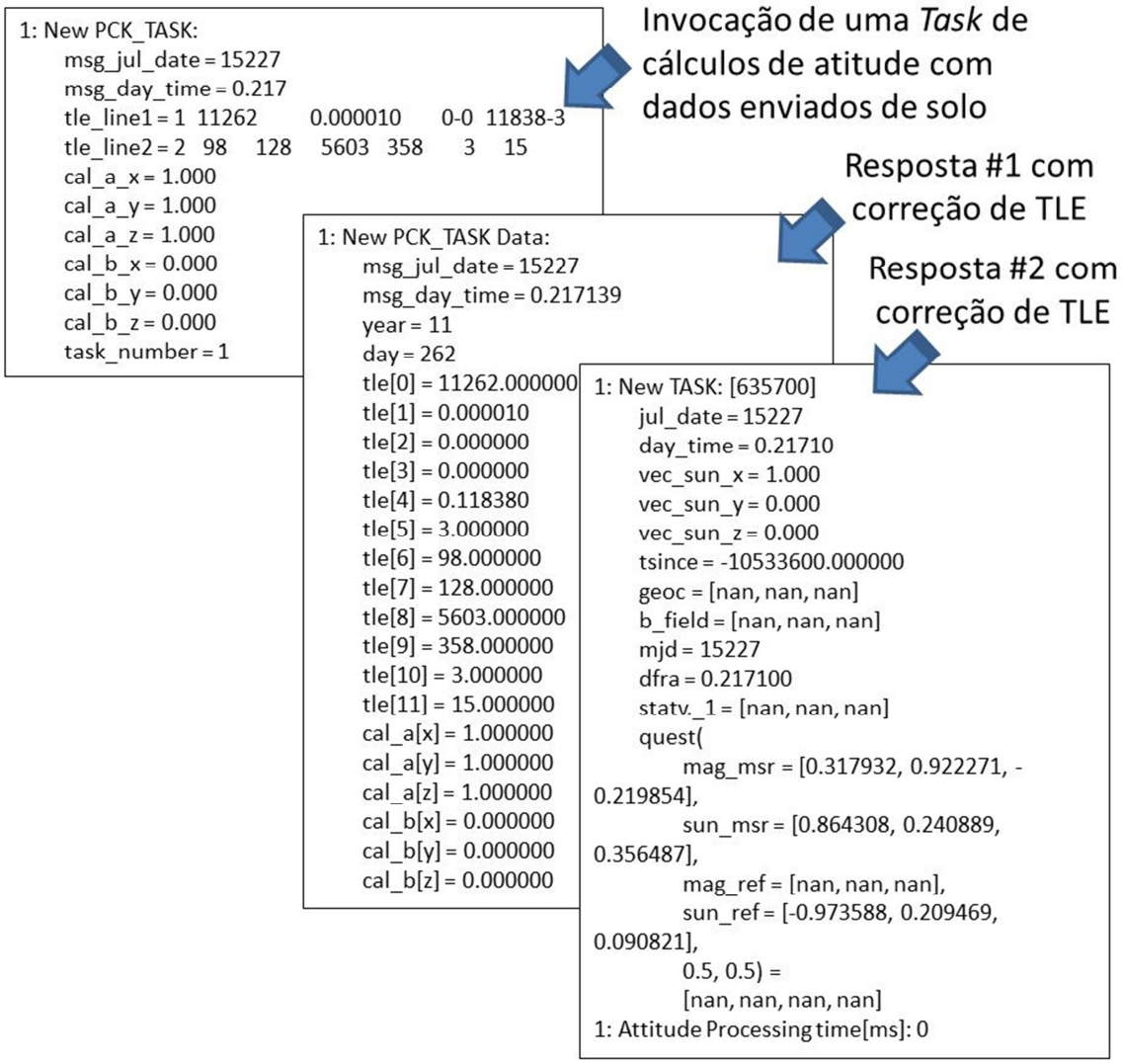


Fonte: Produção do autor.

O teste seguinte, ilustrado na Figura 5.13, verifica a comunicação I2C mediante transmissão de comandos de *tasks* da SDATF para sua execução.

A cada requisição feita pela OBC-P, a SDATF responde com um pacote de dados, contendo as informações de status, valores de engenharia e cálculos realizados, utilizando os magnetômetros da SDATF.

Figura 5.13 – *Handshake* de comunicação entre Master (STM32) e a SDATF



Fonte: Produção do autor.

O último teste realizado, requisitado pelo Dr. Hélio Kuga (DMC – INPE), foi feito para realizar a calibração dos magnetômetros da SDATF. A Figura 5.14 mostra a tela de resposta com a saída de um dos magnetômetros da SDATF.

Figura 5.14 – Resposta do SDATF com status de um magnetômetros.

```
count, time, x, y, z, f(x), f(y), f(z)
0, 676, 00000601, 000000B9, FFFFF9B1, 0.000012, 0.000001, -0.000012
1, 701, 00000587, 00000005, FFFFF983, 0.000011, 0.000000, -0.000012
2, 726, 000004FD, 00000029, FFFFF9AF, 0.000010, 0.000000, -0.000012
3, 751, 0000058B, 00000101, FFFFF95D, 0.000011, 0.000002, -0.000013
4, 776, 00000491, 000000E5, FFFFF9CF, 0.000009, 0.000002, -0.000012
5, 801, 000005B7, 0000013B, FFFFF94D, 0.000011, 0.000002, -0.000013
6, 826, 00000539, 00000047, FFFFFAA5, 0.000010, 0.000001, -0.000010
7, 851, 00000539, 00000087, FFFFFA2F, 0.000010, 0.000001, -0.000011
8, 876, 000005E9, 0000003B, FFFFFA51, 0.000011, 0.000000, -0.000011
9, 901, 00000531, 00000057, FFFFF907, 0.000010, 0.000001, -0.000013
```

Fonte: Produção do autor.

5.5. **Exemplo de teste utilizando Greentea para a ISLP**

Aqui está um exemplo de teste para a ISLP usando o ambiente de testes Greentea, disponível no ambiente mbed. O apendice G mostra esta implementação.

6 CONCLUSÃO

Nesta última seção apresentam-se os comentários finais, sugestões de trabalhos futuros, publicações realizadas e algumas limitações identificadas.

Há hoje um crescente interesse em nanossatélites por poderem embarcar várias missões originalmente idealizadas para satélites de maior porte. Isto é devido ao grau de compactação obtido por avanços em computação e nanotecnologia.

O INPE planeja em curto prazo a migração, quando possível, de alguns experimentos concebidos para microssatélites em nanossatélites, como previsto no Plano Diretor do INPE. Há uma demanda para isto em diversos grupos de pesquisa do CEA no INPE com instrumentos anteriormente instalados em solo, foguetes e balões.

O uso de nanossatélites admite medições *in-situ* bem como a verificação e teste de novos conceitos para eventual utilização em maiores satélites.

Este trabalho focou-se no desenvolvimento integrado de HW/SW para soluções de computação de bordo com cargas úteis científicas em nanossatélite, e utilizou principalmente uma sonda de Langmuir da CEA como estudo de caso.

6.1. **Comentários finais**

Como resultados principais deste trabalho, foi proposta uma arquitetura mínima qual seja: um OBC-P e periféricos para interface com os subsistemas de potência, AOCS, TTC e carga útil de forma a apoiar algumas cargas úteis do CEA.

Adicionalmente, foram obtidos resultados relativos aos seguintes pontos específicos:

- Diversas considerações foram realizadas visando decisões de projeto neste tema, tal como qual ambiente de desenvolvimento, qual processador é mais adequado, qual RTOS, dentre outros.

- Um desenvolvimento de um sistema mínimo de OBC-P, empregando a família de processadores STM32, foi utilizado para validar alguns requisitos como consumo, redes de comunicação e processamento de dados das cargas úteis.
- Um desenvolvimento incremental utilizando modelos foi adotado onde parte-se de um sistema emulado, passa-se por um modelo prototipado e *deploy* para processador *target* de vôo.
- Visando apoio a cargas úteis, um software de *housekeeping* e gerenciamento das cargas úteis foram concebidos em Mbed que provê uma API para RTOS.
- O módulo do OBC-P utilizado da MCUZONE possui características semelhantes aos do iOBC-P da ISIS, memória FLASH NOR e SDRAM, o que permitiu realizar o desenvolvimento e testes de interface das cargas uteis, utilizando um ambiente mais fiel possível ao de vôo.
- Finalmente, o desenvolvimento deste trabalho foi de grande utilidade para a campanha de integração do cubesat NANOBR2.

6.2. **Produção acadêmica realizada**

Foram elaborados três artigos envolvendo este trabalho, apresentados no WETE de 2017 que estão listados a seguir e nos apêndices C, D e E:

- Camargo, L., Dos Santos W.A., Proposta de um Computador de Bordo para o Nanossatélite Alpha Dentro do Programa CTEE, VII WETE, São José dos Campos, disponível online em www3.inpe.br/wete/index.php, 2017.

- Camargo, L., Dos Santos W.A., Computação de Bordo para Cargas Úteis Científicas em Nanossatélites, VII WETE, São José dos Campos, disponível online em www3.inpe.br/wete/index.php, 2017.

- Camargo, L. *et al.,* Estudo e Implementação de um Sistema de Comunicação I2C entre OBC-P e Carga Útil para o Programa

NanosatC-BR, VII WETE, São José dos Campos, disponível online em www3.inpe.br/wete/index.php, 2017.

6.3. **Limitações encontradas e lições aprendidas**

O modulo iOBC-P da ISIS apresentou alguns problemas de interface a nível de protocolo durante a fase de integração das cargas uteis como: a sonda de Langmuir (SLP) e SDATF. Adicionalmente, seu custo é bem maior comparado ao do microcontrolador AT91SAM9G20MDK9G20-EK, MCUZONE.

Outra grande limitação identificada foi a não existência de documentos de ICD (*Interface Control Documents*) para cargas úteis de Rádio Ocultação e o Detector de Elétrons.

Uma lição aprendida para o desenvolvimento de controladores de cargas uteis é: (a) sempre que possível, utilizar como teste o mesmo processador que será utilizado no computador de bordo ou o mais próximo possível e (b) utilizar o mesmo hardware e as mesmas bibliotecas de software.

Outra lição aprendida foi quanto as recomendações para que uma carga útil possa ser agregada a uma OBC-P:

a) Estar dentro dos requisitos de massa;
b) Estar dentro do envolope de potência;
c) Prover algum tipo de barramento de comunicação como I2C ou CAN;
d) Possuir requisitos que possam ser utilizados em operações de verificação e validação;
e) Possuir um ICD (Interface Contro Document);

6.4. **Sugestão de trabalhos e ações futuras**

Como trabalhos futuros relacionados ao tema aqui apresentado, sugerem-se:

- Inclusão de estudos de radiação e sua mitigação de seus efeitos;

- A adaptação deste projeto para o controle de cargas úteis em balões estratosféricos como eventual etapa anterior ao uso de satélites e
- Extensão dos conceitos para fomentar a continuação de missões mais avançadas do programa CTEE, exigirão arquiteturas mínimas de OBC-P as configurações:
    - Missão Beta: (a) Hardware: OBC-P com redundância (2 microcontroladores STM) e (b) Software de bordo: desenvolvimento em mbed (com RTOS)
    - c) Missão Gama: (a) Hardware: OBC-P com redundância (microcontrolador STM + FPGA) e (b) Software – ainda por se definir.

Para ações futuras que contribuirão para *outreach*, maior difusão de conhecimento e evolução do estado da arte, sugerem-se:

- Geração de um sistema mínimo de OBC-P poderá ser disponibilizada eventualmente para a comunidade para uso em ensino, treinamento ou desenvolvimento, tanto em software e hardware.
- Montagem um curso de computação de bordo para nanossatélites, utilizando este sistema como KIT e oferta à comunidade.

REFERÊNCIAS BIBLIOGRÁFICAS

AALBORG UNIVERSITY. **AAUSAT-II**. 2008. Disponível em: http://www.space.aau.dk/aausatii/eng/index.php. Acesso em: 8 abr. 2018.

ANICHE, M. **Test-driven development: teste e design no mundo real**. São Paulo: Casa do Código, 2012. 165p.

ANTUNES, S. **Diy comms and control for amateur space talking and listening to your satellite.** San Francisco, CA: Maker Media, 2015. 156p. ISBN:978-1-4493-1066-0.

BASS, L.; CLEMENTS, P.; KAZMAN, R. **Software architecture in practice.** 2.ed. Reading, MA: Addison Wesley, 2003. 560p. ISBN: 0-321-15495-9.

CLEMENTS, P.; KAZMAN, R.; KLEIN, M. **Evaluating software architectures: methods and case studies**. Reading, MA: Addison Wesley, 2002.

CLIMA ESPACIAL. **EMBRACE: magnetometro**. Disponível em: http://www2.inpe.br/climaespacial/portal/magnetometro-sobre. Acesso em 20 jul. 2017.

CODESCHOOL. **Code school**. Disponível em: https://learntocodewith.me/where-to-learn/code-school/. Acesso em 12 maio 2017.

CAPACITAÇÃO TÉCNICA EM ENGENHARIA ESPACIAL. **CTEE**. Disponível em: https://www.researchgate.net/project/CTEE-Capacitacao-Tecnica-em-Engenharia-Espacial. Acesso em: 28 jul. 2017.

CUBESAT. **Cubesat design specification, revision 13.updated 4-6-2015**. Disponível em: http://www.cubesat.org/images/developers/cds_rev13_final2.pdfhttp://www.cubesat.org/images/developers/cds_rev13_final2.pdf. Acesso em: 28 jul. 2017.

Divisao de Aeronomia Espacial. DAE. Disponível em: http://www.dae.inpe.br/iono. Acesso em: 29 jul. 2017.

DUARTE. R. O.; MARTINS. L. S.; Vale S. R. C. **Sistema de Determinação de Atitude com Tolerância a Falhas (SDATF)**. Belo Horizonte: [S.n.], 2016.

ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE. **Swisscube**. 2011. Disponível em: http://swisscube.epfl.ch. Acesso em: 8 abr. 2018.

EICKHOFF, J. **Onboard computers, onboard software and satellite operations:** an introduction. Berlin: Springer, 2012. 286p.

EMBARCADOS**. Implementando elementos de RTOS no Arduino**. Disponível em: https://www.embarcados.com.br/elementos-de-rtos-no-arduino/. Acesso em: 5 mar. 2017.

FOWLER, M. et al. **Patterns of enterprise application architecture.** Reading, MA: Addison Wesley, 2002.

FREITAS, E. P. **Metodologia orientada a aspectos para a especificação de sistemas tempo-real embarcados distribuídos**. 2007. 171p. Dissertação (Mestrado em Computação) – Universidade Federal do Rio Grande do Sul, Porto Alegre, 2007. Disponivel em: https://www.lume.ufrgs.br/bitstream/handle/10183/10268/000592321.pdf?sequence=1. Acesso em: 5 jul. 2017.

GRIFFITH, R. C. **Mobile cubesat command and control (mc3)**. 2011. 77p. Dissertação (Mestrado) – Naval Postgraduate School, Monterey, California, 2011.

HELVAJIN, H.; JANSON, S.W . **Small satellites**: past, present, and future. Washington: AIAA, 2009. ISBN-13: 978-1884989223

HMELO-SILVER, C. E. Problem-based learning: what and how do students learn? **Educational Psychology Review**, v. 16, n. 3, p. 235-266, 2004.

HORNA, A. P. **Experimento sonda de Langmuir para CUBESAT NCBR2**. São José dos Campos: Instituto Nacional de Pesquisas Espaciais, 2016.

HUDAK, Z. **STM32F103C8T6 board, alias Blue Pill**. 2016. Disponível em: https://os.mbed.com/users/hudakz/code/STM32F103C8T6_Hello. Acesso em: 20 dez. 2017.

INNOVATIVE SOLUTIONS IN SPACE. **ISIS OBC-P Datasheet.DS 1.1**. Holanda, 2014. 37p.

INSTITUTO NACIONAL DE PESQUISAS ESPACIAIS. **Plano diretor do INPE 2016-2019.** São José dos Campos, 2015.

INSTITUTO NACIONAL DE PESQUISAS ESPACIAIS. **Coordenação geral de ciências espaciais e atmosféricas**. 2012. Disponível em: http://www.inpe.br/acessoainformacao/ucea. Acesso em: 12 fev. 2017.

ISTANBUL TECHNICAL UNIVERSITY. **ITUPSAT-1**. 2009. Disponível em: http://www.turkkonferans.org/ITUPSAT.htm. Acesso em: 8 abr. 2018.

LEY, W.; WITTMANN, K.; HALLMANN, W.; **Handbook of space technology**. New York: Wiley, 2009.

LUMBWE, L. **Development of an onboard computer (OBC-P) for a cubesat**. Cape Town: Cape Peninsula University of Technology, 2013.

MBED. **The ARM mbed IoT device platform**. Disponível em: www.mbed.org. Acesso em: 28 jul. 2017.

MELO, T. **Microcontrolandos**. 2012. Disponível em: http://microcontrolandos.blogspot.com.br. Acesso em: 10 maio 2017.

MOORE, R. **How to pick an RTOS, micro digital.** Disponível em: http://www.embedded-computing.com/embedded-computing-design/how-to-pick-an-rtos. Acesso em: 20 jan. 2018

NASA CASI. **Small spacecraft technology state of the art, nasa's small spacecraft technology program**. 2014. Disponível em: http://ntrs.nasa.gov. Acesso em: 18 fev. 2018.

ODRIOZOLA, S. S. **Estudo de irregularidades do plasma ionosférico na região entre as camadas E e F no setor brasileiro a partir de medidas de foguete**. 2017. 218p. Tese (Doutorado em Geofísica Espacial / Ciências do Ambiente Solar-Terrestre) – Instituto Nacional de Pesquisas Espaciais, São José dos Campos, 2016. Disponível em: http://www3.inpe.br/pos_graduacao/cursos/geo/arquivos/teses/tese_siomel_odriozola_2016.pdf. Acesso em: 05 fev. 2018.

PALUSZEK, M.; CASTRO, E.; HYLAND, D. **The Cubesat book**. Plainsboro, NJ: Princeton Satellite Systems, 2010. 33p. ISBN 978-0-9654701-0-0.

QUAKEFINDER. **About Quakesat**. 2018. Disponível em: https://www.quakefinder.com/science/about-quakesat. Acesso em: 10 abr. 2018.

STMICROELECTRONICS. **STM32 32-bit ARM cortex MCUs.** Disponível em: http://www.st.com/en/microcontrollers/stm32-32-bit-arm-cortex-mcus.html. Acesso em: 28 jul. 2017.

TIKAMI, A. **Uma metodologia para re-engenharia de sistemas espaciais aplicada a um picossatélite**. Dissertação (Mestrado em Engenharia Espacial)

- Instituto Nacional de Pesquisas Espaciais (INPE), São José dos Campos, 2016.

TOKYO INSTITUTE OF TECHNOLOGY. **CUTE-1.7**. 2006. Disponível em: http://lss.mes.titech.ac.jp/ssp/cute1.7/projectoverview_e.html. Acesso em: 8 abr. 2018.

WERTZ, J.R.; EVERETT, D. F.; PUSCHELL, J.J. **Space mission engineering**: the new SMAD. Hawthorne, CA: Microcosm, 2011.

APÊNDICE A – BIBLIOTECA DE COMUNICAÇÃO COM SONDA SLP - SLP.H

```
#ifndef MBED_SLP_H
#define MBED_SLP_H

#include "mbed.h"

extern  int addr8bit;
extern char resposta[1];

class Slp{

public:
        Slp(PinName sda, PinName scl);
        void configura(int endereco, int num_blocos, int
indice_bloco);
        int disponivel();
        char escreve_hora();
        char configura_bloco(int numero_blocos);
        char configura_indice_bloco(int indice);
        char * inicia_tx();
        char armazena_memoria();

        I2C* i2c_;
};
#endif
```

APÊNDICE B – BIBLIOTECA DE COMUNICAÇÃO COM SONDA SLP – SLP.CPP

```
#include "Slp.h"
#include "mbed.h"

//#define slave_add ((uint8_t) 0x3C)

//Serial pc(PA_9, PA_10); //tx, rx
//I2C i2c(PB_7,PB_8);     //sda, scl

int addr8bit;
char resposta[1];

Slp::Slp(PinName sda, PinName scl){

    i2c_ = new I2C(sda, scl);

    // frequencia i2c = 100 KHz
    i2c_-> frequency(100000);

}

void Slp::configura(int endereco,int num_blocos, int
indice_bloco)
{
     int addr8bit = endereco << 1; // 8bit I2C address

}

int Slp::disponivel()
{

    char cmd[1];
    char resposta[1];

    cmd[0] = 0xf0;
    i2c_->write(addr8bit,cmd,1);
    i2c_->read(addr8bit,resposta,1);
    return(resposta[0]);

}

char Slp::escreve_hora()
{
```

```
    char cmd[7];
    char resposta[1];

    cmd[0] = 0xf1;
    cmd[1] = 0x01;
    cmd[2] = 0x08;
    cmd[3] = 0x10;
    cmd[4] = 0x00;
    cmd[5] = 0x01;
    cmd[6] = 0x01;

    i2c_->write(addr8bit,cmd,7);
    i2c_->read(addr8bit,resposta,1);

    wait(1);
    return(resposta[0]);

}



char Slp::configura_bloco(int numero_blocos)
{
    char cmd[2];


    cmd[0] = 0xf2;
    cmd[1] = numero_blocos;
    i2c_->write(addr8bit,cmd,1);
    i2c_->read(addr8bit,resposta,1);

    wait(1);
    return(resposta[0]);



}


char Slp::configura_indice_bloco(int indice)
{
    char cmd[3];

    cmd[0] = 0xf3;
    cmd[1] = indice;
    cmd[2] = 0x00;
```

```
    i2c_->write(addr8bit,cmd,3);
    i2c_->read(addr8bit,resposta,1);

    wait(1);
    return(resposta[0]);
}


char * Slp::inicia_tx()
{
    char cmd[1];
    cmd[0] = 0xf4;
    char dados[100];

    i2c_->write(addr8bit,cmd,1);
    i2c_->read(addr8bit,dados,1);

    wait(1);
    return(dados);
}


char Slp::armazena_memoria()
{
}
```

APÊNDICE C – WETE 2017

Proposta de um Computador de Bordo para o Nanossatélite Alpha Dentro do Programa CTEE

CAMARGO, L. A. P.[1], DOS SANTOS, W. A[2]

[1]DAE-CEA-INPE [2]DEA-ETE-INPE

Instituto Nacional de Pesquisas Espaciais, São José dos Campos, SP, Brasil

lazaro.camargo@inpe.br walter.abrahao@inpe.br

Resumo. *Vantagens na adoção de estratégias PBL (Problem-Based Learning) para fundamentação de conceitos são amplamente reconhecidas em diversas áreas do conhecimento. O corpo discente e docente da Pós-Graduação em Engenharia e Tecnologia Espaciais (ETE) do INPE estruturou um PBL denominado CTEE (Capacitação Tecnológica em Engenharia Espacial) com o projeto incremental de série de três nanossatélites. Este trabalho discorre sobre a proposta de computador de bordo (OBC-P) para o primeiro nanossatélite dessa série denominado CTEE-Alpha. O OBC-P visa atender os requisitos mínimos desta missão utilizando métodos e recursos de Engenharia de Sistemas Espaciais e Engenharia de Requisitos.*

Palavras-chave: Nanossatélites. Cubesats. Computador de bordo. Engenharia de Sistemas. Engenharia de Requisitos.

APÊNDICE D – WETE 2017

Computação de Bordo para Cargas Úteis Científicas Em Nanossatélites

CAMARGO, L. [1], DOS SANTOS, W. A.[2]

[1]Instituto Nacional de Pesquisas Espaciais, São José dos Campos, SP, Brasil

[2]Instituto Nacional de Pesquisas Espaciais, São José dos Campos, SP, Brasil

lazaro.camargo@inpe.br

Resumo. *Este trabalho apresenta o desenvolvimento de bibliotecas de software para que uma OBC-P possa gerenciar cargas uteis cientificas em Nanossatélites, utilizando o ambiente de programação mbed, para microcontroladores ARM.*

Palavras-chave: Cargas úteis. Mbed.

APÊNDICE E – WETE 2017

Estudo e Implementação de um Sistema de Comunicação I2C entre OBC-P e Carga Útil para o Programa NanosatC-BR

CAMARGO, L.[1,2], BOSCOLO, G.[3], ESSADO, M.[4], GOMES, C. [1], LIMA, J.[1], MATTIELLO-FRANCISCO, F. [1], DURÃO, O.[1]

[1]Instituto Nacional de Pesquisas Espaciais, São José dos Campos, SP, Brasil
[2]Aluno de Mestrado do curso de Engenharia e Gerenciamento de Sistemas Espaciais - CSE.
[3]Universidade Estadual de Campinas, Campinas, SP, Brasil
[4]EMSISTI Sistemas Espaciais & Tecnologia, São José dos Campos, SP, Brasil

lazaro.camargo@inpe.br

Resumo. *O artigo apresenta o estudo e implementação de um sistema de comunicação I2C entre a OBC-P (on-board computer) e carga útil ISLP - Sonda de Langmuir para o Programa NanosatC-BR – Desenvolvimento de Cubesats, realizado durante mini estágio do Curso de Inverno ETE/INPE 2017. Com o objetivo de fornecer ao aluno um aprofundamento na área de Sistemas Espaciais o trabalho foi executado conforme as etapas: pesquisa, análise, execução e documentação das práticas realizadas. O resultado do trabalho é um conjunto de testes e simulações no que tange ao princípio de funcionamento do barramento I2C, transmissão dos dados e limitações da arquitetura de comunicação. Assim, este trabalho traz algumas das lições aprendidas no período de estágio, as quais poderão ser úteis para estudos futuros e outras aplicações.*

Palavras-chave: I2C; OBC-P; Cubesat; Nanossatélite, NanosatC-Br2.

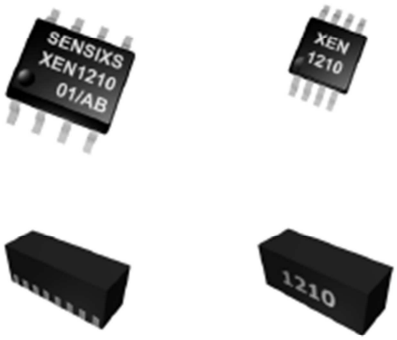
Quando a seção terminar em página ímpar, o verso fica em branco.

APÊNDICE F – SENSOR MAGNÉTICO XEN1210

XEN1210
Magnetic Sensor

Features

- Single axis magnetic measurement
- One chip solution
- 15nT resolution
- Unmatched low offset (Sub-µT) and gain precision
- Wide magnetic field range (±63mT)
- No magnetic hysteresis
- Measurement rate up to 5kSPS
- Low voltage operation (2.5V to 3.3V)
- Single supply
- Serial SPI communication
- -40°C to 125°C Temperature Range
- Surface mount package (1.4x1.6x4mm)
- Package 3D solderable
- Available in tape and reel



General Description

XEN1210 is a magnetic field sensor based on the Hall effect. It uses Sensixs's Q-Hall technology, which with a resolution of 15nT and a magnetic field range of 63mT combines the best of Hall and AMR technology in one.
In position and speed applications where conventional Hall and AMR technology suffer from a large natural offset (drift), its unmatched low offset and offset stability simplifies the design. In contrast to AMR compass sensors no set/reset sequence is required.
The XEN1210 can be used for a wide range of magnetic sensing applications.

A maximum read-out frequency of 5kSPS can be obtained.
The control is fully digital and readout and programming is performed via the SPI bus.

For battery applications, power can be saved by taking single shot measurements. In power-off mode the current draw is limited to below 100nA.

Except for SOIC8 and MSOP8, the XEN1210 is available in SFN8 packages, which can be freely oriented on a PCB in 3 dimensions. By combining a number of XEN1210 sensors in (different) orientations rotary or linear position sensor systems can be designed.

Applications

- Compass Sensors
- Linear Magnetic Field Sensor
- Linear Position and Speed Measurement
- Rotary Position and Speed Measurement
- Current Sensing

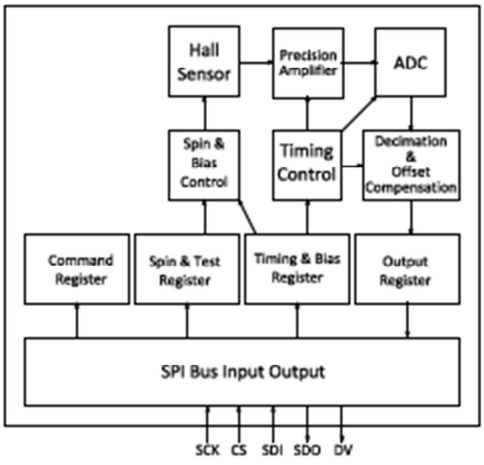


Figure 1: Functional block diagram.

APÊNDICE G – TESTE PARA A ISLP UTILIZANDO O GREENTEA NO AMBIENTE MBED

```
// Teste mbed - I2C - ISLP
 */
// check if I2C is supported on this device
// verifica se o dispositivo suporta I2C
#if !DEVICE_I2C
  #error [NOT_SUPPORTED] I2C not supported on this
platform, add 'DEVICE_I2C' definition to your platform.
#endif

#include "mbed.h"
#include "greentea-client/test_env.h"
#include "unity.h"
#include "utest.h"
#include "islb.h"
#include "islp_test_config.h"

using namespace utest::v1;


// um teste para verificar se a ISLP responde a um comando.
Em caso de falha retorna 0
template<PinName sda, PinName scl, int resposta_esperada>
void test_islp()
{
     islp carga_util(endereco, sda, scl, num_blocos,
indice_blocos)
    int saida = islp.pronta();
     DEBUG_PRINTF("\r\n****\r\nTESTE ISLP : valor recebido
= `%d`\r\n****\r\n",saida);
    TEST_ASSERT_MESSAGE(0 != carga_util.configura(),"Falha
comunicacao com a ISLP");
}

// Teste a inicializacao do objeto I2C
template<PinName sda, PinName scl>
void test_object()
{
    I2C i2c(sda,scl);
    TEST_ASSERT_MESSAGE(true,"problemas com o I2C");
}

utest::v1::status_t test_setup(const size_t
number_of_cases)
```

```
{
    // configura timeout do  Greentea em segundos
    GREENTEA_SETUP(20, "default_auto");
    return verbose_test_setup_handler(number_of_cases);
}

// gerencia falhas nos testes
utest::v1::status_t greentea_failure_handler(const Case
*const source, const failure_t reason)
{
    greentea_case_failure_abort_handler(source, reason);
    return STATUS_CONTINUE;
}

// Casos de teste
Case cases[] = {
    Case("I2C -  Objeto
I2C",test_object<MBED_CONF_APP_I2C_SDA,MBED_CONF_APP_I2C_SC
L>,greentea_failure_handler),
    Case("I2C -  Leitura da
ISLP",test_islp<MBED_CONF_APP_I2C_SDA,MBED_CONF_APP_I2C_SCL
,25,20>,greentea_failure_handler),

};

Specification specification(test_setup, cases);

// Ponto de entrada dos testes
int main()
{
    return !Harness::run(specification);
}
```

PUBLICAÇÕES TÉCNICO-CIENTÍFICAS EDITADAS PELO INPE

Teses e Dissertações (TDI)

Teses e Dissertações apresentadas nos Cursos de Pós-Graduação do INPE.

Manuais Técnicos (MAN)

São publicações de caráter técnico que incluem normas, procedimentos, instruções e orientações.

Notas Técnico-Científicas (NTC)

Incluem resultados preliminares de pesquisa, descrição de equipamentos, descrição e ou documentação de programa de computador, descrição de sistemas e experimentos, apresentação de testes, dados, atlas, e documentação de projetos de engenharia.

Relatórios de Pesquisa (RPQ)

Reportam resultados ou progressos de pesquisas tanto de natureza técnica quanto científica, cujo nível seja compatível com o de uma publicação em periódico nacional ou internacional.

Propostas e Relatórios de Projetos (PRP)

São propostas de projetos técnico-científicos e relatórios de acompanhamento de projetos, atividades e convênios.

Publicações Didáticas (PUD)

Incluem apostilas, notas de aula e manuais didáticos.

Publicações Seriadas

São os seriados técnico-científicos: boletins, periódicos, anuários e anais de eventos (simpósios e congressos). Constam destas publicações o Internacional Standard Serial Number (ISSN), que é um código único e definitivo para identificação de títulos de seriados.

Programas de Computador (PDC)

São as sequências de instruções ou códigos, expressos em uma linguagem de programação compilada ou interpretada, a ser executada por um computador para alcançar um determinado objetivo. São aceitos tanto programas fonte quanto executáveis.

Pré-publicações (PRE)

Todos os artigos publicados em periódicos, anais e como capítulos de livros.