



UNIVERSIDADE FEDERAL DE SANTA CATARINA
CAMPUS FLORIANÓPOLIS
CURSO DE GRADUAÇÃO EM CIÊNCIAS DA COMPUTAÇÃO

Lucas Zacchi de Medeiros

**Análise e implementação de fluxo de automação de testes
dos sistemas embarcados de Satélites Artificiais**

Florianópolis
2022

Lucas Zacchi de Medeiros

**Análise e implementação de fluxo de automação de testes
dos sistemas embarcados de Satélites Artificiais**

Monografia submetida ao Curso de Graduação em
Ciências da Computação da Universidade Federal
de Santa Catarina para a obtenção do título de ba-
charel em Ciências da Computação.

Orientador: Prof. Dr. Eduardo Augusto Bezerra

Coorientador: Prof. Dr. Rafael de Santiago

Florianópolis

2022

RESUMO

Satélites artificiais são projetos que demandam níveis elevados de confiabilidade de seus módulos. Apesar de alguns projetos permitirem atualizações posteriores a firmware e programas, são raros os casos onde é possível revisar e corrigir hardware. Por esse motivo a etapa de testes e AIV (*Assembly, Integration, and Verification*) é crucial para a garantia de que o projeto não corra mais riscos do que os que são inerentes à área. Esses fatores são amplificados quando tratamos de *CubeSats*, que possuem escopo e orçamentos menores quando comparados a grandes projetos governamentais e/ou comerciais. Este trabalho propõe a implementação de um sistema de *workflows* hospedados na plataforma de controle de versionamento *GitHub* que, aliados à funcionalidade *GitHub Actions*, permitirá a execução automatizada de testes no contexto dos planos de *Assembly, Integration, and Verification*(AIV) da missão *FloripaSat-2* e, posteriormente, análise e interpretação dos dados coletados de modo a obter resultados qualitativos e quantitativos da execução.

Palavras-chave: CubeSat. Nanossatélite. FloripaSat-2. AIV.

ABSTRACT

Artificial Satellites are projects that demand highly reliable modules. Despite some missions allowing post-deployment updates to firmware and software, cases where it is possible to perform hardware maintenance are rare. Because of this fact, the *Assembly, Integration and Verification* (AIV) process is crucial to guarantee that the project doesn't face more risks than those which are inherent of a space mission. These factors are intensified when dealing with *CubeSats*, that statistically have lower budgets and scope when compared to large scale governmental and/or commercial space missions. This article proposes the implementation of a test automation workflow system hosted at GitHub that will make use of the GitHub Actions tool to allow automated execution of tests during the AIV step of the FloripaSat-2 mission, and lastly, will allow collection and analysis of data in order to draw relevant conclusions regarding the execution.

Keywords: CubeSat. Nanosatellite. FloripaSat-2. AIV.

LISTA DE FIGURAS

Figura 1 – Renderização do FloripaSat-2.	9
Figura 2 – Lançamentos anuais de Nanossatélites (previsão)	15
Figura 3 – Diagrama de execução de um workflow de testes automatizados . .	20
Figura 4 – Captura de tela do resultado de uma execução de testes automáticos	22

LISTA DE QUADROS

Quadro 1 – Pesquisas realizadas	14
---	----

LISTA DE ABREVIATURAS E SIGLAS

AIV	<i>Assembly, Integration, and Verification</i>
CSA	<i>Canadian Space Agency</i>
NASA	<i>National Aeronautics and Space Administration</i>
SpaceLab	Laboratório de Pesquisa em Tecnologias Espaciais - UFSC
UFSC	Universidade Federal de Santa Catarina

SUMÁRIO

1	INTRODUÇÃO	9
1.1	OBJETIVOS	11
1.1.1	Objetivo Geral	11
1.1.2	Objetivos Específicos	11
1.2	ORGANIZAÇÃO DO TRABALHO	11
2	FUNDAMENTAÇÃO TEÓRICA	12
2.1	CUBESAT	12
2.2	TESTE DE SOFTWARE	12
2.2.1	Teste de Unidade	13
2.3	CMOCKA	13
2.4	MOCKUPS	13
2.5	AUTOMAÇÃO DE TESTES	13
3	TRABALHOS RELACIONADOS	14
3.1	INTEGRATION AND VERIFICATION APPROACH OF ISTSAT-1 CUBESAT (MONTEIRO <i>et al.</i> , 2019)	14
3.2	QUALIFICATION AND VALIDATION TEST METHODOLOGY OF THE OPEN-SOURCE CUBESAT FLORIPASAT-I ("MARCELINO <i>et al.</i> , 2020)	15
3.3	A CRITICAL EMBEDDED SYSTEM CHALLENGE: THE FLORIPASAT- 1 MISSION (MARCELINO <i>et al.</i> , 2020)	16
3.4	CONSIDERAÇÕES	17
4	PROPOSTA	18
4.1	VISÃO GERAL	18
4.2	WORKFLOWS	18
4.2.1	Identificação dos Arquivos	19
4.2.2	Job Matrix	19
4.2.3	Compilação e execução	19
4.3	GITHUB ACTIONS	19
4.3.1	Hospedagem	20
4.3.2	Execução	21
4.3.3	Resultados	21
4.4	RESULTADOS ESPERADOS	21
5	DESENVOLVIMENTO	23
6	CONCLUSÕES	24
6.1	TRABALHOS FUTUROS	24
	REFERÊNCIAS	26

**APÊNDICE A – *WORKFLOW* DE TESTES DE UNIDADE PARA O
MÓDULO OBDH 2.0**

28

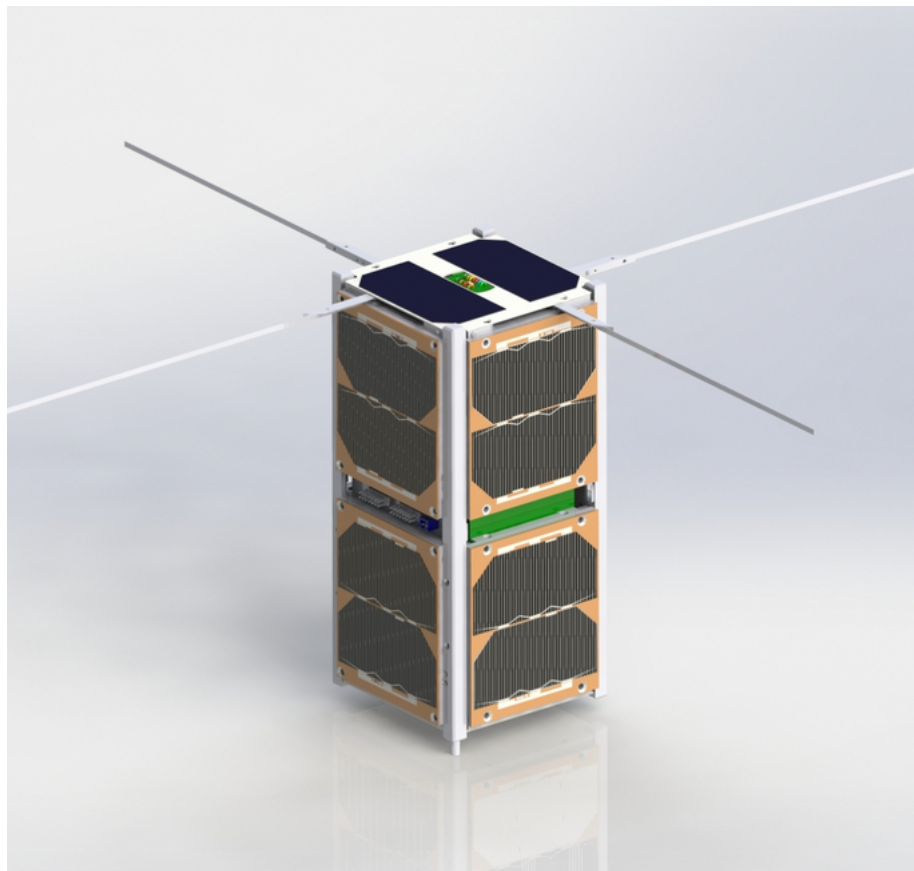
A.1	UNIT-TESTS-DEVICES.YML	28
A.2	DEPLOYJSON.PY	28

1 INTRODUÇÃO

CubeSats são uma classe de artefatos espaciais elaborada com o intuito de reduzir custos e tempo de desenvolvimento, além de providenciar maior acessibilidade ao espaço (JOHNSTONE, 2020). Inicialmente projetados para utilização educacional em universidades (BURT, 2011), são amplamente usados para exploração espacial em órbita terrestre baixa, com altitudes entre 160km e 2000km (ALANAZI; STRAUB, 2019).

CubeSats possuem uma unidade básica (U) de dimensão 10cm x 10cm x 10cm e de até 1kg de massa, mas podem ser configurados em até 24 Unidades (ou 24U) (CANADIAN SPACE AGENCY - CSA, 2018). O FloripaSat-2 (SPACELAB - UFSC, 2021), projeto do Laboratório de Pesquisa em Tecnologias Espaciais da UFSC no qual este trabalho se baseia, é um CubeSat 2U que se encontra no presente momento, em estágio de desenvolvimento ativo. A figura 1 apresenta uma renderização da configuração do FloripaSat-2.

Figura 1 – Renderização do FloripaSat-2.



Fonte: (SPACELAB - UFSC, 2021)

Uma das etapas do desenvolvimento do FloripaSat-2 é a *FlatSat* (SPACELAB - UFSC, 2021), que se trata de uma plataforma de testes onde os módulos do satélite podem ser executados em conjunto para verificar e avaliar o funcionamento de ambos hardware e firmware dos módulos individuais, e ainda a interação entre eles, simulando o funcionamento do satélite em órbita.

O processo de verificação de design é uma etapa crucial em projetos de engenharia. Em tradução livre dos autores, Monteiro et al, dizem que:

Em projetos espaciais, a amplitude e cuidado minucioso dos processos de teste são ainda mais importantes, em vista do nível de confiabilidade a ser imposto no produto final, que é comumente um artefato espacial que precisa resistir à uma gama de ameaças e funcionar em um ambientes inóspitos. (MONTEIRO *et al.*, 2019)

Ainda segundo os autores, o processo de *A/V* (sigla em inglês para Montagem, Integração, e Verificação) tende a ser mais leve em CubeSats, devido à menor escala dos projetos.

No entanto, o risco de um projeto de nanossatélite não é desprezível. Em um estudo realizado sobre os lançamentos de CubeSats entre 2003 e 2015, os autores (PANGA *et al.*, 2016) constataram que cerca de 15% das missões CubeSat nesse período não conseguiram manter comunicações com as estações de controle, resultando em falha da missão após o lançamento.

Por esses motivos, percebe-se então a relevância de um plano de testes bem estruturado para o ciclo de desenvolvimento do FloripaSat-2, pois é necessário elevar o nível de confiabilidade e a garantia de bom funcionamento do sistema, a fim de diminuir a probabilidade de que problemas no satélite resultem em falha da missão.

Durante a participação e contribuição para o projeto, especialmente nos preparativos para a **elaboração dos testes** e considerando os pontos descritos, o crescente interesse se expandiu e a possibilidade de torná-lo um Trabalho de Conclusão de Curso surgiu. O trabalho propõe, então, dois pontos principais:

- A implementação de um sistema de workflows hospedados na plataforma GitHub Actions (GITHUB, n.d.) nos repositórios oficiais do projeto, que possibilite a execução automática de arquivos de teste e do firmware dos módulos do FloripaSat-2 durante a *FlatSat*;
- A análise qualitativa e quantitativa dos dados coletados e dos arquivos de *log* gerados durante a execução dos testes.

Os resultados esperados deste trabalho são a criação de um modelo para automação a ser usado em futuros projetos de desenvolvimento de sistemas embarcados para CubeSats, e um relatório com as conclusões e análises obtidas por meio do estudo dos dados e *logs* coletados.

1.1 OBJETIVOS

1.1.1 Objetivo Geral

O objetivo principal do trabalho é implementar um sistema de fluxos de testes automatizados, hospedados na plataforma GitHub em conjunto com os repositórios dos sistemas e módulos do FloripaSat-2. Eles então, serão aplicados durante o projeto de desenvolvimento do satélite e através dos resultados coletados durante a execução dos testes, e será apresentada uma análise conclusiva sobre a etapa de *FlatSat*.

1.1.2 Objetivos Específicos

1. Implementar *workflows* capazes de automatizar a execução de softwares e testes;
2. Implementar sistemas de geração e armazenamento de dados de *log* das execuções dos *workflows*;
3. Analisar e interpretar os registros coletados para a obtenção de resultados conclusivos sobre a etapa de execução;
4. Disponibilizar códigos-fonte, dados de registro e resultados obtidos.

1.2 ORGANIZAÇÃO DO TRABALHO

Este trabalho está organizado do seguinte modo:

- O capítulo 2 apresenta a definição da metodologia utilizada e a formalização da fundamentação teórica utilizada na elaboração do trabalho;
- O capítulo 3 apresenta uma breve revisão de alguns trabalhos relacionados que abordam o desenvolvimento e testes de CubeSats;
- O capítulo 4 apresenta e descreve o sistema proposto;
- O capítulo 5 apresenta o desenvolvimento e implementação;
- O capítulo 6 contém descrições e análises dos resultados obtidos;
- O capítulo 7 contém as conclusões, considerações finais e os trabalhos futuros.

2 FUNDAMENTAÇÃO TEÓRICA

Este capítulo apresenta a fundamentação teórica utilizada na elaboração do trabalho, trazendo conceitos básicos de nanossatélites e automação de testes, além de um aprofundamento sobre como os temas foram empregados durante o desenvolvimento e implementação do trabalho.

2.1 CUBESAT

CubeSats são uma categoria de satélites miniaturizados cuja principal característica é possuir dimensões e massa reduzidos. De acordo com o documento oficial de padrões de design (JOHNSTONE, 2020), CubeSats devem ter estrutura em formato de cubo de lado 100mm e não podem exceder 1kg de massa.

Essas características correspondem a uma unidade, ou um CubeSat 1U. CubeSats podem ser empilhados em múltiplas unidades, em configurações como 2U, 3U, 6U ou 12U.

O padrão CubeSat foi desenvolvido pela *California Polytechnic State University*, e é uma especificação de projeto de colaboração internacional entre governos, universidades, escolas e o setor privado, com o principal objetivo de conceder acesso ao espaço para cargas pequenas, e possibilitar missões com objetivos diversos, como demonstrações de tecnologia, como foi o caso da missão Mars Cube One, desenvolvida pelo Jet Propulsion Laboratory e enviada à Marte (JPL, 2018), ou com objetivos educacionais, como os CubeSats desenvolvidos pelo SpaceLab da UFSC.

Os CubeSats tradicionalmente possuem custo e risco baixos. Por serem fabricados em sua maioria com peças e materiais provenientes de indústrias convencionais, o orçamento de uma missão não se compara aos orçamentos bilionários de missões convencionais. E do mesmo modo, o risco de se perder um CubeSat em uma missão que resulte em falha não é tão impactante em termos orçamentários e de trabalho perdido. Esse fato faz com que esse tipo de missão seja ideal para ser utilizado com propósitos educativos por instituições de ensino. A missão FloripaSat-1 da UFSC, por exemplo, teve como um dos objetivos treinar e capacitar estudantes de engenharia no processo de desenvolvimento e operação de uma missão espacial (MARCELINO *et al.*, 2020).

2.2 TESTE DE SOFTWARE

Segundo Raul Wazlawick (WAZLAWICK, 2019), quando se trata de teste é importante definir termos que, apesar de parecerem sinônimos, tem significados distintos na área de testes de software:

- Erro (error)

- Defeito (fault)
- Falha (failure)
- Engano (mistake)

Outra distinção importante é entre *verificação* e *validação*. (Continuar)

2.2.1 Teste de Unidade

Testes de unidade são testes de unidade.

2.3 CMOCKA

CMocka is an elegant unit testing framework for C with support for mock objects. It only requires the standard C library, works on a range of computing platforms (including embedded) and with different compilers.

Os testes de firmware do floripasat foram escritos utilizando a biblioteca CMocka para realizar os mockups para simular a interação com o hardware.

2.4 MOCKUPS

Mockups

2.5 AUTOMAÇÃO DE TESTES

Durante o tempo de vida de um projeto de um software ou um sistema de softwares, como é o caso de um projeto de satélites artificiais, os componentes envolvidos invariavelmente passarão por algumas mudanças. Mudanças essas que podem introduzir novos *bugs* em componentes que anteriormente funcionavam. A automação de testes, como definido pelos autores do livro *Software Test Automation - Effective use of test execution tools*, se dá pelo emprego de tecnologias que permitam que um caso de teste seja executado de maneira autônoma, sem intervenção ou monitoramento, de maneira muito mais eficiente (FEWSTER; GRAHAM, 1999) do que se executado via interação humana.

A etapa de testes é essencial em qualquer projeto de engenharia, e pode-se dizer que é ainda mais importante em projetos espaciais, já que erros de projeto, sejam eles de hardware ou firmware, podem resultar em falha da missão.

Este trabalho propõe adotar a ferramenta de automação de testes em uma missão CubeSat, mais especificamente a FloripaSat-2, desenvolvida pelo laboratório SpaceLab da Universidade Federal de Santa Catarina.

A missão FloripaSat-2 vem utilizando os testes de software implementados para validar os seus subsistemas.

3 TRABALHOS RELACIONADOS

No motor de pesquisa acadêmica Google Scholar foram feitas pesquisas associando as áreas de conhecimento de desenvolvimento de nanossatélites, testes de software e automação de testes. As pesquisas foram realizadas ao longo da elaboração do trabalho e diferentes *strings* de busca foram utilizadas. Como critério para escolha das publicações à serem estudadas, foram analisados os resultados apresentados na primeira página do motor de busca. Para determinar a relevância de uma publicação para o trabalho, foi analisado o *abstract* e a introdução dos mesmos.

Quadro 1 – Pesquisas realizadas

<i>String</i>	Resultados
nanosattelite + design	15,200
software + test	4,300,000
software+test+automation	1,970,000
flatsat + cubesat	359
cubesat	29,200
flatsat + nanosatellite	273
cubesat + testbed	3,510

Fonte: Autor.

3.1 INTEGRATION AND VERIFICATION APPROACH OF ISTSAT-1 CUBESAT (MONTEIRO *et al.*, 2019)

Satélites artificiais precisam resistir a uma grande quantidade de adversidades enquanto operam em um ambiente hostil e inóspito. Por isso é essencial que sejam feitos testes extensivos para garantir o nível de confiança necessário para que o objetivos da missão sejam atingidos. Porém, missões de CubeSat tendem a não ser tão rigorosas em suas etapas de montagem, integração e verificação (AIV), fato que é refletido na taxa de missões que acabam em falhas.

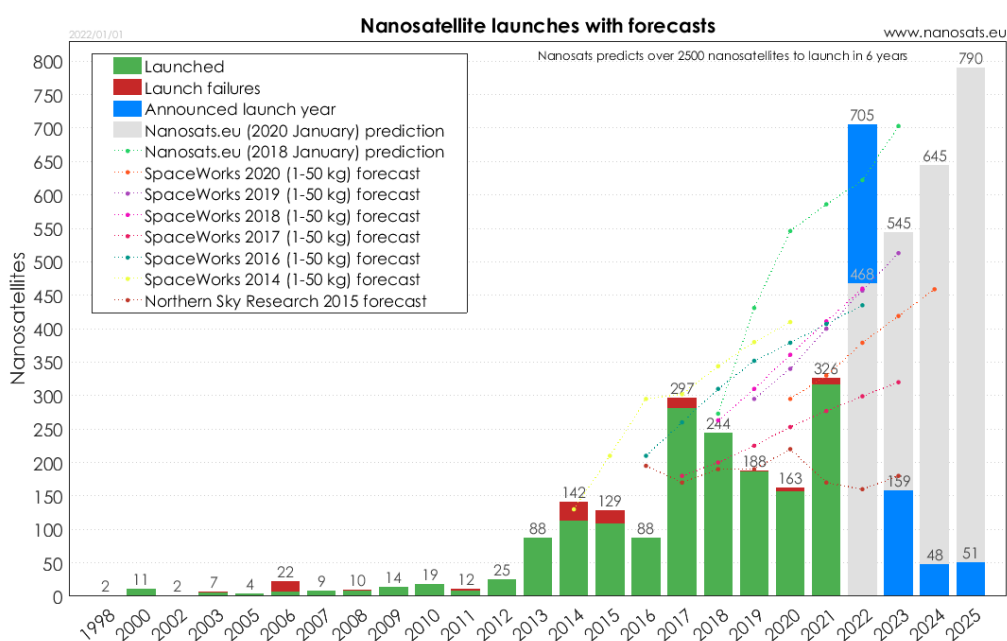
Esse paradigma, no entanto, está mudando a medida que a importância do CubeSats aumenta. Ela vai de simplesmente projetos educacionais, para missões de custos e importância elevados. Um exemplo desse salto é a missão Mars Cube One, ou MarCO (JPL, 2018), desenvolvida pelo *Jet Propulsion Laboratory* da Nasa, que enviou dois CubeSats a Marte, para servirem como satélites de comunicação. Assim, os autores desse artigo apresentam a ideia de que integração e verificação em missões CUBEsats passam a ser encaradas com mais seriedade, como demonstrado na figura 2, onde é possível observar o aumento de lançamento de nanossatélites e, paralelamente, a diminuição da taxa de falhas.

Como estudo de caso, o artigo apresenta o processo de AIV empregado no desenvolvimento do ISTSat-1 (ISTSAT-1... , 2021), o primeiro nanossatélite português,

lançado em 2021 e que utilizou a estratégia de teste através de uma *FlatSat*, que se trata de uma plataforma de testes de *Hardware In the Loop* (HIL).

A equipe do ISTSat-1, segundo os autores, optou por desenvolver a maioria dos subsistemas do satélite e adotou métodos ágeis para as atividades de AIV de forma a diminuir os riscos relacionados a falta de experiência da equipe. Dessa forma foi possível desenvolver e testar os módulos de forma contínua e iterativa, o que possibilitou a descoberta rápida de diversos erros e problemas que necessitavam de ajustes de design. Foi concluído nesse estudo que, caso a equipe tivesse optado por um processo de desenvolvimento em cascata, como tradicionalmente empregado em missões CubeSat, seria mais difícil, senão impossível, ter descoberto os mesmos problemas no design e funcionamento. Concluem então, que a adoção de processos iterativos é útil para projetos educacionais que contam com equipes sem muita experiência, ou sem muitos recursos financeiros.

Figura 2 – Lançamentos anuais de Nanossatélites (previsão)



Fonte: Nanosats Database (KULU, ERIK, 2021)

3.2 QUALIFICATION AND VALIDATION TEST METHODOLOGY OF THE OPEN-SOURCE CUBESAT FLORIPASAT-I ("MARCELINO *et al.*, 2020)

Esta seção e a próxima apresentam dois trabalhos produzidos como resultado da missão FloripaSat-1, uma missão de demonstração desenvolvida inteiramente por estudantes da Universidade Federal de Santa Catarina (UFSC), cujo CubeSat foi lançado em 2019.

O FloripaSat-1 é composto de três módulos distintos, o sistema elétrico - EPS (*Electric Power System*), o módulo de gerência de dados - OBDH (*Onboard Data Handling*) e o módulo de telemetria e telecomandos - TT&C (*Telemetry, Tracking and Command*).

O artigo informa que a missão tinha como objetivo testar tecnologias que possibilitem o desenvolvimento rápido e de baixo custo de satélites espaciais, além de treinar estudantes nas áreas de concepção, implementação e operação de uma missão espacial completa. Além disso, o FloripaSat-1 é um projeto *open source* e as informações de software e hardware dos módulos desenvolvidos estão disponíveis em repositórios públicos, para uso em futuras missões.

Os autores descrevem que o FloripaSat-1 foi desenvolvido com base em projetos de engenharia de sistemas, dividido em Modelos de protótipo (PM), engenharia (EM-I e EM-II) e finalmente o Modelo de Vão (FM). Foram realizados testes em cada uma dessas etapas para validar os sistemas e os resultados, segundo o artigo, foram decisivos na detecção e correção de erros.

Apesar do artigo focar em integração e validação dos módulos a nível de hardware, julga-se relevante para a elaboração deste trabalho, que foca principalmente em software, por descrever as diferentes condições que uma missão espacial é submetida, além da metodologia de desenvolvimento adotada pela equipe do FloripaSat-1, em que a grande maioria dos membros seguiu para a missão FloripaSat-2, a qual este trabalho se baseia.

3.3 A CRITICAL EMBEDDED SYSTEM CHALLENGE: THE FLORIPASAT-1 MISSION (MARCELINO *et al.*, 2020)

Este artigo, elaborado também a partir da missão FloripaSat-1 do SpaceLab, apresenta definições e descrições dos subsistemas do satélite, além de algumas informações e resultados de testes e simulações realizadas nos mesmos.

Como descrito anteriormente, o FloripaSat-1 possui três sub sistemas principais:

- *Electric Power System* (EPS)
- *On-Board Data Handling* (OBDH)
- *Tracking, Telemetry and Command* (TT&C)

O EPS é o módulo responsável por distribuir, coletar e armazenar a energia utilizada pelo satélite. A energia é coletada através de painéis solares e armazenada em uma bateria de íon de lítio. A distribuição da energia é definida a partir de um microcontrolador que analisa os estados de carga e de energia e decide quais módulos permanecerão em operação.

O OBDH é o módulo responsável pela gerência de atividades do satélite. Ele realiza a interface entre todos os subsistemas do satélite. Os dados gerados são empacotados e transmitidos através do TT&C, e dados recebidos são enviados ao OBDH para que ele execute a tarefa requisitada, ou então envie o comando ao módulo requisitado. Este módulo também possui uma memória, para que possam ser futuramente recuperados.

Finalmente, o TT&C é o módulo responsável pela comunicação entre o satélite e as estações de controle na Terra. Ele opera através de dois módulos de rádio, um para a banda VHF e um para UHF. Os comandos a serem enviados e recebidos pelo satélite (*downlink/ uplink*) são transmitidos através do módulo UHF e a banda VHF é reservada para transmissões do tipo *beacon*.

Os autores discorrem ainda sobre os demais módulos e subsistemas do satélite, como outros *payloads* que não foram desenvolvidos ou projetados pela equipe da UFSC e portanto, ficam de fora desse breve resumo.

A relevância deste artigo se dá pelo fato de que o FloripaSat-2 utiliza a mesma estrutura de subsistemas, com os seus módulos sendo sucessores diretos do software e hardware utilizados na missão anterior, operando de forma idêntica ou muito similar.

3.4 CONSIDERAÇÕES

Os trabalhos relacionados escolhidos têm como principal função estabelecer a importância de uma estrutura de testes bem fundamentada. Todos os trabalhos mencionam que a etapa de testes foi importante pra a descoberta e solução de erros que não foram descobertos ou detectados durante os ciclos de desenvolvimento normais.

Além disso, os dois trabalhos produzidos por estudantes do SpaceLab como resultado da missão FloripaSat-1 mostram a importância de CubeSats como ferramenta educacional, servindo para treinar e capacitar estudantes no projeto, desenvolvimento e operação de uma missão espacial completa.

O modelo proposto por este trabalho é uma continuidade desse pensamento: possibilitar mais uma ferramenta de testes que possa resultar em maior confiabilidade dos sistemas embarcados desenvolvidos para o FloripaSat-2, fazendo uso do ambiente de aprendizado e da experiência adquirida durante o desenvolvimento da missão.

4 PROPOSTA

Este capítulo descreve e formaliza o sistema proposto por este trabalho. A motivação para este trabalho surgiu a partir de conversas e reuniões com as equipes de desenvolvimento do FloripaSat-2, e tem como inspiração projetos que utilizam conceitos de integração contínua para automatizar o processo de testes e *deploy* de sistemas.

Inicialmente, como teste de conceito, foi criado um *workflow* para automatizar a execução dos testes dos programas de alguns dos subsistemas do satélite, porém, foi feito de um modo genérico o suficiente para que pudesse ser implementado em todos os outros módulos do FloripaSat-2.

Após verificar o funcionamento e a utilidade destes *workflows*, surgiu a possibilidade de aplicar os mesmos conceitos à FlatSat da missão, de modo a automatizar os testes de software e assim, esta se tornou oficialmente a proposta do trabalho.

Os workflows propostos por este trabalho seguirão a mesma linha de raciocínio e apresentarão funcionalidades semelhantes aos já implementados. As próximas seções apresentam uma descrição do modelo proposto, fazendo comparações, quando possível, com o que já foi implementado.

4.1 VISÃO GERAL

A *FlatSat*, como descrita na seção ??, é uma maneira de testar de maneira rápida e simultânea os subsistemas do satélite. Esta proposta foca em testes de software, utilizando a flatsat como ferramenta de acesso aos sistemas embarcados. Através dela, será possível testar os programas embarcados do FloripaSat-2 de maneira simultânea e em conjunto, analisando a interação entre os diferentes subsistemas, além dos funcionamentos isolados.

Estes *workflows* ainda estão em fase de concepção durante a elaboração deste TCC. Por isso, as próximas seções se baseiam nos fluxos de execução já implementados, que possuem muitos dos mesmos objetivos e características de funcionamento.

4.2 WORKFLOWS

Esta seção e a próxima tomam como exemplo os workflows de automação empregados no módulo OBDH 2.0, já que os fluxos de automação específicos para a FlatSat serão estruturados de maneira similar e seguindo os mesmos princípios. Esta seção apresenta algumas seções de código extraídas para ilustrar e demonstrar o funcionamento do sistema, mas os códigos estão disponíveis de forma pública nos repositórios dos subsistemas, como (SPACELAB, 2020a) e (SPACELAB, 2020b). Eles também são apresentados integralmente ao final deste documento, no Apêndice A.

4.2.1 Identificação dos Arquivos

Os *workflows* são responsáveis por executar uma bateria de testes de unidade, que estão separados em arquivos únicos. Inicialmente, o sistema precisa identificar e compilar cada um desses arquivos de teste. Para isso é empregado um script *Python* que procura nas localizações relevantes arquivos que possuam a nomenclatura apropriada. Neste caso, arquivos cujos nomes terminam como `_test.c`. O trecho de código abaixo, extraído do arquivo *deployJSON.py* e disponível em A.2, demonstra essa busca pelos arquivos relevantes.

Os nomes e localização dos testes encontrados são salvos em um arquivo *.json*, utilizado posteriormente para compilação.

4.2.2 Job Matrix

Para garantir que os testes sejam executados com relativa rapidez, o *workflow* foi estruturado para executar cada teste simultaneamente de forma isolada. Para isso, foi utilizada uma *Job Matrix*, que se trata de uma maneira de dividir um *workflow* em subtarefas que se executam paralelamente. Uma explicação detalhada do funcionamento e configuração de uma *Job Matrix* pode ser encontrada na documentação oficial da ferramenta *GitHub Actions* (GITHUB, n.d.).

Neste caso, o número de subtarefas, ou *jobs*, é definido pela quantidade de testes que foram encontrados pelo passo anterior, para que assim o *workflow* gere a matriz através do resultado da execução do script apresentado anteriormente. O Trecho de código a seguir apresenta essa funcionalidade.

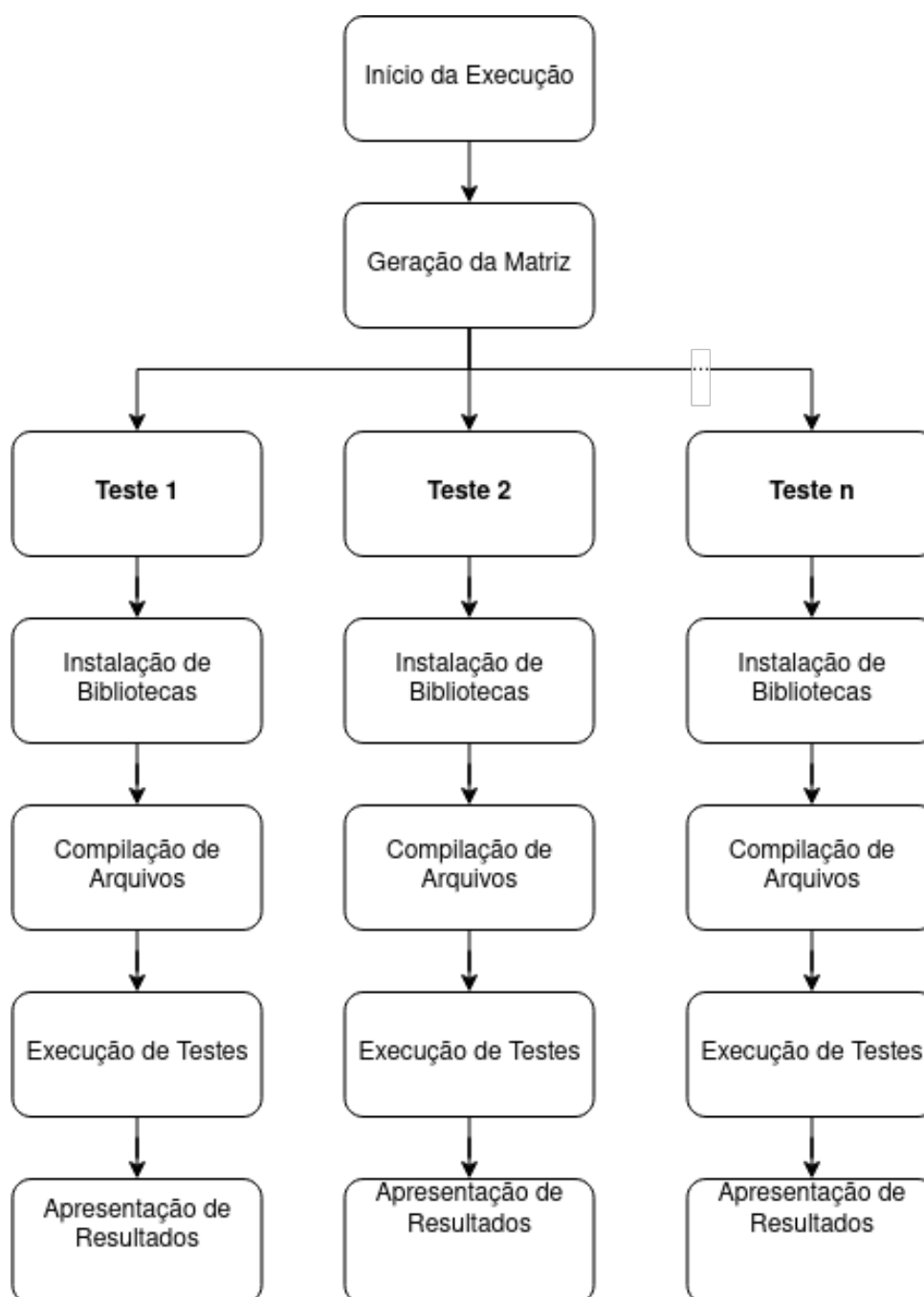
4.2.3 Compilação e execução

A compilação é feita através de um comando *make*, recebendo como entrada cada um dos arquivos encontrados pelo script identificador. A Matriz de subtarefas é organizada de modo que os arquivos sejam divididos de maneira única, ou seja, cada subdivisão do *workflow* é responsável por compilar e executar um único teste. A figura 3 apresenta um diagrama exemplificando o fluxo de execução do processo descrito. O diagrama não representa a identificação dos arquivos de teste.

4.3 GITHUB ACTIONS

Esta seção apresenta uma breve descrição da ferramenta *GitHub Actions*, cuja documentação oficial e completa está disponível em (GITHUB, n.d.). Ela se trata de uma plataforma de CI/CD (*Continuous Integration/Continuous Delivery*) que permite automatizar testes, builds e deploys através da execução de *workflows* que realizam essas tarefas após alterações no repositório como *commits*, ou *pull requests*.

Figura 3 – Diagrama de execução de um workflow de testes automatizados



Fonte: Autor

4.3.1 Hospedagem

Os servidores responsáveis pela execução dos *workflows* são chamados de *runners*. O *GitHub* fornece servidores Windows, Ubuntu e macOS, e cada execução de um workflow ocorre em uma máquina virtual nova. Também é possível usar um servidor próprio, chamados de servidores *self-hosted*. Os *workflows* implementados até o momento são executados em servidores Ubuntu do *GitHub*. Os implementados para a FlatSat, no entanto, serão executados em servidores *self-hosted*, já que será ne-

cessário manter conexão com os módulos do FloripaSat-2 de modo a testar e executar os programas embarcados.

4.3.2 Execução

É possível planejar a execução dos *workflows* e sincronizá-las com ações dentro dos repositórios. Nos casos do OBDH 2.0 e do EPS 2.0, os testes são executados após *commits* em *branches* específicas do repositório como a *master* ou *dev-firmware*, ou após *pull requests*. Também é possível configurar a execução manual, para executar os testes sem realizar nenhuma alteração prévia no repositório.

4.3.3 Resultados

Ao fim da execução dos *workflows*, a ferramenta apresenta um *log* com todos os eventos e mensagens resultantes da execução. Aqui é possível, por exemplo, verificar se houve alguma falha na execução dos testes. Esses registros ficam disponíveis no repositório, e sua utilidade vem do nível de detalhamento que forem configurados. Para o sistema proposto por este trabalho, espera-se um elevado nível de detalhamento dos registros, de modo que os resultados possam ser analisados e seja possível extrair dados conclusivos sobre dados de execução e estatística de falhas e sucessos nos testes. A figura 4 apresenta uma captura de tela de uma execução dos testes durante um *commit* no repositório do OBDH 2.0.

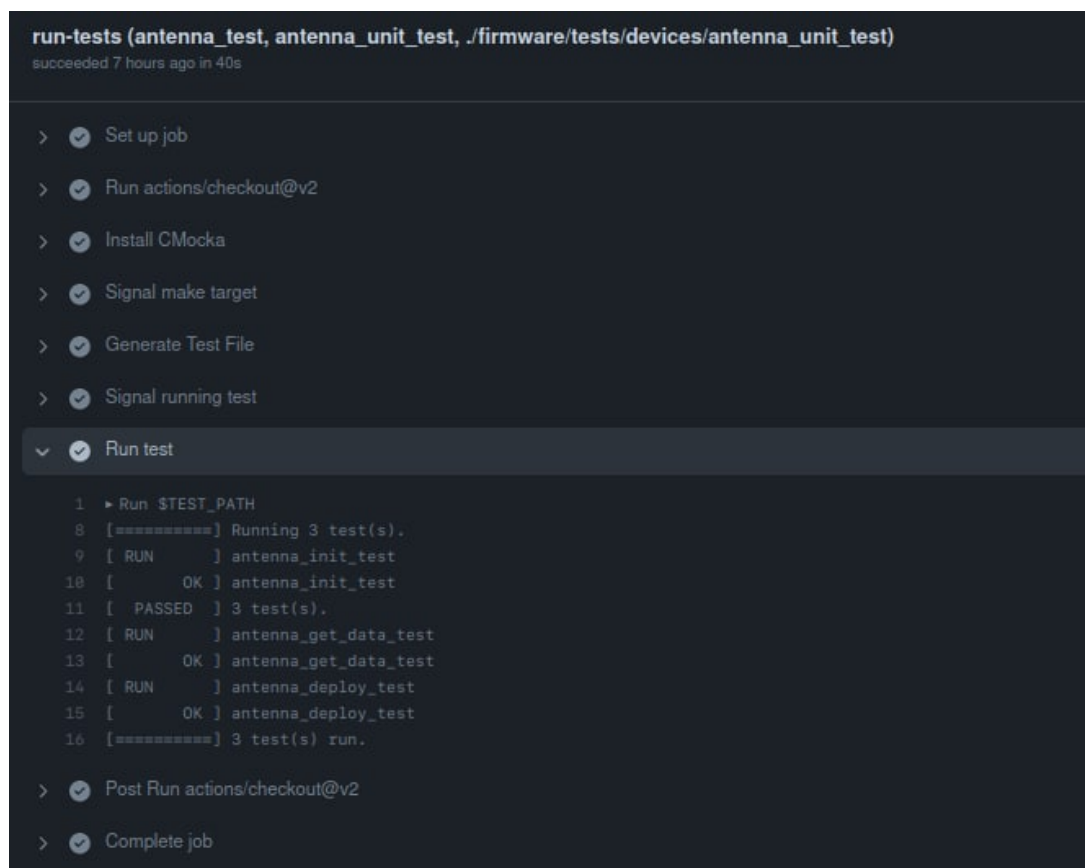
4.4 RESULTADOS ESPERADOS

Os *workflows* de automação descritos anteriormente se provaram benéficos para o ciclo de desenvolvimento dos sistemas do FloripaSat-2. Através deles, é possível testar cada alteração proposta de maneira eficiente e a automação fornece um registro de execuções passadas, onde é possível comparar alterações no código e os resultados que elas provocaram na execução dos testes. Em termos de controle de versão, os testes foram centralizados e disponibilizados para visualização e estudos.

Espera-se que os programas sob teste na FlatSat possam receber os mesmos benefícios e que isso seja potencializado, visto que o escopo da FlatSat é maior, sendo possível testar não somente a execução individual dos sistemas embarcados, mas também todas as funcionalidades que dependem da integração entre os módulos. Dessa forma, será possível, a princípio, testar o software do FloripaSat-2 como um todo, assumindo que todos os seus subsistemas estejam conectados à FlatSat.

Os *workflows* projetados seguirão os mesmos princípios dos que foram descritos anteriormente, de forma a padronizar o uso dentro da missão e para que seja possível aplicar os mesmos princípios de agilidade e eficiência.

Figura 4 – Captura de tela do resultado de uma execução de testes automáticos



The screenshot displays a terminal window for a CI/CD pipeline run titled "run-tests (antenna_test, antenna_unit_test, ./firmware/tests/devices/antenna_unit_test)". The status is "succeeded 7 hours ago in 40s". The pipeline consists of several steps, all of which are marked as successful with green checkmarks:

- > Set up job
- > Run actions/checkout@v2
- > Install CMocka
- > Signal make target
- > Generate Test File
- > Signal running test
- > Run test (expanded)
- > Post Run actions/checkout@v2
- > Complete job

The "Run test" step is expanded, showing the following output:

```
1 ▶ Run $TEST_PATH
8 [=====] Running 3 test(s).
9 [ RUN      ] antenna_init_test
10 [      OK  ] antenna_init_test
11 [ PASSED  ] 3 test(s).
12 [ RUN      ] antenna_get_data_test
13 [      OK  ] antenna_get_data_test
14 [ RUN      ] antenna_deploy_test
15 [      OK  ] antenna_deploy_test
16 [=====] 3 test(s) run.
```

Fonte: OBDH 2.0 (SPACELAB, 2020b)

Uma vez que o sistema esteja implementado, espera-se que seu uso contínuo gere uma quantidade considerável de registros e históricos de execução, que serão estudados para produzir a seção de análise deste projeto que trará então um levantamento sobre a utilidade e o impacto do uso desse sistema para o desenvolvimento da missão.

5 DESENVOLVIMENTO

Nesta seção serão feita a apresentação e descrição dos *softwares* e *scripts* desenvolvidos e utilizados no projeto.

Os testes foram escritos utilizando a biblioteca CMocka descrita na seção 2.3

6 CONCLUSÕES

Os trabalhos relacionados escolhidos apresentam um panorama inicial bastante informativo, quando analisados em conjunto:

- A relevância de missões CubeSat cresce a cada ano, observada pelo crescente volume de lançamentos;
- São projetos com potencial educacional elevado, sendo usados por instituições de ensino para capacitar e treinar estudantes e também desenvolver, testar novas tecnologias e processos de desenvolvimento.

Os dois artigos elaborados pela equipe do FloripaSat-1 demonstram a enorme quantidade de conhecimento gerada, bem como a experiência adquirida em projeto e desenvolvimento de tecnologias espaciais.

Percebeu-se também que, embora exista uma biblioteca ampla sobre testes de hardware de *CubeSats*, a quantidade de material distinto que foque apenas (ou principalmente) em teste de software espacial é comparativamente menor. Espera-se que esse trabalho seja, então, uma boa fonte de informação sobre o potencial que esta área pode apresentar para estudos e inovação.

O estudo destes trabalhos relacionados evidência também a importância de um projeto de testes estruturado. Mais de uma vez, nos artigos selecionados anteriormente foi descrito que a etapa de testes foi importante para a descoberta e correção de erros de design, sejam eles de hardware ou software. É nessa vertente que este trabalho propõe uma nova aproximação para os testes de software, por meio de um sistema de workflows que permita a execução de testes de maneira automática, fazendo uso da *FlatSat* da missão FloripaSat-2 para testar de maneira simultânea, os sistemas embarcados de todos os módulos do satélite, bem como as interações e comunicações entre eles.

Se bem sucedido, esse modelo potencializará a descoberta e correção de erros e possibilitará o desenvolvimento de programas mais confiáveis e de forma mais rápida.

6.1 TRABALHOS FUTUROS

Como sugestão e planejamento de trabalhos futuros, sobretudo dando continuidade à este relatório durante a disciplina de Trabalho de Conclusão de Curso II, sugere-se alguns pontos de aprofundamento:

- Aprofundar a pesquisa sobre testes de software, trazendo uma breve análise do estado da arte;

- Trazer mais trabalhos relacionados sobre testes de software embarcado, e software para aplicações espaciais;
- Implementar e apresentar o sistema de *workflows* à ser utilizado em conjunto com a FlatSat do FLoripaSat-2;
- Armazenar, preservar, e analisar os registros e histórico de execuções dos fluxos de testes automatizados;
- Redigir um estudo de caso trazendo as principais informações e conclusões, tendo como base os dados coletados.

REFERÊNCIAS

"MARCELINO, Gabriel Mariano; MORSCH, Filho Edemar; SARA, Vega-Martinez; PIO, De Mattos André Martins; ORIEL, Seman Laio; KESSLER, Slongo Leonardo; AUGUSTO, Bezerra Eduardo". "Qualification and validation test methodology of the open-source CubeSat FloripaSat-I". **"Journal of Systems Engineering and Electronics"**, v. 31, n. 6, p. 1230–1244, 2020. DOI: 10.23919/JSEE.2020.000103. Disponível em: <https://ieeexplore.ieee.org/document/9316404>.

ALANAZI, Abdulaziz; STRAUB, Jeremy. Engineering Methodology for Student-Driven CubeSats. **Aerospace**, v. 6, n. 5, 2019. ISSN 2226-4310. DOI: 10.3390/aerospace6050054. Disponível em: <https://www.mdpi.com/2226-4310/6/5/54>.

BURT, Robert. **Distributed Electrical power systems in cubesat applications**. 2011. Diss. (Mestrado) – Utah State University.

CANADIAN SPACE AGENCY - CSA. **What is a CubeSat**. [S.l.: s.n.], 2018. asc-csa.gc.ca/eng/satellites/cubesat/what-is-a-cubesat.asp. Acessado em 30/08/2021.

FEWSTER, Mark; GRAHAM, Dorothy. **Software test automation**. [S.l.]: Addison-Wesley Reading, 1999.

GITHUB. **GitHub Actions**. [S.l.: s.n.], n.d. github.com/features/actions. Acessado em 01/09/2021.

ISTSAT-1. [S.l.: s.n.], 2021. <https://istsat-one.tecnico.ulisboa.pt/~istsat-one.daemon/>. Acessado em 28/02/2022.

JOHNSTONE, Alicia. **CubeSat Design Specification (1U – 12U) REV 14 CP-CDS-R14**. San Luis Obispo, CA, EUA, 2020.

JPL. **Mars Cube One (MarCO)**. [S.l.: s.n.], 2018. <https://www.jpl.nasa.gov/missions/mars-cube-one-marco>. Acessado em 21/02/2022.

KULU, ERIK. **Nanosats Database**. [S.l.: s.n.], 2021. <https://www.nanosats.eu/>. Acessado em 08/09/2021.

MARCELINO, Gabriel Mariano; VEGA-MARTINEZ, Sara; SEMAN, Laio Oriel; KESSLER SLONGO, Leonardo; BEZERRA, Eduardo Augusto. "A Critical Embedded System Challenge: The FloripaSat-1 Mission". **"IEEE Latin America Transactions"**, v. 18, n. 02, p. 249–256, 2020. DOI: 10.1109/TLA.2020.9085277. Disponível em: <https://ieeexplore.ieee.org/document/9085277>.

MONTEIRO, João P.; ROCHA, Rui M.; SILVA, Alexandre; AFONSO, Rúben; RAMOS, Nuno. Integration and Verification Approach of ISTSat-1 CubeSat. **Aerospace**, v. 6, n. 12, 2019. ISSN 2226-4310. DOI: 10.3390/aerospace6120131. Disponível em: <https://www.mdpi.com/2226-4310/6/12/131>.

PANGA, WJ; BO, B; MENG, X; YU, XZ; GUO, J; ZHOU, J. Boom of the CubeSat: A Statistic Survey of CubeSats Launch in 2003–2015. *In*: PROCEEDINGS of the 67th International Astronautical Congress (IAC), Guadalajara, Mexico. [S.l.: s.n.], 2016. P. 26–30.

SPACELAB. **Electrical Power System 2.0**. [S.l.: s.n.], 2020a. <https://github.com/spacelab-ufsc/eps2>. Acessado em 11/03/2022.

SPACELAB. **On-Board Data Handling 2.0**. [S.l.: s.n.], 2020b. <https://github.com/spacelab-ufsc/obdh2>. Acessado em 11/03/2022.

SPACELAB - UFSC. **FloripaSat-2 Doc**. [S.l.: s.n.], 2021. <https://spacelab-ufsc.github.io/floripasat2-doc/slb-fsat2-doc-v0.2.pdf>. Acessado em 30/08/2021.

WAZLAWICK, Raul Sidnei. **Engenharia de Software: Conceitos e Práticas**. 2. ed. [S.l.]: Elsevier, 2019. ISBN 978-8535260847.

APÊNDICE A – *WORKFLOW* DE TESTES DE UNIDADE PARA O MÓDULO OBDH 2.0

Este capítulo apresenta os códigos necessários para a execução do *workflow* de automação de testes de unidade do módulo OBDH 2.0 (SPACELAB, 2020b) do FloripSat-2

A.1 UNIT-TESTS-DEVICES.YML

A.2 DEPLOYJSON.PY