ISSN 2447-6102

Article

Arquitetura de Software da Missão do OBDH do Cubesat Aldebaran 1

Marques, J.G.B.[1]*, Santos, J.L.[2], Silva, L.C.[3], Júnior, C.A.[4], Júnior, J.R.[5] and Lopes, M.[6]

[1] Engenharia Aeroespacial, Universidade Federal do Maranhão, Maranhão, Brasil; jgb.marques@discente.ufma.br
[2] Engenharia Aeroespacial, Universidade Federal do Maranhão, Maranhão, Brasil; joao.lcs@discente.ufma.br
[3] Engenharia Aeroespacial, Universidade Federal do Maranhão, Maranhão, Brasil; claudio.luis@ufma.br
[4] Engenharia Aeroespacial, Universidade Federal do Maranhão, Maranhão, Brasil; carlos.brito@ufma.br
[5] Engenharia Elétrica, Universidade Federal do Maranhão, Maranhão, Brasil; jrbp.junior@ufma.br
[6] Engenharia Aeroespacial, Universidade Federal do Maranhão, Maranhão, Brasil; lopes.marcus@ufma.br
***** Correspondence: jgb.marques@discente.ufma.br

Received: Outubro/2025; Accepted: Novembro/2025; Published: Novembro/2025

Resumo: Missões de CubeSat acadêmicas enfrentam desafios no desenvolvimento de software de voo robusto com recursos limitados. Este artigo aborda essa questão ao apresentar um estudo de caso sobre o desenvolvimento e a validação do software para o sistema On-Board Data Handling (OBDH) do Aldebaran 1, um CubeSat 1U. A arquitetura de software, adaptada da plataforma de código aberto FloripaSat-2 com FreeRTOS, foi projetada para gerenciar o deploy de antenas, processar telecomandos e armazenar dados de forma persistente. A validação do sistema foi realizada por meio de testes de sistema fim a fim (end-to-end), utilizando estações terrestres para simular um ciclo operacional completo. Os testes confirmaram a correta implementação da arquitetura, incluindo a cronologia do deploy, o processamento de comandos e a persistência de dados após reinicializações. Notavelmente, a estratégia de gerenciamento de energia foi simplificada para controle remoto do rádio devido a limitações de hardware. Este trabalho apresenta um ciclo de desenvolvimento pragmático e bem-sucedido, servindo como uma referência técnica e um conjunto de lições aprendidas que são contribuições valiosas para outros projetos universitários de CubeSat.

Palavras-chave: cubesat; OBDH; software embarcado; Aldebaran 1.

Abstract: Academic CubeSat missions face significant challenges in developing robust flight software with limited resources. This paper addresses this issue by presenting a case study on the development and validation of the software for the On-Board Data Handling (OBDH) system of Aldebaran 1, a 1U CubeSat. The software architecture, adapted from the open-source FloripaSat-2 platform with FreeRTOS, is designed to manage antenna deployment, process telecommands, and persistently store mission data. System validation was performed through end-to-end system tests, utilizing ground stations to simulate a complete operational cycle. The tests confirmed the correct implementation of the architecture, including the deployment timeline, command processing, and data persistence across reboots. Notably, the power management strategy was simplified to remote radio control due to hardware limitations. This work presents a pragmatic and successful development cycle, serving as a technical reference and a set of lessons learned that provide valuable contributions to other university-led CubeSat projects.

Keywords: cubesat; OBDH; flight software; Aldebaran 1.

1. Introdução

Os CubeSats representam uma revolução no acesso ao espaço, sendo caracterizados pelo baixo custo, flexibilidade e pela capacidade de integrar componentes comerciais de prateleira (COTS). Essa abordagem, pioneira no final da década de 1990, viabilizou uma gama diversificada de aplicações que vão desde o sensoriamento remoto, com constelações como a da Planet Labs que imageiam a Terra diariamente, até missões interplanetárias como a MarCO da NASA, que demonstrou a viabilidade da comunicação de CubeSats no espaço profundo, Cratere et al. (2024). Particularmente no meio acadêmico, os CubeSats se consolidaram como uma plataforma educacional inestimável, permitindo que estudantes de graduação e pós-graduação participem do ciclo completo de uma missão espacial — do projeto à operação —, promovendo uma formação prática e multidisciplinar em engenharia e ciências espaciais.

Este movimento de miniaturização é um pilar da economia do "New Space", que prioriza o desenvolvimento ágil e a redução drástica de custos. Contudo, a mesma miniaturização e o uso de COTS impõem desafios significativos. O ambiente espacial é inerentemente hostil, com radiação ionizante capaz de causar falhas em componentes eletrônicos (Single Event Upsets - SEUs), vácuo e ciclos térmicos extremos. Adicionalmente, as severas restrições de tamanho, peso, energia e custo limitam diretamente a capacidade computacional disponível, exigindo que o software de voo (FSW) seja não apenas funcional, mas também extremamente eficiente e, acima de tudo, robusto e tolerante a falhas, Konecny et al. (2015). A criticidade do software é tamanha que falhas de FSW são apontadas como uma das principais causas de perda de missões de CubeSats, tornando a sua confiabilidade um fator determinante para o sucesso, Bouwmeester et al. (2022).

Neste contexto, o subsistema de processamento de dados a bordo (OBDH) atua como o "cérebro" do satélite, sendo responsável por executar telecomandos, gerenciar a telemetria, controlar a carga útil e garantir a autonomia do sistema. Para gerenciar essa complexidade, a comunidade de desenvolvimento de CubeSats convergiu para um conjunto de melhores práticas. A adoção de arquiteturas de software modulares e em camadas tornou-se um padrão, pois facilita a reutilização de código entre missões e simplifica a manutenção, Avanzi et al. (2018). Essa abordagem tipicamente envolve uma camada de abstração de hardware (HAL), sobre a qual opera um sistema operacional de tempo real (RTOS) que gerencia o escalonamento determinístico de tarefas concorrentes. Sobre o RTOS, reside a camada de aplicação, onde as lógicas específicas da missão são implementadas.

A escolha de uma plataforma de software com herança de voo (flight heritage) é outra estratégia crucial para mitigar riscos, especialmente em projetos com recursos limitados, como os universitários. Plataformas de código aberto, como a do FloripaSat-2, desempenham um papel vital neste ecossistema, oferecendo uma base de código testada e validada que acelera o desenvolvimento e permite que novas equipes foquem nos requisitos únicos de suas missões, Marcelino et al. (2024). Complementarmente, a validação do FSW por meio de testes de sistema em um ambiente representativo é uma etapa indispensável no ciclo de vida do desenvolvimento. Nesses testes, o software embarcado é validado em seu hardware final (o modelo de engenharia) enquanto se comunica via rádio com uma estação terrestre, permitindo a verificação funcional fim a fim do sistema antes do lançamento.

É neste cenário que o projeto Aldebaran 1, ilustrado na Figura 1, se insere. Desenvolvido na Universidade Federal do Maranhão (UFMA), o Aldebaran 1 é um CubeSat 1U com uma missão primariamente educacional. Adicionalmente, o projeto inclui uma missão secundária de prova de conceito para um sistema de retransmissão de mensagens de emergência de pequenas embarcações na costa do Maranhão, utilizando tecnologia LoRa, e uma plataforma para coleta de dados de sensores embarcados, Júnior et al. (2022) e Silva et al. (2022). Diante dos desafios expostos e alinhado com as melhores práticas da comunidade, a equipe do Aldebaran 1 optou por uma metodologia de desenvolvimento pragmática, baseada na adaptação da plataforma de código aberto do FloripaSat-2.

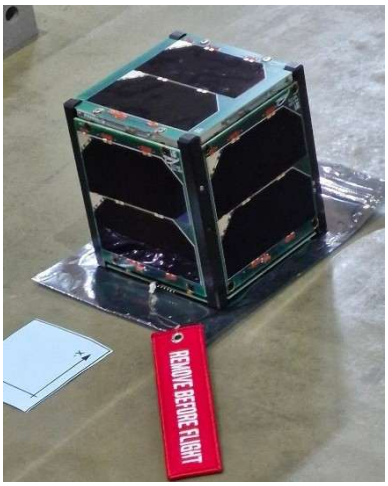

Figura 1. Aldebaran 1. Feito pelo autor

Este artigo apresenta um estudo de caso completo do desenvolvimento e validação do software de missão do OBDH do Aldebaran 1. As principais contribuições deste trabalho são: (1) a apresentação detalhada de uma arquitetura de FSW para um CubeSat 1U, servindo como uma referência técnica para outros projetos acadêmicos; (2) a análise de uma decisão de engenharia crítica — a simplificação da estratégia de gerenciamento de energia em resposta a limitações de hardware — e seu impacto na robustez do sistema; e (3) a demonstração de um ciclo de validação funcional utilizando uma estação terrestre de pequeno porte e uma estação terrestre de grande porte, confirmando a prontidão do software para a missão. O restante deste artigo está estruturado da seguinte forma: a Seção 2 detalha a metodologia de desenvolvimento e a arquitetura de implementação; a Seção 3 e 4 apresenta e discute os resultados dos testes de validação; e, finalmente, a Seção 5 sumariza as conclusões e aponta direções para trabalhos futuros.

2. Materiais e Métodos

Esta seção detalha a metodologia de desenvolvimento, a arquitetura de hardware e software, a implementação das funcionalidades do software de voo (FSW) e o ambiente de validação utilizado no projeto Aldebaran 1.

2.1. Metodologia de Desenvolvimento de Software

O desenvolvimento de FSW para CubeSats é um campo que, apesar da ausência de frameworks universalmente padronizados, possui melhores práticas consolidadas na literatura. A abordagem moderna prioriza arquiteturas modulares e em camadas para facilitar a reutilização e a manutenção, Avanzi et al. (2018). Além disso, o uso de sistemas operacionais de tempo real (RTOS) é fundamental para garantir o escalonamento determinístico de tarefas críticas, Cratere et al. (2024).

A decisão metodológica central do projeto foi a adaptação de uma arquitetura de software de código aberto com herança de voo (flight heritage): a plataforma desenvolvida pelo SpaceLab da UFSC para o CubeSat FloripaSat-2, Marcelino et al. (2024). Essa escolha estratégica permitiu mitigar riscos e acelerar o ciclo de desenvolvimento ao aproveitar uma base de código já testada em condições de voo. O FSW foi construído de forma incremental sobre essa plataforma, com cada funcionalidade sendo desenvolvida e testada individualmente antes da integração final.

O sistema operacional FreeRTOS, nativo da plataforma adaptada, foi mantido como núcleo do FSW. A escolha foi validada por ser uma solução leve, com baixo consumo de memória (low footprint) e amplamente suportada pela comunidade, características ideais para sistemas embarcados com recursos restritos como o do Aldebaran 1.

2.2. Arquitetura de Hardware e Software

A implementação do FSW está intrinsecamente ligada à plataforma de hardware do Aldebaran 1, cujos principais componentes são descritos a seguir.

- Plataforma de Hardware (Materiais): o "cérebro" do Aldebaran 1 é o subsistema OBDH, baseado no microcontrolador MSP430F6659 da Texas Instruments. A escolha desta família de microcontroladores foi reforçada por sua compatibilidade com a plataforma FloripaSat-2 e por seu histórico de uso em missões espaciais, que oferece um grau de robustez à radiação relevante para missões em órbita baixa, Larsen et al. (2016). A arquitetura de hardware, ilustrada na Figura 2, espelha a do FloripaSat-2, adotando um design com redundância de comunicação que emprega dois subsistemas de TTC idênticos (TTC_0 e TTC_1). O TTC_0 atua como receptor principal de telecomandos (uplink), enquanto o TTC_1 serve como transmissor principal de telemetria (downlink), ambos comunicando-se com o OBDH via barramento SPI.

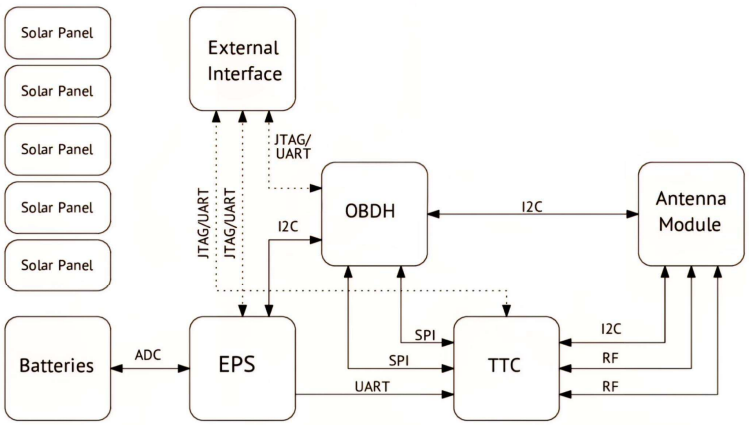

Figura 2. Diagrama de blocos do Aldebaran 1. Adaptado de Marcelino et al. (2024)

- Arquitetura de Software: a arquitetura do FSW foi estruturada em três camadas distintas para garantir modularidade e portabilidade, conforme ilustrado na Figura 3.
    a. Camada de Abstração de Hardware (HAL): a camada mais baixa, que interage diretamente com os periféricos do microcontrolador (SPI, I2C, UART, GPIOs). Ela fornece uma interface padronizada para as camadas superiores, de modo que o software da aplicação não precise conhecer os detalhes de baixo nível do hardware.
    b. Sistema Operacional de Tempo Real (RTOS): a camada intermediária, composta pelo FreeRTOS. É responsável pelo escalonamento de tarefas, gerenciamento de memória e mecanismos de comunicação inter-tarefas (filas, semáforos), garantindo o comportamento determinístico do sistema.
    c. Camada de Aplicação: a camada superior, onde as funcionalidades específicas da missão Aldebaran 1 são implementadas como tarefas independentes do RTOS. Isso inclui o gerenciamento do deploy, processamento de comandos, armazenamento de dados, entre outros.

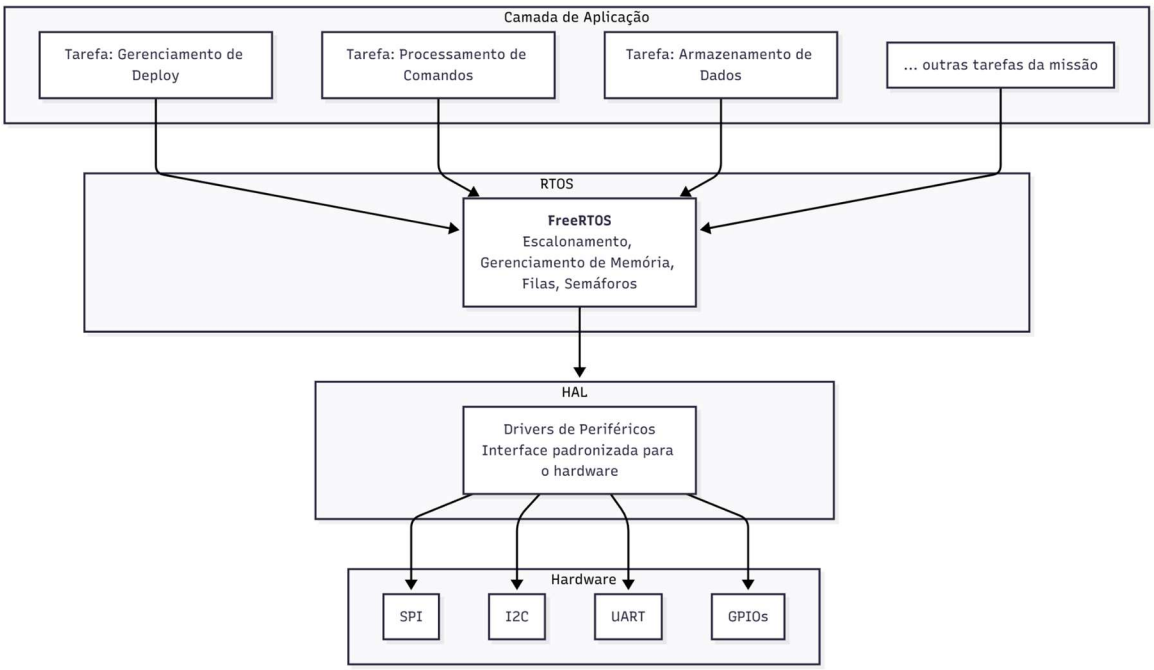

Figura 3. Hierarquia de software do Aldebaran 1. Feito pelo autor

2.3. Implementação das Funcionalidades do FSW

As funcionalidades primárias da missão foram implementadas como tarefas distintas na camada de aplicação.

2.3.1. Deployment Management

A lógica de deployment é uma das funcionalidades mais críticas da fase inicial da missão e é gerenciada por uma tarefa de alta prioridade do FreeRTOS, executada uma única vez na inicialização do satélite, após a liberação dos kill-switches. O fluxo de execução, ilustrado no fluxograma da Figura 4, segue uma sequência rigorosa para garantir a segurança da missão. A tarefa imediatamente ativa uma flag de "modo de hibernação" na memória, que restringe todas as operações de transmissão para evitar interferências. Em seguida, o estado das antenas é verificado.

Se os sensores indicarem que as antenas ainda não estão abertas, a tarefa inicia um temporizador de 45 minutos. Este período de espera é uma prática padrão em missões de CubeSats para permitir que o satélite se afaste do veículo lançador e de outros satélites ejetados na mesma órbita, minimizando o risco de colisões ou interferência de RF. Ao final da contagem, a rotina comanda a ativação dos mecanismos de queima (burn-wire), que liberam as antenas.

O sucesso da operação é então verificado através da leitura dos sensores de fim de curso (limit switches). Somente se uma transição de estado válida for detectada, indicando que as antenas estão totalmente abertas, a flag de hibernação é desativada na memória persistente, a tarefa de deployment é suspensa para liberar recursos de CPU, e o escalonador do RTOS inicia as demais tarefas do sistema. Em caso de falha no procedimento, o satélite permanece em modo de hibernação. Esta estratégia de fail-safe garante que, mesmo em caso de falha crítica no deployment, o satélite não se tornará uma fonte de interferência.

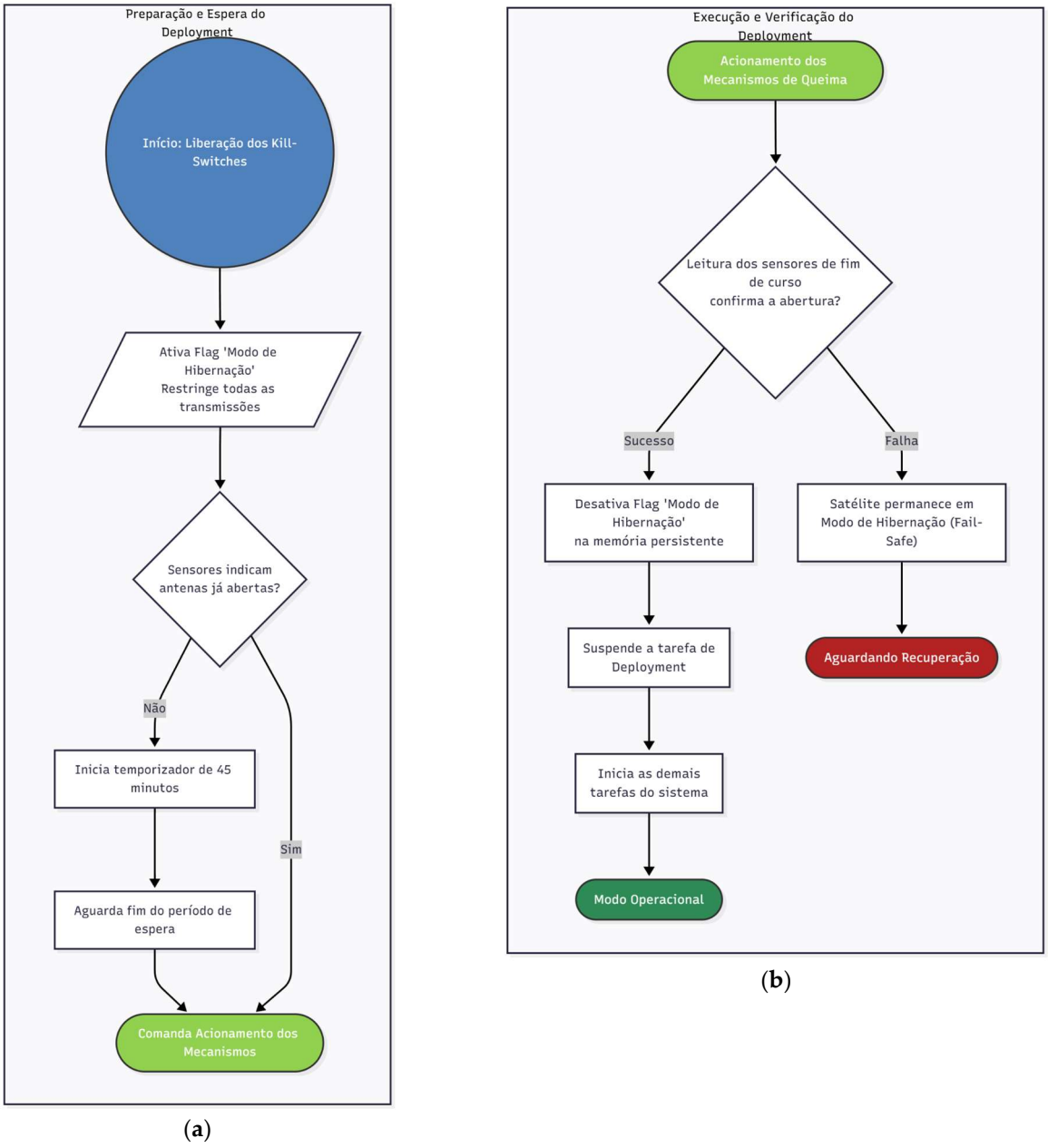

Figura 4. Fluxo de execução do deployment management: (**a**) Fluxo de preparação e espera do deployment; (**b**) Fluxo de execução e verificação do deployment. Feito pelo autor.

2.3.2. Processamento dos Telecomandos

O fluxo de processamento de telecomandos é o coração da interação do satélite com a estação de solo. Ele se inicia com a recepção de pacotes pelo subsistema TTC_o, que os retransmite ao OBDH via barramento SPI. Uma tarefa dedicada no OBDH aguarda por novos dados no barramento e, ao recebê-los, inicia um processo de validação em duas etapas para garantir a segurança e a integridade dos comandos:

1. Verificação de Identidade (ID): O primeiro byte do pacote é comparado com uma lista de IDs de comandos válidos. Se o ID não for reconhecido, o pacote é descartado imediatamente.
2. Autenticação de Mensagem (HMAC): Se o ID for válido, o FSW calcula um Código de Autenticação de Mensagem baseado em Hash (HMAC) sobre o conteúdo do pacote, utilizando uma chave secreta pré-compartilhada. O resultado é comparado com o HMAC recebido no pacote. Se os valores não corresponderem, o pacote é descartado, prevenindo a execução de comandos corrompidos ou maliciosos.

Se ambas as validações forem bem-sucedidas, a lógica correspondente ao comando é executada. Para fins de diagnóstico e rastreabilidade, o FSW armazena uma cópia do último comando validado (ID e callsign do remetente) na memória flash, permitindo sua recuperação posterior através de um comando específico. A resposta, estruturada com

o ID do comando original, o callsign "Aldebaran" e os dados de retorno, é então encaminhada ao TTC_1 para transmissão via downlink. Os principais telecomandos implementados e suas estruturas de pacotes estão detalhados na Tabela 1 e descritos a seguir.

Tabela 1. Comandos de telemetria implementados e suas estruturas de pacotes.

Comando	Estrutura do pacote
Ping	0x30 + Callsign[7]
Entrar em hibernação	0x43 + Callsign[7] + HMAC[20]
Sair da hibernação	0x32 + Callsign[7] + HMAC[20]
Receber dados dos barcos	0x4C + Callsign[7] + Localização[16] + HMAC[20]
Receber dados de PCBs	0x4D + Callsign[7] + PCB ID[1] + Dados[80] + HMAC[20]
Transmitir dados	0x53 + Callsign[7] + Sub-ID[1] + HMAC[20]
Configuração do rádio	0x51 + Callsign[7] + Tempo[1] + Potência[1] + HMAC[20]

- Ping (ID 0x30): é o comando mais simples, utilizado para verificar a conectividade do link de comunicação. Não requer HMAC, pois não altera o estado do sistema. Ao recebê-lo, o FSW apenas formula uma resposta de confirmação (acknowledgment) e a envia para o downlink.
- Entrar/Sair da Hibernação (IDs 0x43 e 0x32): estes comandos permitem que a estação de solo controle o estado de transmissão do satélite. O comando "Entrar em hibernação" ativa a flag de hibernação, que impede as transmissões do rádio. O comando "Sair da hibernação" reverte esse processo, retornando o satélite ao modo operacional normal.
- Receber Dados (Barco e PCB) (IDs 0x4C e 0x4D): estes comandos são o mecanismo de uplink para os dados da missão. O comando "Receber dados dos barcos" carrega um pacote de 16 bytes contendo dados de localização e o insere na fila de armazenamento persistente. O comando "Receber dados de PCBs" permite o uplink de um bloco de telemetria maior (80 bytes) de uma placa de circuito impresso específica, identificado pelo campo PCB ID.
- Transmitir Dados (ID 0x53): este é o comando mais versátil para o downlink de informações. Ele funciona como um multiplexador de telemetria, utilizando um sub-identificador em seu pacote para solicitar a transmissão de diferentes conjuntos de dados. Conforme o sub-ID fornecido, o FSW pode transmitir: (1) o conteúdo completo da fila de dados de embarcações, (2) o buffer de telemetria da PCB 1, (3) o buffer de telemetria da PCB 2, ou (4) o log do último telecomando executado.
- Configurar Rádio (ID 0x51): este comando é a principal ferramenta para o gerenciamento de energia. Ele permite que a estação de solo ajuste remotamente dois parâmetros cruciais do rádio de downlink: o nível de potência de transmissão e o intervalo de tempo entre as transmissões automáticas de beacon.

2.3.3. Armazenamento dos Dados de Missão

A persistência de dados críticos no Aldebaran 1 é assegurada por um sistema de gerenciamento de memória em duas camadas, detalhado na Tabela 2, que desacopla a operação em tempo real da escrita em flash. Na inicialização do sistema, todos os parâmetros de configuração e estado da missão são carregados da memória flash não volátil para uma estrutura de dados espelhada na RAM. O software de voo opera exclusivamente sobre esta cópia em RAM, garantindo acesso rápido e minimizando o desgaste da flash. Periodicamente, a cada 20 minutos, uma tarefa de baixo nível sincroniza o estado da RAM de volta para a flash, garantindo a persistência entre reinicializações.

Este modelo suporta estruturas de dados complexas, como a fila circular de 80 bytes implementada para armazenar os últimos cinco registros de localização de embarcações. Os ponteiros de controle da fila também são persistidos, assegurando sua integridade. Quando um comando de transmissão é recebido, todo o bloco de dados da fila é enviado, e a fila é subsequentemente reiniciada. A mesma estratégia de persistência é aplicada a outros parâmetros vitais, como o estado de hibernação, as configurações do rádio e o log do último telecomando executado, garantindo que o satélite sempre retome sua operação a partir de um estado conhecido e válido.

Tabela 2. Mapa da memória flash.

Segmento	Posição	Descrição
B	11	Flag de estado (operacional/hibernação)
B	12-14	Controle da fila (início, fim e contador)
B	15-22	Último telecomando válido (ID + callsign)

B	23-102	Fila dos dados de localização das embarcações (5 x 16 bytes)
C	84	Intervalo de transmissão dos beacons
C	85	Potência de transmissão do rádio
C	0-79	Buffer de telemetria da PCB 1
D	0-79	Buffer de telemetria da PCB 2

2.3.4. Gerenciamento de Energia

O Aldebaran 1 não implementa um sistema de gerenciamento de energia dinâmico complexo, como uma máquina de estados. A decisão de engenharia de adotar uma abordagem mais simples foi uma resposta pragmática aos desafios identificados durante os testes de integração, nomeadamente a imprecisão dos sensores de corrente e a sobrecarga do barramento SPI.

Portanto, o controle sobre o principal consumidor de energia — o rádio de downlink — é realizado via telecomando. O FSW permite que a estação de solo ajuste remotamente dois parâmetros cruciais: a potência de transmissão e o intervalo de tempo entre os beacons. Essa capacidade oferece uma forma indireta, porém eficaz e altamente confiável, de gerenciar o balanço energético da missão. Em cenários de baixa geração de energia, a estação de solo pode comandar uma redução na potência de transmissão ou aumentar o intervalo entre beacons, conservando energia de forma controlada e previsível.

2.4. Validação das Funcionalidades do FSW

A validação funcional da arquitetura de software foi realizada através de testes de sistema fim a fim. Esta abordagem permite a verificação da interação entre o software de voo, executando em seu hardware final (o modelo de engenharia), e a infraestrutura de comunicação em solo, representando um ciclo operacional completo. O ambiente de teste, ilustrado nas Figuras 5 e 6, foi composto por:

- Hardware Sob Teste (HUT): uma unidade de engenharia do OBDH do Aldebaran 1, idêntica à que será embarcada no satélite. O HUT foi integrado ao hardware do subsistema TTC para prover a interface de comunicação por rádio.
- Estação Terrestre de Desenvolvimento: para os testes iniciais e de depuração, foi desenvolvida uma estação de solo de pequeno porte, como mostra a Figura 5. Ela consiste em uma placa de desenvolvimento ESP32 e um rádio LoRa, desenvolvido na IDE Arduino que fornece uma interface gráfica para o envio de comandos e a visualização de telemetria em tempo real.
- Estação Terrestre de Grande Porte: para a validação final em um ambiente de maior fidelidade, os testes foram replicados utilizando a estação terrestre de grande porte da UFMA (Figura 6). Esta estação possui antenas de alto ganho e sistemas de rádio mais potentes, permitindo validar a comunicação em condições mais próximas às de uma passagem real do satélite.

A metodologia de teste consistiu na execução de um conjunto de procedimentos para validar os principais requisitos funcionais em ambos os ambientes de estação terrestre:

1. Teste de Ciclo de Vida Pós-Lançamento: Simulação da sequência completa desde a inicialização do HUT, passando pelo período de espera de 45 minutos, o comando de deploy das antenas e a transição para o modo operacional, com cada etapa sendo confirmada pela telemetria recebida.
2. Teste de Comunicação Fim a Fim: Envio sequencial de todos os telecomandos listados na Tabela 1 e verificação da integridade e correção das respostas recebidas. Foram incluídos testes com HMACs inválidos para confirmar que os pacotes eram corretamente descartados.
3. Teste de Persistência de Dados: Execução de comandos de armazenamento de dados, seguida por um ciclo de energia (power-cycle) do OBDH para confirmar que o estado anterior (configurações do rádio, conteúdo da fila, log do último TC) foi corretamente recuperado a partir da memória flash.

Os dados coletados durante os testes incluíram os logs de telemetria recebidos pela estação de solo e o estado da memória do microcontrolador, lido através de uma interface de depuração.

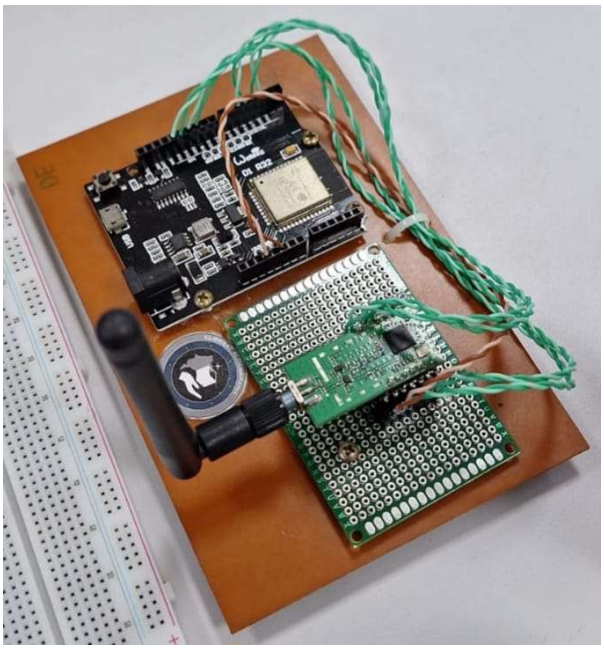

Figura 5. Estação terrestre de desenvolvimento. Feito pelo autor

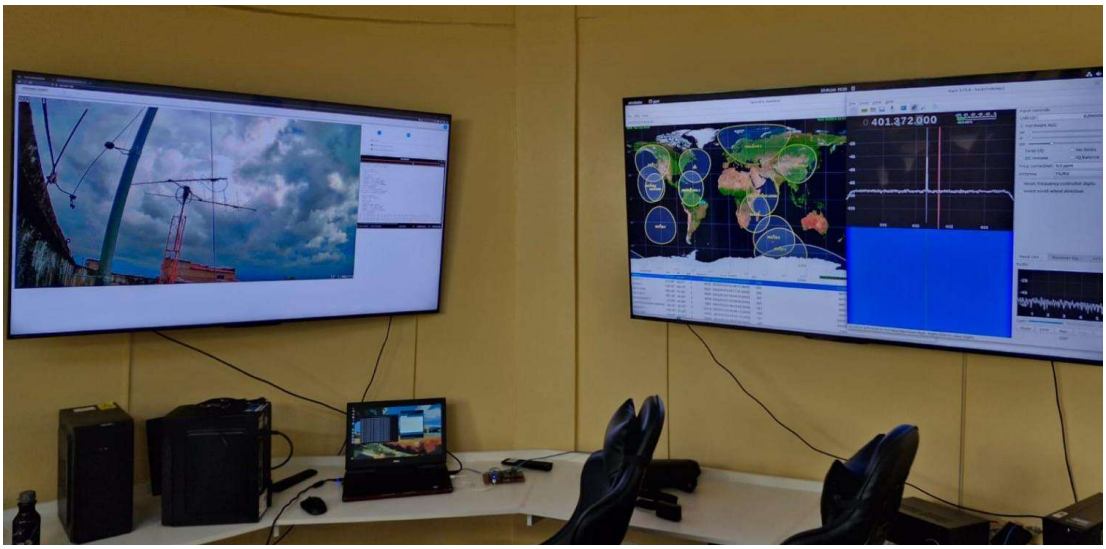

Figura 6. Estação terrestre de validação. Feito pelo autor

3. Resultados

Esta seção apresenta os resultados da validação funcional da arquitetura de software, obtidos através de testes de sistema fim a fim. Os testes foram conduzidos utilizando o modelo de engenharia do OBDH integrado ao subsistema TTC, que se comunicava via rádio com duas estações terrestres distintas: uma de desenvolvimento e uma de grande porte. Os resultados confirmaram o sucesso da implementação de todas as funcionalidades críticas do FSW, conforme sumarizado na Tabela 3. A seguir, os principais resultados são detalhados.

Tabela 1. Comandos de telemetria implementados e suas estruturas de pacotes.

Funcionalidade Testada	Critério de Sucesso	Resultado
Gerenciamento de Deploy	Cumprimento da cronologia de 45 min e confirmação de abertura.	Verificado
Processamento de Comandos	Resposta correta a todos os TCs, incluindo rejeição de HMAC inválido.	Verificado
Armazenamento (Fila de Dados)	Escrita e leitura correta da fila de 80 bytes.	Verificado

Funcionalidade Testada	Critério de Sucesso	Resultado
Persistência após Reset	Recuperação do estado completo após power-cycle.	Verificado

3.1. Validação do Ciclo de Vida Pós-Lançamento e Comunicação

A validação da sequência de deploy e da cadeia de comunicação foi realizada de forma integrada. O teste da rotina de deploy foi executado múltiplas vezes para garantir sua repetibilidade. Em todos os casos, após a inicialização do OBDH, a cronologia de 45 minutos foi verificada com sucesso, seguida pela confirmação de abertura das antenas, recebida via telemetria pela estação de solo. A transição da flag de "modo de hibernação" de ATIVADO para DESATIVADO foi observada na telemetria somente após a simulação dos sensores de fim de curso, validando a lógica de segurança implementada.

Após a transição para o modo operacional, a cadeia de comunicação foi validada pelo envio sequencial de todos os telecomandos listados na Tabela 1. As respostas recebidas pela estação de solo corresponderam exatamente ao formato e conteúdo esperados. Um ponto crucial da validação foi o teste da lógica de autenticação HMAC: comandos com HMACs deliberadamente corrompidos foram enviados, e observou-se que o FSW os descartou corretamente, não gerando resposta, o que confirma a segurança do protocolo de comando.

3.2. Validação do Armazenamento e Persistência de Dados

A validação da persistência de dados, um requisito crítico para a confiabilidade da missão, Bouwmeester et al. (2022), foi um ponto central dos testes. O comando para armazenar dados de embarcações foi enviado múltiplas vezes para preencher a fila circular na RAM. O estado da fila foi então verificado através do comando de transmissão de dados, que retornou o bloco de 80 bytes corretamente, com os dados na ordem esperada.

Em seguida, foi realizado um teste de ciclo de energia (power-cycle) no OBDH para validar a eficácia do modelo de gerenciamento de memória em duas camadas. Após a reinicialização do hardware, a estação de solo enviou comandos para verificar o estado do satélite. Observou-se que as configurações do rádio e o log do último telecomando executado foram corretamente recuperados da memória flash, correspondendo aos valores anteriores ao reset. Este resultado valida de forma conclusiva a capacidade do FSW de retomar sua operação a partir de um estado conhecido e válido, mesmo após uma falha de energia.

4. Discussão

Os resultados apresentados na seção anterior, obtidos por meio de testes de sistema fim a fim (end-to-end), confirmam que a arquitetura de software desenvolvida para o Aldebaran 1 atende aos requisitos funcionais da missão. Nesta seção, discutimos a interpretação desses resultados, suas implicações no contexto mais amplo do desenvolvimento de CubeSats acadêmicos e as direções para pesquisas futuras.

A estratégia de adaptar a plataforma de código aberto do FloripaSat-2, Marcelino et al. (2024), provou ser uma decisão metodológica acertada. Os resultados positivos na validação funcional em solo de funcionalidades críticas, como o gerenciamento de deploy e o processamento de comandos, demonstram que a reutilização de software com herança de voo é um caminho eficaz para acelerar o desenvolvimento e mitigar riscos em projetos com recursos limitados. Essa abordagem permitiu que a equipe de desenvolvimento focasse nos aspectos únicos da missão Aldebaran 1, como a implementação da fila de dados persistente, em vez de despender tempo reinventando a infraestrutura básica do FSW, uma prática que se alinha com as recomendações de modularidade e reuso destacadas por Avanzi et al. (2018).

Uma das descobertas mais significativas deste trabalho foi a validação da robustez do modelo de gerenciamento de memória em duas camadas. A capacidade do sistema de recuperar seu estado completo após um power-cycle, como demonstrado nos testes de persistência, é um resultado crucial. A falha na persistência de dados de estado é uma causa comum de anomalias em órbita, Bouwmeester et. (2022). A implementação bem-sucedida de uma fila circular persistente para dados de missão, utilizando um modelo que minimiza o desgaste da flash, representa uma contribuição prática para outros desenvolvedores que enfrentam desafios similares de armazenamento de dados em microcontroladores COTS.

Talvez a implicação mais importante deste trabalho reside na "lição aprendida" com o gerenciamento de energia. A tentativa inicial de implementar uma estratégia complexa foi frustrada por limitações identificadas durante a integração do hardware, um cenário comum em projetos de CubeSat que dependem de componentes COTS, Larsen et al. (2016). A decisão de pivotar para uma solução mais simples e robusta — o controle remoto do rádio — não deve ser vista como uma limitação, mas como uma validação de uma hipótese de trabalho pragmática: em sistemas críticos com recursos limitados, a simplicidade e a confiabilidade superam a complexidade. Este resultado contrasta com abordagens mais ambiciosas de gerenciamento autônomo de energia, mas oferece um modelo mais atingível e robusto para missões universitárias, onde a continuidade operacional é o objetivo principal.

Para trabalhos futuros, a arquitetura validada do Aldebaran 1 serve como uma base sólida (baseline). A próxima fase do projeto, a operação em órbita, fornecerá dados valiosos sobre o desempenho do software no ambiente espacial real, permitindo uma análise comparativa com os resultados dos testes em solo aqui apresentados.

5. Conclusão

Este artigo apresentou um estudo de caso completo sobre o desenvolvimento, implementação e validação da arquitetura de software de missão para o OBDH do CubeSat Aldebaran 1. A metodologia, baseada na adaptação da plataforma de código aberto do FloripaSat-2 com FreeRTOS, provou ser uma estratégia eficaz, permitindo acelerar o desenvolvimento e focar nos requisitos específicos da missão.

As principais funcionalidades do sistema — incluindo o gerenciamento de deploy, o processamento seguro de telecomandos e o armazenamento persistente de dados — foram implementadas com sucesso. A validação, conduzida através de testes de sistema fim a fim com estações terrestres, confirmou que a arquitetura de software atende de forma robusta a todos os requisitos funcionais, demonstrando sua prontidão para a fase operacional.

A principal contribuição deste trabalho é a demonstração de um ciclo de desenvolvimento pragmático e bem-sucedido, que serve como uma referência técnica e um conjunto de lições aprendidas para a comunidade acadêmica de CubeSats. Em particular, a decisão de simplificar a estratégia de gerenciamento de energia em favor de uma solução mais robusta e controlável destaca a importância da adaptação e do pragmatismo no contexto de projetos com recursos limitados.

O projeto Aldebaran 1 não apenas cumpre seus objetivos de missão, mas também estabelece uma base de conhecimento e uma plataforma de hardware e software reutilizável para futuras missões espaciais na UFMA. A validação final da arquitetura ocorrerá em órbita, o que permitirá uma análise aprofundada do desempenho do software em condições operacionais reais, consolidando as contribuições deste trabalho.

Financiamento: Esta pesquisa foi financiada pela Agência Espacial Brasileira (AEB).

Agradecimentos: Os autores gostariam de agradecer o suporte institucional fornecido pela Universidade Federal do Maranhão (UFMA). Somos particularmente gratos ao Laboratório de Eletrônica e Sistemas Embarcados Espaciais (LABESEE), que proveu o ambiente e os recursos para a concepção e o desenvolvimento deste trabalho. Sinceros agradecimentos também se estendem a todos os membros e colaboradores do projeto Aldebaran, cuja dedicação e esforço coletivo foram vitais para superar os desafios aqui apresentados.

Conflitos de Interesse: Os autores declaram não haver conflito de interesses. Os financiadores não tiveram nenhum papel no desenho do estudo; na coleta, análise ou interpretação dos dados; na redação do manuscrito; ou na decisão de publicar os resultados.

Referências

1. Cratere, A.; Gagliardi, L.; Sanca, G.; Golmar, F.; Dell'Olio, R. On-board computer for cubesats: State-of-the-art and future trends. IEEE Access 2024, 12, 99537–99569. doi:10.1109/ACCESS.2024.3412536.
2. Konecny, S.; Püschel, T.; Baturkin, V.; Dets, I. Development of a generic on-board software for small satellites. Acta Astronautica 2015, 117, 263–273. doi:10.1016/j.actaastro.2015.08.016.
3. Bouwmeester, J.; Guo, J.; Zandbergen, B. In-orbit data of cubesat subsystem reliability. IEEE Aerosp. Electron. Syst. Mag. 2022, 37, 4–17. doi:10.1109/MAES.2022.3179357.
4. Avanzi, A.; Noca, M.; Togni, G.; Ferrario, M. A modular software architecture for cubesat based on freertos. In Proceedings of the 2018 IEEE 5th International Workshop on Metrology for AeroSpace (MetroAeroSpace), Rome, Italy, 20-22 June 2018; pp. 40-45. doi:10.1109/MetroAeroSpace.2018.8467576.
5. Marcelino, G.M.; Schuck, G.P.; de Souza, J.C.; Bezerra, E.A. Floripasat-2: An open-source platform for cubesats. IEEE Embed. Syst. Lett. 2024, 16, 77–80. doi:10.1109/LES.2023.3323087.
6. Júnior, C.A.R.B.; et al. The cubesat mission aldebaran 1. In Proceedings of the IAA-LA CubeSat Workshop & IAA-LA Symposium on Small Satellites, Brasília, Brazil, 2022.
7. Silva, L.C.O.; et al. An experimental evaluation of long-range technology in a stratospheric sonde for educational cubesat mission aldebaran-1. In Proceedings of the IAA-LA CubeSat Workshop & IAA-LA Symposium on Small Satellites, Brasília, Brazil, 2022.
8. Larsen, J.S.; Jørgensen, J.L.; Christiansen, F. High-performance and reliable on-board computers for nano-satellites. J. Aerosp. Eng. 2016, 29, 04015019. doi:10.1061/(ASCE)AS.1943-5525.0000517.