

Cubesat da equipe TwoKósmos

Desenvolvimento de Cubesat para criação de GPS brasileiro

Kislan de Lima Souza
Gabriel Gonçalves Moreira
Letícia Cerbaro Barbosa
Arthur Alves Tavares
Credson Isaac Lopes dos Santos
2025

Resumo

Este projeto propõe o desenvolvimento de um protótipo satelital de um sistema nacional de georreferenciamento por meio de uma constelação de CubeSats, com o objetivo de reduzir a dependência do Brasil de tecnologias estrangeiras, especialmente em serviços estratégicos como o GPS. A iniciativa visa criar um "GPS brasileiro" acessível, eficiente e soberano, utilizando satélites em órbita heliossíncrona a 600 km de altitude, equipados com sensores embarcados, módulos de posicionamento ESP32, relógios OCXO (com viabilidade futura para relógios atômicos CSAC), comunicação via LoRaWAN, com pacotes de dados de sincronismo cronológico e medição de distância por propagação de onda portadora. O sistema operará com uma lógica de rede hierárquica baseada em *Cluster Tree*, com reconfiguração dinâmica e estratégias de hibernação e descarte seguro por drag sail, promovendo sustentabilidade orbital. A constelação deverá conter de 24 a 48 CubeSats para garantir cobertura total do território nacional além de todo continente americano. A equipe do IFRN – Campus Parnamirim – é composta por quatro estudantes e orientada por um professor-tutor, combinando conhecimentos em informática, eletrônica, escrita científica e simulação 3D. O projeto enfatiza a viabilidade técnica, eficiência energética, resiliência do sistema e independência tecnológica, representando um passo estratégico para a soberania e inovação espacial do Brasil.

Sumário

1. Declaração de problema da missão.....	1
2. Objetivos da missão e identificação do mérito científico.....	1
3. Funções e responsabilidades da equipe.....	2
4. Métodos.....	3
5. Detalhes operacionais.....	5
6. Requisitos e restrições do projeto.....	6
7. Cronograma preliminar de desenvolvimento e plano de trabalho.....	7
8. Descrição de todos os subsistemas essenciais e do subsistema de missão.....	8
8.1 Subsistema de Suprimento de Energia (EPS).....	8
8.2 Subsistema de Controle Térmico.....	8
8.3 Subsistema de Carga Útil (Payload).....	8
8.4 Subsistema de Controle de Bordo (OBC).....	9
8.5 Subsistema de Telemetria e Comunicação.....	9
8.6 Subsistema de Armazenamento de Dados.....	10
8.7 Subsistema Estrutural e Mecânico.....	10
9. Procedimento de montagem.....	10
10. Projeto mecânico.....	12
11. Projeto eletrônico.....	12
12. Fluxograma dos códigos desenvolvidos.....	13
13. Registro de dados.....	14
14. Procedimento de execução da missão.....	14
15. Dados coletados e transmitidos pela payload de missão.....	15
15.1 Payload 01.....	15
15.2 Payload 02.....	15
15.3 Payload 03.....	15
15.4 Payload 04.....	16
16. Descrição e resultados dos testes e simulações:.....	16
16.1 Caracterização física.....	16
16.2 Robustez mecânica.....	16
16.3 Robustez eletrônica e magnética.....	17
16.4 Robustez térmica.....	18
16.5 Captura de dados de telemetria.....	18
16.6 Captura de dados de missão.....	18
16.7 Armazenamento de dados.....	19
16.8 Transmissão de dados conforme descrito no apêndice 1 do edital.....	19
Referências.....	42

Lista de Figuras

Figura 1 - Vista frontal da estrutura mecânica do CubeSat.....	12
Figura 2 - Vista isométrica da estrutura mecânica do CubeSat.....	12
Figura 3 - Diagrama Elétrico do Cubesat.....	13
Figura 4 - Fluxograma do Código executado pelo ESP32.....	13
Figura 5 - Resultado de teste de envio de dados.....	14
Figura 6.1 - Estrutura Final do Cubesat.....	17
Figura 6.2 - Estrutura Final do Cubesat.....	17
Figura 7 - Imagem das Placas de Fenolites no Cubesat.....	17
Figura 8 - Diretório “logs” do armazenamento interno do ESP32 contendo os dados de telemetria captados.....	18
Figura 9 - Visualização dos dados de telemetria captados.....	19
Figura 10 - Exibição da conclusão da captura de dados telemétricos no console.....	19
Figura 11 - Resultado de teste de envio de dados.....	20

1. Declaração de problema da missão

Uma das principais problemáticas enfrentadas pelo Brasil atualmente é a dependência de tecnologias estrangeiras, que hoje representam vulnerabilidade estratégica. Com a crescente integração tecnológica às atividades econômicas, sociais e militares, o país torna-se suscetível a tensões geopolíticas e restrições comerciais. Um exemplo é o uso de serviços de internet via satélite, como o Starlink, controlado por empresa norte-americana, que pode expor informações sensíveis e reforçar a dependência externa (NAKAGAWA, s.d.).

Ademais, uma situação ainda mais crítica é a dependência dos sistemas GNSS estrangeiros, como *NAVSTAR-GPS*, *GLONASS*, *Galileo* e *BeiDou/BDS*, essenciais para agricultura de precisão, transporte, aviação e operações militares. Em cenários de conflito ou decisões políticas unilaterais, o acesso a esses sistemas pode ser restringido, comprometendo a segurança e a economia nacional.

A principal iniciativa brasileira nesse campo é o projeto GeoLocal, estruturado com bases fixas em solo, repetidora espacial e alvo móvel terrestre (MACHO, 2018; FRIOLANI; PAMBOUKIAN, 2020). Embora inovador, os testes apontaram limitações como menor precisão, forte dependência de infraestrutura terrestre, necessidade de comunicação ativa constante e ausência de posicionamento direto em tempo real ao usuário, o que reduz sua escalabilidade quando comparado ao GPS tradicional. que, apesar de tecnicamente mais caro, oferece mais precisão, autonomia, escalabilidade e suporte direto à navegação em tempo real.

Diante desse cenário, evidencia-se a necessidade de uma solução nacional acessível, eficiente e soberana. O desenvolvimento de um sistema brasileiro de navegação por satélite constitui passo estratégico para a autonomia tecnológica, a segurança nacional e o fortalecimento da soberania do país.

2. Objetivos da missão e identificação do mérito científico

Diante da crescente necessidade de independência tecnológica e segurança estratégica, o grupo TwoKósmos propõe o desenvolvimento de um sistema nacional de posicionamento baseado em CubeSats — pequenos satélites amplamente utilizados em comunicação, monitoramento ambiental e coleta de dados. A proposta consiste na criação de um sistema brasileiro de navegação por satélite, eficiente e acessível, capaz de atender às demandas do território nacional sem dependência de tecnologias estrangeiras.

O conceito possui respaldo técnico-científico consistente. Experiências como a constelação Xona PULSAR™ demonstram que satélites em órbita baixa (LEO) não apenas complementam os sistemas GNSS tradicionais, mas podem superá-los em intensidade de sinal, imunidade a interferências, precisão e rapidez de convergência de algoritmos como o PPP (Precise Point Positioning) (PISONI et al., 2025). Esses resultados reforçam a viabilidade do uso de CubeSats em um sistema nacional de navegação.

O grupo reconhece que, para assegurar funcionamento contínuo, precisão e cobertura eficaz, é indispensável a implementação de uma constelação de satélites. Como os satélites em órbita baixa (LEO) possuem campo de visão limitado e deslocam-se rapidamente sobre a superfície terrestre, a cobertura individual é restrita. Para que um receptor determine sua posição com precisão, é necessário receber sinais simultâneos de pelo menos quatro satélites — três para calcular latitude, longitude e altitude, e um quarto para corrigir o erro do relógio interno. Dessa forma, apenas uma rede coordenada com múltiplos satélites garante visibilidade constante em qualquer região, além de proporcionar redundância operacional, atualização mais frequente dos dados de navegação e maior robustez sistêmica. A constelação Xona PULSAR™ demonstra esse potencial ao alcançar precisão de até 11 cm e sinais 170 vezes mais intensos que os sistemas tradicionais (PISONI et al., 2025).

Com a execução da missão, serão estabelecidos os fundamentos de um sistema nacional denominado RPS — Regional Position System — baseado em órbitas polares heliosíncronas e cobertura regional. Trata-se de uma solução escalável e de menor custo, capaz de promover avanços na autonomia tecnológica brasileira e fortalecer a infraestrutura estratégica nacional, abrindo espaço para novas iniciativas científicas e comerciais no setor aeroespacial.

3. Funções e responsabilidades da equipe

A equipe TwoKósmos é formada por quatro estudantes do terceiro ano do Técnico Integrado em Informática do IFRN — Campus Parnamirim, orientados pelo Prof. Credson Isaac Lopes dos Santos, coordenador do CVT-E.

Integrante	Formação / Vínculo	Área de Conhecimento	Responsabilidade Principal
Kislan de Lima Souza	Técnico Informática IFRN	Informática, Astronomia, Modelagem 3D	Modelagem tridimensional do CubeSat e simulações gráficas estruturais
Gabriel Gonçalves Moreira	Técnico Informática IFRN	Informática, Escrita Científica	Elaboração dos relatórios técnicos de cada fase da olimpíada
Letícia Cerbaro Barbosa	Técnico Informática IFRN	Informática, Eletrônica	Desenvolvimento e construção do circuito eletrônico do CubeSat
Arthur Alves Tavares	Técnico Informática IFRN	Programação, Matemática	Software de controle do módulo de georreferenciamento e cálculos de posicionamento
Credson Isaac Lopes dos Santos	Professor Tutor Grad. Redes de Computadores	Redes, Computação, Gestão de Projetos	Orientação técnica e coordenação do projeto CVT-E

4. Métodos

O projeto propõe o desenvolvimento de um protótipo de sistema nacional de georreferenciamento baseado em uma constelação de CubeSats em órbita heliossíncrona. Esse tipo de órbita, sincronizada com o Sol, é amplamente utilizado em missões de observação da Terra devido à consistência temporal, versatilidade e ampla cobertura global (AEROENGENHARIA, s.d.; SANCHEZ, 2009), como exemplificado pelo satélite CBERS. A altitude média prevista é de 600 km, com inclinação entre 97° e 98°, garantindo estabilidade orbital, previsibilidade operacional e cobertura eficiente sobre o território brasileiro com regime regular de iluminação solar (SANCHEZ, 2009).

Por se tratar de um protótipo, serão utilizados componentes comerciais de prateleira com adaptações. Cada CubeSat será equipado com sensores BMP280 (pressão, temperatura e altitude estimada), MPU6050 (aceleração e rotação angular) e DHT22 (temperatura e umidade relativa), além de painéis solares e microcontrolador ESP32, responsável pela lógica de processamento e posicionamento. Esses parâmetros permitem determinar o estado do satélite, sua dinâmica orbital e fornecer dados auxiliares ao sistema de trilateração.

Além disso, outro componente essencial é o relógio, pois a posição é calculada a partir dos dados de tempo em que o sinal é enviado e recebido pelo

satélite. Entretanto, um relógio comum não é o suficiente, devido ele não ser preciso para os cálculos e seu uso pode ocasionar em um resultado equivocado. Por isso, o ideal seria utilizar relógios atômicos, reconhecidos por sua grande precisão na medição de tempo e sincronização em sistemas de navegação modernos. Um relógio atômico é, a princípio, composto por um mecanismo que oscila e outro que conta tais oscilações. Como os átomos absorvem e emitem luz em determinadas frequências, eles produzem um padrão de oscilação mais estável que qualquer mecanismo feito por seres humanos — isto reduz o risco de dessincronização que um relógio comum possui, que pode causar erros na medição temporal (NIST, s.d.; LUTWAK, 2009). O relógio atômico ideal seria o Chip-Scale Atomic Clock (CSAC), por oferecer uma ampla faixa de temperaturas de operação, aquecimento/travamento atômico rápidos e estabilidade de frequência superior em ambientes extremos — adequado para sistemas militares e industriais que exigem altíssima precisão e baixo consumo de energia (MICROCHIP, s.d.), os sistemas satelitais modernos sincronizam seus tempo de gps usando césio 133.

Entretanto, esse modelo possui um preço elevado de aproximadamente US\$ 3.000 a US\$ 10.000, visto isso, um modelo mais acessível seria o Oven-Controlled Crystal Oscillator que custa em torno de US\$ 70 e tem uma estabilidade de 10^{-8} a 10^{-9} por variação de temperatura, no entanto, deve-se destacar que esse modelo não é atômico, mas sim, baseado na oscilação do quartzo, que mesmo que não seja tão preciso quanto o atômico, é extremamente mais preciso que um relógio comum. Logo, é a opção com mais custo benefício para o projeto.

O detalhamento da arquitetura de comunicação, topologia de rede e critérios de dimensionamento encontra-se descrito no **Apêndice A**.

O tempo de sincronia do segmento espacial será dividido em dois contadores a fim de se ter o tempo sideral e o tempo GPS, ambos baseados no relógio de oscilação adquirido pela equipe e terá sua performance configurada com base nos sistemas satelitais pré-existentes contendo um contador aproximado do tempo UTC (Tempo Universal Coordenado), incluindo um contador de semanas — a partir da semana 0 — e um contador de segundos a ser sincronizado todas as semanas a partir do segmento de controle via acionamento remoto. O tempo sideral tem a função de georreferenciar o satélite no espaço sideral a partir do sincronismo com estações de controle terrestre, enquanto o tempo GPS é utilizado para localizar um objeto do segmento usuário na terra a partir do tempo de propagação do sinal

A alimentação será realizada por *pack* de baterias de íon-lítio em configuração 1S2P, com duas células Panasonic NCR18650B (3,7 V e 3400 mAh cada), totalizando 3,7 V, 6800 mAh e 25,16 Wh. A escolha se justifica pela elevada densidade energética, estabilidade térmica e confiabilidade.

O consumo estimado entre 400 e 600 mA proporciona autonomia de 11 a 17 horas. Em órbita heliossíncrona a 600 km, com ciclos médios de 58 minutos de iluminação e 35 minutos de sombra, painéis solares de GaAs com eficiência de 29,5% geram até 6,7 W líquidos, permitindo recarga completa em aproximadamente 3,75 horas. A adoção de hibernação parcial e *duty cycle* assegura autossustentação energética, ainda sujeita à validação em órbita.

5. Detalhes operacionais

O sistema é baseado em uma constelação de CubeSats em órbita heliossíncrona, entre 500 e 800 km de altitude. Nessa configuração, cada satélite pode cobrir aproximadamente 3,5 milhões de km² por passagem, com um raio de visada em solo de cerca de 1.050 km. Considerando que o território nacional possui mais de 8,5 milhões de km² e que a geolocalização exige visibilidade simultânea de ao menos quatro satélites por ponto, estima-se a necessidade de uma constelação composta por 24 a 48 CubeSats, distribuídos em 3 a 4 planos orbitais, com 8 a 12 satélites por plano. Essa arquitetura assegura cobertura contínua e confiável.

A implantação pode ocorrer de forma escalonada, não sendo obrigatória a disponibilidade inicial de todos os satélites. Em fase preliminar, balões estratosféricos podem ser utilizados para simular o ambiente operacional e validar subsistemas como telemetria, comunicação LoRa, sincronização temporal e algoritmos de posicionamento, além de testar a estrutura em rede e o desempenho dos sensores em condições reais.

A constelação opera sob lógica hierárquica em clusters. Cada grupo possui um nó coordenador responsável pela organização da rede local, controle de tráfego e sincronização temporal. O coordenador emite sinais periódicos de anúncio (ADV), enquanto os demais satélites confirmam sua operacionalidade por meio de pacotes de reconhecimento (ACK). Cada cluster mantém ainda um vice-coordenador, previamente definido com base em critérios de desempenho e disponibilidade energética. Em caso de degradação do coordenador principal, ocorre transição

automática de liderança, garantindo continuidade operacional. O satélite degradado passa a exercer função secundária, reduzindo sua carga computacional e energética.

O sistema contempla também um protocolo de descarte seguro para unidades em falência funcional irreversível. Nesses casos, é acionada uma vela de arrasto (*drag sail*), dispositivo que se desdobra aumentando a área exposta ao arrasto atmosférico em órbita terrestre baixa, reduzindo gradualmente a altitude até a reentrada abaixo de aproximadamente 200 km, evitando a geração de detritos espaciais (ORBITAL TRANSPORTS, s.d.). Missões como o a CanX-7, que implantou quatro velas de aproximadamente 1 m² e antecipou significativamente a sua reentrada (EO PORTAL, s.d.-a), e o InflateSail, que com vela de cerca de 10 m² reduziu o tempo de reentrada para aproximadamente 72 dias (EO PORTAL, s.d.-b), demonstram a eficácia desse mecanismo passivo e de baixo custo.

Adicionalmente, os CubeSats operam com modo automatizado de hibernação energética. Ao ultrapassarem o território nacional, sensores e transmissores são temporariamente desativados, permanecendo ativos apenas os subsistemas essenciais de controle e geração de energia. Essa estratégia permite a recarga das baterias a partir das placas solares até o retorno à área de interesse.

Esses mecanismos — reconfiguração de liderança, descarte controlado e hibernação energética — garantem robustez, autonomia e maior vida útil à constelação, mesmo diante das limitações inerentes ao ambiente orbital.

Por fim, a Fase 3 tem como objetivo testar a acurácia do relógio de oscilação na manutenção do sincronismo e na viabilização do cálculo da distância entre o segmento espacial e o segmento usuário com base no tempo de propagação do sinal. Também será validado o envio remoto de comandos semanais para reset do contador de segundos do relógio atômico, transmitidos do segmento de controle ao segmento espacial durante a operação em órbita.

6. Requisitos e restrições do projeto

As condições ambientais em órbita impõem desafios significativos. Os CubeSats estarão sujeitos a variações extremas de temperatura, exigindo componentes com alta tolerância térmica e estabilidade de frequência. Além disso, como a alimentação depende exclusivamente dos painéis solares, é indispensável

otimizar o consumo energético por meio de estratégias como modo de repouso automático quando o satélite estiver fora da zona de interesse geográfico.

A vida útil limitada dos CubeSats impõe requisitos adicionais quanto à continuidade do sistema. Para assegurar sua funcionalidade ao longo do tempo, devem ser adotadas estratégias de substituição autônoma de satélites coordenadores por vice-coordenadores previamente definidos, bem como mecanismos seguros de descarte orbital que evitem a geração de detritos espaciais.

A maioria dos dispositivos microSD comerciais não resiste ao ambiente espacial, sendo necessária a utilização de modelos industriais com maior tolerância à radiação, temperatura e ciclos de escrita, como o Swissbit Industrial microSD, com capacidades de 4 a 32 GB, operação entre -40°C e $+85^{\circ}\text{C}$, suporte a correção de erros (ECC) e alta resistência mecânica, compatível com o ESP32 (SWISSBIT, s.d.).

O sistema deve garantir autossustentação energética em órbita heliossíncrona, considerando ciclos de 58 minutos de sol e 35 minutos de sombra. A recarga das baterias deve ocorrer em aproximadamente 3,75 horas de exposição solar, com implementação de *duty cycle* e hibernação parcial para manter apenas subsistemas essenciais ativos.

Outra restrição refere-se à resolução temporal do relógio de oscilação (casas decimais), que influencia diretamente a precisão do cálculo do tempo de propagação dos sinais entre os segmentos espacial, usuário e controle.

7. Cronograma preliminar de desenvolvimento e plano de trabalho

O cronograma da missão foi elaborado com base em critérios técnicos e logísticos, contemplando a disponibilidade de materiais, acesso ao equipamento e o tempo para a execução de cada etapa do projeto. Conforme detalhado no **Apêndice B**, a tabela cronológica organiza o desenvolvimento das fases compatíveis com o calendário da OBSAT, especificando datas, atividades, entregas parciais e responsabilidades referentes a cada membro da equipe, com o objetivo de garantir um fluxo de trabalho eficiente e coordenado.

A equipe, vinculada ao Instituto Federal de Educação, Ciência e Tecnologia do Rio Grande do Norte (IFRN), dispõe de facilidades institucionais que favorecem a execução do projeto em âmbitos como a obtenção de materiais para a produção do CubeSat, diminuindo o tempo necessário para a realização de algumas etapas e acesso a infraestrutura técnica. Entre os recursos disponíveis, destaca-se a

impressora 3D, possibilitando maior agilidade nas etapas de fabricação e montagem do CubeSat.

O planejamento também leva em consideração o tempo de prototipação e programação, dimensionando com base nas competências técnicas atuais dos integrantes. Dessa forma, o cronograma foi estruturado de modo realista e adaptável, visando o cumprimento integral das metas propostas até o final do ano.

8. Descrição de todos os subsistemas essenciais e do subsistema de missão

O CubeSat desenvolvido no âmbito deste projeto é composto por um conjunto de subsistemas integrados, projetados para garantir a operação contínua, a aquisição confiável de dados e a comunicação eficiente com a estação terrestre. Cada subsistema desempenha uma função específica dentro da arquitetura da missão, assegurando a execução adequada dos objetivos propostos.

8.1 Subsistema de Suprimento de Energia (EPS)

O subsistema de suprimento de energia fornece alimentação elétrica contínua a todos os componentes do CubeSat por meio de uma bateria de íon-lítio em configuração 1S2P. Ademais, estratégias de economia de energia, como hibernação parcial, *duty cycle* inteligente e painéis fotovoltaicos, serão implementadas para maximizar a vida útil do sistema e assegurar a autossustentação energética do CubeSat. A representação 3D da placa deste subsistema encontra-se no **Apêndice G**.

8.2 Subsistema de Controle Térmico

O controle térmico da missão é realizado de forma passiva, por meio da utilização de materiais termoestáveis e isolantes. A aplicação de fita Kapton sobre os componentes eletrônicos reduz a influência de variações térmicas abruptas, enquanto o isolamento interno contribui para manter os dispositivos dentro das faixas operacionais seguras. Essa abordagem é compatível com as limitações energéticas e estruturais do CubeSat, garantindo estabilidade térmica ao longo do ciclo operacional.

8.3 Subsistema de Carga Útil (Payload)

O subsistema de carga útil da missão é dividido em plano de controle e plano de dados.

O plano de controle compreende o conjunto de informações obtidas por sensores ambientais e inerciais responsáveis pela aquisição dos dados experimentais. O sensor BMP280 realiza medições de pressão atmosférica, temperatura e estimativa de altitude. O sensor DHT22 coleta dados de umidade relativa do ar e temperatura. O MPU6050 fornece informações de aceleração linear e rotação angular. Esses dados permitem avaliar o comportamento dinâmico do CubeSat e caracterizar as condições ambientais durante a operação da missão. O plano de controle também incorpora dados provenientes do segmento de controle, especificamente as informações de tempo de propagação obtidas pelas estações terrestres fixas.

O plano de dados é responsável pelo georreferenciamento. Ele inclui informações do contador de tempo em segundos e da semana GPS, fornecendo subsídios para a validação dos algoritmos de posicionamento e sincronização temporal empregados pelo sistema de georreferenciamento sideral e global, aplicado neste projeto em escala regional.

Além disso, este subsistema incorpora também os circuitos de temporização e sincronização, fundamentais para aplicações de georreferenciamento. A arquitetura prevê a utilização de um oscilador de cristal controlado por forno (OCXO), que oferece elevada estabilidade de frequência e permite a validação dos algoritmos de posicionamento. O sistema foi concebido de forma modular, possibilitando a futura integração de um relógio atômico do tipo Chip-Scale Atomic Clock (CSAC), ampliando a precisão temporal e a robustez do sistema sem a necessidade de alterações estruturais significativas no controle de bordo. A representação 3D da placa deste subsistema encontra-se no **Apêndice H**.

8.4 Subsistema de Controle de Bordo (OBC)

O subsistema de controle de bordo é centralizado no microcontrolador ESP32, responsável pela coordenação global das operações do CubeSat. Entre suas funções estão o gerenciamento da carga útil, o controle dos modos de operação, o processamento dos dados coletados e a otimização do consumo energético. A representação 3D da placa deste subsistema encontra-se no **Apêndice I**.

8.5 Subsistema de Telemetria e Comunicação

O subsistema de telemetria e comunicação é responsável pela organização, transmissão e recepção das informações da missão. Os dados coletados pela carga

útil são enviados ao microcontrolador ESP32 por meio do barramento I²C, estruturados em pacotes de telemetria e transmitidos via módulo de comunicação LoRa E32-433T20D, que oferece comunicação de longo alcance com baixo consumo energético.

Este subsistema compreende também a estação terrestre, responsável pela recepção, decodificação, visualização e armazenamento dos pacotes transmitidos pelo CubeSat. A estação terrestre é composta por módulos de comunicação LoRa integrados a 4 microcontroladores ESP32 e a um sistema computacional, permitindo o monitoramento da missão, a validação da integridade dos dados recebidos e o envio das informações. A representação 3D da placa deste subsistema encontra-se no **Apêndice J**.

8.6 Subsistema de Armazenamento de Dados

O armazenamento local das informações da missão é realizado por meio de um cartão microSD, conectado ao microcontrolador ESP32. Esse subsistema permite o registro contínuo dos dados coletados pela carga útil, assegurando redundância em situações de indisponibilidade temporária do enlace de comunicação. O uso de armazenamento local contribui para a confiabilidade do sistema e para a preservação dos dados experimentais.

8.7 Subsistema Estrutural e Mecânico

A estrutura mecânica do CubeSat foi fabricada por impressão 3D em filamento ABS e tem como função acomodar, proteger e organizar os componentes eletrônicos da missão. O projeto estrutural garante alinhamento adequado, estabilidade mecânica e integração entre os subsistemas. Adicionalmente, é integrada uma vela de arrasto destinada ao descarte controlado ao final da missão, contribuindo para a mitigação de detritos orbitais e atendendo às boas práticas de sustentabilidade espacial.

9. Procedimento de montagem

Nesta etapa do projeto, realizou-se a montagem funcional dos subsistemas do CubeSat com o objetivo de validar a operação individual e integrada dos componentes eletrônicos. Paralelamente, a estrutura mecânica definitiva do satélite foi modelada em ambiente de software Blender e posteriormente fabricada por meio de impressão 3D utilizando filamento ABS. Com a placa de circuito impresso (PCB) já finalizada, os testes passaram a ser realizados diretamente na própria placa,

proporcionando maior fidelidade ao sistema final e permitindo uma avaliação mais precisa da integração entre os componentes.

Inicialmente, foi realizada a validação individual de cada sensor da carga útil. Cada dispositivo — incluindo sensores ambientais e inerciais — foi conectado separadamente ao microcontrolador ESP32, de modo a verificar seu funcionamento, comunicação via barramento apropriado e correta aquisição dos dados. Essa etapa permitiu identificar possíveis falhas de conexão, incompatibilidades de alimentação ou erros de leitura antes da integração completa do sistema.

Em seguida, foi testado o subsistema de telemetria e comunicação de forma isolada. Para isso, estabeleceu-se a comunicação entre dois módulos LoRa, configurados para transmissão e recepção de dados, a fim de validar o enlace de comunicação, o alcance funcional e a integridade das mensagens trocadas entre os dispositivos.

Após a validação individual dos sensores e do enlace de comunicação, procedeu-se à integração dos subsistemas. Nessa etapa, os dados coletados pelos sensores foram processados pelo microcontrolador ESP32 e encapsulados em pacotes de telemetria, que passaram a ser transmitidos pelo módulo LoRa. A estação receptora, também montada em bancada, foi utilizada para confirmar o recebimento correto das informações enviadas, validando o fluxo completo de aquisição, processamento e transmissão dos dados da missão.

Esse procedimento de montagem em bancada e integração funcional permitiu verificar o comportamento conjunto dos subsistemas, identificar eventuais inconsistências e garantir a confiabilidade do sistema antes da transição para a etapa de montagem estrutural definitiva do CubeSat e do desenvolvimento da placa eletrônica dedicada.

No que se refere à temporização e sincronização do sistema, destaca-se que o relógio de oscilação previsto na arquitetura final ainda não foi obtido. Como solução provisória para viabilizar os testes iniciais e a validação dos algoritmos de georreferenciamento, está sendo utilizado um contador interno para a geração das marcações temporais durante os experimentos em bancada. Essa abordagem permite a continuidade do desenvolvimento e a verificação do funcionamento dos processos de aquisição e transmissão de dados (Visualizar código do contador no **Apêndice D**). Em etapas futuras do projeto, pretende-se integrar uma interface de programação de aplicações (API) externa para obtenção de referências temporais mais precisas,

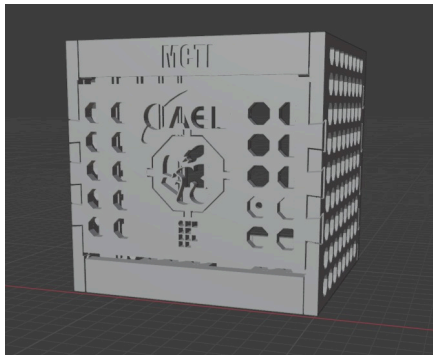
faproximando o comportamento do sistema das condições previstas para a operação com o relógio de oscilação no controle de bordo até adquirir o mesmo.

10. Projeto mecânico

A estrutura mecânica do CubeSat será fabricada por meio de impressão 3D, utilizando filamento do tipo ABS, conforme o modelo apresentado nas Figuras 1 e 2. Esse material foi selecionado por apresentar boa resistência mecânica, estabilidade dimensional e facilidade de fabricação, características adequadas para protótipos e testes experimentais.

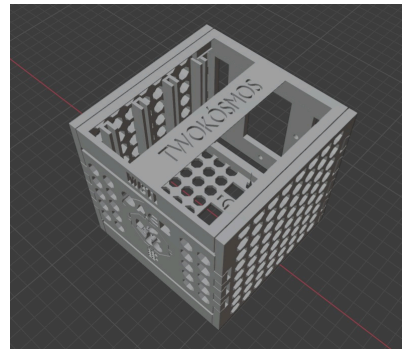
No interior da carcaça, serão integradas placas estruturais produzidas com o mesmo material, cuja função será organizar, fixar e separar os diferentes subsistemas do CubeSat. Essa disposição interna permitirá uma montagem modular, garantindo alinhamento adequado dos componentes, proteção mecânica e maior confiabilidade estrutural ao longo da missão.

Figura 1 - Vista frontal da estrutura mecânica do CubeSat



Fonte: autoria própria.

Figura 2 - Vista isométrica da estrutura mecânica do CubeSat



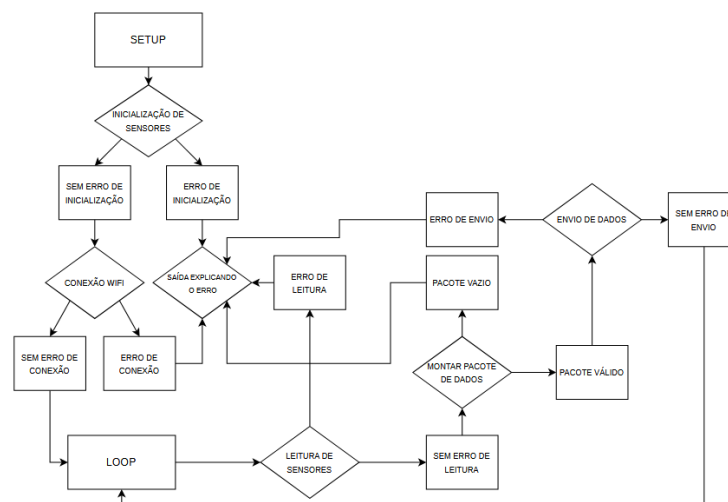
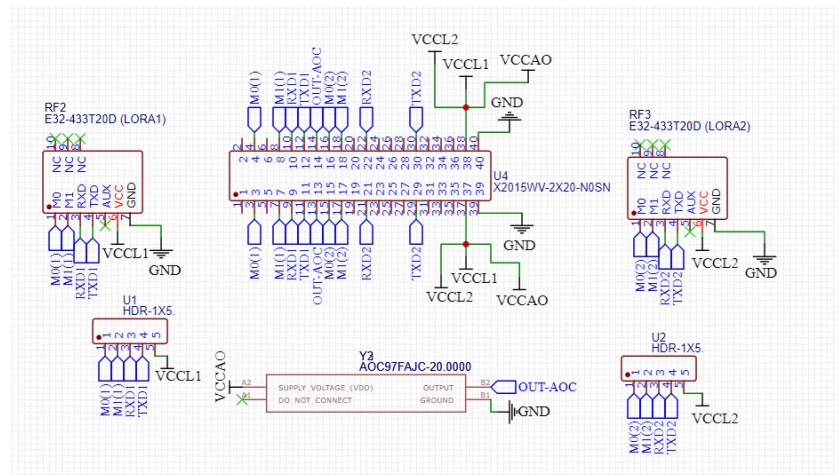
Fonte: autoria própria.

11. Projeto eletrônico

O sistema eletrônico do CubeSat foi projetado para garantir a integração funcional entre os subsistemas e a validação progressiva da missão. Atualmente, os componentes já estão implementados em uma placa de circuito impresso (PCB), permitindo testes mais próximos da configuração final e maior confiabilidade na validação do sistema.

A alimentação proveniente da bateria é direcionada ao ESP32, que atua como unidade central de controle, gerenciando energia, processamento de dados e

O módulo de rádio E32-433T20D é integrado por interface UART (TX/RX), possibilitando a transmissão dos dados processados pelo ESP32 via rádio, validando o fluxo completo de aquisição, processamento e envio das informações durante os testes da missão.



13.Registro de dados

Após a aquisição dos dados pelos sensores da carga útil, as informações são processadas pelo microcontrolador ESP32 e registradas no armazenamento local por meio de um cartão microSD. Paralelamente, os dados são organizados em pacotes de telemetria e transmitidos via módulo LoRa e encaminhadas ao servidor teste da OBSAT (https://obsat.org.br/teste_post/index.php), conforme os procedimentos estabelecidos pela organização. A Figura 5 ilustra o registro dos dados enviados e recebidos no servidor de teste da OBSAT.

Figura 5 - Resultado de teste de envio de dados

Servidor de teste - OBSAT 									
Essa página irá mostrar todas as requisições enviadas para o endereço https://obsat.org.br/servidor_testes/envio_bipes.php . Ou https://obsat.org.br/servidor_testes/envio.php (envio sem o bipes). Caso essa requisição apresentar algum erro, ele será exibido. Os dados dessa página serão apagados a cada 1 hora. Caso você encontre algum erro, relate-o no servidor.									
Equipe	Bateria	Temperatura	Pressao	Giroscopio	Acelerometro	Payload	Datetime	Tamanho	Status
25	3.901978	31.1	1013.25	0.05,0.06,0.07	0.01,0.02,9.81	[3.901978,31.1,76.1,1013.25]	2026-04-03 17:31:52	16	Sucesso!
25	3.901978	31.1	1013.25	0.05,0.06,0.07	0.01,0.02,9.81	[3.901978,31.1,76.8,1013.25]	2026-04-03 17:31:42	16	Sucesso!
25	3.872967	31.1	1013.25	0.05,0.06,0.07	0.01,0.02,9.81	[3.872967,31.1,75.7,1013.25]	2026-04-03 17:31:32	16	Sucesso!
25	3.905201	31.1	1013.25	0.05,0.06,0.07	0.01,0.02,9.81	[3.905201,31.1,75.3,1013.25]	2026-04-03 17:31:23	16	Sucesso!

Fonte: autoria própria.

14.Procedimento de execução da missão

A execução da missão foi estruturada para garantir a operação integrada dos subsistemas, a aquisição confiável de dados e a validação do método de geolocalização, tanto em cenário ideal quanto em testes em solo.

Após a energização, o ESP32 inicializa sensores, comunicação, armazenamento e temporização. O sistema então entra em modo de aquisição periódica, registrando medições ambientais e inerciais associadas a marcas temporais. Os dados são processados e armazenados no cartão microSD.

Em seguida, são organizados em pacotes de telemetria e transmitidos via módulo LoRa. O instante de recepção é registrado, permitindo a análise da diferença de tempo entre envio e recebimento do sinal.

A geolocalização baseia-se na relação $d = c \cdot \Delta t$, em que c representa a velocidade de propagação do sinal eletromagnético e Δt o tempo de propagação medido. Com múltiplas medições ou estações, aplica-se a técnica de diferença de tempo de chegada dos sinais, refinando a estimativa de posição. Os dados inerciais e ambientais atuam como suporte para validação e consistência dos resultados obtidos.

Durante a operação, são empregados protocolos de economia de energia, incluindo hibernação parcial, definição de duty cycle e priorização de modos de baixo consumo, otimizando o uso da energia disponível. Na fase atual, a sincronização temporal utiliza referências externas para validação contínua dos algoritmos.

15. Dados coletados e transmitidos pela payload de missão

Os dados da missão estão organizados em quatro tipos de pacotes (payloads): um destinado ao cumprimento dos requisitos da Olimpíada OBSAT e três voltados à prova de conceito do funcionamento do RPS.

15.1 Payload 01

O Payload 01 também é transmitido em direção descendente, do segmento espacial para o segmento usuário, por meio do módulo LoRa E32-433T20D na frequência de 435 MHz. Esse pacote contém o identificador do satélite, a semana GPS e o tempo GPS em segundos, ou ciclos do relógio de oscilação. Essas informações permitem ao segmento usuário calcular a distância ao satélite com base no tempo de propagação do sinal, viabilizando o georreferenciamento regional.

15.2 Payload 02

O Payload 02 é transmitido em direção descendente do segmento espacial para o segmento de controle, utilizando o módulo LoRa na frequência de 440 MHz. Ele inclui também inclui o identificador do satélite, junto dos dados ambientais e inerciais coletados pelos sensores BMP280, DHT22 e MPU6050, organizados em pacotes padronizados com informações sensoriais e temporais. Esse payload tem como finalidade o monitoramento do sistema e a manutenção dos clusters do segmento espacial, também responsável pelas informações relacionadas a vida útil e manutenção da rede satelital.

15.3 Payload 03

O Payload 03 corresponde à comunicação em direção ascendente, do segmento de controle para o segmento espacial. Esse pacote consiste em comandos remotos enviados semanalmente com o objetivo de sincronizar o sistema, incluindo o identificador da estação do segmento controle e o reset do contador do relógio atômico embarcado. Além disso, esse comando também é responsável pela

sincronização periódica do sistema temporal, reiniciando o contador de segundos e mantendo a referência de semanas GPS iniciada na semana 0.

15.4 Payload 04

O Payload 04 corresponde à comunicação em direção descendente, do segmento espacial para o controle da missão OBSAT, sendo transmitido via protocolo Wi-Fi. Esse pacote contém dados ambientais, físicos e inerciais utilizados na estimativa de localização do CubeSat ao longo da operação. As medições são realizadas pelos sensores BMP280, responsável por pressão atmosférica, temperatura e estimativa de altitude; DHT22, que mede umidade relativa e temperatura; e MPU6050, que fornece dados de aceleração linear e rotação. Após a aquisição, os dados são encaminhados ao microcontrolador ESP32 para processamento. Fluxograma dos payloads e Sistema Satelital no **Apêndice E**.

Destaca-se que, na fase atual do projeto, a marcação temporal é realizada com base em referências externas por meio de API. Na configuração final da missão, está prevista a integração de um relógio atômico do tipo Chip-Scale Atomic Clock (CSAC), que proporcionará maior estabilidade temporal e redução das incertezas associadas ao processo de localização, também responsável pelas informações relacionadas a vida útil e manutenção da rede satelital.

16. Descrição e resultados dos testes e simulações:

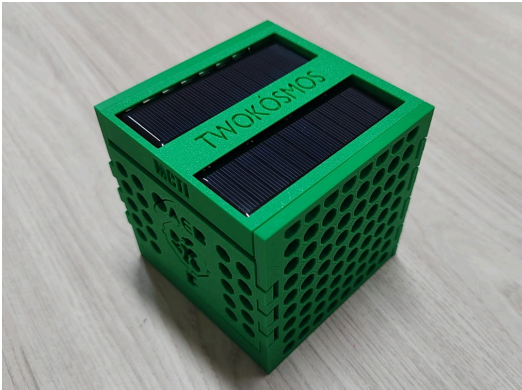
16.1 Caracterização física

A versão final do CubeSat desenvolvido obteve 1U de volume e 350 gramas de massa.

16.2 Robustez mecânica

A robustez mecânica do CubeSat foi garantida por um projeto estrutural simples e eficiente, desenvolvido em modelagem tridimensional, com previsão de fabricação em ABS por impressão 3D, conforme ilustrado nas Figuras 6.1 e 6.2. O material foi selecionado por sua resistência e estabilidade, suportando vibrações e pequenos impactos durante manuseio, transporte e testes. A estrutura interna mantém os componentes eletrônicos firmemente fixados e organizados, enquanto as placas internas reforçam o conjunto, assegurando a integridade e a confiabilidade do sistema ao longo da missão.

Figura 6.1 - Estrutura Final do Cubesat



Fonte: autoria própria

Figura 6.2 - Estrutura Final do Cubesat

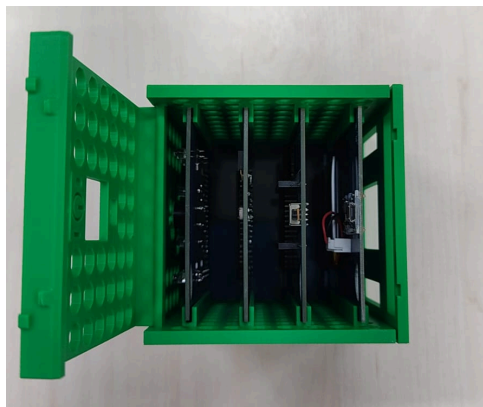


Fonte: autoria própria.

16.3 Robustez eletrônica e magnética

A montagem dos componentes eletrônicos em múltiplas placas de fenolite perfuradas garante a fixação adequada dos circuitos, reduzindo falhas de conexão e curtos-circuitos, conforme representado por meio da Figura 7. Os componentes foram distribuídos entre essas placas de acordo com seus respectivos subsistemas, favorecendo a organização, manutenção e integração do sistema. O uso do fenolite, aliado à aplicação de fita Kapton sobre os componentes, contribui para o isolamento térmico, atenuando os efeitos de variações bruscas de temperatura durante a operação. A soldagem direta dos condutores à placa assegura maior estabilidade elétrica e mecânica, aumentando a confiabilidade do CubeSat ao longo da missão.

Figura 7 - Imagem das Placas de Fenolites no Cubesat



Fonte: autoria própria.

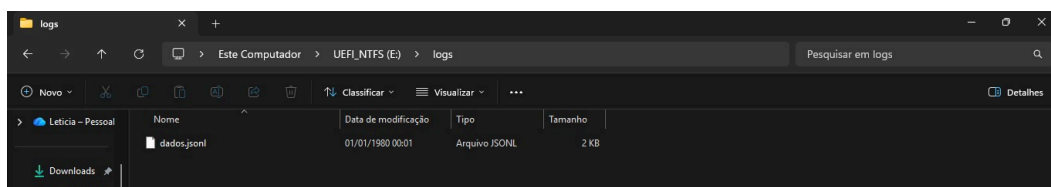
16.4 Robustez térmica

A robustez térmica do CubeSat será garantida por um sistema de controle térmico passivo, baseado no uso de materiais isolantes e termoestáveis. A fita Kapton aplicada sobre os componentes eletrônicos reduz os efeitos de variações térmicas abruptas e protege os circuitos contra gradientes de temperatura. Internamente, placas de depron atuam como isolante adicional, diminuindo as trocas térmicas com o ambiente externo. Essa configuração mantém os módulos eletrônicos dentro de uma faixa segura de operação, assegurando estabilidade e confiabilidade ao longo da missão.

16.5 Captura de dados de telemetria

Os testes de captura de dados de telemetria validaram o fluxo completo de aquisição, organização e envio das informações coletadas pelos sensores BMP280, DHT22 e MPU6050. As medições ambientais, inerciais e o nível de carga da bateria são organizados pelo ESP32 em formato JSONL e encaminhados via Wi-Fi ao servidor de testes da OBSAT. Durante os testes, os dados recebidos apresentaram valores coerentes com as grandezas físicas medidas, sem falhas relevantes no processo de aquisição ou transmissão, conforme ilustrado nas Figuras 8.

Figura 8 - Diretório “logs” do armazenamento interno do ESP32 contendo os dados de telemetria captados

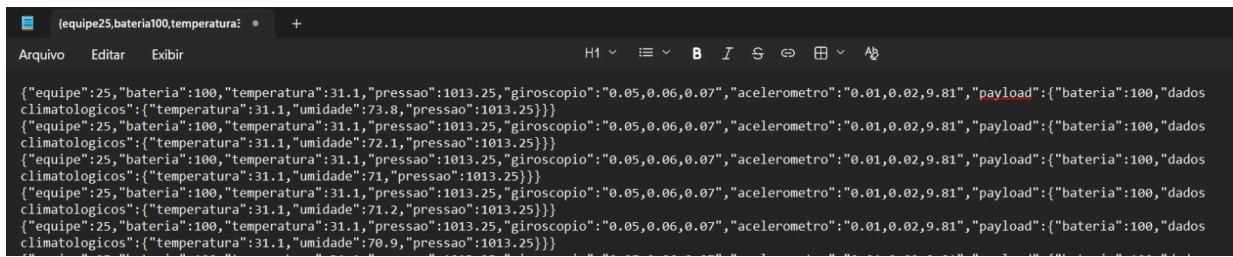


Fonte: autoria própria.

16.6 Captura de dados de missão

Os dados de missão correspondem diretamente aos dados de telemetria coletados pela carga útil do CubeSat, uma vez que as medições ambientais e inerciais constituem a base para os cálculos de localização propostos pelo experimento. Durante os testes, essas informações foram capturadas com sucesso por meio dos sensores BMP280, DHT22 e MPU6050. As leituras de temperatura, pressão atmosférica, umidade, aceleração e rotação foram registradas de forma contínua. Os dados capturados mostraram-se consistentes e adequados para a análise do experimento proposto, conforme apresentado na Figura 9 abaixo.

Figura 9 - Visualização dos dados de telemetria captados

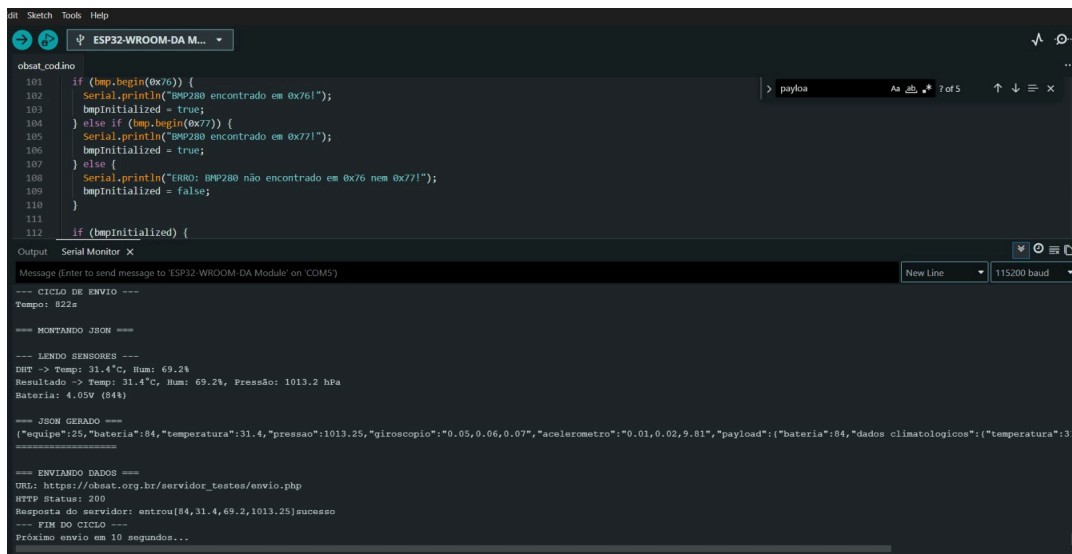


Fonte: autoria própria.

16.7 Armazenamento de dados

O armazenamento dos dados de missão, após capturados pelos sensores, é realizado de forma contínua no cartão microSD conectado ao microcontrolador ESP32. O processo de armazenamento e a correta gravação dos dados podem ser verificados por meio dos registros exibidos durante os testes e pela recepção das informações na estação terrestre, como ilustrado na Figura 10 a seguir.

Figura 10 - Exibição da conclusão da captura de dados telemétricos no console



Fonte: autoria própria.

16.8 Transmissão de dados conforme descrito no apêndice 1 do edital

A transmissão dos dados de telemetria do CubeSat TwoKósmos, realizada por meio de requisições HTTP ao servidor de testes da OBSAT, obteve status “Sucesso!”, conforme a Figura 11, em conformidade com o Apêndice 1 do edital, com o envio dos campos de identificação da equipe, nível de bateria e dados ambientais e inerciais da missão. A transmissão e recepção dos demais payloads do sistema RPS dependem da entrega do relógio de oscilação síncrona; na fase atual, os testes são realizados com dados simulados.

Figura 11 - Resultado de teste de envio de dados

Servidor de teste - OBSAT 										
Essa página irá mostrar todas as requisições enviadas para o endereço https://obsat.org.br/servidor_testes/envio_bipes.php . Ou https://obsat.org.br/servidor_testes/envio.php (envio sem o bipes). Caso essa requisição apresentar algum erro, ele será exibido. Os dados dessa página serão apagados a cada 1 hora. Caso você encontre algum erro, relate-o no servidor.										
Equipe	Bateria	Temperatura	Pressao	Giroscopio	Acelerometro	Payload	Datetime	Tamanho	Status	
25	3.901978	31.1	1013.25	0.05,0.06,0.07	0.01,0.02,9.81	[3.901978,31.1,76.1,1013.25]	2026-04-03 17:31:52	16	Sucesso!	
25	3.901978	31.1	1013.25	0.05,0.06,0.07	0.01,0.02,9.81	[3.901978,31.1,76.8,1013.25]	2026-04-03 17:31:42	16	Sucesso!	
25	3.872967	31.1	1013.25	0.05,0.06,0.07	0.01,0.02,9.81	[3.872967,31.1,75.7,1013.25]	2026-04-03 17:31:32	16	Sucesso!	
25	3.905201	31.1	1013.25	0.05,0.06,0.07	0.01,0.02,9.81	[3.905201,31.1,75.3,1013.25]	2026-04-03 17:31:23	16	Sucesso!	

Fonte: autoria própria.

17. Lista de materiais

Cubesat:

- Microcontrolador ESP32 (controle de bordo)
- Sensores:
 - BMP280 (pressão, temperatura e altitude)
 - DHT22 (umidade e temperatura)
 - MPU6050 (acelerômetro e giroscópio)
- 2x Módulo de comunicação LoRa E32-433T20D
- Cartão microSD (armazenamento local de dados)
- Bateria Lítio-Íon
- Placas Fotovoltaicas
- Placas de fenolite perfuradas para montagem dos circuitos
- Fita Kapton para proteção e isolamento térmico
- Estrutura mecânica impressa em 3D (filamento ABS)
- Vela de arrasto para descarte controlado

Base Terrestre:

- 4x E32-433T20D Módulo Lora
- 4x ESP32

Apêndices

Apêndice A

A metodologia adotada será baseada no protocolo Cluster Tree, observável na Figura A.1, em que há um nó coordenador e nós de roteamento intermediário (sub-coordenadores), podendo cada nó Full Function Device (FFD) comunicar-se com aglomerados de nós de função reduzida, Reduced Function Devices (RFD), e formar conjuntos conhecidos como cluster, em que, de maneira hierárquica, um nó coordenador assume a posição de raiz desta árvore (SANTOS, 2017).

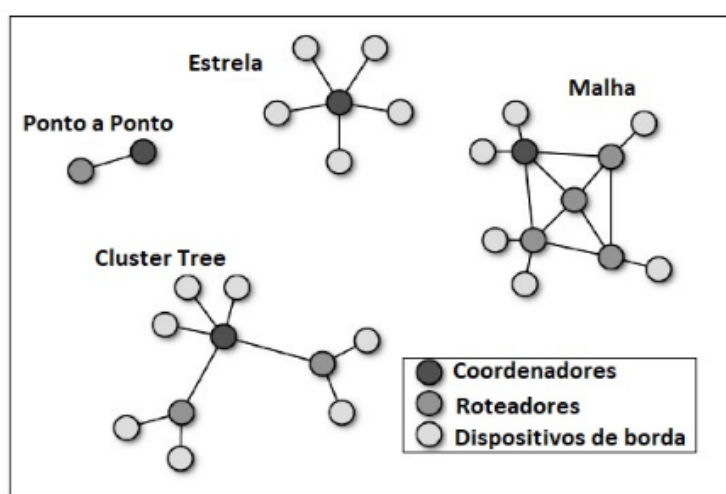
A comunicação entre os CubeSats e os nós terrestres será baseada em LoRaWAN (Long Range Wide Area Network), um protocolo de comunicação sem fio e de longo alcance e baixo consumo energético — ideal para redes de sensores distribuídos. O controle da rede será gerenciado por meio de sinais ADV (Advertisement), que são pacotes enviados pelos coordenadores para anunciar sua presença e organização da rede, e ACK (Acknowledgment), que são mensagens de confirmação de recebimento enviadas pelos nós após a recepção bem-sucedida de dados.

O desenvolvimento do sistema de georreferenciamento por CubeSats envolve uma série de requisitos técnicos e restrições operacionais que orientam suas escolhas de projeto. Em primeiro lugar, há limitações orçamentárias que restringem o uso de componentes de alto custo, como os relógios atômicos CSAC, cuja elevada precisão seria ideal para a sincronização temporal, mas cujo preço inviabiliza sua adoção em larga escala. Diante disso, opta-se por osciladores de cristal controlados por forno (OCXO), que, apesar de menos precisos, oferecem uma relação mais equilibrada entre desempenho e viabilidade econômica.

Do ponto de vista da comunicação, o sistema será derivado do protocolo LoRaWAN, cuja compatibilidade com a topologia Cluster Tree permite estabelecer uma rede hierárquica eficiente, com baixo consumo energético e maior resistência a interferências. Será empregado o transceptor LoRa SX127x, capaz de operar com alta sensibilidade e atingir picos de até 140 mA durante a transmissão, o que exige atenção ao dimensionamento energético do satélite. Para otimizar o tráfego de

dados em um ambiente com severas restrições de largura de banda, o protocolo 6LoWPAN será adotado, permitindo a compressão e fragmentação de pacotes IPv6, o que viabiliza a comunicação eficiente em redes de baixa taxa de transmissão. Finalmente, para garantir a qualidade da geolocalização oferecida, o sistema deverá apresentar alta disponibilidade e precisão de posicionamento, sendo os resultados avaliados por meio do coeficiente PDOP (Position Dilution of Precision), que estima a acurácia com base na geometria relativa dos satélites em relação ao receptor.

Figura A.1



Fonte: Adaptada (FALUDI, 2010).

Imagem de **SANTOS, C. I. L.** *Dependabilidade aplicada a protocolo hierárquico baseado no LEACH*. UFERSA, 2013. Dissertação de Mestrado.

Apêndice B

Cronograma Anual da Missão e Projeto de Trabalho			
Fases da OBSAT	Etapas	Responsabilidades	Datas
Fase 0	Estudos de conceitos e tecnologias e Formulação do Objetivo.	Toda a Equipe	24 de Fevereiro a 30 de Setembro de 2025
Fase 1	Produção e envio do relatório.	Toda a Equipe	
Fase 2	Pesquisa sobre sistemas GNSS, constelações de CubeSats/Definição da arquitetura do sistema regional de georreferenciamento.	- Pesquisa sobre sistemas GNSS, constelações de CubeSats: Leticia Cerbaro e Kislán Souza. - Definição da arquitetura do sistema regional de	01 de Novembro a 30 de Novembro

		georreferenciamento: Arthur Tavares e Gabriel Gonçalves.	
	Desenvolvimento de placa de carga útil com relógio atômico e rádio transmissor Lora/Escolha dos componentes e sensores a serem implementados.	- Desenvolvimento de placa de carga útil com relógio atômico e rádio transmissor Lora: Leticia Cerbaro. - Escolha dos componentes e sensores a serem implementados: Kislân Souza.	1 de Dezembro a 31 de Dezembro
	Prototipação da estrutura do CubeSat por impressão 3D/Testes dos sensores para validação de funcionamento.	- Testes dos sensores para validação de funcionamento: Leticia Cerbaro. - Prototipação da estrutura do CubeSat por impressão 3D: Kislân Souza.	01 de Janeiro a 31 de Janeiro
	Testes de sensores, validação da comunicação entre módulos LoRa e integração dos subsistemas para transmissão de dados de telemetria para a estação terrestre/Envio do projeto.	- Testes de sensores, validação da comunicação entre módulos LoRa e integração dos subsistemas para transmissão de dados de telemetria para a estação terrestre: Arthur Tavares e Leticia Cerbaro. - Envio do projeto: Toda a equipe.	01 de Fevereiro a 04 de Abril
Fase 3	Lançamento - evento regional.	Toda a Equipe	04 de Abril a 24 de Maio
	Produção de relatório atualizado com dados do lançamento regional	Toda a Equipe	24 de Maio a Agosto
Fase 4	Adaptação do CubeSat	- Montagem do CubeSat: Leticia Cerbaro e Kislân Souza. - Telemetria: Arthur Tavares e Gabriel Gonçalves. - Testes de funcionamento: Toda a equipe.	

	Lançamento sub-orbital - evento nacional	Toda a Equipe	Entre Outubro e Novembro de 2026
--	---	---------------	---

Apêndice C

```
#include <WiFi.h>
#include <HTTPClient.h>
#include <ArduinoJson.h>
#include <Wire.h>
#include <Adafruit_Sensor.h>
#include <Adafruit_BMP280.h>
#include <DHT.h>
#include <MPU6050_tockn.h>

// =====
// CONFIGURAÇÕES
// =====

const char* SSID = "Familia";
const char* PASSWORD = "Luzia7278";
const char* SERVER_URL = "https://obsat.org.br/servidor_testes/envio.php";

unsigned long week = 0;
unsigned long lastweek = 0;
const unsigned long weektime = 604800000;

const int EQUIPE = 25;
const unsigned long SEND_INTERVAL_MS = 10000;

// =====
// DEFINIÇÕES DOS SENSORES
// =====

// DHT
#define DHTPIN 2
#define DHTTYPE DHT22
#define BATT_PIN 35 // Seu pino de bateria

DHT dht(DHTPIN, DHTTYPE);

// BMP280
Adafruit_BMP280 bmp;
#define BMP_I2C_ADDRESS 0x76 // Tente 0x76 se 0x77 não funcionar

// MPU6050
MPU6050 mpu6050(Wire);
```

```

// =====
// VARIÁVEIS GLOBAIS
// =====

unsigned long lastSend = 0;
bool bmpInialized = false;
bool mpulnialized = false;
bool dhtInialized = false;

// =====
// INICIALIZAÇÃO DOS SENSORES
// =====

void initSensors() {
  Serial.begin(115200);
  delay(2000); // Aguarda inicialização do Serial
  Serial.println("\n=== INICIALIZANDO SENSORES ===");

  // Inicializa I2C
  Wire.begin(21, 22); // SDA=21, SCL=22 para ESP32
  delay(100);
  Serial.println("I2C inicializado");

  // Teste comunicação I2C com dispositivos comuns
  Serial.println("\n--- SCAN I2C ---");
  byte error, address;
  int nDevices = 0;

  for(address = 1; address < 127; address++) {
    Wire.beginTransmission(address);
    error = Wire.endTransmission();

    if (error == 0) {
      Serial.print("Dispositivo I2C encontrado no endereço 0x");
      if (address < 16) Serial.print("0");
      Serial.print(address, HEX);

      // Identifica dispositivos conhecidos
      if (address == 0x68) Serial.println(" -> Provavelmente MPU6050");
      else if (address == 0x76 || address == 0x77) Serial.println(" -> Provavelmente BMP280");
      else if (address == 0x27 || address == 0x3F) Serial.println(" -> Provavelmente LCD");
      else Serial.println(" -> Desconhecido");

      nDevices++;
    }
  }
}

```

```

if (nDevices == 0) {
  Serial.println("NENHUM dispositivo I2C encontrado!");
}
Serial.println("--- FIM SCAN ---\n");

// Inicializa DHT
dht.begin();
dhtInitialized = true;
Serial.println("DHT inicializado");

// Inicializa BMP280 - TENTA AMBOS ENDEREÇOS
if (bmp.begin(0x76)) {
  Serial.println("BMP280 encontrado em 0x76!");
  bmpInitialized = true;
} else if (bmp.begin(0x77)) {
  Serial.println("BMP280 encontrado em 0x77!");
  bmpInitialized = true;
} else {
  Serial.println("ERRO: BMP280 não encontrado em 0x76 nem 0x77!");
  bmpInitialized = false;
}

if (bmpInitialized) {
  bmp.setSampling(Adafruit_BMP280::MODE_NORMAL,
                  Adafruit_BMP280::SAMPLING_X2,
                  Adafruit_BMP280::SAMPLING_X16,
                  Adafruit_BMP280::FILTER_X16,
                  Adafruit_BMP280::STANDBY_MS_500);
}

// Inicializa MPU6050
// Tenta detectar se o MPU6050 está presente
Wire.beginTransmission(0x68);
byte mpuError = Wire.endTransmission();

if (mpuError == 0) {
  Serial.println("MPU6050 detectado no endereço 0x68");
  mpu6050.begin();

  // Pequena pausa antes da calibração
  delay(1000);

  Serial.println("Calibrando MPU6050 (não mexa o sensor por 3 segundos)...");
  mpu6050.calcGyroOffsets(true); // true = mostra progresso no Serial
  mpuInitialized = true;
  Serial.println("MPU6050 calibrado e pronto!");
} else {
  Serial.println("ERRO: MPU6050 não encontrado no endereço 0x68");
  mpuInitialized = false;
}

```

```

}

Serial.println("\n=== STATUS DOS SENSORES ===");
Serial.print("DHT22: ");
Serial.println(dhtInitialized ? "OK" : "FALHA");
Serial.print("BMP280: ");
Serial.println(bmpInitialized ? "OK" : "FALHA");
Serial.print("MPU6050: ");
Serial.println(mpuInitialized ? "OK" : "FALHA");
Serial.println("=====\n");
}

// =====
// LEITURA DA BATERIA REAL
// =====

float readBatteryVoltage() {
    int leituraADC = analogRead(BATT_PIN);
    float voltage = (leituraADC / 4095.0) * 3.3 * 2.0;

    // Mapeia: 4.2V = 100% | 3.2V = 0%
    float porcentagem = (voltage - 3.2) / (4.2 - 3.2) * 100.0;

    //Travas de segurança para não sair de 0-100
    if (porcentagem > 100.0) porcentagem = 100.0;
    if (porcentagem < 0.0) porcentagem = 0.0;

    Serial.print("Bateria: ");
    Serial.print(voltage, 2);
    Serial.print("V (");
    Serial.print((int)porcentagem);
    Serial.println("%)");

    // Retorna a porcentagem para o JSON
    return porcentagem;
}

// =====
// LEITURA DOS SENSORES
// =====

bool readDHT(float &temp, float &hum) {
    hum = dht.readHumidity();
    temp = dht.readTemperature();

    if (isnan(hum) || isnan(temp)) {
        Serial.println("Falha na leitura do DHT!");
        return false;
    }
}

```

```

Serial.print("DHT -> Temp: ");
Serial.print(temp, 1);
Serial.print("°C, Hum: ");
Serial.print(hum, 1);
Serial.println("%");
return true;
}

bool readBMP(float &temp, float &pres) {
    if (!bmpInitialized) {
        Serial.println("BMP280 não inicializado!");
        return false;
    }

    temp = bmp.readTemperature();
    pres = bmp.readPressure() / 100.0; // Pa para hPa

    if (isnan(temp) || isnan(pres)) {
        Serial.println("Falha na leitura do BMP280!");
        return false;
    }

    Serial.print("BMP -> Temp: ");
    Serial.print(temp, 1);
    Serial.print("°C, Pressão: ");
    Serial.print(pres, 1);
    Serial.println(" hPa");
    return true;
}

bool readMPU(float &ax, float &ay, float &az, float &gx, float &gy, float &gz) {
    if (!mpuInitialized) {
        Serial.println("MPU6050 não inicializado!");
        return false;
    }

    mpu6050.update();
    ax = mpu6050.getAccX();
    ay = mpu6050.getAccY();
    az = mpu6050.getAccZ();
    gx = mpu6050.getGyroX();
    gy = mpu6050.getGyroY();
    gz = mpu6050.getGyroZ();

    Serial.print("MPU -> Acel: [");
    Serial.print(ax, 2); Serial.print(", ");
    Serial.print(ay, 2); Serial.print(", ");
    Serial.print(az, 2); Serial.print("], Giro: [");

```

```

Serial.print(gx, 2); Serial.print(", ");
Serial.print(gy, 2); Serial.print(", ");
Serial.print(gz, 2); Serial.println("]");

return true;
}

// =====
// LEITURA COMPLETA DOS SENSORES
// =====

bool readAllSensors(float &temp, float &hum, float &pres,
                    float &ax, float &ay, float &az,
                    float &gx, float &gy, float &gz) {

    Serial.println("\n--- LENDO SENSORES ---");

    bool dhtOK = false;
    bool bmpOK = false;
    bool mpuOK = false;

    // Tenta ler DHT
    float tempDHT = NAN, humDHT = NAN;
    if (dhtInitialized) {
        dhtOK = readDHT(tempDHT, humDHT);
    }

    // Tenta ler BMP
    float tempBMP = NAN, presBMP = NAN;
    if (bmpInitialized) {
        bmpOK = readBMP(tempBMP, presBMP);
    }

    // Tenta ler MPU
    float accX = 0, accY = 0, accZ = 0, gyroX = 0, gyroY = 0, gyroZ = 0;
    if (mpuInitialized) {
        mpuOK = readMPU(accX, accY, accZ, gyroX, gyroY, gyroZ);
    }

    // Combina leituras (prioriza DHT para temperatura/humidade)
    if (dhtOK) {
        temp = tempDHT;
        hum = humDHT;
    } else if (bmpOK) {
        temp = tempBMP;
        hum = 50.0; // Valor padrão se DHT falhar
    } else {
        temp = 25.0; // Valor de fallback
        hum = 50.0;
    }
}

```

```

}

if (bmpOK) {
    pres = presBMP;
} else {
    pres = 1013.25; // Pressão padrão ao nível do mar
}

if (mpuOK) {
    ax = accX; ay = accY; az = accZ;
    gx = gyroX; gy = gyroY; gz = gyroZ;
} else {
    // Valores simulados para MPU
    ax = 0.01; ay = 0.02; az = 9.81;
    gx = 0.05; gy = 0.06; gz = 0.07;
}

Serial.print("Resultado -> Temp: ");
Serial.print(temp, 1);
Serial.print("°C, Hum: ");
Serial.print(hum, 1);
Serial.print("%, Pressão: ");
Serial.print(pres, 1);
Serial.println(" hPa");

// Considera OK se pelo menos um sensor funcionou
return dhtOK || bmpOK;
}

// =====
// MONTAGEM DO JSON
// =====

String montarJson() {
    Serial.println("\n=== MONTANDO JSON ===");

    StaticJsonDocument<2048> doc;

    // Variáveis para os dados dos sensores
    float temperatura, umidade, pressao;
    float accX, accY, accZ;
    float gyroX, gyroY, gyroZ;

    // Lê dados dos sensores
    bool leituraOK = readAllSensors(temperatura, umidade, pressao,
                                    accX, accY, accZ,
                                    gyroX, gyroY, gyroZ);

    if (!leituraOK) {

```

```

Serial.println("AVISO: Usando valores simulados para todos sensores");
temperatura = 25.5;
umidade = 60.0;
pressao = 1013.25;
accX = 0.01; accY = 0.02; accZ = 9.81;
gyroX = 0.05; gyroY = 0.06; gyroZ = 0.07;
}

// Lê tensão da bateria
float bateria = readBatteryVoltage();

// ORDEM EXATA EXIGIDA PELO SERVIDOR
doc["equipe"] = EQUIPE;
doc["bateria"] = (int)bateria;
doc["temperatura"] = temperatura;
doc["pressao"] = pressao;

// Giroscópio e Acelerômetro como strings
char giroStr[30], acelStr[30];
snprintf(giroStr, sizeof(giroStr), "%.2f,%.2f,%.2f", gyroX, gyroY, gyroZ);
snprintf(acelStr, sizeof(acelStr), "%.2f,%.2f,%.2f", accX, accY, accZ);

doc["giroscopio"] = giroStr;
doc["acelerometro"] = acelStr;

// PAYLOAD
JsonObject payload = doc.createNestedObject("payload");
payload["bateria"] = (int)bateria;

JsonObject clim = payload.createNestedObject("dados climatologicos");
clim["temperatura"] = temperatura;
clim["umidade"] = umidade;
clim["pressao"] = pressao;

// SERIALIZAÇÃO FINAL
String json;
serializeJson(doc, json);

Serial.println("\n=== JSON GERADO ===");
Serial.println(json);
Serial.println("=====");

return json;
}

// =====
// CONEXÃO WIFI
// =====

```

```

void conectaWifi() {
    Serial.println("\n=== CONECTANDO WIFI ===");
    Serial.print("SSID: ");
    Serial.println(SSID);

    WiFi.begin(SSID, PASSWORD);

    int tentativas = 0;
    while (WiFi.status() != WL_CONNECTED && tentativas < 20) {
        delay(500);
        Serial.print(".");
        tentativas++;
    }

    if (WiFi.status() == WL_CONNECTED) {
        Serial.println("\nWiFi CONECTADO!");
        Serial.print("IP: ");
        Serial.println(WiFi.localIP());
    } else {
        Serial.println("\nERRO: WiFi NÃO CONECTADO!");
    }
}

// =====
// ENVIO HTTP
// =====

void enviar(const String& json) {
    if (json.length() == 0) {
        Serial.println("ERRO: JSON VAZIO!");
        return;
    }

    // Verifica WiFi
    if (WiFi.status() != WL_CONNECTED) {
        Serial.println("WiFi desconectado, reconectando...");
        conectaWifi();
        if (WiFi.status() != WL_CONNECTED) {
            Serial.println("Não foi possível reconectar. Pulando envio.");
            return;
        }
    }

    HTTPClient http;

    http.begin(SERVER_URL);
    http.addHeader("Content-Type", "application/json");
    http.setTimeout(10000);

```

```

Serial.println("\n=== ENVIANDO DADOS ===");
Serial.print("URL: ");
Serial.println(SERVER_URL);

int code = http.POST(json);

Serial.print("HTTP Status: ");
Serial.println(code);

if (code > 0) {
    String resposta = http.getString();
    Serial.print("Resposta do servidor: ");
    Serial.println(resposta);
} else {
    Serial.print("Erro no envio: ");
    Serial.println(http.errorToString(code));
}

http.end();
}

// =====
// SETUP / LOOP
// =====

void setup() {
    delay(2000); // Aguarda inicialização

    lastweek = millis();

    Serial.println("\n\n=== INICIO DO SISTEMA ===");

    // Inicializa sensores
    initSensors();

    // Conecta WiFi
    conectaWifi();

    lastSend = millis();
    Serial.println("\n=== SISTEMA PRONTO ===");
}

void loop() {
    unsigned long now = millis();

    if (now - lastweek >= weektime){
        week++;
        lastweek = now;
    }
}

```

```

if (millis() - lastSend >= SEND_INTERVAL_MS) {
    lastSend = millis();

    Serial.println("\n\n--- CICLO DE ENVIO ---");
    Serial.print("Tempo: ");
    Serial.print(millis() / 1000);
    Serial.println("s");

    String json = montarJson();
    enviar(json);

    Serial.println("--- FIM DO CICLO ---");
    Serial.print("Próximo envio em ");
    Serial.print(SEND_INTERVAL_MS / 1000);
    Serial.println(" segundos...");
}

delay(1000);
}

```

Apêndice D

O módulo do contador abaixo implementa a marcação temporal utilizada pelo sistema na fase atual do projeto, servindo como solução provisória até a integração do relógio de oscilação OCXO ao controle de bordo. O contador é dividido em dois registros: o número da semana GPS, incrementado a cada 604.800 segundos, e o tempo decorrido dentro da semana atual, ambos baseados no tempo interno do ESP32. Essa estrutura replica o formato de temporização dos sistemas GNSS tradicionais, permitindo a validação dos algoritmos de sincronização e posicionamento mesmo sem o hardware de relógio definitivo.

```

unsigned long week = 0;
unsigned long lastweek = 0;
const unsigned long weektime = 604800000; // 7 dias em ms

void setup(){
    Serial.begin(115200);
    lastweek = millis();
}

void loop(){
    unsigned long now = millis();

```

```

    unsigned long tempoNestaSemana = now - lastweek; // tempo desde que a
    semana começou

    // Se passou 1 semana
    if (tempoNestaSemana >= weektime){
        week++;          // aumenta a semana
        lastweek = now;   // reseta o marco da semana
        tempoNestaSemana = 0; // zera o contador da semana atual
    }

    Serial.println("-----");
    Serial.print("Numero da semana: ");
    Serial.println(week);

    Serial.print("Tempo NESTA semana (desde que começou): ");
    Serial.println(tempoNestaSemana);
}

```

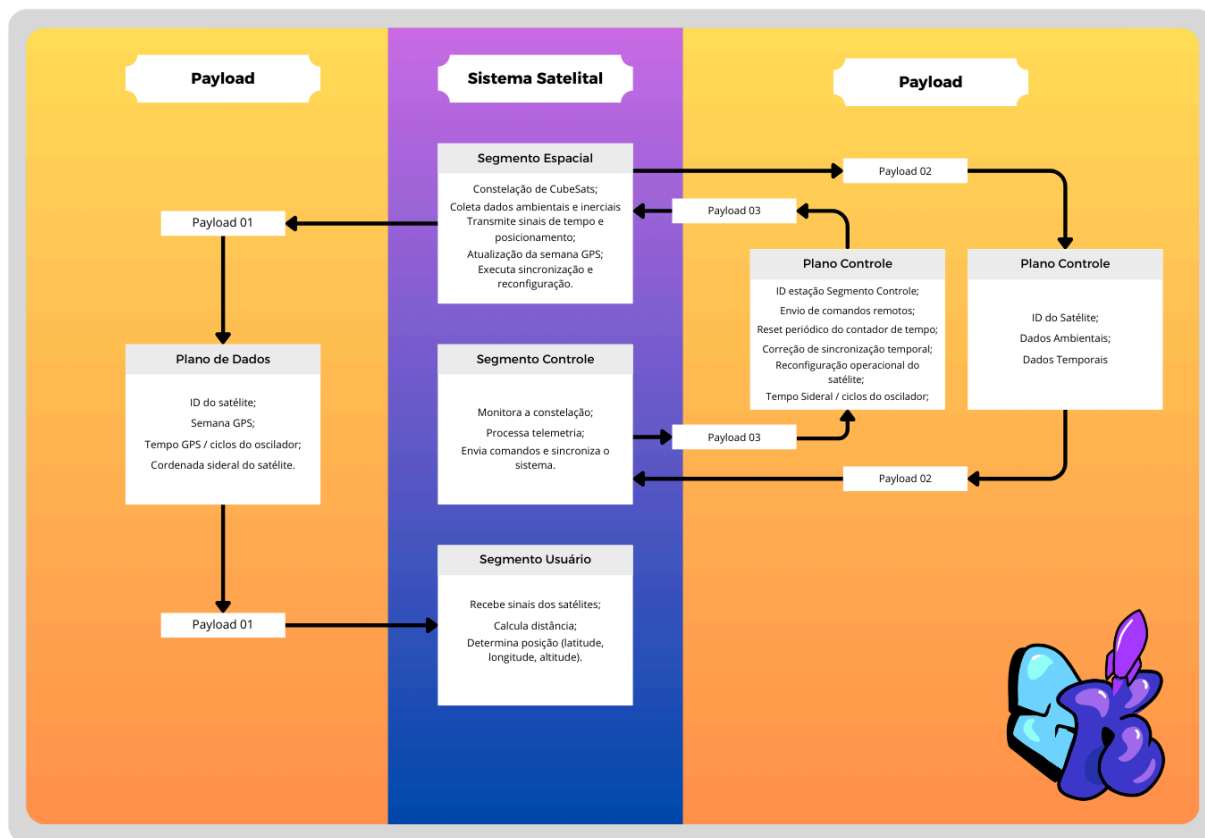
Resultado no Console:

```

Tempo NESTA semana (desde que começou): 4169
-----
Numero da semana: 0
Tempo NESTA semana (desde que começou): 4169
-----
Numero da semana: 0
Tempo NESTA semana (desde que começou): 4180
-----
Numero da semana: 0
Tempo NESTA semana (desde que começou): 4180

```

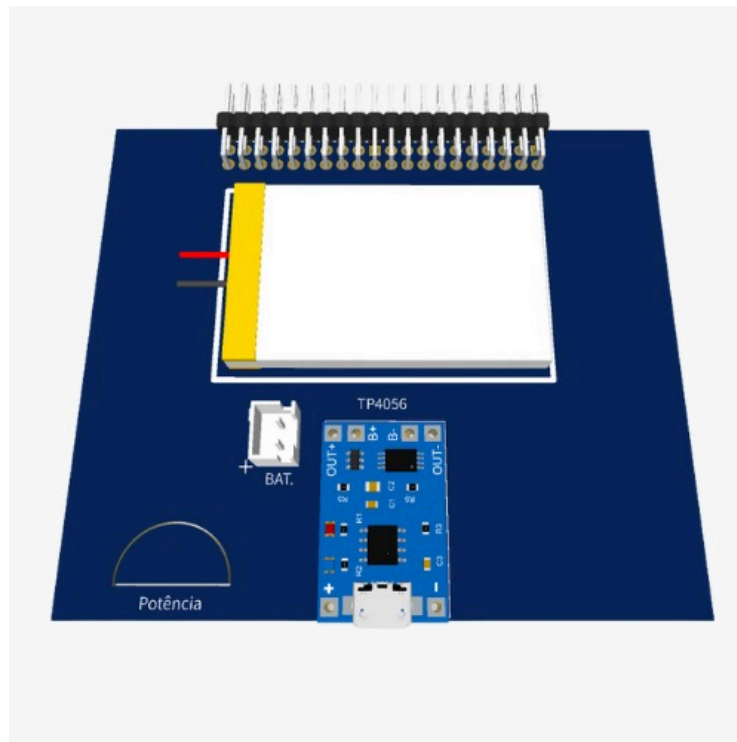
Apêndice E



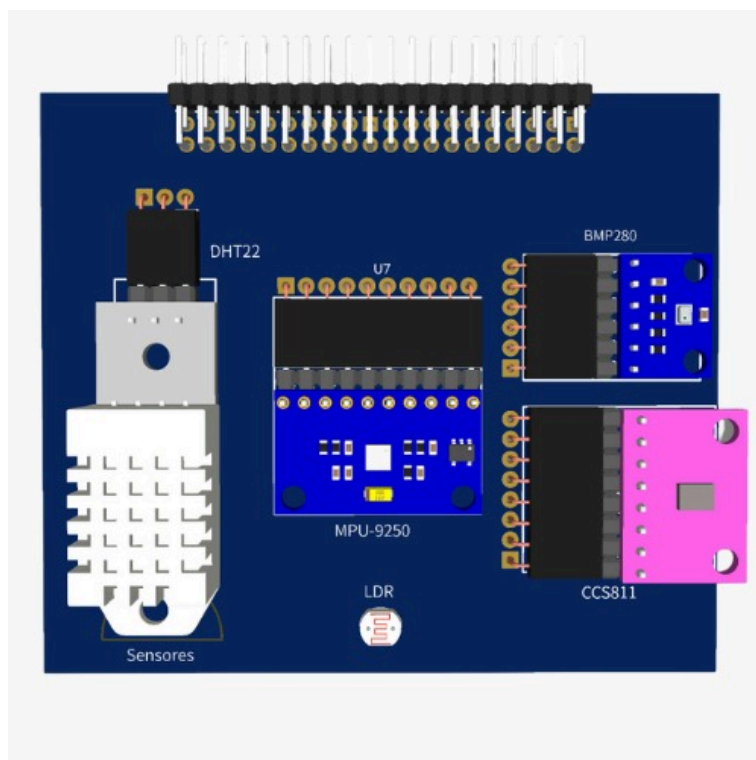
Apêndice F

TWOKÓSMOS			
SOBRE OS SEGMENTOS			
Segmentos	Função Principal	Plano de Controle	Plano de Dados
Satelital/Espacial	Coleta de dados, cálculo de posicionamento e transmissão dos payloads	Processa dados dos sensores e gerencia modos de operação, hibernação e duty cycle	Transmite identificador, semana GPS e tempo GPS para viabilizar o georreferenciamento (Payload 01)
Controle	Monitoramento da missão e envio de comandos de sincronização ao segmento espacial	Envia comandos de reset do contador do relógio e sincronização temporal (Payload 03)	—
Usuário	Recebe o sinal do satélite e calcula sua posição via tempo de propagação	—	Recebe Payload 01 e calcula a distância ao satélite com base no tempo de propagação

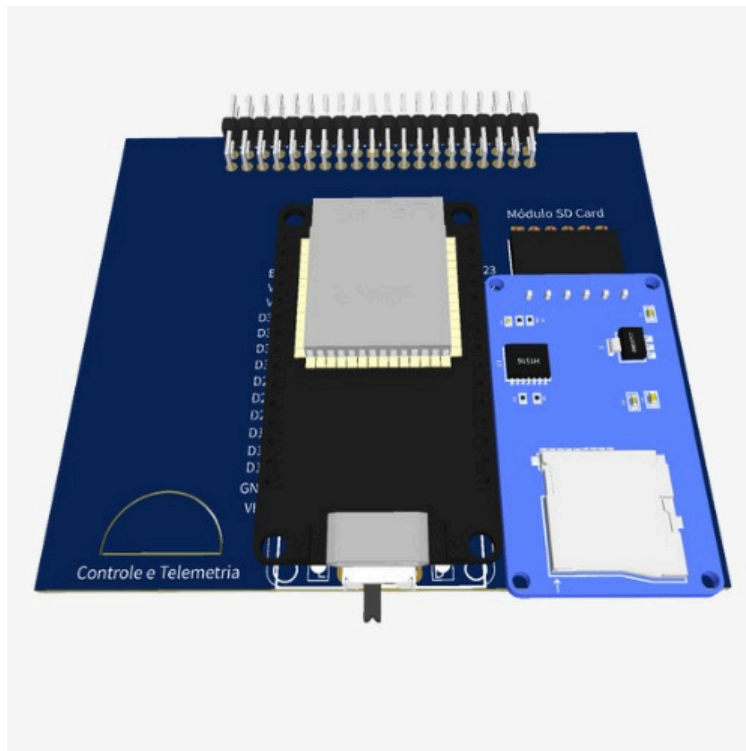
Apêndice G



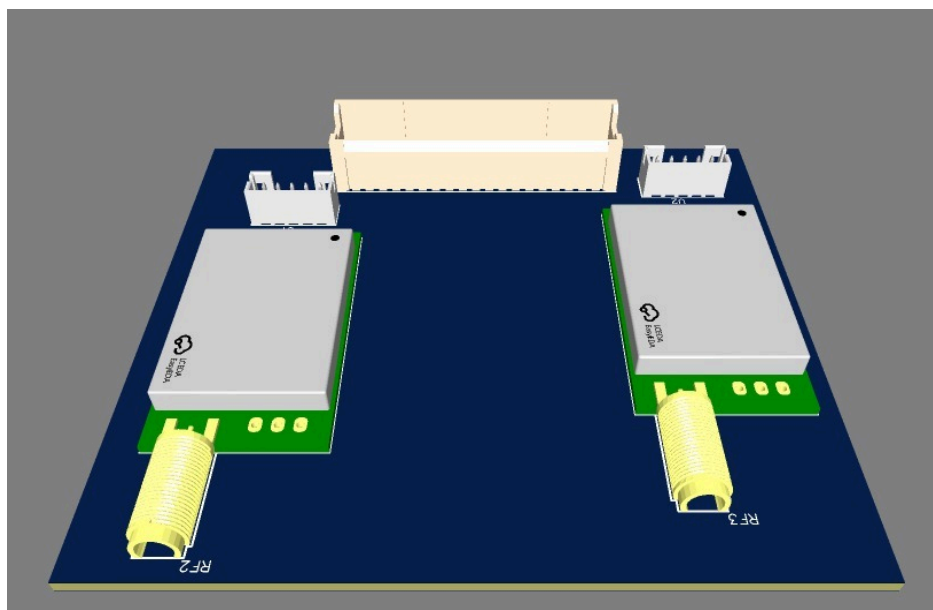
Apêndice H



Apêndice I



Apêndice J



Apêndice K

O módulo de trilateração tridimensional abaixo implementa o princípio matemático central do sistema RPS. A partir das posições conhecidas de quatro estações terrestres e das distâncias até o satélite, o algoritmo resolve o sistema de equações e retorna as coordenadas espaciais do CubeSat. Na fase atual, as distâncias utilizadas são simuladas, representando um cenário simétrico em que o

satélite se encontra equidistante de todas as estações. Nas fases seguintes, esses valores serão substituídos pelas distâncias reais, calculadas a partir do tempo de propagação do sinal LoRa conforme a relação $d = c \cdot \Delta t$, integrando o módulo ao fluxo completo de posicionamento do sistema.

```
//Aqui eu defino uma estrutura 'ponto' com coordenadas espaciais x, y e z
struct Point {
    double x, y, z;
};

void setup() {
    Serial.begin(115200);
    //exemplo de localizações das bases terrestres
    std::array<Point,4> base = {{
        {0, 0, 0},
        {2, 0, 0},
        {2, 1, 0},
        {0, 1, 0}
    }};
    //aqui estão as distâncias dos pontos das bases para o topo da uma pirâmide, o
satélite
    std::array<double,4> d = {
        2.5, 2.5, 2.5, 2.5
    };

    Point apex;
    if (findApex(base, d, apex)) {
        Serial.println("Apex: ");
        Serial.println(apex.x);
        Serial.println(apex.y);
        Serial.println(apex.z);
    } else {
        Serial.println("Sem solução válida\n");
    }
}

void loop() {
    delay(10);
}

//esta função é encarregada de resolver um sistema linear
bool solve2x2(double A[2][2], double b[2], double& x, double& y) {
    double det = A[0][0]*A[1][1] - A[0][1]*A[1][0];

    //esta parte verifica se o determinante é aproximadamente zero
    if (std::abs(det) < 1e-9)
        return false;
```

```

    x = ( b[0]*A[1][1] - b[1]*A[0][1]) / det;
    y = (-b[0]*A[1][0] + b[1]*A[0][0]) / det;
    return true;
}

bool findApex(
    const std::array<Point,4>& base,
    const std::array<double,4>& d,
    Point& apex
) {
    // Usar ponto zero como referência
    double x1 = base[0].x;
    double y1 = base[0].y;

    double A[2][2];
    double bvec[2];

    // Construir duas equações (usando pontos 1 e 2)
    for (int i = 1; i <= 2; i++) {
        double xi = base[i].x;
        double yi = base[i].y;

        A[i-1][0] = 2 * (xi - x1);
        A[i-1][1] = 2 * (yi - y1);

        bvec[i-1] =
            d[i]*d[i] - d[0]*d[0]
            + xi*xi - x1*x1
            + yi*yi - y1*y1;
    }

    double x, y;
    if (!solve2x2(A, bvec, x, y)) {
        return false;
    }

    // Calcular z
    double dx = x - x1;
    double dy = y - y1;
    double z2 = d[0]*d[0] - dx*dx - dy*dy;

    if (z2 < 0) return false;

    double z = std::sqrt(z2);

    // Escolher solução acima da base
    apex = {x, y, z};
    return true;
}

```

}

Referências

1. AEROENGENHARIA. O que é órbita heliossíncrona?. Disponível em: <https://aeroengenharia.com/glossario/o-que-e-heliosynchronous-orbit/>.
2. EO PORTAL. CanX-7 Mission. Disponível em: <https://www.eoportal.org/satellite-missions/canx-7#CanX7.html.3>.
3. EO PORTAL. InflateSail Mission. Disponível em: <https://www.eoportal.org/satellite-missions/inflatesail#boom-folding-method>.
4. FRIOLANI, Leticia Gatti; PAMBOUKIAN, Sergio Vicente Denser. Estudo da topologia do sistema Geolocal: um sistema de navegação brasileiro independente de GNSS. [S.l.]: Universidade Presbiteriana Mackenzie, 2020. Disponível em: https://www.mackenzie.br/fileadmin/ARQUIVOS/Public/1-mackenzie/universidade/lab-oratorios/labgeo/2020/Trabalhos/IC_-_ESTUDO_DA_TOPOLOGIA_DO_SISTEMA_GEOLOCAL_V10.pdf.
5. LUTWAK, Robert. The Chip-Scale Atomic Clock – Recent Developments. In: JOINT MEETING OF THE IEEE INTERNATIONAL FREQUENCY CONTROL SYMPOSIUM AND THE EUROPEAN FREQUENCY AND TIME FORUM, 23., 2009, Besançon. Proceedings... [S.l.]: IEEE, 2009. p. 573-577.
6. MACHO, Eduardo Perez. Implantação de um sistema brasileiro de georreferenciamento independente de outros sistemas de navegação por satélite: o Geolocal. 2018. 105 f. Dissertação (Mestrado em Ciências e Aplicações Geoespaciais) – Programa de Pós-Graduação em Ciências e Aplicações Geoespaciais, Universidade Presbiteriana Mackenzie, São Paulo, 2018.
7. MICROCHIP. CSAC SA65 – Chip Scale Atomic Clock. Disponível em: <https://www.microchip.com/en-us/product/csac-sa65>.
8. MONICO, Joao Francisco Galera. Posicionamento pelo GNSS: descrição, fundamentos e aplicações – Editora UNESP. São Paulo, 2008.
9. NAKAGAWA, F. Starlink é 16a em internet no Brasil, mas líder em satélite e 1/3 dos clientes estão no norte. Disponível em: <https://www.cnnbrasil.com.br/blogs/fernando-nakagawa/economia/negocios/starlink-e-16a-em-internet-no-brasil-mas-lider-em-satelite-e-1-3-dos-clientes-estao-no-norte/>.
10. NATIONAL INSTITUTE OF STANDARDS AND TECHNOLOGY – NIST. Como funcionam os relógios atômicos. Disponível em: https://www-nist-gov.translate.goog/atomic-clocks/how-do-atomic-clocks-work?_x_tr_sl=en&_x_tr_tl=pt&_x_tr_hl=pt&_x_tr_pto=tc.
11. ORBITAL TRANSPORTS. The Science Behind Drag Sails: A Solution for Satellite Deorbiting. Disponível em: <https://www.orbitaltransports.com/the-science-behind-drag-sails-a-solution-for-satellite-deorbiting/>.
12. PISONI, Fabio et al. Combined Navigation and Tracking with Applications to Low Earth Orbit Satellites. Eng. Proc., Basel, v. 88, n. 22, 28 mar. 2025. Disponível em: <https://doi.org/10.3390/engproc2025088022>.
13. SANCHEZ, Diogo Merguizo. Dinâmica ressonante de alguns satélites artificiais terrestres no sistema Terra-Lua-Sol. 2009. 92 f. Dissertação (Mestrado em Física Aplicada) – Programa de Pós-Graduação em Física, Instituto de Geociências e Ciências Exatas, Universidade Estadual Paulista, Rio Claro, 2009.
14. SANTOS, Credson Isaac Lopes dos. Dependabilidade Aplicada a Protocolo Hierárquico Baseado no LEACH. 2017. 66 f. Dissertação (Mestrado em Ciência da Computação) – Programa de Pós-Graduação em Ciência da Computação,

Universidade do Estado do Rio Grande do Norte e Universidade Federal Rural do Semiárido, Natal, 2017.

15. SWISSBIT. SFCA032GH2AD4TO-I-GS-246-STD – Compact Flash Card Industrial. Mouser Electronics. Disponível em: <https://br.mouser.com/ProductDetail/Swissbit/SFCA032GH2AD4TO-I-GS-246-STD?q=s=IKkN%2F947nfDjaGtDGuRhow%3D%3D>.