UNIVERSIDADE FEDERAL DO ABC
BACHARELADO EM ENGENHARIA DE INFORMAÇÃO

Leonardo de Souza Oliveira

PROTOCUBE

Implementação e Integração de Subsistemas de um Nanossatélite

Santo André, SP
2023

LEONARDO DE SOUZA OLIVEIRA

PROTOCUBE

Implementação e Integração de Subsistemas de um Nanossatélite

Relatório para o Trabalho de Graduação em Engenharia de Informação pela Universidade Federal do ABC, como requisito para obtenção do título de Bacharel em Engenharia de Informação.

Orientador: Prof. Dr. Leandro Baroni
Coorientador: Prof Dr. João Henrique Kleinschmidt

Santo André, SP
2023

Agradecimentos

Ao meu orientador, Prof. Dr. Leandro Baroni, por proporcionar o auxílio necessário para que eu pudesse realizar este trabalho.

Ao meu coorientador, Prof. Dr. João Henrique Kleinschmidt, por também me apoiar sempre que necessário durante a realização deste trabalho.

Aos coordenadores das disciplinas do Trabalho de Graduação da Engenharia de Informação, Marco Aurélio Cazzaroto e Engenharia Aeroespacial, Cesar Freire e Heloise Fazzolari, pelos apoios que permitiram com que este trabalho se tornasse realidade.

A todos os professores que me acompanharam ao longo desta graduação.

Aos meus amigos Saulo e Victor, que sempre colaboraram e torceram pelo meu sucesso durante a graduação.

Aos meus colegas da Equipe Sirius UFABC, pelos momentos de cooperação e amizade que passamos nos últimos anos.

Aos meus familiares: mãe, pai, irmão e avó, pelo apoio e incentivos incondicionais.

À minha namorada Bárbara, por sempre me apoiar e ajudar a superar dificuldades.

Epígrafe

“The cosmos is within us. We are made of star stuff. We are a way for the universe to know itself.” Carl Sagan.

Resumo

O setor aeroespacial vem experimentando uma transformação na última década em direção à reimaginação do conceito de um satélite. Antes uma área dominada por grandes empresas e agências governamentais, a ascensão dos nanossatélites traz consigo, além de novas possibilidades de mercado, o acesso mais democrático ao espaço para instituições de ensino de todo o mundo. Com isso, geram-se novas oportunidades de aprimoramento técnico nas universidades através de implementações desses pequenos satélites. O presente projeto visa o desenvolvimento do ProtoCube, um protótipo de integração de subsistemas de nanossatélites baseado em componentes Commercial Off-The-Shelf (COTS). Foi realizada a implementação lógica, física e computacional de um sistema composto por Suprimento de Energia, Computador de Bordo e sensores inerciais. O desempenho do sistema integrado foi aferido por meio de testes físicos dos componentes e testes de aquisição de dados pelos sensores com transmissão serial para uma plataforma com o software MATLAB, permitindo a visualização das medições em tempo real. Os resultados positivos demonstraram a capacidade de integração desses componentes e abriram caminho para que trabalhos futuros de expansão do ProtoCube sejam colocados em prática.

Palavras-chave: nanossatélite; subsistemas; computador de bordo; suprimento de energia; microcontrolador; sensor inercial; commercial off-the-shelf.

Abstract

The aerospace sector has undergone a transformation in the past decade towards reimagining the concept of a satellite. Once dominated by large companies and government agencies, the rise of nanosatellites brings not only new market possibilities but also more democratic access to space for educational institutions worldwide. This creates new opportunities for technical advancement in universities through the implementation of these small satellites. The current project aims to develop ProtoCube, a prototype for integrating nanosatellite subsystems based on Commercial Off-The-Shelf (COTS) components. The logical, physical, and computational implementation of a system composed of Electrical Power System, Onboard Computer, and inertial sensors was carried out. The performance of the integrated system was evaluated through physical testing of components and data acquisition tests by sensors with serial transmission to a platform with MATLAB software, enabling real-time visualization of measurements. Positive results demonstrated the integration capability of these components and paved the way for future expansion work on ProtoCube to be put into practice.

Keywords: nanosatellite; subsystems; electrical power system; onboard computer; microcontroller; inertial sensor; commercial off-the-shelf.

Lista de Figuras

Lista de Tabelas

SUMÁRIO

1 Introdução

Uma tendência crescente da adoção de projetos de nanossatélites vem atingindo, principalmente, universidades e instituições acadêmicas. Como apontado em (ROMAN-GONZALEZ; VARGAS-CUENTAS, 2016), os países da América do Sul estão realizando uma movimentação de interesse pela tecnologia dos pequenos satélites, principalmente como uma porta de entrada, já que seus custos são consideravelmente inferiores do que a da indústria espacial geral. O desenvolvimento de conhecimento e tecnologia nessa área é fundamental para que seja difundido um mercado espacial ativo na América do Sul nos próximos anos, propiciando investimentos e o aprimoramento das indústrias locais.

Com essa visão em mente, é apresentado neste trabalho o projeto **ProtoCube**, que visa desenvolver conceitos em variadas áreas do projeto de um sistema embarcado, onde nanossatélites se encaixam como tecnologias promissoras e com diversos desenvolvimentos em andamento, como pode ser verificado em (AEB, 2023).

Durante o período destinado ao desenvolvimento deste projeto, será implementado um protótipo de integração de diferentes subsistemas essenciais para o funcionamento de um pequeno satélite. Para isso, serão utilizados conceitos de engenharia, eletrônica e computação para que seja possível desenvolver o trabalho proposto.

Serão apresentados nas próximas seções os detalhes do desenvolvimento do ProtoCube, cujo nome foi inspirado nas chamadas “protoestrelas”, denominação dada a formações de gases e poeira interestelar que por fim darão vida a estrelas no futuro.

A seguir, serão disponibilizados a justificativa e os objetivos deste projeto, fornecendo um horizonte mais definido das ambições desejadas.

1.1 Justificativa

Como já citado, o desenvolvimento de tecnologia espacial vem sendo muito impactado mundialmente pelo movimento conduzido pelos nanossatélites. Uma indústria nacional requer uma cadeia longa de produtos e serviços que agreguem valor. No caso do Brasil, esse conglomerado ainda está muito limitado às demandas fornecidas pelo o INPE (Instituto Nacional de Pesquisas Espaciais) e pelo Instituto de Aeronáutica e Espaço (IAE)/Departamento de Ciência e Tecnologia Aeroespacial (DCTA), como apresentado em (SCHMIDT, 2011). Em adição às instituições citadas anteriormente, a VISIONA Tecnologia Espacial, joint-venture entre a Embraer Defesa & Segurança e a Telebras, vem se posicionando para também ocupar esse papel na cadeia de serviços. (SILVA, 2017; GALILEU, 2023)

Além desse fator, o exercício dos conceitos e fundamentos necessários aqui aprimorados podem ser aplicados em diversas áreas da engenharia, desde sistemas embarcados para aviação e eletrônica em geral, até mesmo para o desenvolvimento de software, tanto embarcado quanto de aplicação.

1.2 Objetivos

O objetivo geral deste trabalho é apresentar uma implementação de subsistemas que são essenciais para o funcionamento de um nanossatélite. Importante salientar que, além do funcionamento adequado dos subsistemas isolados, a integração entre eles é fundamental para o sucesso da missão. Portanto, o objetivo geral deste trabalho também envolve avaliar a atuação conjunta desses diferentes subsistemas implementados.

Os objetivos específicos incluem:

- Realizar uma revisão bibliográfica sobre projetos de nanossatélites desenvolvidos por instituições acadêmicas;

- Seleção de subsistemas para desenvolvimento do sistema embarcado;
- Seleção de componentes COTS (“Commercial Off-The-Shelf”);
- Implementação lógica, física e computacional dos subsistemas;
- Testes e análise dos resultados obtidos.

2 Desenvolvimento

Nas seções seguintes, serão abordados inúmeros tópicos relacionados às tarefas que foram essenciais para a realização deste projeto. Tanto descrições de componentes como detalhes de implementação serão alvos de discussão, culminando nos testes e resultados obtidos através do ProtoCube. Para isso, primeiramente, será apresentada a metodologia empregada, na qual também envolve a referência a projetos desenvolvidos em outros centros acadêmicos.

2.1 Metodologia

A metodologia deste trabalho foi dividida em seis fases, às quais compreendem as etapas que decorreram desde a definição dos subsistemas a serem implementados até a análise dos resultados obtidos pelos sensores embarcados:

1. Revisão de trabalhos desenvolvidos em instituições acadêmicas voltados à implementação de sistemas embarcados utilizados em nanossatélites;
2. Verificação de componentes COTS no mercado considerando-se as especificações técnicas necessárias para o funcionamento adequado dos subsistemas integrados;
3. Implementação lógica, física e computacional dos componentes. Esta etapa inclui a organização lógica dos subsistemas, montagem física e desenvolvimento de software embarcado e de aplicação para operação do Computador de Bordo

e recebimento de dados pelo software MATLAB;

4. Testes de desempenho dos subsistemas sob determinadas condições de operação, que são baseadas em etapas reais presentes ao longo das atividades de uma missão;

5. Testes e análise dos resultados obtidos através dos sensores embarcados no projeto;

6. Discussão de procedimentos e/ou componentes que poderiam ser aprimorados ou anexados em um projeto futuro.

A seguir, serão apresentados os detalhes dos processos realizados para confecção deste trabalho, incluindo informações referentes aos componentes do sistema, processos de montagem e implementação física e desenvolvimento de software utilizado tanto para funcionamento do computador de bordo quanto do recebimento e processamento dos dados recebidos através da plataforma MATLAB.

2.2 Revisão da Literatura

Desde o início da década de 1980 já eram dados alguns passos em direção ao desenvolvimento de pequenos satélites por universidades ao redor do mundo. No entanto, quando foi definido o design de referência do CubeSat em 1999 por Stanford University e California Polytechnic State University, uma nova corrida espacial começou a ser definida, desta vez, acompanhada também pela iniciativa privada com demandas comerciais.

Como citado em (CARVALHO; ESTELA; LANGER, 2020), entre 2006 e 2018, centenas de pequenos satélites foram lançados por universidades, evidenciando a capacidade dos programas dessas instituições, mas também demonstrando o grande interesse de órgãos americanos no financiamento de projetos estudantis, como pela agência espacial americana (NASA). Na Figura 1, pode-se conferir essa evolução em instituições acadêmicas.

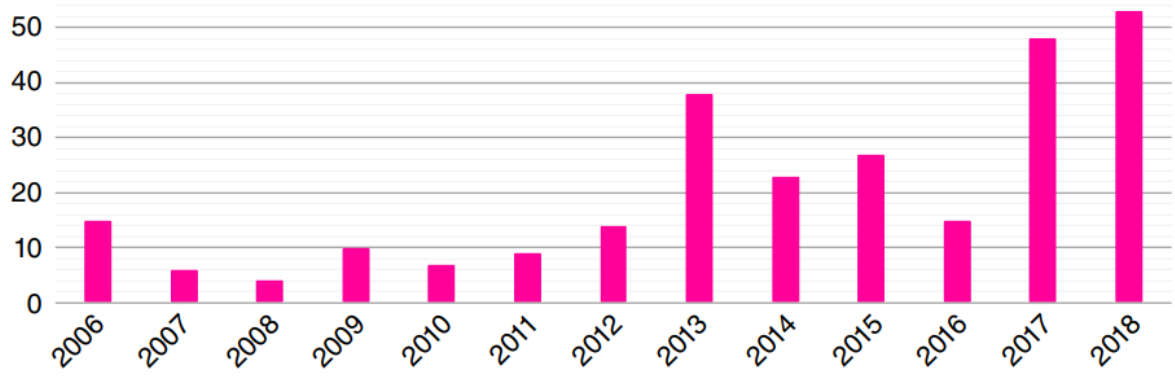

Figura 1: Evolução dos lançamentos de satélites de até 10kg por universidades entre 2006 e 2018. **Fonte: (CARVALHO; ESTELA; LANGER, 2020), p. 311.**

O aprendizado baseado em projetos também se mostrou bastante efetivo para o aprimoramento dos conhecimentos e das experiências em equipe de estudantes de diversas universidades ao redor do mundo, que impactaram no lançamento de programas universitários com o intuito de promover o desenvolvimento dos seus estudantes. (CARVALHO; ESTELA; LANGER, 2020)

Para exemplificar algumas experiências bem sucedidas que impulsionaram algumas instituições acadêmicas, podem ser citados os casos da Aalborg University e University of Tokio.

Para a Universidade de Aalborg, pode-se citar a experiência desenvolvida por seus estudantes ao longo das duas primeiras décadas deste século, em que desenvolveram os projetos AAUSAT-II, que carregava um sistema de determinação e controle experimental, além de um detector de raios gama, e o AAUSAT3, que possuía um Automatic Identification System (AIS) como carga útil. Suas contribuições se estendem para subsistemas e cargas úteis de diversas instituições vizinhas.

Já a respeito da Universidade de Tóquio, vale mencionar as missões XI, iniciadas na década de 2000, onde tanto o desenvolvimento e a produção de satélites como também a maioria das tarefas de gestão eram realizadas por estudantes. Com par-

cerias que vão desde o Observatório Astronômico Nacional do Japão, até empresas privadas comerciais do ramo espacial, como a Axelspace Corporation, a Universidade de Tóquio já conta, nos últimos anos, com instalações para desenvolvimento de missões tanto científicas como para observações da Terra.

Essa coleção de exemplos de investimentos em material humano nas universidades pode ser encontrada em (CARVALHO; ESTELA; LANGER, 2020).

Os subsistemas necessários para garantir o funcionamento das missões citadas geralmente são compostos por:

- Computador de Bordo (OBC, do inglês On-Board Computer);
- Suprimento de Energia (EPS, do inglês, *Electrical Power System*);
- Telemetria, Telecomando e Rastreio (TT&C, do inglês *Telemetry, Tracking, and Control*);
- Determinação e Controle de Atitude (ADC, do inglês *Attitude Determination and Control*);
- Carga Útil (*Payload*).

Para este trabalho, foi escolhida uma abordagem envolvendo a integração entre EPS, OBC e sensores inerciais comumente utilizados em subsistemas ADC. A partir da operação conjunta entre esses elementos, os dados adquiridos pelos sensores iniciais podem ser transmitidos e visualizados por um software de elevada capacidade gráfica que, neste caso, foi escolhido o MATLAB, da empresa MathWorks. Nas próximas seções, serão apontados os elementos escolhidos para permitir o sucesso da confecção deste projeto.

2.3 Componentes dos Subsistemas

A utilização de componentes COTS em projetos aeroespaciais vêm sido amplamente adotada nos últimos anos, principalmente nanossatélites que operam em órbitas baixas (LEO), onde a radiação solar possui uma ação menos nociva aos dispositivos eletrônicos. Apesar disso, a eletrônica COTS segue despreparada para esses ambientes hostis, requisitando estratégias para viabilizar sua implementação em missões reais. (SELČAN; KIRBIŠ; KRAMBERGER, 2016)

Visando promover o desenvolvimento dos objetivos do presente trabalho dentro do prazo e orçamento disponíveis, foram adotados componentes COTS disponíveis no mercado. Portanto, para a implementação real de subsistemas de um nanossatélite que, de fato, almeje operar nas condições adversas dos espaço, a substituição dos elementos escolhidos e a utilização de técnicas de aprimoramento de confiabilidade, como as mencionadas anteriormente, devem ser adotadas a fim de garantir o sucesso de uma missão aeroespacial.

2.3.1 Sistema de Suprimento de Energia (EPS)

Um dos requisitos fundamentais para a realização de uma missão é o abastecimento de energia para todos os componentes embarcados em um nanossatélite. Esse é o papel que desempenha o Sistema de Suprimento de Energia (EPS).

O elemento central desse subsistema encontra-se na figura da bateria, que deve ser capaz de fornecer energia elétrica a partir de sua energia química armazenada, garantindo o funcionamento do nanossatélite ao longo de sua vida útil. Outra função a qual o EPS é frequentemente atribuído é a de geração de energia para carregamento do conjunto de baterias. Para isto, módulos compostos por células fotovoltaicas realizam a conversão da energia transportada por fótons em energia elétrica. Dessa maneira, é possível recarregar as baterias durante a missão, prolongando a longevidade de suas atividades.

Tendo em mente os papéis referenciados acima, para este projeto, foram selecionados os seguintes componentes COTS para o desenvolvimento de um subsistema de suprimento de energia:

- Conjunto de Baterias 18650 de íon de lítio (Li-Ion);
- Módulo de Gerenciamento de Baterias (BMS, do inglês, *Battery Management System*);
- Conjunto de módulos compostos por células fotovoltaicas;
- Diodos para garantir a proteção adequada das células fotovoltaicas;
- Módulo regulador de tensão para limitar a tensão fornecida para carga do conjunto de baterias;
- Conjunto composto por fonte chaveada e módulo regulador de tensão e corrente (estes elementos foram adicionados ao projeto para fornecer uma via alternativa para o carregamento do conjunto de baterias, não sendo objetos de implantação em um sistema embarcado de uma missão espacial);
- Proteção contra sobrecorrente.

2.3.1.1 Baterias 18650

A respeito do conjunto de baterias, foram selecionadas 2 baterias recarregáveis 18650 de íon de lítio (Li-Ion). As baterias 18650 são amplamente encontradas no mercado, onde pode-se escolher os mais variados modelos de capacidades. Para este projeto, foram utilizadas baterias com capacidades de 8000mAh. Na tabela 1, são apresentadas características tipicamente encontradas em *datasheets* para esse tipo de bateria, onde mais informações podem ser obtidas em (EEMB CO., LTD., 2010).

Tabela 1: Características típicas para baterias 18650.

Tensão Nominal	3.7V
Impedância Interna	$\leq$ 70mΩ
Tensão de Corte de Descarga	3.0V
Tensão Máxima de Carga	4.20$\pm$0.05V
Corrente de Carga Padrão	0.52A
Corrente de Carga Rápida	1.3A
Corrente de Descarga Padrão	0.52A
Composição	Li-Ion

O nome do modelo dessas baterias está relacionado a suas dimensões, em que cada célula cilíndrica possui diâmetro de 18mm e comprimento de 65mm. Na Figura 2, tem-se um exemplo de uma bateria 18650.

Figura 2: Exemplo de bateria 18650.

A célula 18650 possui uma tensão máxima de carga de 4,2V, portanto, para garantir o suprimento de energia tanto do computador de bordo quanto dos demais componentes do sistema, foi optado por uma composição de 2 baterias 18650 em série. O conjunto de 2 baterias permite uma tensão máxima de 8,4V com a mesma autonomia de 8000mAh.

Para garantir que a composição de baterias esteja segura contra curto-circuitos, subtensão, sobretensão e sobrecorrente, é de extrema importância que seja adotado o uso de Sistemas de Gerenciamento de Baterias (BMS, do inglês, Battery Management Systems). Sua adoção permite a proteção contra as situações mencionadas

anteriormente e, além disso, geralmente garante o balanceamento de tensão entre as células Li-Ion.

2.3.1.2 Módulo de Gerenciamento de Baterias (BMS)

Cumprindo a função de proteção como BMS, foi selecionado o módulo HX-2S-D20, que possui capacidade de assegurar 2 células 18650 contra curto-circuito, sobrecarga e descarga excessiva (Veja Figura 3). Esse modelo em específico não possui balanceamento de carga, responsável por igualar os níveis de carga em ambas as baterias. Com a ausência dessa funcionalidade, é necessário carregar as 2 baterias de forma externa ao projeto (lembrando-se de se ter atenção aos limites de corrente de carga), para garantir que ambas as células sejam inseridas no sistema deste projeto em níveis de carga balanceados previamente. A partir do momento em que o BMS passa a gerenciar o conjunto de baterias, os ciclos de carga e descarga são realizados em nível de igualdade. Uma Tabela com algumas das características do módulo escolhido encontra-se na Tabela 2.

Tabela 2: Características do Módulo BMS HX-2S-D20.

Modelo	HX-2S-D20
Faixa de Tensão de Sobrecarga	4.25-4.35 V + 0.05 V
Faixa de Tensão de Subtensão	2.5-3.0 V + 0.05 V
Máxima Corrente de Operação	13 A
Tensão de Carregamento	8.4-9 V
Corrente Quiescente	<10 μA
Impedância Interna	<300 Ohm
Temperatura de Operação	-40°C à +50°C
Comprimento (mm)	44.77
Largura (mm)	18.85
Altura (mm)	3.5
Peso (g)	4

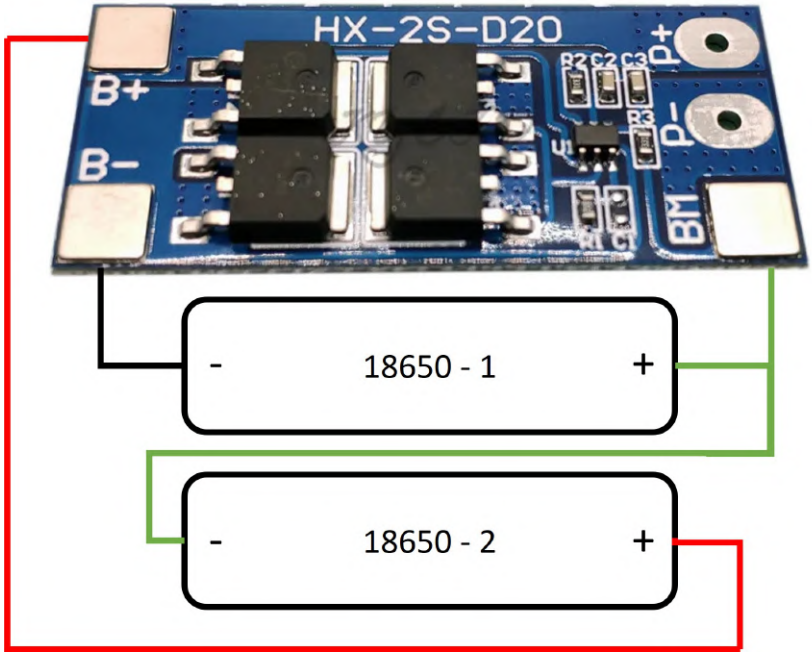

Figura 3: Módulo BMS com esquema de interligação com 2 Baterias Li-Ion 18650.
Fonte: próprio autor.

2.3.1.3 Células Fotovoltaicas

Com o objetivo de permitir que o EPS possua capacidade de reabastecimento energético durante o andamento da missão, foi optada pela seleção de pequenos módulos de células fotovoltaicas, que possuem dimensões apropriadas para a instalação fixa às estruturas de um nanossatélite, ou seja, até 1U (10 centímetros). A escolha da combinação e das características de células fotovoltaicas para uma missão é crucial para proporcionar um sistema de geração de energia capaz de abastecer as células de baterias do EPS de acordo com os requisitos estipulados pela missão, visto que um desempenho dos painéis fotovoltaicos inferior à demanda energética necessária se torna um fator limitante para ao seu tempo de serviço.

Para este trabalho, foi escolhida uma configuração composta por 3 pares de módulos fotovoltaicos conectados em paralelo, de modo a proporcionar uma potência

de geração maior em relação ao que apenas 1 par é capaz de produzir. A Figura 4 exibe um esquema no qual é possível visualizar a configuração mencionada. Nele, também constam os chamados “Diodos de Bypass” e “Diodos de Bloqueio”, cujos papéis foram desempenhados por diodos 1N4007.

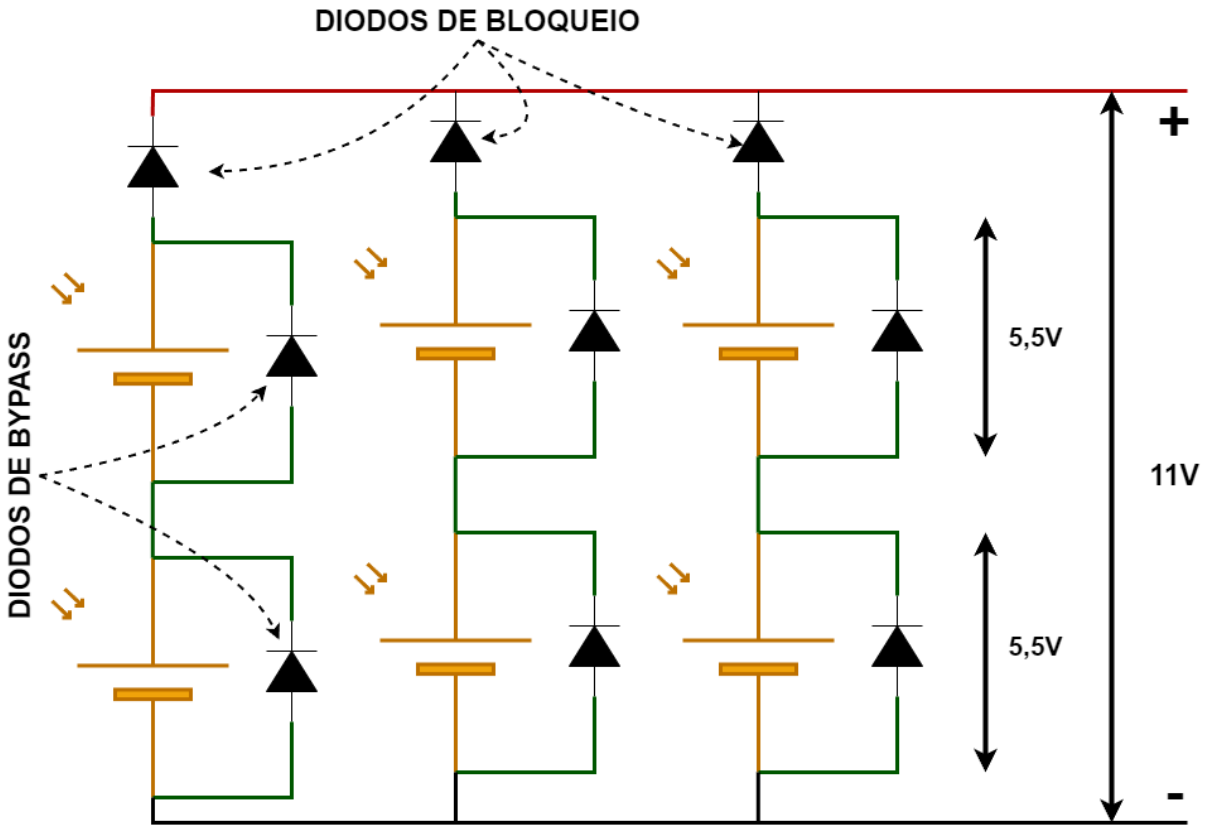

Figura 4: Diagrama da configuração de células fotovoltaicas utilizada para este trabalho. Na imagem, encontram-se também os diodos 1N4007, que possuem as funções de bloqueio e de bypass, adicionados ao conjunto de módulos para proporcionar maior segurança e desempenho.
Fonte: próprio autor.

A escolha das dimensões 100x30mm para as células fotovoltaicas foi baseada nas medidas padrão de nanossatélites, que utilizam como referência de padronização de medidas 1U (100x100x100mm), sendo possível encontrá-los até em configurações 6U, ou seja, dimensões equivalentes a 6 cubos de lado 100mm. Uma lista de especificações técnicas do módulo mencionado pode ser encontrada na Tabela 3.

Tabela 3: Especificações técnicas de cada módulo de células fotovoltaicas.

Marca	OEM
Tensão de Operação	5,5V
Potência	0,33W
Corrente	60mA
Material	Silicatos / Metal
Tamanho	100mm x 28mm Profundidade x 3mm
Peso	10g

2.3.1.4 Regulador de Tensão

A partir das especificações das células fotovoltaicas, uma composição de 2 células em série é capaz de converter energia solar em elétrica gerando uma tensão de por volta de 11V (2x5,5V). Em consequência disto, considerando que o módulo BMS e o par de baterias 18650 possuirão tensão de carregamento de 8,4V, é necessário um dispositivo de regulação que reduza a diferença de potencial para essa voltagem máxima. O regulador de tensão step-down ajustável D-SUN baseado no regulador de comutação MP1584, onde mais informações podem ser obtidas em (MONOLITHIC POWER SYSTEMS, 2011). Uma imagem do regulador de tensão mencionado pode ser encontrada na Figura 5

Figura 5: Regulador de Tensão Step Down Mini-360.

2.3.1.5 Fonte Chaveada

Apesar de não ser um dos componentes embarcados em um nanossatélite real, foi adicionado ao projeto uma fonte de alimentação auxiliar externa para facilitar o carregamento das baterias durante os períodos de testes dos subsistemas. A fim de atingir esse objetivo, foi selecionada uma pequena fonte chaveada 12V 3A para carregamento rápido das baterias sem que elas precisem ser removidas de sua estrutura física implementada. A Figura 6 exibe uma imagem da fonte mencionada.

Figura 6: Fonte chaveada 12V 3A para carregamento rápido das baterias.

2.3.1.6 Regulador de Tensão e Corrente

O Módulo BMS HX-2S-D20 não possui um limitador de corrente embutido, porém, possui uma proteção contra sobrecorrente, que desliga o módulo para garantir que as baterias 18650 não sejam danificadas com a corrente alta que a fonte exigiria. Vale relembrar que, como as baterias 18650 possuem resistências internas abaixo de 70mΩ, a corrente elétrica resultante do carregamento direto por fontes prejudica a vida útil das baterias. De acordo com a Tabela 1, a corrente de carregamento nominal deverá ser em torno de 0,5A. Com isso, para garantir a manutenção da qualidade das baterias, foi adicionado um regulador CC/CV baseado no conversor XL4015 (XLSEMI, 2018) entre a fonte chaveada e o regulador de tensão 8,4V. A imagem desse regulador

pode ser encontrada na Figura 7.

Figura 7: Regulador CC/CV utilizado para limitar a corrente de carga das baterias através da fonte de alimentação auxiliar.

2.3.2 Computador de Bordo (OBC) e Sensores

Todos os subsistemas que compõem o sistema espacial como o tratado neste trabalho são imprescindíveis para que o projeto seja bem sucedido, no entanto, pode-se considerar o Computador de Bordo (OBC) como o “cérebro do nanossatélite”. Ele é responsável por gerenciar e processar os dados adquiridos, supervisionar a operação dos demais componentes e garantir que todas as etapas da missão sejam seguidas de acordo com o planejamento.

2.3.2.1 Computador de Bordo

Para cumprir esse papel central neste projeto, foi optado por uma Plataforma de Desenvolvimento desenvolvida pela fabricante DOIT, que utiliza como base o módulo microcontrolador SoC ESP-WROOM-32, da fabricante chinesa Espressif Systems. A plataforma em questão é denominada **ESP32 DevKit v1**, cuja imagem pode ser encontrada na Figura 8. Esse microcontrolador poderoso conta com uma série de periféricos que permitem a aquisição de dados a partir dos mais variados sensores, além do controle de atuadores, que podem ser indicados como requisitos de missão. Na Tabela 4, pode-se conferir algumas das especificações técnicas do módulo ESP-WROOM-32. Mais informações podem ser obitidas em (ESPRESSIF SYSTEMS (SHANGHAI) CO., LTD., 2023b).

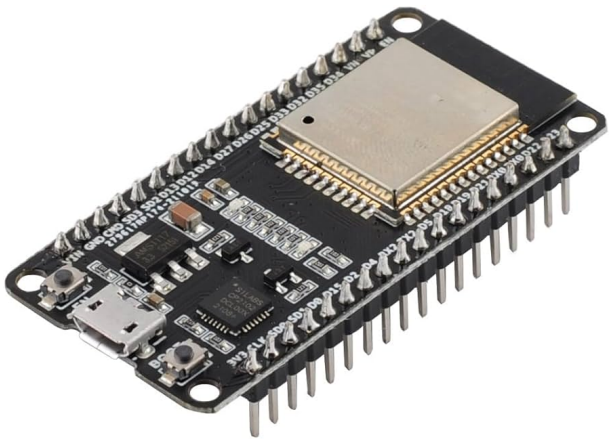

Figura 8: Plataforma de Desenvolvimento DOIT ESP32 DevKit v1.

Tabela 4: Especificações Técnicas do ESP-WROOM-32.

Wi-Fi	Protocolos	802.11 b/g/n (802.11n até 150 Mbps)
	Faixa de frequência central do canal de operação	2412~2484 MHz
Bluetooth	Protocolos	Especificação Bluetooth v4.2 BR/EDR e Bluetooth LE
	Radio	Recetor NZIF com sensibilidade de -97 dBm
Hardware	Interfaces de módulos	Cartão SD, UART, SPI, SDIO, I2C, LED PWM, Motor PWM, I2S, IR, contador de impulsos, GPIO, sensor tátil capacitivo, ADC, DAC
	Cristal integrado	Cristal de 40 MHz
	Flash SPI integrado	4 MB
	Tensão de operação	3.0 V ~ 3.6 V
	Corrente de operação	Média: 80 mA
	Corrente mínima fornecida pela fonte de alimentação	500 mA
	Faixa de temperatura ambiente de operação recomendada	-40°C ~ +85°C
	Tamanho do envólucro	18 mm x 25.5 mm x 3.10 mm

O DevKit v1 possui um regulador de tensão acoplado a sua plataforma, com isso, a sua alimentação externa pode ser realizada através de uma fonte de energia que opera em uma faixa de 6 a 20V, apesar de que é recomendado permanecer dentro do intervalo de 7 a 12V. O suprimento de energia na região dos 8,4V através do EPS é mais do que suficiente para manter a operação estável do OBC composto por esse microcontrolador da Espressif.

Dois periféricos indicados na Tabela 4 são de suma importância para a implementação deste trabalho: I2C e UART.

- **I2C**

  O protocolo I2C se trata de um protocolo de comunicação muito difundido tanto para a aquisição de dados de sensores como também para a escrita. A comunicação utilizando este protocolo é estabelecida a partir do uso de 2 sinais: SDA e SCL.

  O SDA consiste no canal de comunicação em si, onde os dados podem ser enviados e/ou recebidos em uma configuração half-duplex, ou seja, permite uma comunicação de 2 vias, porém, não simultâneas. Uma alternativa full-duplex seria a utilização do protocolo SPI, no entanto, o módulo MPU-9250, que possui os sensores necessários para este trabalho, possui apenas comunicação através do protocolo I2C.

  O canal SCL constitui simplesmente no sinal de *clock*, que, através da combinação com o sinal SDA, possibilita a correta interpretação das mensagens recebidas.

  O fato desse protocolo possuir característica half-duplex requer que o computador de bordo implemente elementos de espera para transmitir suas requisições para os sensores caso o barramento I2C estiver ocupado com uma transmissão. Na Figura 9, tem-se uma imagem que exemplifica uma transmissão realizada pelo protocolo I2C.

- **UART**

  O protocolo UART consiste na tradicional comunicação serial assíncrona, onde bits são transmitidos em sequência sem um sinal de *clock* disponibilizado.

  Uma comunicação simples através desse protocolo requer apenas 2 canais: RX e TX. Os bits podem ser transmitidos e recebidos simultaneamente, visto que possuem canais de transmissão diferentes, porém, o tamanho de cada pacote é bem limitado. Há ainda uma variação do protocolo em que existem mais 2 canais para controle do fluxo de dados pelas aplicações envolvidas na comunicação (RTS e CTS), possibilitando que os componentes informem uns aos outros os

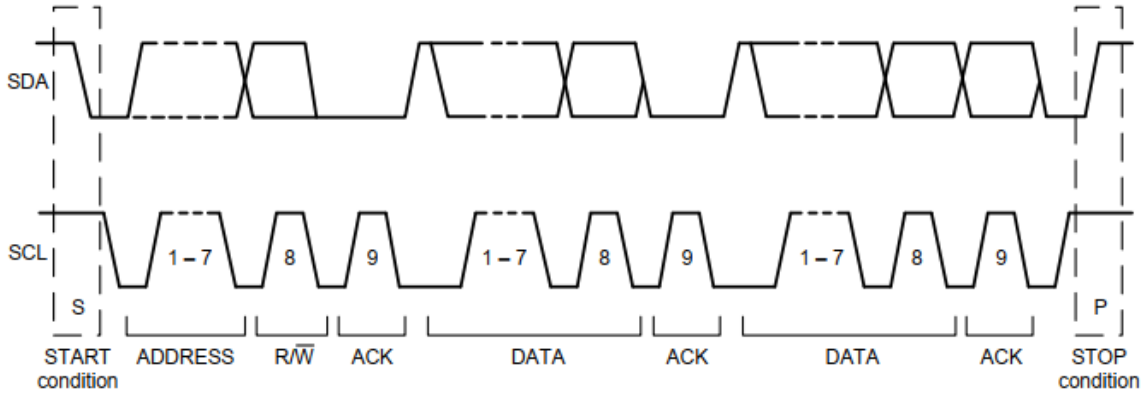

Figura 9: Exemplo de transmissão de dados através do protocolo I2C.

Start Bit (1 bit)	Data Frame (5 to 9 Data Bits)	Parity Bits (0 to 1 bit)	Stop Bits (1 to 2 bits)

Figura 10: Exemplo de Transmissão serial a partir do protocolo UART.

momentos que desejam ou não receber uma nova transmissão.

Na Figura 10, é possível visualizar um exemplo de transmissão de dados a partir do protocolo UART.

2.3.2.2 Sensores Inerciais

Devido ao seu fator de forma reduzido e ser composto pelos 3 sensores inerciais requisitados para este projeto, foi escolhido o módulo MPU-9250, composto por acelerômetro, giroscópio e magnetômetro. Isto permite ser obtido um IMU 9-DOF, ou seja, uma unidade de medição inercial de 9 graus de liberdade (do inglês, 9 Degrees of Freedom Inertial Measurement Unit). Com este tipo de dispositivo, é possível que sejam realizadas medições de diferentes fontes para, com o tratamento devido, fornecer informações de orientação do corpo no qual está implantado. Portanto, o MPU-9250 se tornou uma excelente opção para essa tarefa. Na Figura 11, tem-se uma imagem do

módulo MPU-9250 utilizado neste projeto. Mais informações a respeito desse módulo estão disponíveis em (INVENSENSE INC., 2016).

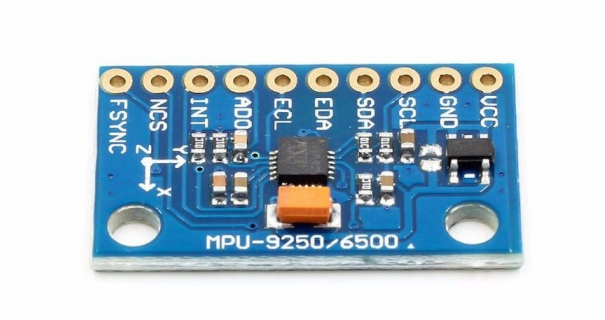

Figura 11: Módulo MPU-9250 composto por sensores acelerômetro, giroscópio e magnetômetro.

2.3.2.3 Conversor UART - Micro USB

Os dados das medições realizadas pelos sensores do módulo MPU-9250 são processadas pelo OBC (ESP32 DevKit v1), porém, é necessária uma interface de comunicação capaz de permitir com que esses dados sejam transmitidos para o computador que está aguardando a recepção para exibição dos resultados. Quem cumpre este papel é um módulo composto por uma interface USB/Serial CP2102, cujas especificações técnicas podem ser encontradas em (SILICON LABORATORIES, 2004). Uma imagem do módulo escolhido encontra-se na Figura 12.

Figura 12: Conversor USB-Serial CP2102, utilizado como interface entre o OBC e o computador executando o software Matlab.

2.4 Implementação do Projeto

Tendo em mãos os componentes necessários para o desenvolvimento do projeto, é possível separar a implementação em 3 áreas diferentes:

- **Lógica:** organização lógica dos subsistemas e as relações entre eles;
- **Física:** majoritariamente voltada ao hardware dos componentes. É composta tanto pela organização física dos elementos quanto pelas tarefas de soldagem e montagem;
- **Computacional:** desenvolvimento e implementação de algoritmos tanto para execução de tarefas pelo OBC quanto para a visualização dos resultados através do software Matlab.

Nas próximas seções, serão expostos detalhes sobre a implementação deste projeto. Serão exibidas diversas imagens dos componentes utilizados e informações a respeito dos processos realizados a fim de construir o protótipo dos subsistemas

integrados.

2.4.1 Lógica

Considerando os objetivos atribuídos para a missão (Seção 2.2), pode-se organizar os subsistemas a serem explorados em um diagrama com suas relações destacadas.

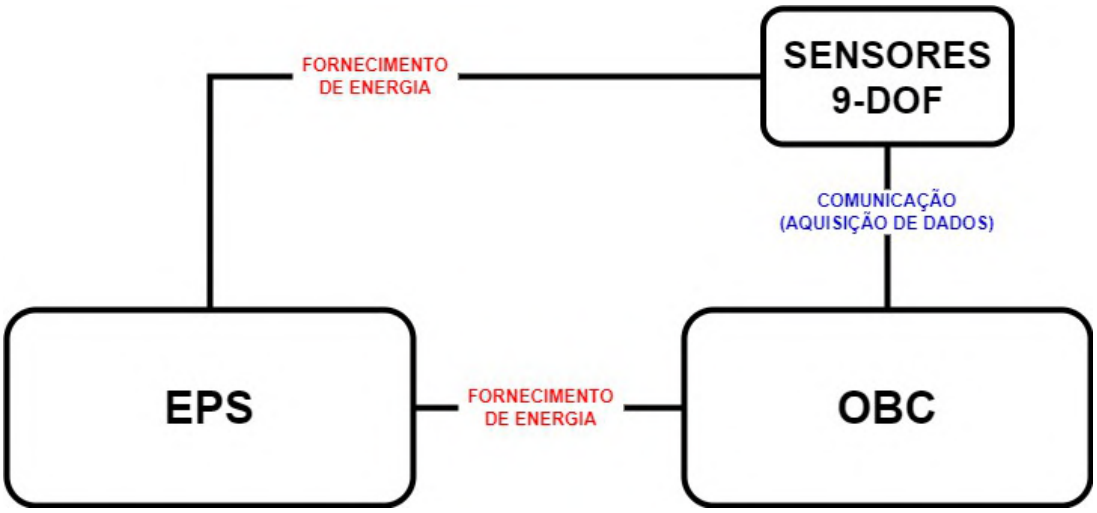

Figura 13: Diagrama Lógico dos Subsistemas implementados e suas relações.
Fonte: próprio autor.

Na Figura 13, o subsistema de Suprimento de Energia é representado pelo bloco "EPS", o subsistema de Computador de Bordo pelo bloco "OBC" e o módulo composto pelos sensores acelerômetro, giroscópio e magnetômetro pelo bloco "SENSORES 9-DOF". Também são apresentadas as relações entre essas entidades:

- **Fornecimento de Energia:** o EPS tem como função principal abastecer tanto o OBC como os Sensores 9-DOF;
- **Comunicação (Aquisição de Dados):** o OBC se relaciona com os Sensores 9-DOF através da aquisição dos dados medidos pelos sensores. Para tanto, o OBC realiza solicitações de leitura para os sensores através do barramento de comunicação I2C.

Dessa forma, é criada uma visualização de alto nível, abstraindo detalhes técnicos e possibilitando uma compreensão geral a respeito dos papéis de cada subsistema.

2.4.2 Física

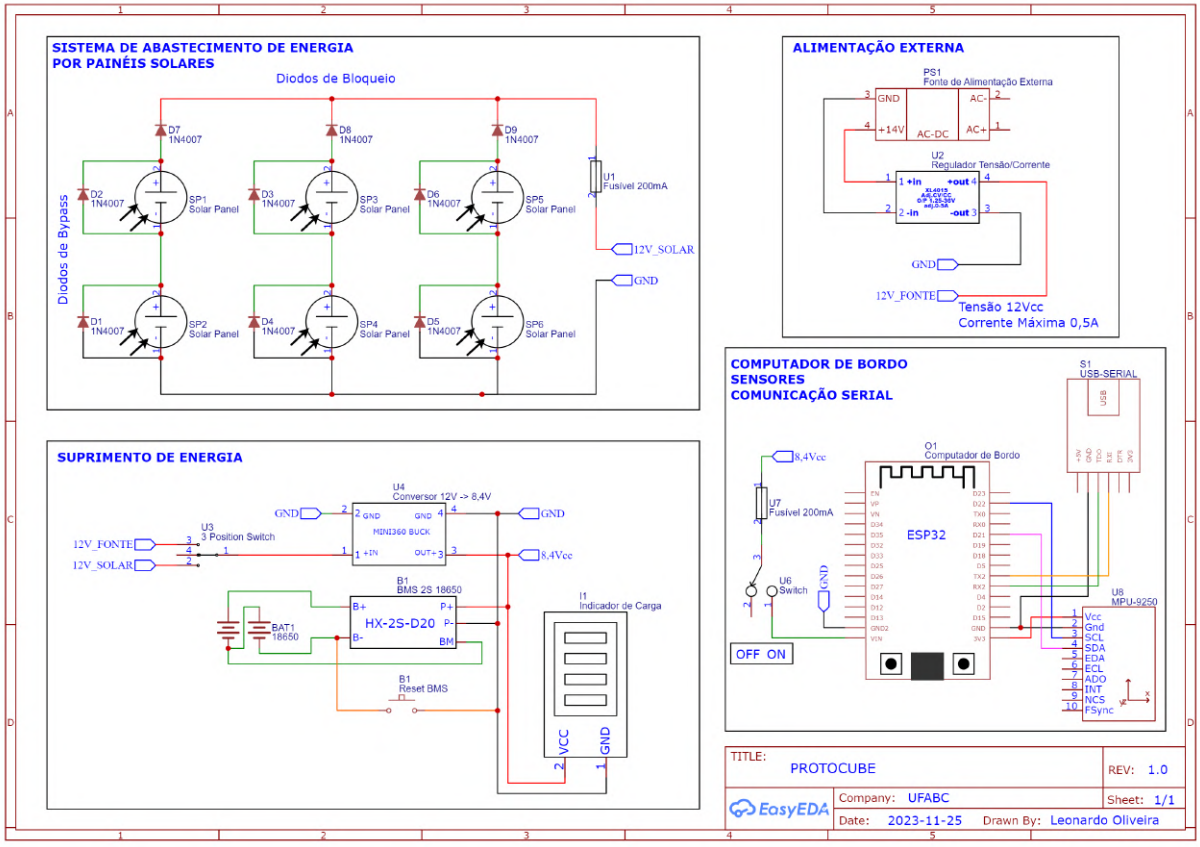

Figura 14: Esquemático geral da implementação física do ProtoCube.
Fonte: próprio autor.

A implementação física deste projeto inclui desde o planejamento e organização física à soldagem de componentes. Ao longo do desenvolvimento, foram sendo aprimoradas técnicas e estratégias de como posicionar da maneira mais eficiente os elementos de maneira que fossem facilitados os testes e a manutenção. O estabelecimento de itens de proteção, como fusíveis, são garantias extras para evitar situações inesperadas e indesejáveis.

Após a finalização dos trabalhos físicos, a Figura 14 dispõe de um esquemático que aborda todos os componentes e subsistemas implementados. Para facilitar a identificação dos subsistemas, estes foram separados e destacados, proporcionando uma visualização mais ágil, objetiva e organizada.

Além do esquemático geral do projeto, na Figura 15, tem-se uma imagem que exibe todos os componentes do ProtoCube. Nela, encontram-se todos os subsistemas que faziam parte do planejamento, inclusive com a adição de switches (botões) para facilitar a operação do protótipo. Com o objetivo de facilitar a identificação a partir das imagens, foram adicionadas etiquetas com os nomes dos componentes físicos.

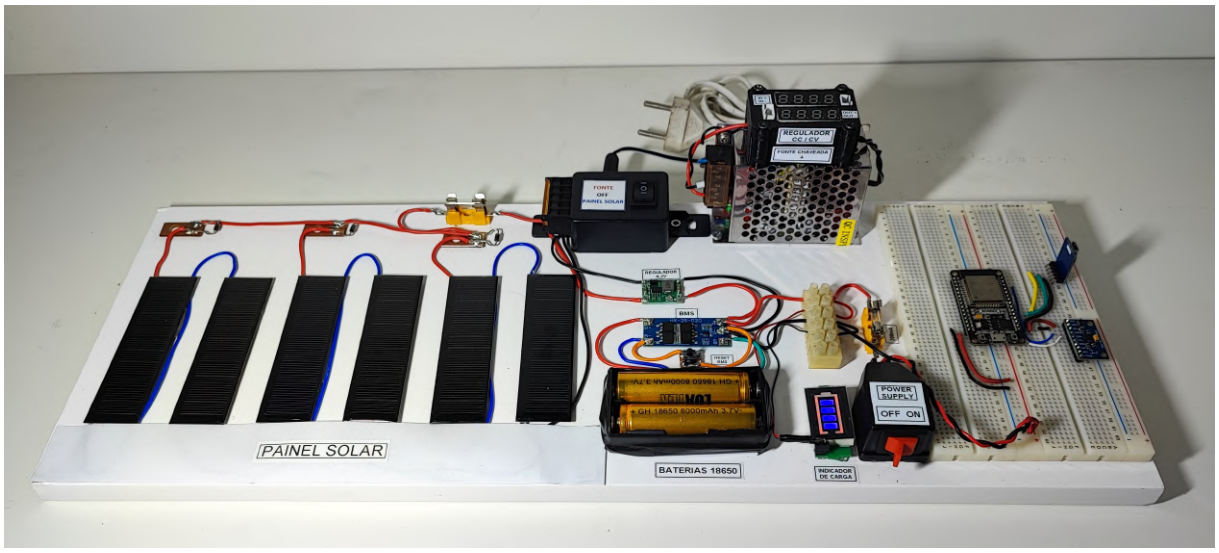

Figura 15: Visão Geral da Implementação Física do ProtoCube.

O detalhamento de cada um dos subsistemas será fornecido nas próximas seções de forma separada, permitindo com que os conceitos e os objetos de estudo tenham enfoques adequados.

2.4.2.1 Sistema de Abastecimento de Energia por Painéis Solares

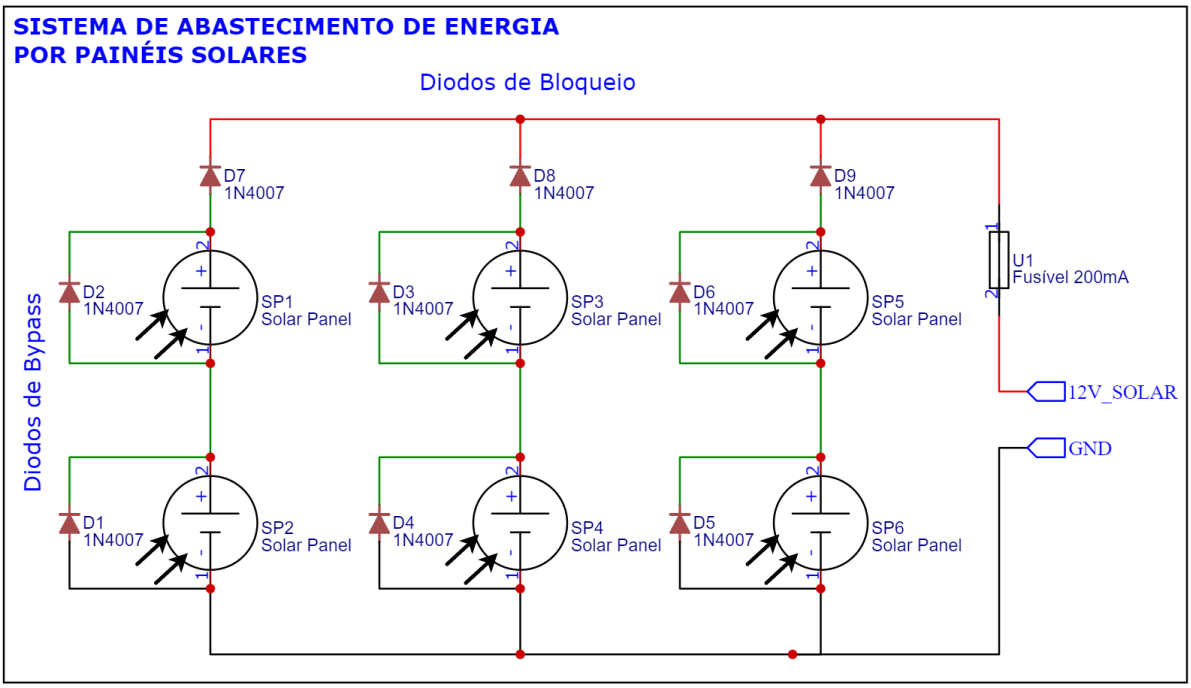

Figura 16: Esquemático do Sistema de Abastecimento de Energia por células fotovoltaicas.

Para confeccionar o sistema de abastecimento de energia por células fotovoltaicas, foi necessário realizar uma combinação dos módulos para que fosse possível ser gerada uma tensão de operação e corrente maiores. Cada módulo composto por células fotovoltaicas permite, teoricamente, proporcionar uma diferença de potencial de 5,5V, corrente máxima de 60mA. Logo, para satisfazer os requisitos da missão, foi adotada uma configuração de três pares de módulos em série conectados em paralelo, como pode ser verificada no diagrama esquemático fornecido na Figura 16.

Levando em conta fatores de segurança e proteção para os módulos de células fotovoltaicas, foram adicionados diodos de Bypass paralelamente a cada um dos módulos, evitando que defeitos sejam propagados para os vizinhos em série e que o sombreamento de células individuais provocasse regiões quentes, conhecidas como *“hot spots”*. Para tal, foram empregados diodos 1N4007.

Quando está sendo implementado um sistemas de painéis solares, também há a

necessidade de colocar em prática outra estratégia de proteção para células fotovoltaicas: diodos de bloqueio. A possibilidade da ocorrência de correntes reversas sobre os módulos, seja provocadas por outros pares de módulos ou pelo conjunto de baterias do EPS, adiciona esses diodos como requisito de segurança. Da mesma forma como para os diodos de bypass, também foram adotados diodos 1N4007 para os diodos de bloqueio.

É importante salientar que, quando utilizados diodos de bloqueio, deve-se levar em conta a tensão de queda desses diodos. A queda de tensão dos diodos implementados no sistema encontrava-se em torno de 0,58V, comum a diodos 1N4007. Uma alternativa para minimizar essa queda de tensão seria a substituição dos diodos de bloqueio por um MOSFET, que é capaz de reduzir em 10 vezes esse valor.

Todos os diodos mencionados foram soldados ao circuito, sendo visualmente identificáveis para o caso dos diodos de bloqueio e ocultos por detrás dos módulos de células fotovoltaicas para o caso dos diodos de *bypass*.

A imagem do recorte do sistema de abastecimento de energia por painéis solares pode ser analisado na Figura 17.

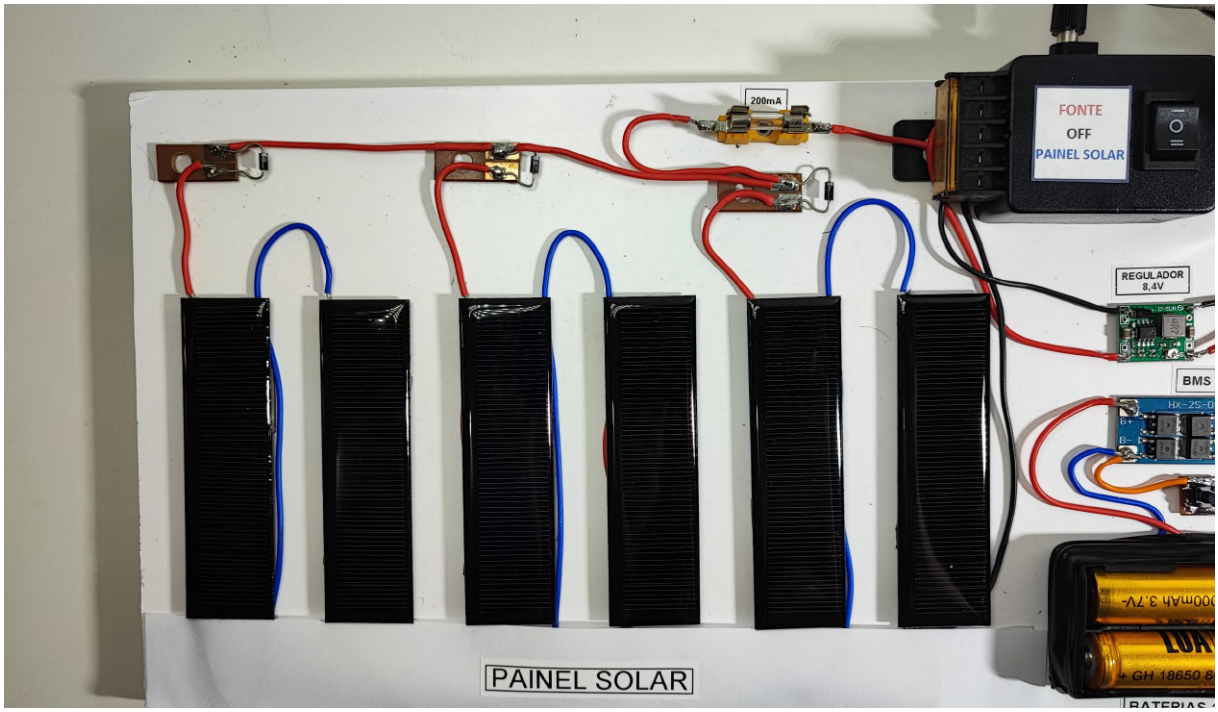

Figura 17: Visão superior do Abastecimento de Energia por Células Fotovoltaicas.

2.4.2.2 Alimentação Auxiliar Externa

O carregamento das baterias 18650 é facilitado através do sistema de alimentação auxiliar externa adicionado ao ProtoCube. Apesar desse tipo de sistema não ser candidato a ser embarcado em um nanossatélite real, foi de suma importância para agilizar a operação e os testes deste projeto. Na Figura 18, pode-se verificar a existência de dois componentes: Fonte de alimentação chaveada e um regulador de tensão e corrente (CC/CV).

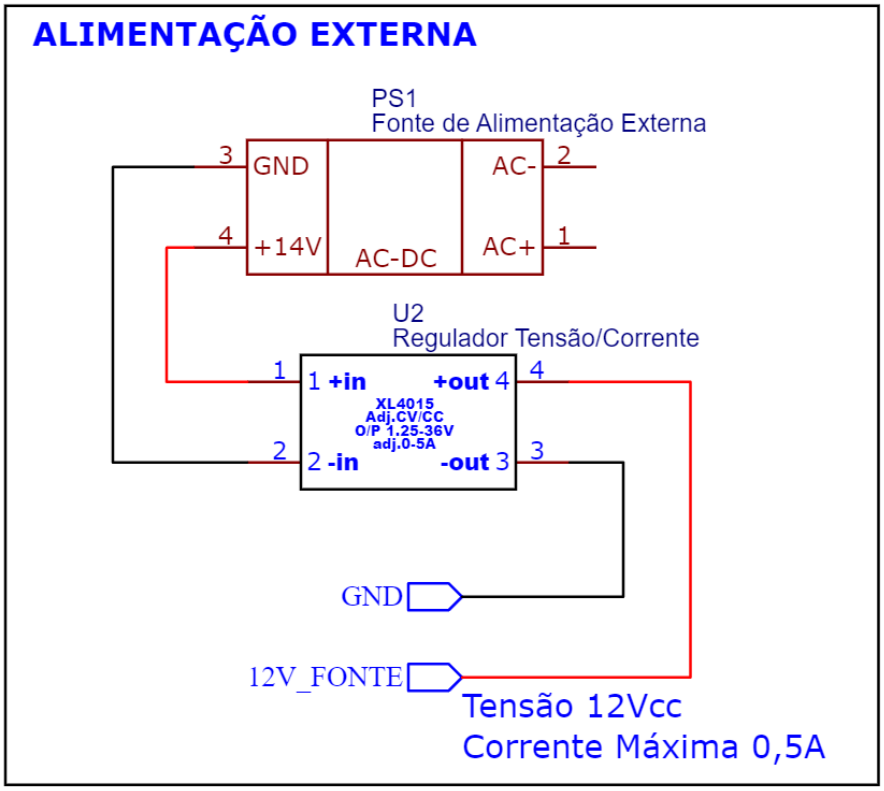

Figura 18: Esquemático do Sistema de Alimentação Externa.

A fonte chaveada foi ajustada para uma saída de 14V com o intuito de permitir maior liberdade de ajuste de tensão no regulador CC/CV. O ajuste da tensão do regulador foi configurada para 12V, enquanto a corrente máxima alvo foi determinada em 0,5A. Esses ajustes foram possíveis devido à existência de dois potenciômetros posicionados na parte traseira do módulo regulador XL4015.

A corrente máxima foi definida em 0,5A baseando-se nas especificações técnicas das baterias 18650 (Tabela 1), que trabalham com uma corrente de carga típica em torno desse valor. Buscar estar de acordo com as fichas técnicas dos componentes utilizados é imprescindível para assegurar a integridade e a vida útil dos mesmos.

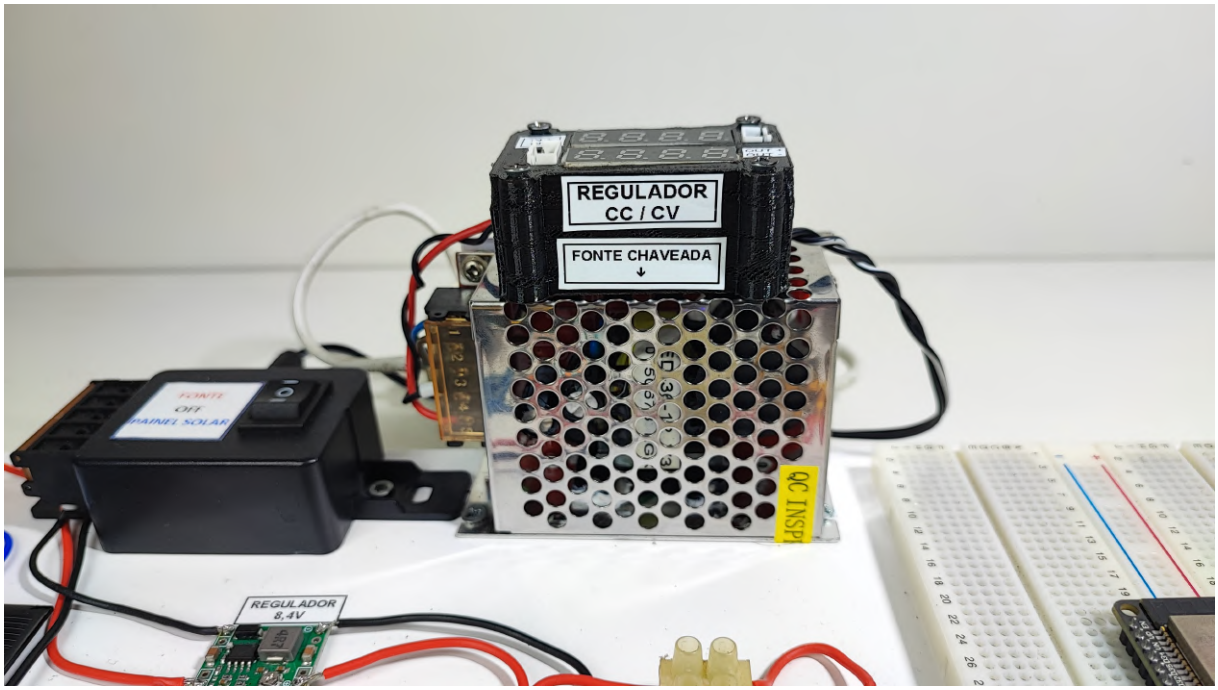

Figura 19: Conjunto composto por fonte chaveada e regulador de tensão e corrente. À esquerda, também é possível observar o botão de alternância dos modos de carregamento das baterias: alimentação externa ou painéis solares.

Para que o sistema de reabastecimento de energia do EPS possua praticidade na alternância dos meios de fornecimento, foi acrescentado uma chave de 3 posições, que propicia a fácil ativação dos modos disponíveis:

- **Fonte de Alimentação Auxiliar Externa:** carregamento das baterias realizado a partir da fonte chaveada;
- **Painéis Solares:** carregamento das baterias por meio das células fotovoltaicas do ProtoCube;
- **OFF:** sistemas de carregamento das baterias desligado.

Na Figura 19, são exibidas a fonte chaveada, o regulador CC/CV e o botão de alternância dos modos de abastecimento.

2.4.2.3 Suprimento de Energia

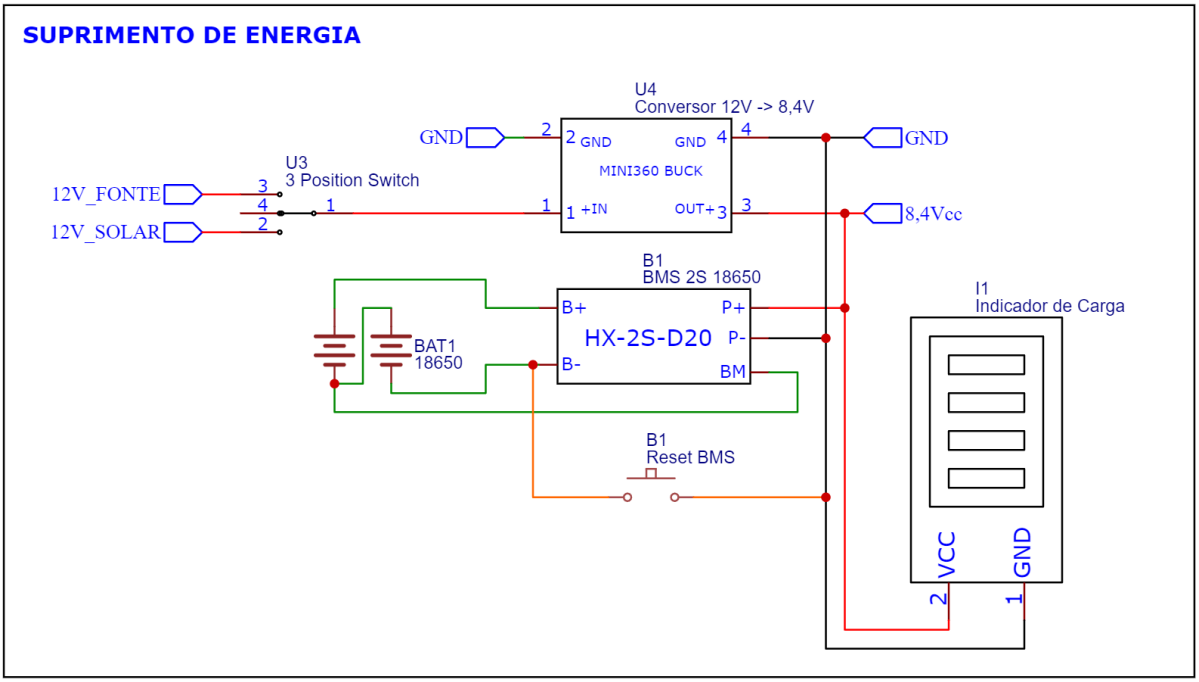

Figura 20: Esquemático do Suprimento de Energia do ProtoCube.

Os principais componentes responsáveis pelo fornecimento de energia para o OBC e sensores estão apresentados no esquemático da Figura 20. Nela pode-se obervar:

- Regulador de Tensão step down para 8,4V;
- Duas baterias 18650 8000mAh;
- Módulo BMS HX-2S-D20 para proteção das baterias 18650;
- Botão de Reset para o BMS;
- Indicador de carga das baterias.

Os componentes foram devidamente soldados e interconectados para suprir a demanda energética do OBC. Um detalhe adicionado ao protótipo está no botão de

reset do BMS. Quando as baterias 18650 são totalmente removidas e reinseridas no sistema ou se há a ativação da proteção contra sobrecarga, o MOSFET do módulo não se recupera do estado desconectado automaticamente. Para retornar ao estado ativo, deve-se reiniciar um procedimento de carga das baterias ou fechar um contato entre os pólos B- e P-. O botão de reset cumpre esse papel de fechar o contato necessário para reativar o BMS.

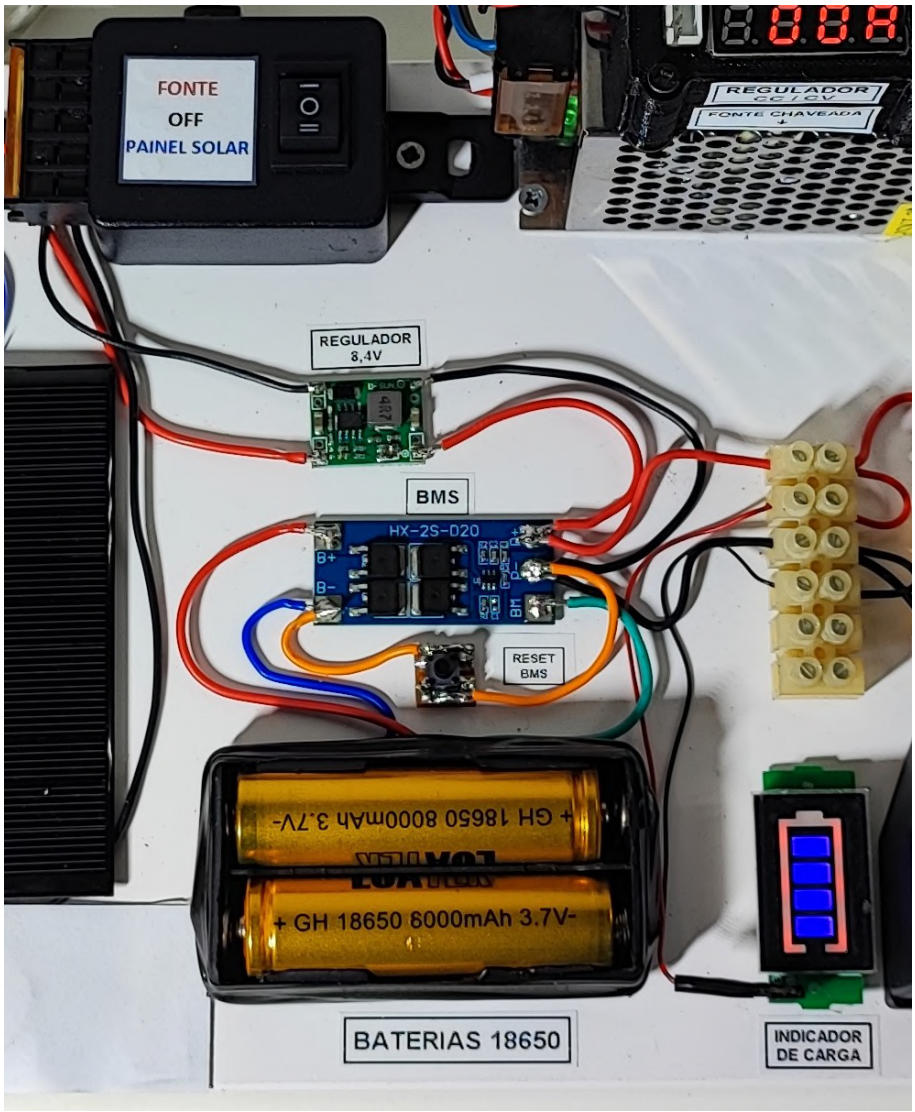

Figura 21: Visão superior dos componentes de Suprimento de energia do ProtoCube.

Importante relembrar que o módulo HX-2S-D20 não possui a funcionalidade de balanceamento de carga. Portanto, foi necessário carregar as baterias 18650 de maneira independente até que estivessem com nível de tensão similar (dentro do intervalo de 0,01V). A partir de então, pode-se carregar as baterias a partir do BMS normalmente, notando-se que os ciclos de carga e descarga são realizados de forma equilibrada entre ambas as células.

Para facilitar a identificação do nível de carga das baterias, foi acrescentado um módulo indicador de carga, que fornece uma visualização da porcentagem de carga disponível no conjunto de baterias 18650. A Tabela 5 exibe detalhes dos valores convertidos para os níveis do indicador.

Tabela 5: Valores mínimos para indicação do nível de carga.

#	25%	50%	75%	100%
1S	3,3V	3,5V	3,7V	3,9V
2S	6,6V	7,0V	7,4V	7,8V
3S	9,9V	10,5V	11,1V	11,7V
4S	13,2V	14V	14,8V	15,6V
5S	16,5V	17,5V	18,5V	19,5V
6S	19,8V	21V	22,2V	23,4V
7S	23,1V	24,5V	25,9V	27,3V
8S	26,4V	28V	29,6V	31,2V

2.4.2.4 Computador de Bordo, Sensores e Comunicação Serial

Os últimos componentes a serem detalhados compreendem o conjunto formado pelo OBC, Sensores 9-DOF (módulo MPU-9250) e interface de comunicação serial. Como já mencionado anteriormente, o subsistema de Computador de Bordo age como o centro de processamento de todos os dados adquiridos, além ser passível de realizar comandos que podem, por exemplo, acionar atuadores de um subsistema de controle de atitude.

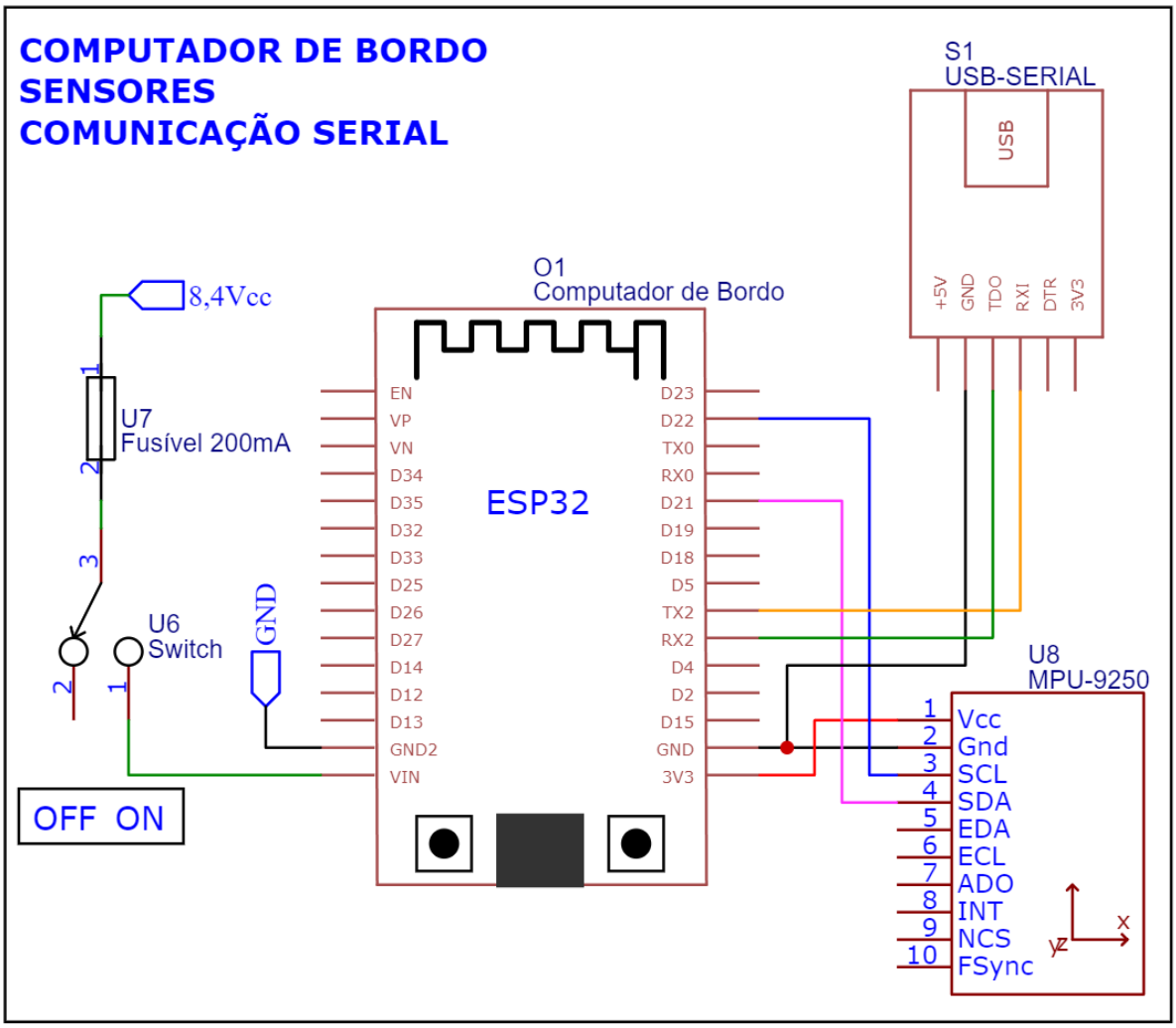

Figura 22: Esquemático do Computador de Bordo, dos sensores e da interface de comunicação serial.

Na Figura 22, pode-se identificar o módulo ESP32 DevKit v1 e suas conexões através dos periféricos I2C e UART. A comunicação através do protocolo I2C possibilita ao OBC a aquisição dos valores medidos pelos sensores acelerômetro, giroscópio e magnetômetro, enquanto a comunicação serial permite que esses dados sejam enviados a um computador para apuração dos resultados através do software Matlab.

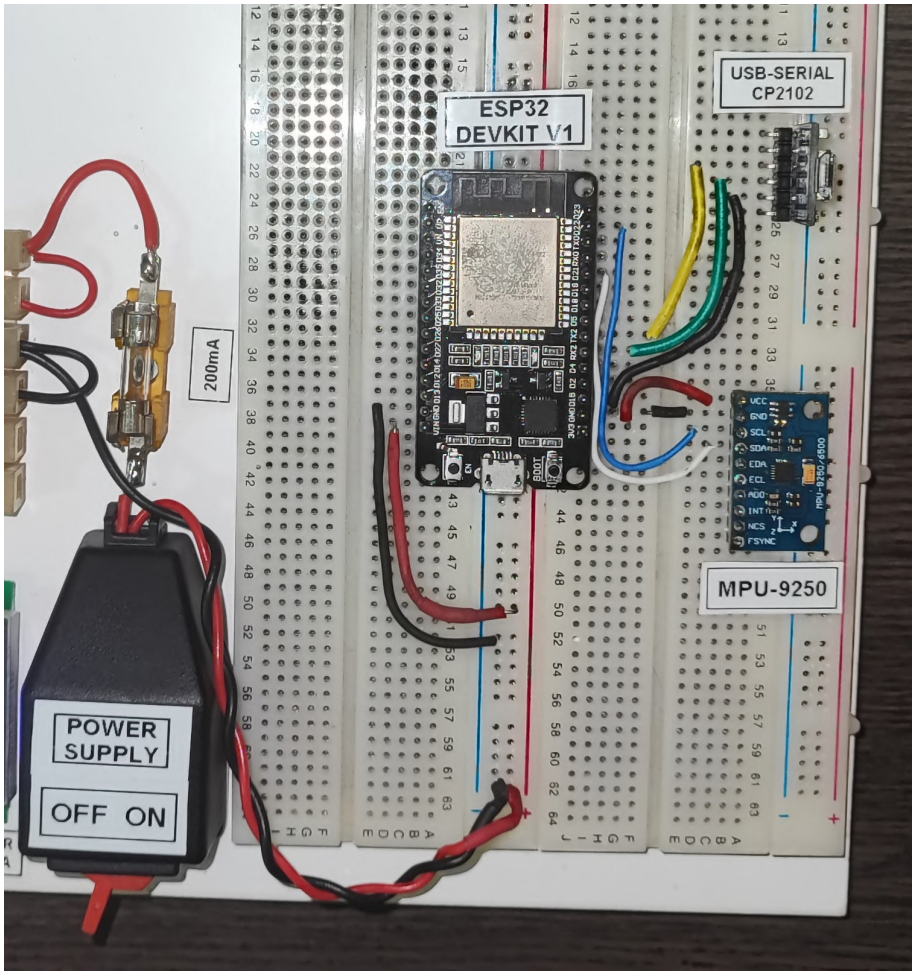

Figura 23: Visão superior do OBC, dos sensores e da interface de comunicação serial.

Como item adicional a implementação requisitada, foi inserido um switch com função Liga e Desliga da alimentação dos componentes aqui destacados. Com isso, é possível interromper completamente o funcionamento do OBC para permitir, por exemplo, que as baterias do EPS sejam carregadas sem interferência desses elementos.

Na Figura 23, estão dispostos fisicamente todos os componentes mencionados nesta seção, inclusive um fusível de proteção para correntes a partir de 200mA.

Com isso, são finalizadas as informações a respeito da implementação física do ProtoCube, restando, apenas, o detalhamento da implementação computacional dos algoritmos de processamento do OBC e do MATLAB.

2.4.3 Computacional

Para que toda a implementação física projetada e construída cumpra seu propósito, é necessário o desenvolvimento de códigos a serem executados no OBC e no Matlab. Para realizar essa tarefa, serão discutidos os detalhes a respeito nas duas subseções a seguir.

2.4.3.1 Computador de Bordo e Sensores Inerciais

O módulo ESP32 da *Espressif Systems* conta com uma plataforma robusta, composta por diversos periféricos úteis para agilizar o desenvolvimento de produtos, que, aliada à disponibilização do framework ESP-IDF para SoCs da Espressif, proporciona um ambiente de desenvolvimento acelerado e eficiente. Esta eficiência também se relaciona com a sua linguagem base para a implementação de aplicações: a linguagem C (Figura 24). Com isso, é possível executar tarefas eficientes em tempo e energia nos projetos envolvendo a gama de produtos ESP32. Uma alternativa mais recente, porém menos eficiente no momento, seria o uso do MicroPython, uma implementação de Python 3 para microcontroladores (<https://micropython.org/>). Mais informações sobre o ESP-IDF podem ser encontradas em (ESPRESSIF SYSTEMS (SHANGHAI) CO., LTD., 2023a).

A fim de permitir a comunicação adequada com os sensores disponíveis no módulo MPU-9250, foi utilizada como base o trabalho desenvolvido por Simon Werner, disponível em (WERNER, 2023), que proporcionou grande auxílio para este projeto.

A estrutura do código pode ser dividida em 3 seções:

- Drivers para interação com o módulo MPU-9250;
- Biblioteca para comunicação serial;
- Script principal, envolvendo também as etapas de calibração do módulo de sen-

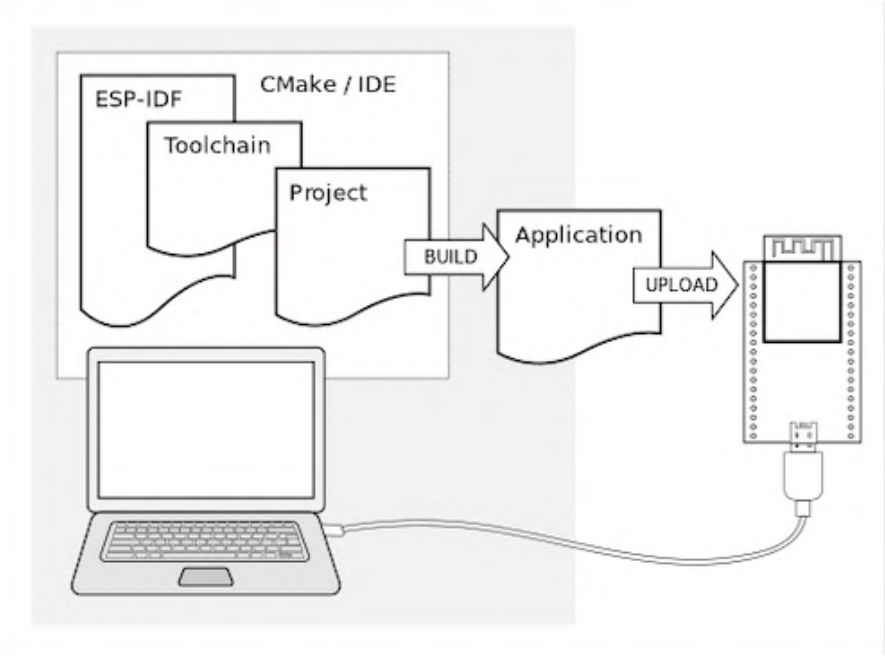

Figura 24: Diagrama das etapas do desenvolvimento de projetos para módulos ESP32.

sores 9-DOF.

Os drivers para o módulo MPU-9250 incluem, além dos parâmetros de inicialização do barramento I2C, validações da comunicação dos sensores e o mapeamento dos registros disponíveis tanto para escrita como leitura. O mapa dos registros encontra-se disponível em (INVENSENSE INC., 2013).

Para a comunicação I2C, é importante considerar os endereços I2C dos componentes, pois, durante o processo de transmissão, esse valor é utilizado para identificar os destinatários da comunicação, evitando assim que mais de um elemento responda a mesma solicitação. Os endereços foram definidos como a seguir:

- **MPU-9250 (acelerômetro e giroscópio):** 0x68;
- **AK8963 (magnetômetro):** 0x0c (o sensor magnetômetro atua como um dispositivo independente, possuindo, portanto, um endereço I2C diferente do MPU-9250).

Os seguintes parâmetros de escala completa para os sensores acelerômetro e giroscópio foram definidos através da configuração:

- **Acelerômetro:** $\pm 4g$
- **Giroscópio:** $\pm 250^{\circ}/s$

Como o magnetômetro possui intervalo de escala completa fixo em $\pm 4800\mu T$, não foi necessário ajuste. É importante levar em conta esses parâmetros durante os testes para que não sejam excedidos os valores mencionados.

Outro parâmetro decisivo para a comunicação I2C com esses sensores é a frequência do *master clock*, que foi definida em 200.000 Hz. O sinal do *clock* é fundamental para o reconhecimento adequado das instruções transmitidas através do barramento.

Para a comunicação serial através da UART entre o OBC e o computador executando o software Matlab, os seguintes parâmetros foram considerados:

- **Baud Rate:** 115200;
- **Data Bits:** 8;
- **Paridade:** desabilitada;
- **Bits de Parada:** 1;
- **Flow Control:** desabilitado.
- **Pino RX:** 16;
- **Pino TX:** 17.

Definindo-se os 5 primeiros parâmetros em ambos os endpoints, a comunicação pode ser realizada com sucesso.

No script principal ("main.c"), encontram-se todas as importações de bibliotecas

necessárias, que incluem, por exemplo, a biblioteca "uart.h" (com as funções relacionadas à interface serial), a biblioteca "mpu9250.h" (drivers e funções relacionadas ao módulo 9-DOF) e as bibliotecas referentes ao FreeRTOS, o Sistema Operacional de Tempo Real que atua no plano de fundo de todos processos executados pelo microcontrolador.

O script "main.c" atua com a seguinte lógica:

1. É verificado se o parâmetro "CONFIG_CALIBRATION_MODE" está definido como verdadeiro ou falso no menu de configuração do ESP-IDF SDK. Em caso negativo, segue-se para as fases seguintes. Porém, se positivo, o algoritmo entra em modo de calibração para identificar os valores de referência para o módulo MPU-9250 utilizado e descrito nos sub itens a seguir.
   (a) Calibração do giroscópio mantendo-o estático;
   (b) Calibração do acelerômetro através do direcionamento dos seus eixos para cada um dos sentidos possíveis durante 10 segundos;
   (c) Calibração do magnetômetro rotacionando o dispositivos em todos os eixos até que os valores máximos e mínimos não se alterem.
2. A variável de calibração "cal" é declarada a fim de ser utilizada para a correção dos valores adquiridos pelos sensores;
3. MPU-9250 e UART inicializados;
4. É acessado um laço "while" onde todos os processos a seguir são repetidos até a reinicialização ou desligamento dos componentes envolvidos (ESP32, MPU-9250 e USB-Serial CP2102).
   (a) Aquisição das medições do Acelerômetro, giroscópio e magnetômetro;
   (b) Transformação dos eixos dos valores adquiridos para a mesma orientação dos eixos do magnetômetro;

(c) Envio de uma mensagem composta pela string "data:" seguida de todos as componentes x, y e z dos valores adquiridos pelos sensores. Esse envio é realizado através da UART para o computador com o script de recebimento dos dados em execução no Matlab.

A variável de calibração obtida durante os testes com o módulo MPU-9250 utilizado neste projeto encontra-se a seguir:

```
calibration_t cal = {
.mag_offset = {.x = -13.234375, .y = 80.871094, .z = -38.414062},
.mag_scale = {.x = 0.980424, .y = 1.006865, .z = 1.013324},
.accel_offset = {.x = 0.039083, .y = 0.070368, .z = 0.136644},
.accel_scale_lo = {.x = 1.016390, .y = 1.036406, .z = 1.069427},
.accel_scale_hi = {.x = -0.978780, .y = -0.962060, .z = -0.925534},

.gyro_bias_offset = {.x = -0.947105, .y = 2.625738, .z = 0.588435}
};
```

Os códigos para o OBC encontram-se disponíveis no repositório do Github em (OLIVEIRA, 2023), com o script "main.c" e o "uart.c" também expostos nos Apêndices A e B, respectivamente.

2.4.3.2 MATLAB

A utilização do MATLAB para este projeto recai sobre a necessidade de uma melhor visualização e compreensão dos dados adquiridos. A partir do MATLAB, foi possível receber os dados, transmitidos por comunicação serial UART pelo OBC, contendo os valores medidos pelos sensores 9-DOF. Com isso, a visualização gráfica em tempo real foi implementada, permitindo com que o usuário avalie prontamente o comportamento das medições ao longo do período de teste.

A seguir, é descrita a estrutura dos códigos implementados no MATLAB, apontando descrições dos seus funcionamentos:

- **main.m:** composto pelo script principal de execução. Compreende todas as declarações de variáveis, processos de leitura dos dados pela porta serial, adequação dos dados recebidos para estrutura de manipulação no MATLAB e exibição dos gráficos em tempo real, de acordo com a evolução da recepção no tempo;
- **processData.m:** recebe os dados recebido pela comunicação serial e os organiza em *arrays* para que sejam manipulados adiante;
- **g2MetersPerSecSquared.m:** converte o array de valores obtidos pelo acelerômetro de $G's$ para m/s^2.

O andamento da implementação pode ser enumerada nos passos abaixo:

1. "main.m" é executado;
2. Os parâmetros da comunicação serial são definidos de acordo com os preestabelecidos no OBC. Neste caso, os únicos parâmetro não padrão que precisam ser ajustados são:
    - comPort = 'COM4';
    - baudRate = 115200;
    - configureTerminator(s, "LF"). Este parâmetro define '\n' como elemento de terminação de linha.
3. Inicialização dos Arrays de dados;
4. Definição do contador de tempo, garantindo um *timeout* de 20 segundos em caso da ausência de recebimento de dados via porta serial;
5. Inicialização dos gráficos para que sejam exibidos em tempo real no decorrer da

execução;

6. Criação do loop para recebimento dos dados enquanto a porta serial esteja aberta e haja transmissão de dados dentro do *timeout* definido:

   (a) Leitura dos dados através da porta serial;

   (b) Armazenamento e concatenação dos dados em uma matriz;

   (c) Processamento dos dados recebidos através da função 'processData()', retornando 3 *arrays* separados: accelArray, gyroArray, magArray;

   (d) Determinação do tempo decorrido desde o início da execução e concatenando-o com um array 'timeArray' composto pelos valores double correspondentes ao tempo em que cada amostra de dados foi recebida e armazenada.;

   (e) Atualização dos gráficos baseando-se nos novos dados recebido;

7. Encerramento da execução após o timeout estabelecido ou desconexão da porta serial.

   Os códigos desenvolvidos encontram-se disponíveis nos Apêndices C, D e E.

2.5 Testes e Resultados

Com toda a implementação lógica, física e computacional executada e descrita, nesta seção, serão abordados os testes realizados para validar o funcionamento dos subsistemas do projeto ProtoCube. Foram realizados testes das seguintes naturezas:

- Tensão de capacidade máxima das baterias 18650;
- Desempenho de descarga do EPS;
- Consumo exigido por OBC e Sensores 9-DOF
- Capacidade de carga das baterias a partir da conversão de energia solar em

elétrica por meio das células fotovoltaicas;

- Aquisição dos sensores 9-DOF.

2.5.1 Tensão de capacidade máxima das baterias 18650

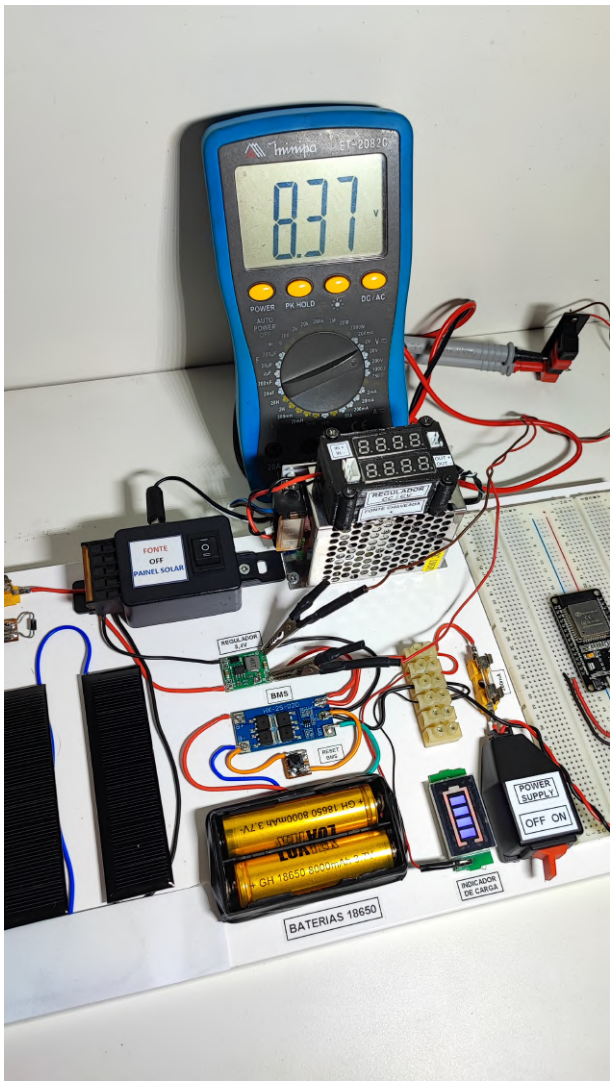

Figura 25: Teste de tensão máxima das baterias.

Foi realizado o teste de tensão máxima das baterias 18650, sendo mantida uma tensão levemente abaixo da máxima recomendada de 8,4V. Demonstra também a capacidade de redução do regulador de tensão MP1584, que, após ajuste de sua

tensão de saída para 8,4V, não sofreu desvio durante todo o período deste projeto.

2.5.2 Desempenho de Descarga do Sistema de Suprimento de Energia (EPS)

Com o intuito de avaliar o desempenho do EPS durante um evento de descarga, foi realizado um teste utilizando um conjunto de resistores com resistência equivalente de 25Ω e potência de 5W. Foi adotada uma carga diferente do conjunto OBC e sensores devido ao tempo necessário para acompanhar uma variação considerável no nível de tensão das baterias. Com as resistências mencionadas, foi possível acelerar este processo e avaliar o comportamento de descarga ao longo de uma janela de 10 minutos. A Figura 26 exibe o conjunto de elementos do teste.

Para avaliar o comportamento de descarga do EPS, foram registradas medidas em intervalos de 30 segundos durante um período total de 10 minutos, utilizando com o referência um cronômetro posicionado ao lado do conjunto de teste. Importante ressaltar que, antes do início do teste, o estado do reabastecimento de energia do EPS estava no modo “OFF”, ou seja, tanto a alimentação externa quanto o sistema de células fotovoltaicas estavam inativos. A Figura 27 exibe imagens do início e do fim do período avaliado.

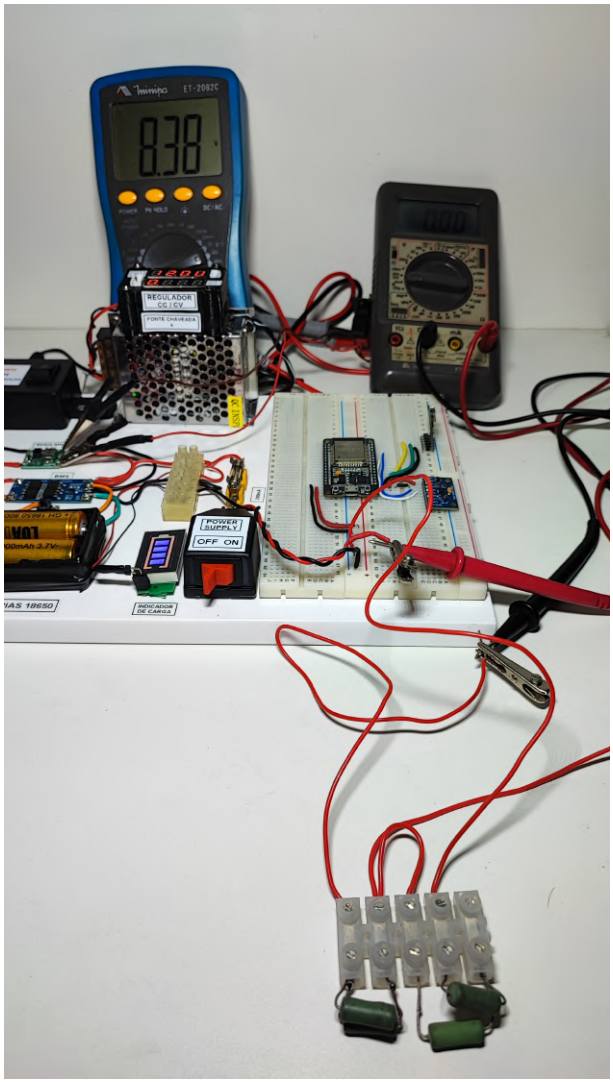

Figura 26: Conjunto de teste de descarga do EPS. Além do conjunto de resistores que possuem 25 ohms, vê-se o multímetro a esquerda configurado para leitura da tensão das baterias e outro à direita configurado para avaliar corrente consumida pela carga.

Figura 27: Imagens do início e do final, respectivamente, do teste de descarga do EPS.

Com base nos resultados alcançados, foi possível obter uma tabela e um gráfico do decaimento da tensão ao longo do tempo do teste, que podem ser conferidos na Tabela 6 e na Figura 28.

Tabela 6: Resultados do teste de descarga das baterias com resistência de 25Ω. Deve-se levar em conta que o instante inicial representa um momento imediatamente anterior à ativação da carga ôhmica.

Tempo decorrido (s)	Tensão das Baterias (V)	Corrente Consumida (A)
0	8,35	0
30	8,14	0,32
60	8,09	0,32
90	8,07	0,32
120	8,05	0,32
150	8,03	0,32
180	8,02	0,32
210	8,01	0,32
240	8,00	0,31
270	7,99	0,32
300	7,99	0,31
330	7,98	0,31
360	7,97	0,31
390	7,97	0,31
420	7,96	0,31
450	7,96	0,31
480	7,95	0,31
510	7,95	0,31
540	7,95	0,31
570	7,94	0,31
600	7,94	0,31

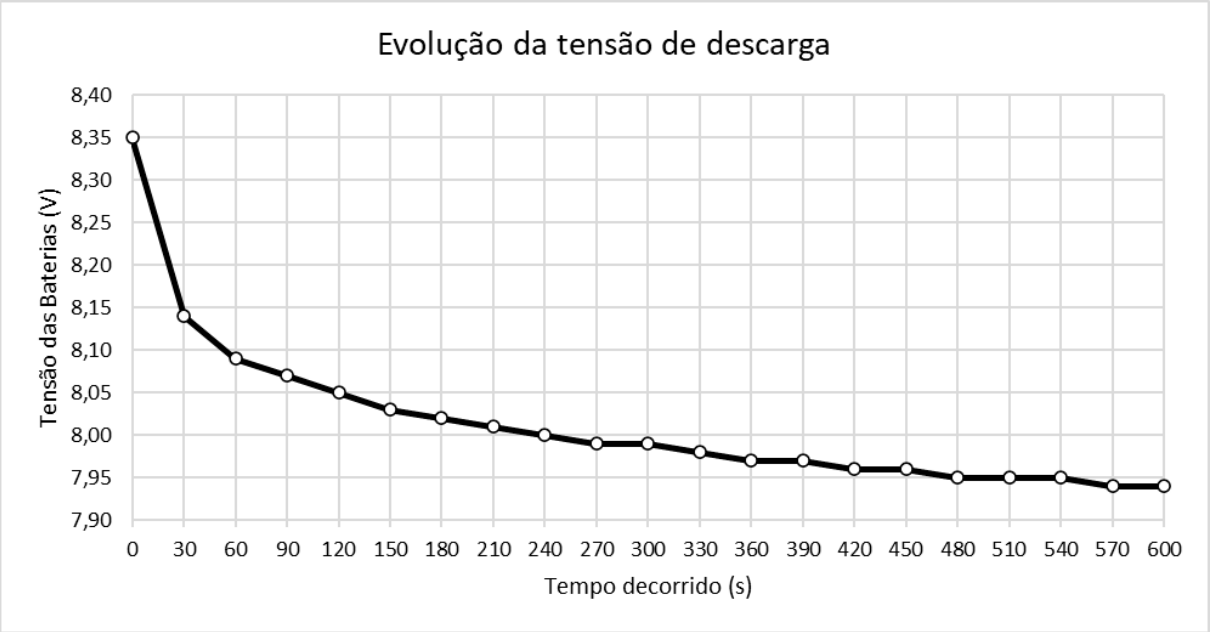

Figura 28: Teste de Descarga do EPS com resistência ôhmica de 30Ω.

2.5.3 Consumo exigido por Computador de Bordo e Sensores Inerciais

Outro teste fundamental para se analisar a capacidade de suprimento de energia do EPS está na verificação do consumo exigido pela combinação entre OBC e sensores inerciais. Foi verificado um consumo estável no valor de 50mA, que pode ser verificado na Figura 29.

Considerando que o conjunto de baterias 18650, responsável por fornecer a energia necessária para o funcionamento dos componentes do projeto, possui capacidade de carga de até 8000mAh, pode-se estimar uma duração aproximada máxima de 160 horas (mais de 6 dias) de atividades do ProtoCube sem recarga do EPS.

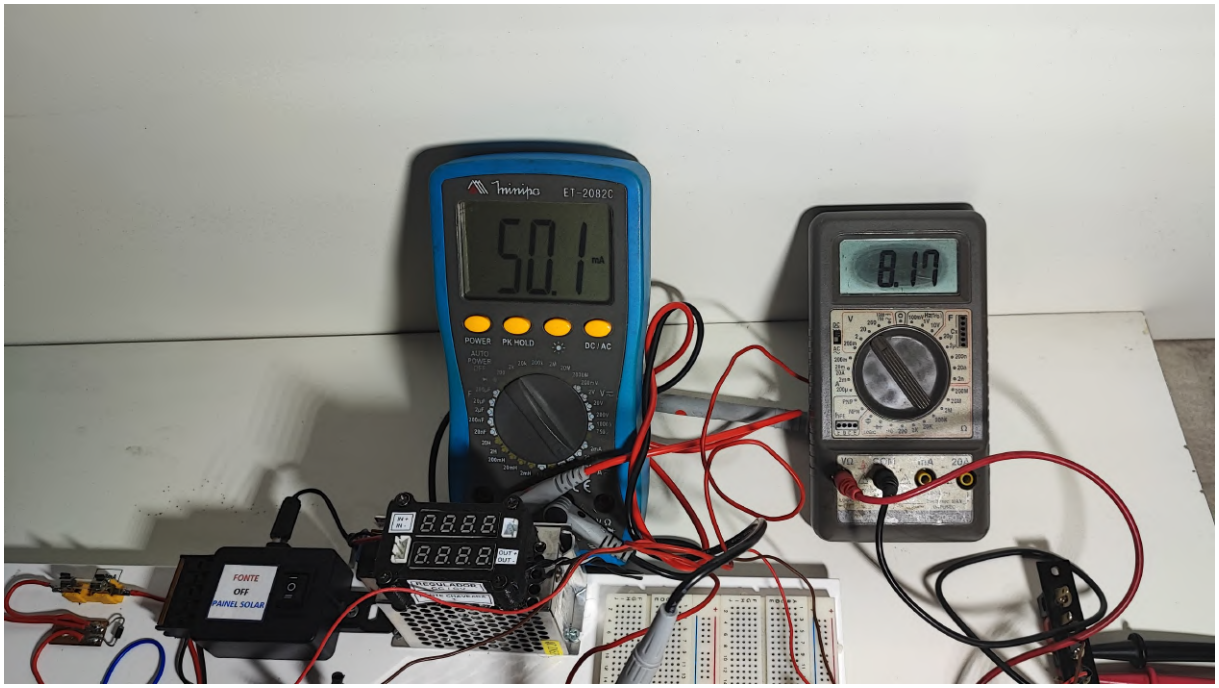

Figura 29: Teste de consumo do conjunto OBC e Sensores 9-DOF.

2.5.4 Capacidade de carga das baterias por células fotovoltaicas

Para que se pudesse avaliar o desempenho da implementação do sistema de reabastecimento de energia por células fotovoltaicas, foi realizado um teste cujo objetivo foi analisar a resposta dos painéis solares frente a um ambiente de alta exposição de luz.

Para isto, foi utilizada uma caixa com interior preenchido com papel alumínio como aparato bloqueador, visando favorecer a reflexão da luz e uma maior intensidade em consequência. Foram executadas duas aberturas na face superior da caixa para o encaixe de um refletor e uma lanterna com altas potências e apenas uma região exposta na base da caixa, onde as células fotovoltaicas poderiam receber a luz incidente. O aparato revestido de alumínio encontra-se na Figura 30.

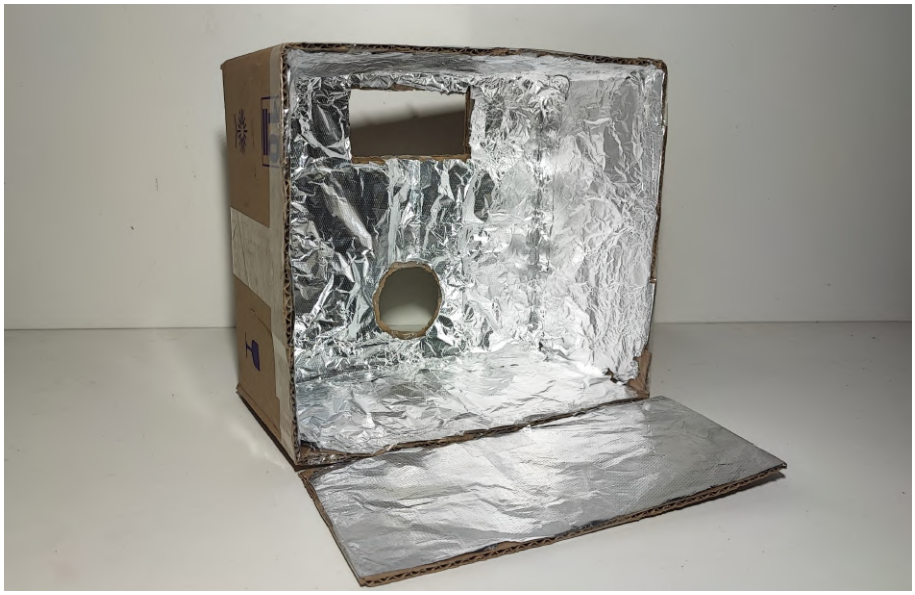

Figura 30: Aparato bloqueador revestido de papel alumínio utilizado no teste de geração de energia dos painéis solares.

Além da estratégia mencionada acima, também foi utilizado um luxímetro, dispositivo responsável por realizar medição de iluminância. O luxímetro empregado está destacado na Figura 31.

A configuração do teste foi montada e, durante a sua realização, nenhuma carga ou componente foi inserida ao EPS, apenas visando verificar o desempenho de geração de energia e carregamento do conjunto de baterias. Uma imagem da configuração do teste pode ser encontrada na Figura 32.

Figura 31: Luxímetro MT-30, da marca R&D Instruments, usado nos testes de capacidade de geração dos painéis solares.

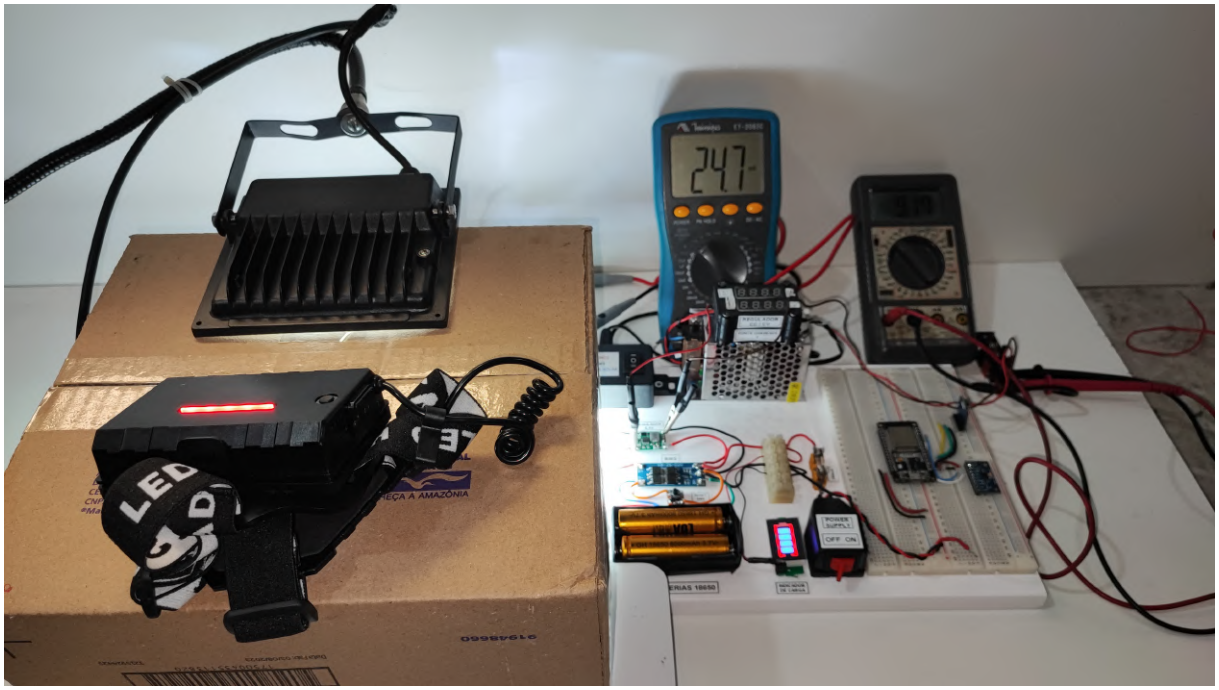

Figura 32: Visão superior do teste de carregamento via painéis solares.

Através desse teste, foi possível verificar os seguintes resultados:

- Tensão de carga produzida: 9,17V;
- Corrente elétrica produzida: 24,7mA;
- Iluminância: 47900 lx.

As imagens dos resultados obtidos encontram-se nas Figuras 33 e 34.

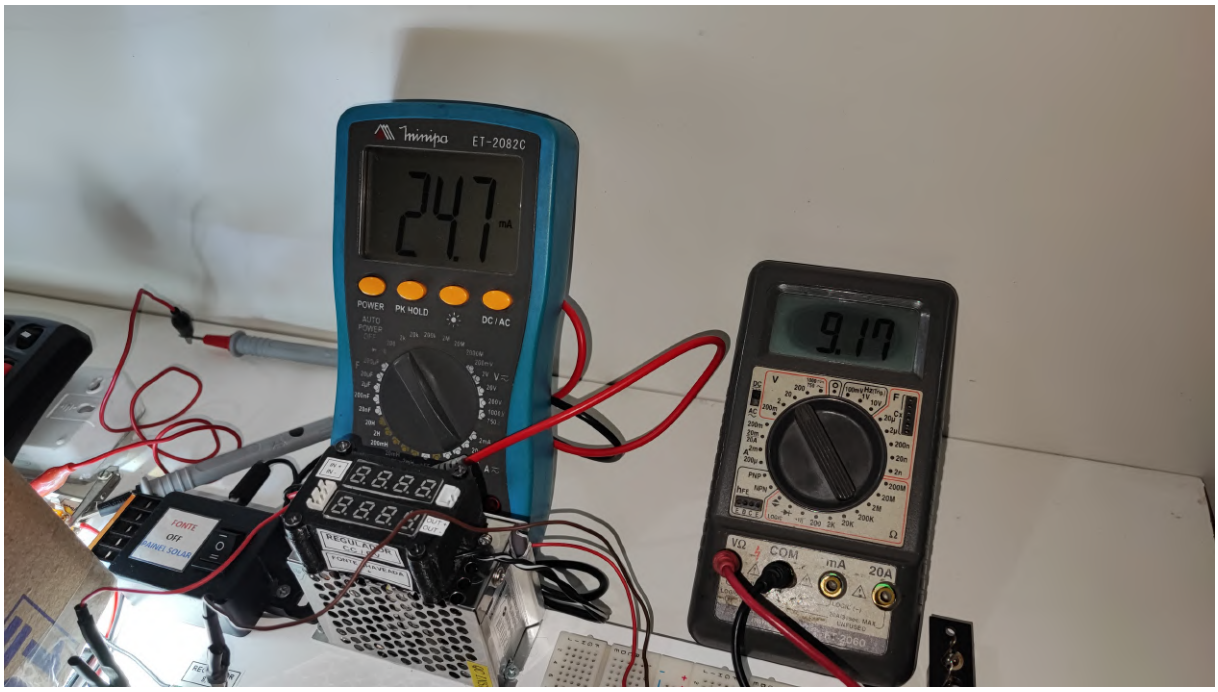

Figura 33: Medições de tensão e corrente geradas pelas células fotovoltaicas durante teste.

Figura 34: Medição realizada pelo luxímetro da iluminância detectada durante o teste.

2.5.5 Aquisição dos Sensores Inerciais

O último teste realizado para este projeto consiste na obtenção prática de resultados dos Sensores 9-DOF a partir de cenários pré determinados. Será realizada a seguinte bateria de exercícios:

- Rotações em +90°para cada um dos três eixos, retornando para o estado original ao final. A orientação deslocada em +90 graus deverá permanecer estática durante 30 segundos antes de retornar à posição inicial.

Será habilitado o fornecimento de energia e a consequente inicialização do OBS após 10 segundos da execução do script ”main.m” no MATLAB. Após isso, mais 10 segundos serão aguardados antes do início dos movimentos. O teste será encerrado aos 60 segundos, desligando a chave de fornecimento de energia do OBS.

Nas Figuras 35 e 36, podem ser verificadas imagens dos testes dos sensores 9-DOF.

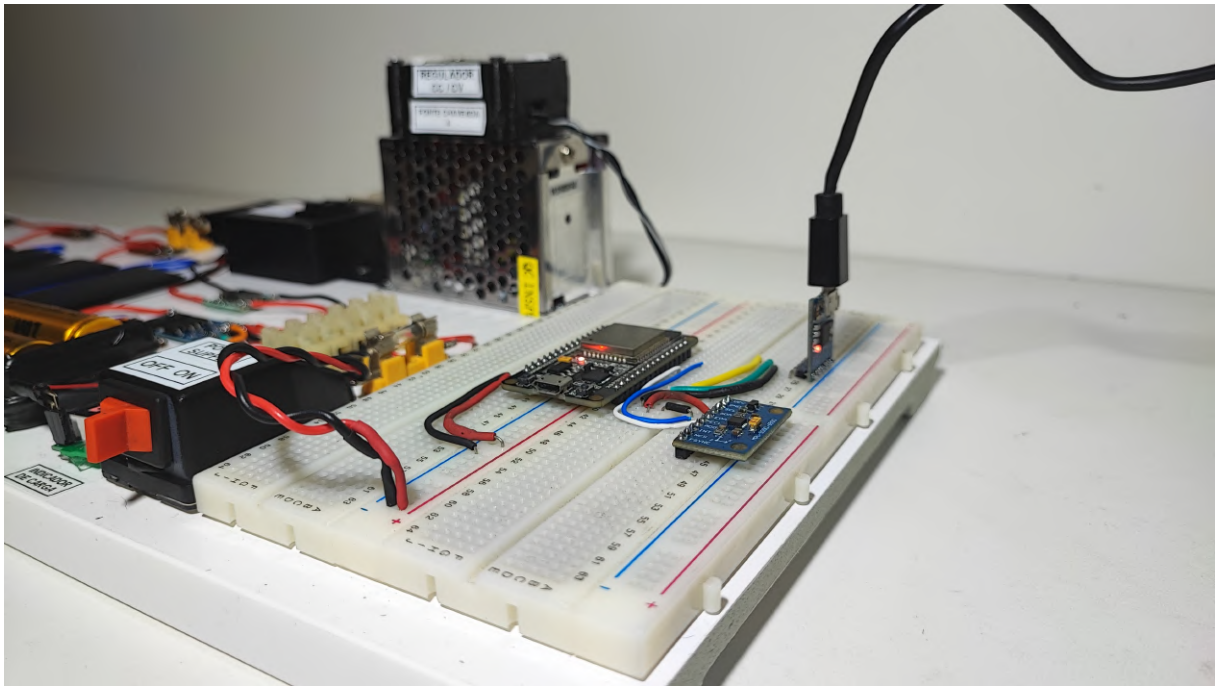

Figura 35: Teste em andamento para análise da aquisição de dados pelo sensores 9-DOF.

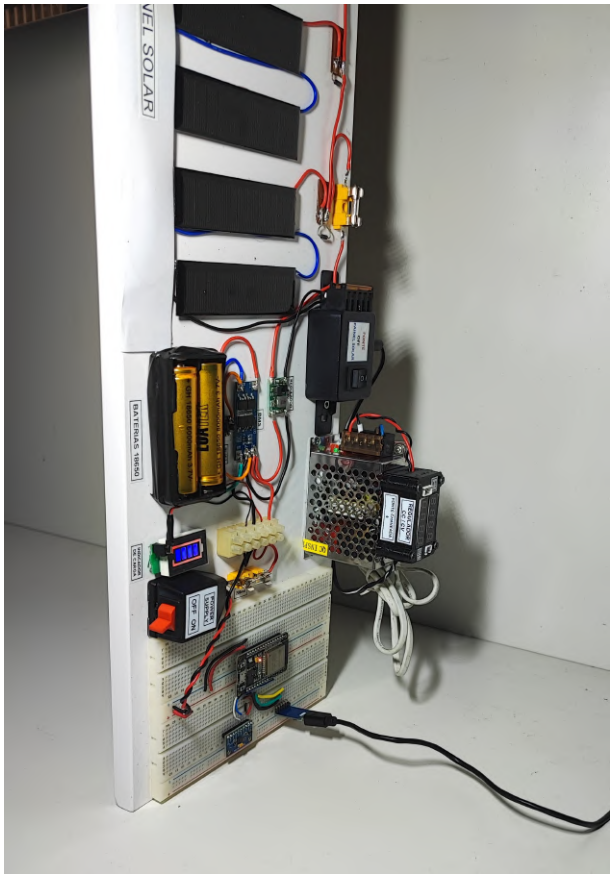

Figura 36: Momento do teste de aquisição dos sensores 9-DOF quando a aceleração máxima encontrava-se no sentido positivo do eixo Y.

A seguir, são apresentados os resultados para cada um dos seis cenários avaliados durante o teste.

2.5.5.1 Rotação $+x$

Iniciando-se a partir do cenário em que ocorre rotação positiva em relação ao eixo x, foi obtido o gráfico exibido na Figura 37.

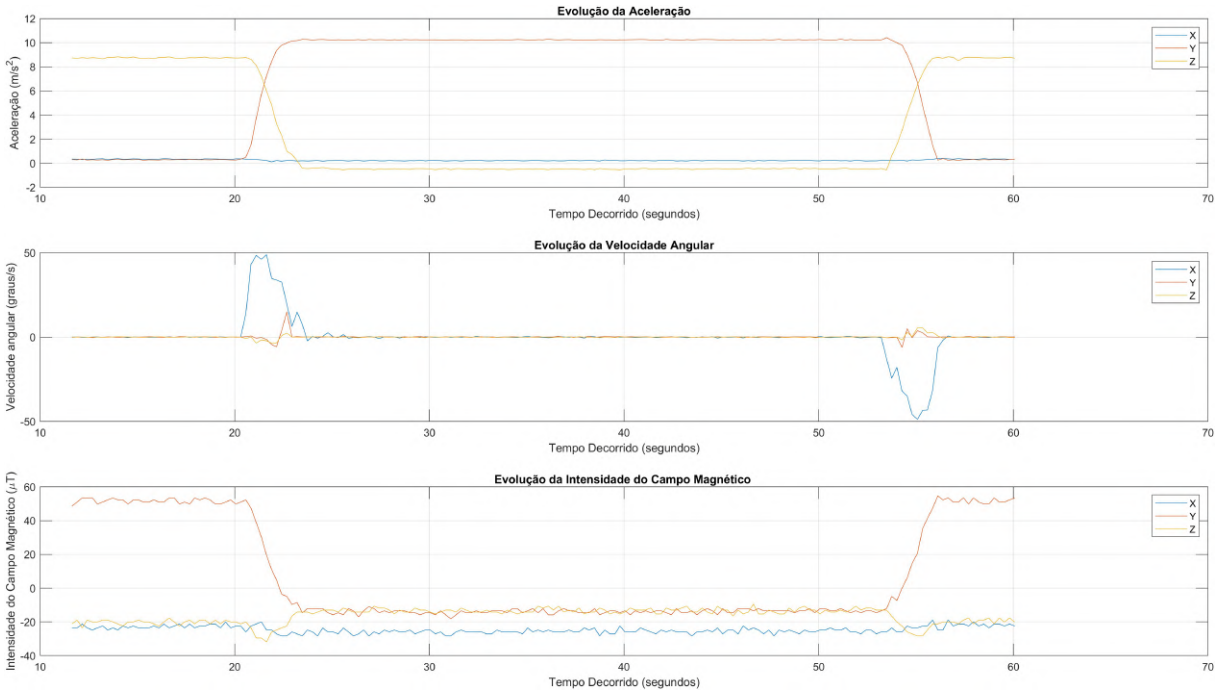

Figura 37: Rotação positiva em torno do eixo x do MPU-9250.

2.5.5.2 Rotação $-x$

Iniciando-se a partir do cenário em que ocorre rotação negativa em relação ao eixo x, foi obtido o gráfico exibido na Figura 38.

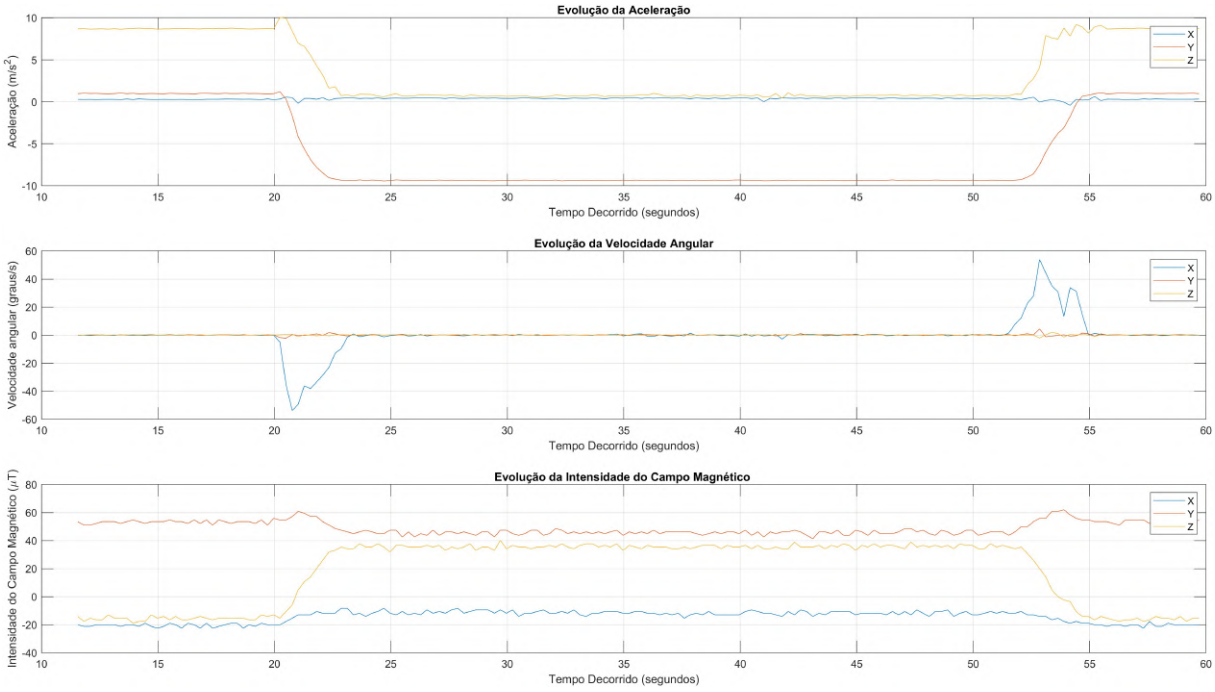

Figura 38: Rotação negativa em torno do eixo x do MPU-9250.

2.5.5.3 Rotação $+y$

Iniciando-se a partir do cenário em que ocorre rotação positiva em relação ao eixo y, foi obtido o gráfico exibido na Figura 39.

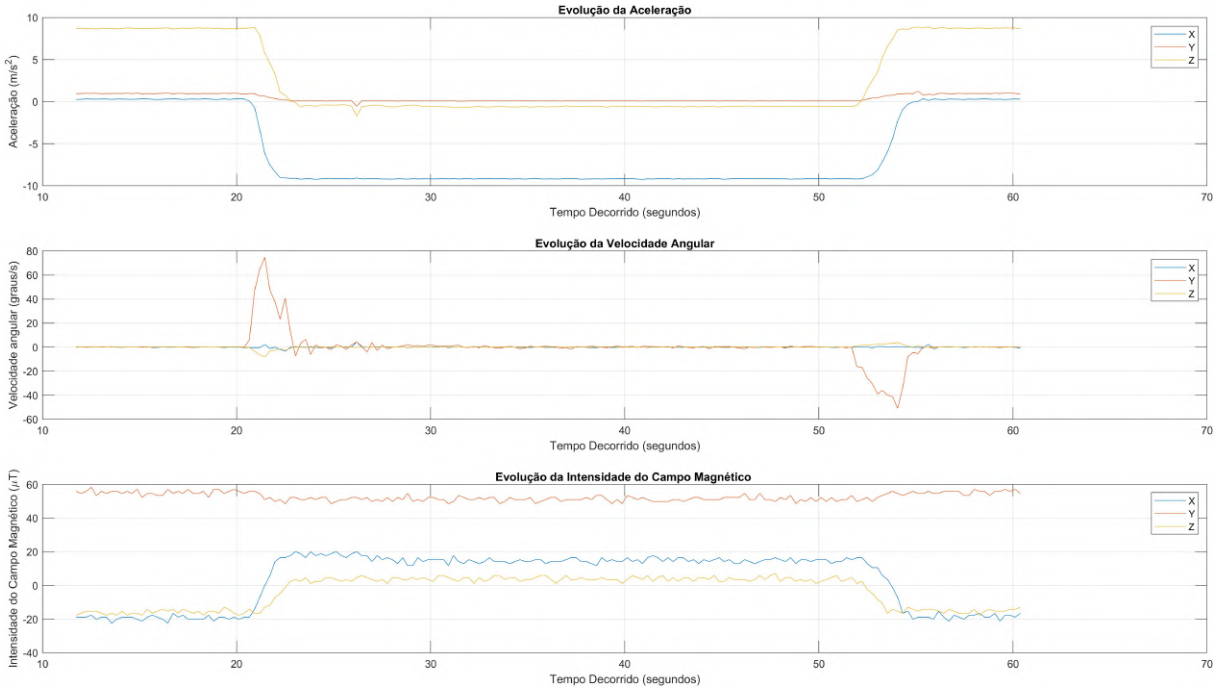

Figura 39: Rotação positiva em torno do eixo y do MPU-9250.

2.5.5.4 Rotação $-y$

Iniciando-se a partir do cenário em que ocorre rotação negativa em relação ao eixo y, foi obtido o gráfico exibido na Figura 40.

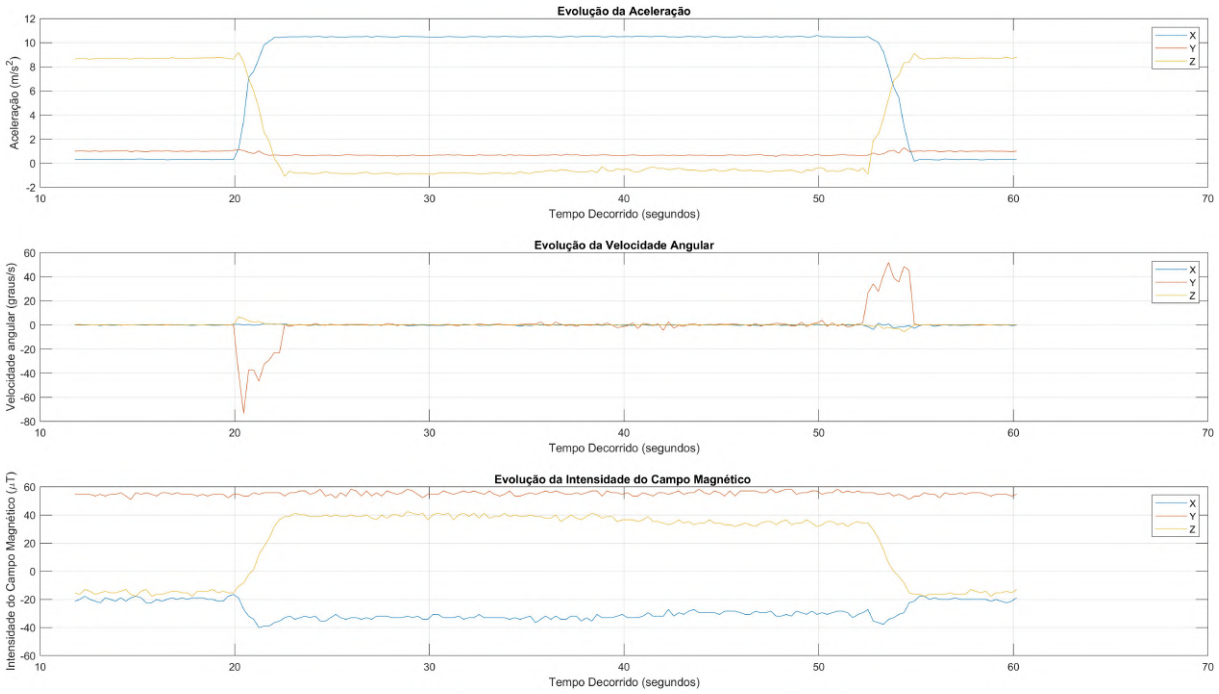

Figura 40: Rotação negativa em torno do eixo y do MPU-9250.

2.5.5.5 Rotação $+z$

Iniciando-se a partir do cenário em que ocorre rotação positiva em relação ao eixo z, foi obtido o gráfico exibido na Figura 41.

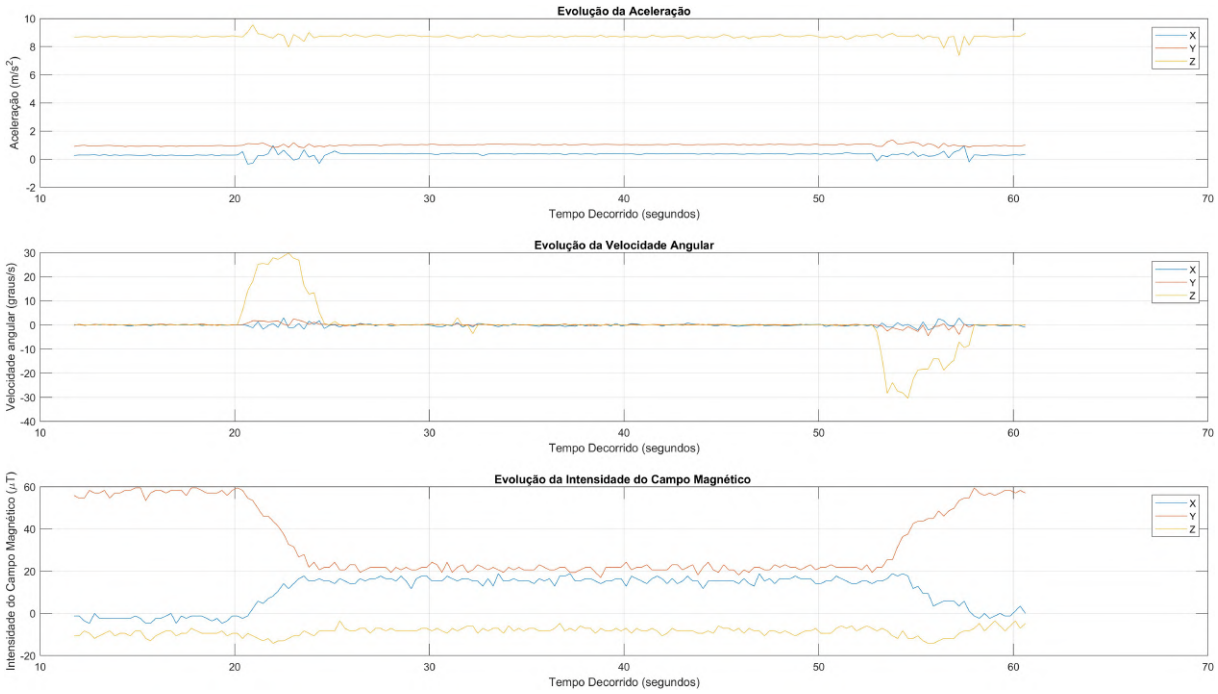

Figura 41: Rotação positiva em torno do eixo z do MPU-9250.

2.5.5.6 Rotação $-z$

Iniciando-se a partir do cenário em que ocorre rotação negativa em relação ao eixo z, foi obtido o gráfico exibido na Figura 42.

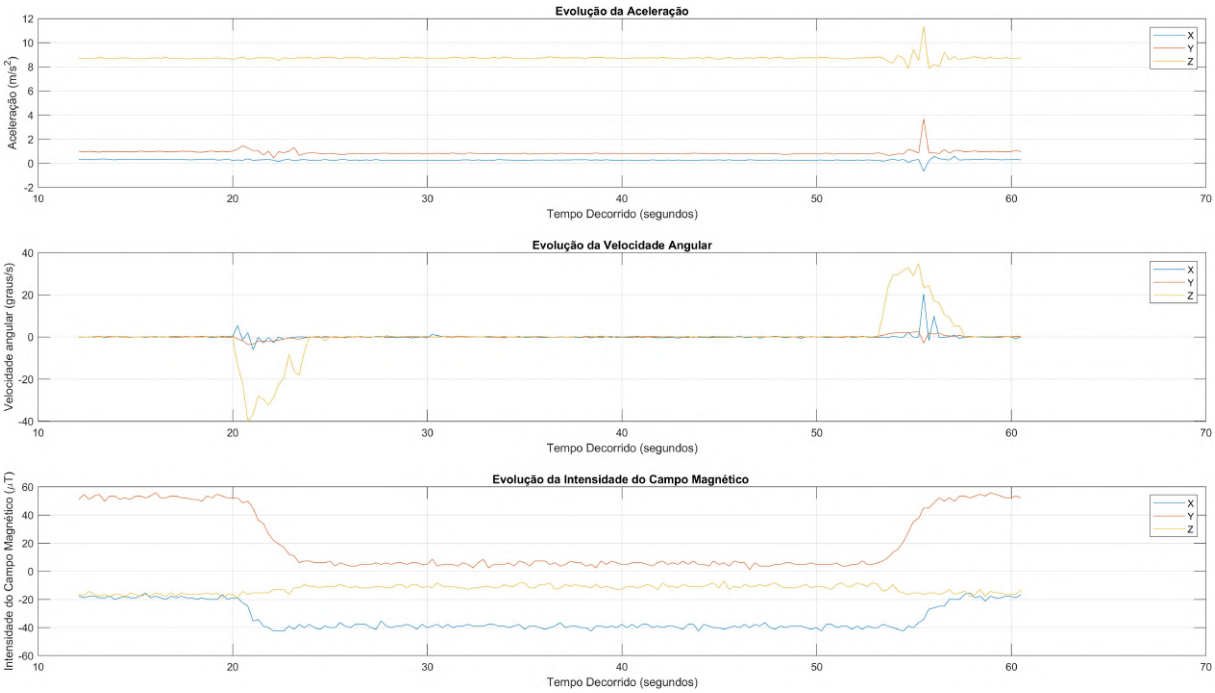

Figura 42: Rotação negativa em torno do eixo z do MPU-9250.

2.6 Análise dos Resultados Obtidos

Os resultados obtidos na seção 2.5 permitiram comprovar a capacidade operacional do ProtoCube nas diversas tarefas às quais ele se propõe atuar.

Em um dos testes realizados, foi avaliado o desempenho de descarga do conjunto de baterias 18650, com seus resultados disponibilizados através da Tabela 6 e da Figura 28. A partir dessa Figura, é facilmente identificável o comportamento exponencial da relação entre a tensão das baterias sendo descarregadas e o tempo decorrido. Pode-se notar que este comportamento está em acordo com o início do processo de descarga tal qual verifica-se no modelo fornecido pela equação de Arrhenius, como

indicado em (THIRUGNANAM; SAINI; KUMAR, 2012). A Figura 43 fornece um panorama do funcionamento de um ciclo de carga e descarga de uma bateria Li-Ion, como as utilizadas neste projeto.

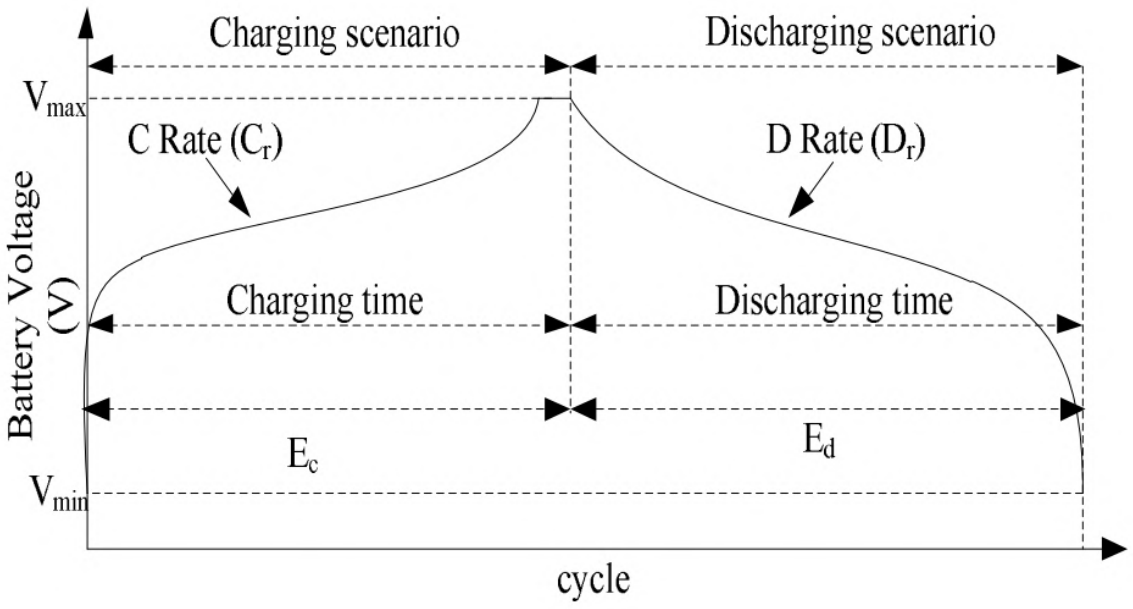

Figura 43: Características de desempenho da bateria Li-Ion para um ciclo.
Fonte: (THIRUGNANAM; SAINI; KUMAR, 2012), p. 3.

A combinação dos resultados de outros dois testes permite serem tiradas conclusões a respeito da vida útil que o projeto ProtoCube poderia receber durante uma atividade real, considerando os componentes escolhidos e avaliados. Através do teste de consumo exigido pelo OBC e pelos sensores, foi verificado que é necessária uma corrente de aproximadamente 50mA a 8,17V para manter sua operação adequada, ou seja, cerca de 400mW.

A experiência realizada com a geração de energia por meio das células fotovoltaicas promoveu uma corrente elétrica de aproximadamente 25mA para a carga das baterias. Com isso, pode-se inferir que a quantidade e as especificações técnicas das células fotovoltaicas escolhidas seriam capazes de proporcionar o carregamento das 2 baterias 18650 Li-Ion, mas com uma taxa relativamente baixa. Para fazer uma com-

paração, essas baterias geralmente são tipicamente carregadas com uma corrente de 500mA, portanto, cerca de 20 vezes maior.

Outra conclusão que pode ser obtida reside no balanço energético do EPS para que seja autossuficiente, sem depender de fontes de carregamento externas. Como o conjunto OBC e Sensores 9-DOF consome cerca do dobro do que os painéis foram capazes de gerar nas condições apresentadas, pode-se considerar que o sistema não seria autossuficiente, no entanto, teria a vida útil da sua missão prolongada, principalmente se fossem adotadas estratégias de economia de energia.

Toda missão espacial possui um prazo para encerramento, que pode ser estendido dependendo das tecnologias e/ou estratégias implementadas. Uma alternativa para garantir uma vida útil da missão mais extensa seria, após monitoração do nível das baterias, determinar um limiar inferior para a tensão das baterias de forma que, se as suas condições se aproximassem de um nível crítico, poderia ser ativado o modo deep sleep ou hibernation durante um determinado intervalo de tempo. Com isso, os painéis solares conseguiriam suficientemente carregar as baterias apesar da sua corrente gerada baixa, prolongando o tempo da missão. Após esse tempo determinado, o chip poderia ser acordado novamente, restaurando as funções dos seus subsistemas e prosseguindo com as atividades da missão. Importante ressaltar que uma estratégia de hibernação dependerá dos objetivos e requisitos da missão, levando em conta se seria aceitável a interrupção de sua operação.

A respeito dos dados adquiridos pelo Sensores 9-DOF, verifica-se a concordância com os resultados esperados, principalmente para os sensores acelerômetro e giroscópio. Ruídos são inerentes das medições dessas grandezas, podendo ter diversas fontes, como calibrações e temperatura. Como mencionado em (FARAHAN et al., 2022), o tratamento dos dados utilizando filtros, como o de Kalman, permitem estimar o estado de um sistema dinâmico linear ou não linear afetado por ruído.

Houveram medições onde foram apresentadas oscilações semelhantes a outli-

ers fora dos momentos de transição, que podem ser notados, por exemplo, nos testes do eixo z. Isso se deve ao movimento de rotação ter sido efetuado manualmente, portanto, se tornando passível de leves desvios.

3 Conclusão

A partir do projeto ProtoCube, foi possível aprimorar conhecimentos de diversas áreas distintas, partindo de conceitos aeroespaciais a eletrônica e programação. Os resultados alcançados por este trabalho indicaram que o desafio da integração entre múltiplos sistemas é desafiador, porém, extremamente recompensador assim que dados são recebidos com sucesso ou quando abordagens diferentes surtem o efeito desejado. A partir dos conceitos adquiridos e fundamentos exercitados, o ProtoCube pode seguir para uma implementação mais avançada, aprimorando seus componentes e expandindo suas dimensões para um dia se tornar um protótipo completo de um nanossatélite.

Existem vários desafios que ainda estão para serem solucionados e desenvolvidos, como o processamento e a determinação de atitude baseando-se nos dados dos sensores e a criação de um sistema de Telemetria e Telecomando. Pretende-se, através desses conceitos, possibilitar o desenvolvimento de uma tecnologia mais ambiciosa para o cenário de missões espaciais reais.

Estes desafios são objetos de estudo para trabalhos futuros, que visarão ampliar a experiência de integração de sistemas e a cooperação com outros alunos e entusiastas de diversas áreas da tecnologia.

Referências

AEB. *Nanossatélites movimentam o Programa Espacial Brasileiro*. 2023. Disponível em: <https://www.gov.br/aeb/pt-br/assuntos/noticias/nanossatelites-movimentam-o-programa-espacial-brasileiro>.

CARVALHO, R. A. d.; ESTELA, J.; LANGER, M. *Nanosatellites: Space and Ground Technologies, Operations and Economics*. [S.l.]: John Wiley Sons, 2020. ISBN 9781119042037; 1119042038.

EEMB CO., LTD. *Lithium-ion Battery DATA SHEET*. [S.l.], 2010.

ESPRESSIF SYSTEMS (SHANGHAI) CO., LTD. *ESP-IDF Programming Guide*. [S.l.], 2023.

ESPRESSIF SYSTEMS (SHANGHAI) CO., LTD. *ESP32 WROOM 32 Datasheet*. [S.l.], 2023.

FARAHAN, S. B. et al. 9-dof imu-based attitude and heading estimation using an extended kalman filter with bias consideration. *Sensors*, MDPI, v. 22, n. 9, p. 3416, 2022.

GALILEU. *As promessas do VCUB1, 1º nanossatélite brasileiro lançado no espaço | Espaço | Galileu*. 2023. <https://revistagalileu.globo.com/ciencia/espaco/noticia/2023/04/as-promessas-do-vcub1-1o-nanossatelite-brasileiro-lancado-no-espaco.ghtml>. (Acessado em 19/12/2023).

INVENSENSE INC. *MPU-9250 Register Map and Descriptions Revision 1.4*. [S.l.], 2013.

INVENSENSE INC. *MPU-9250 Product Specification Revision 1.1*. [S.l.], 2016.

MONOLITHIC POWER SYSTEMS. *MP1584*. [S.l.], 2011.

OLIVEIRA, L. d. S. *Projeto ProtoCube*. [S.l.]: GitHub, 2023. <https://github.com/leosoli/Projeto-TG-ProtoCube>.

ROMAN-GONZALEZ, A.; VARGAS-CUENTAS, N. I. Nanosatellites: Actual mission that can perform. In: . [S.l.: s.n.], 2016.

SCHMIDT, F. d. H. *Desafios e oportunidades para uma indústria espacial emergente: o caso do Brasil*. [S.l.], 2011.

SELČAN, D.; KIRBIŠ, G.; KRAMBERGER, I. Increasing reliability of cots-based eps subsystems for nanosatellites. In: *Proc. 4S Symp*. [S.l.: s.n.], 2016.

SILICON LABORATORIES. *CP2102 SINGLE-CHIP USB TO UART BRIDGE*. [S.l.], 2004.

SILVA, C. F. da. O mercado brasileiro de micro e nano satélites. *Centro de Estudos Estratégicos do Exército: Análise Estratégica*, v. 7, n. 1, p. 15–23, 2017.

THIRUGNANAM, K.; SAINI, H.; KUMAR, P. Mathematical modeling of li-ion battery for charge/discharge rate and capacity fading characteristics using genetic algorithm approach. In: IEEE. *2012 IEEE Transportation Electrification Conference and Expo (ITEC)*. [S.l.], 2012. p. 1–6.

WERNER, S. *MPU9250 driver for the ESP8266 and the ESP32*. [S.l.]: GitHub, 2023. <https://github.com/psiphi75/esp-mpu9250>.

XLSEMI. *XL4015 5A 180KHz 36V Buck DC to DC Converter Datasheet*. [S.l.], 2018.

APÊNDICES

A main.c (ESP-IDF / C)

```
1  /*
2   *
    *
3   *  Copyright 2018 Simon M. Werner
                                                *
4   *
    *
5   *  Licensed under the Apache License, Version 2.0 (the "License");
              *
6   *  you may not use this file except in compliance with the License.
             *
7   *  You may obtain a copy of the License at
                                       *
8   *
    *
9   *      http://www.apache.org/licenses/LICENSE-2.0
                                  *
10  *
    *
11  *  Unless required by applicable law or agreed to in writing, software
          *
12  *  distributed under the License is distributed on an "AS IS" BASIS,
            *
13  *  WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or
    implied. *
14  *  See the License for the specific language governing permissions and
          *
15  *  limitations under the License.
                                                *
```

```
16  *
        *
17 */
18
19 /**
20  * @file main.c
21  * @brief Brief description of your modifications.
22  *
23  * Detailed description of the modifications and improvements made to
        the
24  * original code by Simon M. Werner.
25  *
26  * Modifications:
27  * - Removed library AHRS, since it is no longer the goal of this work
28  * - Added library to deal with the UART initialization and writing
        proccesses
29  * - Removed axis transformation for the magnetometer readings since
30  *   accelerometer and gyroscope values should be in the magnetometer
        reference
31  * - Adjusted calibration parameters to meet the MPU-9250 tested
32  * - Delivery of data acquired through UART in a 5 Hz rate
33  *
34  * Acknowledgments:
35  * - Thank you to Simon M. Werner for the original code.
36  *
37  * License:
38  * - This modified code is also licensed under the Apache License,
        Version 2.0.
39  */
40
41
42
43 #include <stdio.h>
44 #include <string.h>
```

```
45 #include <stdlib.h>
46
47 #include "freertos/FreeRTOS.h"
48 #include "freertos/task.h"
49 #include "freertos/queue.h"
50
51 #include "esp_log.h"
52 #include "esp_system.h"
53 #include "esp_err.h"
54 #include "esp_task_wdt.h"
55
56 #include "driver/i2c.h"
57
58 #include "mpu9250.h"
59 #include "calibrate.h"
60 #include "common.h"
61
62 #include "libs/uart.h"
63
64 static const char *TAG = "main";
65
66 #define I2C_MASTER_NUM I2C_NUM_0 /*!< I2C port number for master dev */
67
68 calibration_t cal = {
69     .mag_offset = {.x = -13.234375, .y = 80.871094, .z = -38.414062},
70     .mag_scale = {.x = 0.980424, .y = 1.006865, .z = 1.013324},
71     .accel_offset = {.x = 0.039083, .y = 0.070368, .z = 0.136644},
72     .accel_scale_lo = {.x = 1.016390, .y = 1.036406, .z = 1.069427},
73     .accel_scale_hi = {.x = -0.978780, .y = -0.962060, .z = -0.925534},
74
75     .gyro_bias_offset = {.x = -0.947105, .y = 2.625738, .z = 0.588435}
76     };
77
78
79
```

```
80 static void transform_accel_gyro(vector_t *v)
81 {
82   float x = v->x;
83   float y = v->y;
84   float z = v->z;
85
86   v->x = y;
87   v->y = x;
88   v->z = -z;
89 }
90
91
92 void run_imu(void)
93 {
94
95   // Inicialização do barramento I2C e verificações iniciais do estado
       de funcionamento do MPU-9250 utilizando os parâmetros de calibração
        fornecidos
96   i2c_mpu9250_init(&cal);
97
98   // Inicialização da UART para envio dos dados adquiridos
99   init_uart();
100
101  uint64_t i = 0;
102  while (true)
103  {
104    vector_t va, vg, vm;
105
106    // Aquisição das medições do Acelerômetro, giroscópio e magnetômetro
         .
107    ESP_ERROR_CHECK(get_accel_gyro_mag(&va, &vg, &vm));
108
109    // Transformação dos eixos dos valores adquiridos para a mesma
         orientação dos eixos do magnetômetro
110    transform_accel_gyro(&va);
```

```
111     transform_accel_gyro(&vg);
112
113
114     // Impressão a cada 10 iterações
115     if (i++ % 10 == 0)
116     {
117
118       ESP_LOGI(TAG, "accel: x %2.3f, y %2.3f, z %2.3f (g)", va.x, va.y,
              va.z);
119       ESP_LOGI(TAG, "gyro: x %2.3f, y %2.3f, z %2.3f (degrees/s)", vg.x,
               vg.y, vg.z);
120       ESP_LOGI(TAG, "mag: x %2.3f, y %2.3f, z %2.3f (uT)", vm.x, vm.y,
              vm.z);
121
122       ESP_LOGI(TAG, "data:%2.3f,%2.3f,%2.3f,%2.3f,%2.3f,%2.3f,%2.3f,%2.3
              f,%2.3f", va.x, va.y, va.z, vg.x, vg.y, vg.z, vm.x, vm.y, vm.z)
              ;
123
124       // Atribuir memória para a mensagem string
125       size_t message_size = snprintf(NULL, 0, "data:%2.3f,%2.3f,%2.3f
              ,%2.3f,%2.3f,%2.3f,%2.3f,%2.3f,%2.3f\n", va.x, va.y, va.z, vg.x
              , vg.y, vg.z, vm.x, vm.y, vm.z) + 1;
126       char* message = (char*)malloc(message_size);
127
128       // Gerar a string da mensagem e guardá-la na memória reservada
129       snprintf(message, message_size, "data:%2.3f,%2.3f,%2.3f,%2.3f,%2.3
              f,%2.3f,%2.3f,%2.3f,%2.3f\n", va.x, va.y, va.z, vg.x, vg.y, vg.
              z, vm.x, vm.y, vm.z);
130
131       // Envio da mensagem atraves da UART
132       send_uart(message);
133
134       // Liberacao da memoria alocada
135       free(message);
136
```

```
137
138         send_uart("\n");
139
140
141
142         // Intervalo de amostragem
143         vTaskDelay(200 / portTICK_PERIOD_MS);
144       }
145
146       pause();
147     }
148 }
149
150 static void imu_task(void *arg)
151 {
152
153 #ifdef CONFIG_CALIBRATION_MODE
154     calibrate_gyro();
155     calibrate_accel();
156     calibrate_mag();
157     ESP_LOGI("main-calibrate", "A calibração foi concluída com sucesso.\n"
        );
158 #else
159     run_imu();
160 #endif
161
162     // Exit
163     vTaskDelay(100 / portTICK_PERIOD_MS);
164     i2c_driver_delete(I2C_MASTER_NUM);
165
166     vTaskDelete(NULL);
167 }
168
169 void app_main(void)
170 {
```

```
171   // start i2c task
172   xTaskCreate(imu_task, "imu_task", 4096, NULL, 10, NULL);
173 }
```

B uart.c (ESP-IDF / C)

```c
#include <stdio.h>
#include <string.h>
#include "freertos/FreeRTOS.h"
#include "freertos/task.h"
#include "esp_log.h"
#include "driver/gpio.h"
#include "driver/uart.h"

#define UART_NUM UART_NUM_2
#define BUF_SIZE 1024
#define TASK_MEMORY 1024 * 2
#define DELAY 1000
static const char *TAG = "UART";

#define TX_PIN (GPIO_NUM_17)
#define RX_PIN (GPIO_NUM_16)




void init_uart(void)
{
    uart_config_t uart_config = {
        .baud_rate = 115200,
        .data_bits = UART_DATA_8_BITS,
        .parity = UART_PARITY_DISABLE,
        .stop_bits = UART_STOP_BITS_1,
        .flow_ctrl = UART_HW_FLOWCTRL_DISABLE,
        .source_clk = UART_SCLK_APB,
    };

    uart_param_config(UART_NUM, &uart_config);
    uart_set_pin(UART_NUM, TX_PIN, RX_PIN, UART_PIN_NO_CHANGE,
       UART_PIN_NO_CHANGE);
```

```
34      uart_driver_install(UART_NUM, BUF_SIZE, BUF_SIZE, 0, NULL, 0);
35
36      ESP_LOGI(TAG, "init uart completed!");
37  }
38
39
40  void send_uart(char *data)
41  {
42      uart_write_bytes(UART_NUM_2, data, strlen(data));
43  }
```

C main.m (MATLAB)

```
%% AQUISIÇÃO DE MEDIDAS DE SENSORES 9-DOF

clear; clc;

%% PARÂMETROS DE COMUNICAÇÃO SERIAL
comPort = 'COM4';
baudRate = 115200;
timeout = 20;  % Timeout de 20 segundos sem comunicação

%% Criação do objeto de porta serial com um timeout
s = serialport(comPort, baudRate, 'Timeout', timeout);

% Abertura e configuração do terminador 'LF' (\n)
configureTerminator(s, "LF");  % Set the line terminator

fprintf('Lendo a porta %s...\n', comPort);

%% Initialização dos Arrays de dados
receivedDataCell = {};
accelArray = [0 0 0];
gyroArray = [0 0 0];
magArray = [0 0 0];
timeArray = [0];

totalElapsedTime = 0;

%% Execução das Leituras

```

```
29
30 try % Leitura da porta serial
31     % Criação de tic (contador de tempo de timeout)
32     startTime = tic;
33     totalElapsedTime = 0;  % Inicialização do tempo total
           decorrido
34
35     % Supressão temporária do Warning
36     warning('off', 'MATLAB:serial:SerialFscanf:
           unsuccessfulRead');
37
38     % Inicialização do gráfico com subplots
39     figure('units','normalized','outerposition',[0 0 1 1]);
40
41     subplot(3, 1, 1);
42     plotHandleAccel = plot(timeArray, accelArray(:, 1),
           timeArray, accelArray(:, 2), timeArray, accelArray(:,
           3));
43     title('Evolução da Aceleração');
44     xlabel('Tempo Decorrido (segundos)');
45     ylabel('Aceleração (m/s^2)');
46     legend('X', 'Y', 'Z');
47     grid on;
48
49     subplot(3, 1, 2);
50     plotHandleGyro = plot(timeArray, gyroArray(:, 1),
           timeArray, gyroArray(:, 2), timeArray, gyroArray(:, 3)
           );
51     title('Evolução da Velocidade Angular');
```

```
52      xlabel('Tempo Decorrido (segundos)');
53      ylabel('Velocidade angular (graus/s)');
54      legend('X', 'Y', 'Z');
55      grid on;
56
57      subplot(3, 1, 3);
58      plotHandleMag = plot(timeArray, magArray(:, 1), timeArray
            , magArray(:, 2), timeArray, magArray(:, 3));
59      title('Evolução da Intensidade do Campo Magnético ');
60      xlabel('Tempo Decorrido (segundos)');
61      ylabel('Intensidade do Campo Magnético (\muT)');
62      legend('X', 'Y', 'Z');
63      grid on;
64
65      accelArray = [];
66      gyroArray = [];
67      magArray = [];
68      timeArray = [];
69
70      while isvalid(s) && toc(startTime) < timeout
71          try
72              data = readline(s);
73          catch
74              % Lidando com Warning de timeout do readline
75              warning('Timeout reached. No data received.');
76              data = '';
77          end
78
79          % Exibindo os dados recebidos
```

```
80          fprintf('Received data: %s\n', data);
81
82          % Verificar se os dados não são "proximo" ou apenas
               LF
83          if ~strcmp(data, 'proximo') && ~isequal(data, 10) &&
               ~all(isspace(data))
84              % Armazenamento dos dados numa variável
                   temporária
85              tempCell = {data};
86
87              % Concatenando o Array temporario com o Array de
                   dados principal
88              receivedDataCell = [receivedDataCell; tempCell];
89
90              % Processamento dos dados e atualização dos
                   Arrays
91              [accelArray, gyroArray, magArray] = processData(
                   data, accelArray, gyroArray, magArray);
92
93              % Store the elapsed time in the timeArray
94              elapsed_time = toc(startTime);
95              totalElapsedTime = totalElapsedTime +
                   elapsed_time;
96              timeArray = [timeArray; totalElapsedTime];
97
98
99              % PLOTS DURANTE EXECUÇÃO
100
101             % Update the acceleration plot using handles
```

```
102             plotHandleAccel(1).XData = timeArray;
103             plotHandleAccel(1).YData = accelArray(:, 1);
104 
105             plotHandleAccel(2).XData = timeArray;
106             plotHandleAccel(2).YData = accelArray(:, 2);
107 
108             plotHandleAccel(3).XData = timeArray;
109             plotHandleAccel(3).YData = accelArray(:, 3);
110 
111             % Update the gyroscope plot using handles
112             plotHandleGyro(1).XData = timeArray;
113             plotHandleGyro(1).YData = gyroArray(:, 1);
114 
115             plotHandleGyro(2).XData = timeArray;
116             plotHandleGyro(2).YData = gyroArray(:, 2);
117 
118             plotHandleGyro(3).XData = timeArray;
119             plotHandleGyro(3).YData = gyroArray(:, 3);
120 
121             % Update the magnetometer plot using handles
122             plotHandleMag(1).XData = timeArray;
123             plotHandleMag(1).YData = magArray(:, 1);
124 
125             plotHandleMag(2).XData = timeArray;
126             plotHandleMag(2).YData = magArray(:, 2);
127 
128             plotHandleMag(3).XData = timeArray;
129             plotHandleMag(3).YData = magArray(:, 3);
130 
```

```
131             if length(timeArray) < 2
132                 min_timeArray = 0
133             else
134                 min_timeArray = min(timeArray);
135             end
136
137             xlim([min_timeArray, max(timeArray)]);
138             drawnow;
139
140
141
142             startTime = tic;
143         end
144
145         pause(0.001);
146     end
147
148     % Restauração do estado do Warning
149     warning('on', 'MATLAB:serial:SerialFscanf:
            unsuccessfulRead');
150
151 catch
152     disp('Ocorreu um erro durante o recebimento dos dados.');
153 end
154
155 clear s;
156 disp('Porta serial fechada.');
157
158
```

```
159 h = msgbox('Porta serial fechada. A comunicação serial foi
        encerrada.', 'Porta fechada');
160 currentPosition = get(h, 'Position');
161 newPosition = [currentPosition(1), currentPosition(2), 400,
        50];
162 set(h, 'Position', newPosition);
163 hText = findobj(h, 'Type', 'Text');
164 font_size = 14;
165 set(hText, 'FontSize', font_size);
166
167
168 % set(gca, 'Color', 'w');
169 % f = gcf;
170 % exportgraphics(f,'exportedPlot.jpg','Resolution',300)
```

D processData.m (MATLAB)

```
function [accelArray, gyroArray, magArray] = processData(data
    , accelArray, gyroArray, magArray)

    dataValues = str2double(strsplit(extractAfter(data, 'data
        :'), ','));

    if contains(data, 'data')
        accelData = g2MetersPerSecSquared(dataValues(1:3)); %
            aceleração em m/s^2
        gyroData = dataValues(4:6);
        magData = dataValues(7:9);

        if isempty(accelArray)
            accelArray = zeros(0, 3);
        end
        if isempty(gyroArray)
            gyroArray = zeros(0, 3);
        end
        if isempty(magArray)
            magArray = zeros(0, 3);
        end

        accelArray = [accelArray; accelData];
        gyroArray = [gyroArray; gyroData];
        magArray = [magArray; magData];
    end
end
```

E g2MetersPerSecSquared.m (MATLAB)

```
function acceleration_mps2 = g2MetersPerSecSquared(
    acceleration_gs)
    % Constantes
    g_to_mps2 = 9.80665;  % Aceleração padrão devido à
        gravidade em m/s^2

    % Convertendo cada elemento de g's para m/s^2
    acceleration_mps2 = acceleration_gs * g_to_mps2;
end
```