Desenvolvimento do Sistema de Gerenciamento de Bateria Inteligente Integrado com ROS

Luis Gustavo Souza Serafim - 54167

Dissertação apresentada à Escola Superior de Tecnologia e Gestão de Bragança para a obtenção do grau de Mestre em Engenharia Industrial, no âmbito do programa de Dupla Diplomação com a Universidade Tecnológica Federal do Paraná

Trabalho realizado sob a orientação de:

Professor André Chaves Mendes, D.Sc.

Professor João Alberto Fabro, D.Sc.

Bragança

2022-2023

Desenvolvimento do Sistema de Gerenciamento de Bateria Inteligente Integrado com ROS

Luis Gustavo Souza Serafim - 54167

Dissertação apresentada à Escola Superior de Tecnologia e Gestão de Bragança para a obtenção do grau de Mestre em Engenharia Industrial, no âmbito do programa de Dupla Diplomação com a Universidade Tecnológica Federal do Paraná

Trabalho realizado sob a orientação de:

Professor André Chaves Mendes, D.Sc.

Professor João Alberto Fabro, D.Sc.

Bragança

2022-2023

A possibilidade é uma pequena partícula de esperança; sem coragem para mudar o mundo, estamos destinados a ser engolidos por ele.

Okabe Rintaro

Dedicatória

Dedico este trabalho aos meus pais Anésio e Ariolita, cujo apoio inabalável e amor sem limites foram a minha luz orientadora ao longo de toda a minha jornada. O altruísmo e dedicação dos meus pais me inspiraram a buscar a excelência e a sempre dar o meu melhor. Estou profundamente grato por tudo o que fizeram por mim e espero deixá-los orgulhosos com este trabalho.

Dedico também este trabalho à minha querida namorada Maria Eduarda, que apesar da distância entre nós, tem sido uma fonte constante de apoio e força, sempre presente para me levantar e me fazer seguir em frente nos momentos mais difíceis. Seu amor têm sido um farol de esperança, e sou imensamente grato por isso, e a você dedico este trabalho de todo coração.

Dedico este trabalho aos meus queridos amigos, pelo incentivo e apoio ao longo desta jornada. Sua lealdade e camaradagem foram fundamentais para meu sucesso. Tenho orgulho de ter vocês como meus amigos e irmãos e espero poder inspirá-los e motivá-los como fizeram por mim.

Com profunda admiração, Luis Gustavo Souza Serafim.

Agradecimentos

Gostaria de agradecer aos meus estimados mentores André Chaves e João Fabro, cujo apoio e orientação foram fundamentais para o sucesso deste projeto. As suas crenças no meu potencial e as suas vontade de me desafiar a ir além dos meus limites inspiraram-me a dar o meu melhor. Os seus incentivos permitiram-me prosseguir a minha paixão pela eletrônica. Sou profundamente grato por suas dedicações e comprometimento com meu crescimento acadêmico e espero honrar os seus legados continuando a aprender e crescer em minha área.

Gostaria de expressar minha gratidão ao Instituto Politécnico de Bragança (IPB) e à Universidade Tecnológica Federal do Paraná (UTFPR) por me proporcionarem a oportunidade de cursar um programa de Dupla Diplomação. Seus compromissos com a excelência acadêmica e seus ambiente de apoio foram fundamentais para o meu crescimento pessoal e profissional, e sou profundamente grato por suas contribuições em minha jornada.

Muito Obrigado!

Abstract

This dissertation presents the development of an Intelligent Battery Management System (BMSi) integrated with the Robot Operating System (ROS). The proposed BMSi not only incorporates essential protections but also enables data logging and real-time battery information collection, providing data such as voltages (including individual cells), current, charge, battery and MOSFET temperatures, as well as fault detections. It also contains all necessary protections to ensure its safe use. The proposed system was tested on the Magni robot, which previously used lead-acid batteries, and with the proposed BMSi using a Li-Ion battery, resulted in a considerable increase in its autonomy while providing the robot with complete knowledge of battery information, enabling informed decision-making. The BMSi developed in this dissertation has the potential to contribute to the advancement of BMS technologies, particularly in the context of robotic applications, with significant implications for the development of more affordable robotic systems, with potential applications in various sectors, including manufacturing, healthcare, and transportation.

Keywords: Intelligent Battery Management System (BMSi), Robot Operating System (ROS), Robotic Systems, Li-Ion Batteries, Magni Robot, ESP32, BQ76942 Battery Monitor and Protector.

Resumo

esta dissertação apresenta o desenvolvimento de um Sistema de Gerenciamento de Bateria Inteligente (BMSi) integrado com o Sistema Operacional de Robôs (ROS). O BMS proposto não só incorpora proteções essenciais, mas também permite o registro de dados e a coleta de informações da bateria em tempo real, sendo capaz de fornecer dados como tensões (incluindo células individuais), corrente, carga, temperatura da bateria e dos MOSFETs, bem como detecções de falhas. Contem também todas as proteções necessárias para garantir o seu uso seguro. Os teste do sistema proposto foram realizados no robô Magni, que anteriormente utilizava baterias de chumbo-ácido, e com o uso do BMSi proposto com bateria de Li-Íon, resultou em um aumento considerável da sua autonomia, enquanto disponibiliza ao robô total conhecimento sobre as informações da bateria possibilitando tomada de decisões informada. O BMSi desenvolvido nesta dissertação tem o potencial de contribuir para o avanço das tecnologias de BMS, particularmente no contexto de aplicações robóticas, com implicações significativas para o desenvolvimento de sistemas robóticos mais acessíveis, com potenciais aplicações em diversos setores, incluindo manufatura, saúde e transporte.

Palavras-chave: Sistema de Gerenciamento de Bateria Inteligente (BMSi), Robot Operating System (ROS), Sistemas Robóticos, Baterias de Li-Íon, Robô MAGNI, ESP32, Monitor e Protetor de Bateria BQ76942.

Conteúdo

Lista de Tabelas

Lista de Figuras

Lista de Algoritmos

Acrônimos

BMS Battery Management System (Sistema de Gerenciamento de Bateria).

BMSi Intelligent Battery Management System (Sistema de Gerenciamento de Bateria Inteligente).

BMSiv2 Intelligent Battery Management System Version 2 (Sistema de Gerenciamento de Bateria Inteligente - segunda versão).

CeDRI Research Centre in Digitalization and Intelligent Robotics.

CHG Charge (Carga).

CI Integrated Circuit (Circuito Integrado).

COV Cell Overvoltage (Sobretensão das Células).

CUV Cell Undervoltage (Subtensão das Células).

DSG Discharge (Descarga).

EMI Electromagnetic Interference (Interferência Eletromagnética).

ESD Electrostatic Discharge (Descargas Eletrostáticas).

ESP-IDF Espressif IoT Development Framework (Framework de Desenvolvimento IoT da Espressif).

ESTiG Escola Superior de Tecnologia e Gestão.

I2C Inter-Integrated Circuit (Circuito Inter-Integrado).

IDE Integrated Development Environment (Ambiente de Desenvolvimento Integrado).

IHM Human-Machine Interface (Interface Homem-Máquina).

IPB Instituto Politécnico de Bragança.

OCC Overcurrent Charge (Sobrecorrente em Carga).

OCD Overcurrent Discharge (Sobrecorrente em Descarga).

OTC Overtemperature Charge (Sobretemperatura de Carga).

OTD Overtemperature Discharge (Sobretemperatura de Descarga).

OTFET Overtemperature of MOSFETs (Sobretemperatura dos MOSFETs).

PCB Printed Circuit Board (Placa de Circuito Impresso).

RDS(on) Drain-Source On-Resistance (Resistência de Dreno-Fonte do MOSFET Conduzindo).

ROS Robot Operating System (Sistema Operacional de Robô).

Rthja Junction-to-Ambient Thermal Resistance (Resistência Térmica de Junção-Ambiente).

SCD Short Circuit Discharge (Curto Circuito em Descarga).

SDIO Secure Digital Input Output (Entrada e Saída Digital Segura).

SI Sistema Internacional das Unidades.

SMD Surface Mount Device (Dispositivo de Montagem em Superfície).

SOC State of Charge (Estado de Carga).

SOF State of Function (Estado de Função).

SOH State of Health (Estado de Saúde).

SPI Serial Peripheral Interface (Interface Periférica Serial).

Tj Junction Temperature (Temperatura de Junção).

Tjmax Maximum Junction Temperature (Temperatura Máxima da Junção).

TRM Technical Reference Manual (Manual de Referência Técnica).

UC Unidade Curricular.

UTC Undertemperature Charge (Baixa Temperatura em Carga).

UTD Undertemperature Discharge (Baixa Temperatura em Descarga).

UTFPR Universidade Tecnológica Federal do Parana.

VGS Gate-Source Voltage (Tensão Porta-Fonte).

Capítulo 1

Introdução

1.1 Contextualização

As baterias de Li-Íon, compostas por múltiplas células em série e em paralelo, desempenham um papel crucial no armazenamento de energia em uma variedade de aplicações, desde dispositivos eletrônicos portáteis até veículos elétricos. No entanto, o desbalanceamento entre as células devido a diferenças de capacidade e características internas pode comprometer a eficiência e a segurança dessas baterias durante o carregamento e descarregamento, sendo necessário o uso de sistemas de gerenciamento de bateria (BMS). Além de equilibrar as células, o BMS desempenha um papel fundamental na proteção das baterias contra sobrecarga, sobredescarga, sobrecorrente e variações de temperatura, garantindo um desempenho consistente e seguro.

Os BMS comumente disponíveis no mercado podem enfrentar limitações em sua capacidade de se comunicar com outros dispositivos, adicionando complexidade aos sistemas em que são implementados. Sistemas de Gerenciamento de Bateria Inteligentes podem superar algumas dessas limitações, mas geralmente têm um custo mais alto e utilizam protocolos como CAN e RS485, que, embora robustos e confiáveis, podem ser mais complexos de implementar. Esses protocolos podem não ser ideais para todos os sistemas robóticos, dependendo das necessidades específicas do sistema. Portanto, o desenvolvimento de um

BMS inteligente (BMSi) de fácil implementação e que seja de baixo custo comparado aos sistemas existentes, que também possa se comunicar efetivamente com outros dispositivos em um sistema robótico, pode ser uma solução valiosa.

O robô autônomo Magni da Ubiquity, que é o foco deste projeto, utiliza baterias de chumbo-ácido. Este robô enfrenta limitações operacionais, como baixa autonomia, devido à capacidade limitada das baterias. Além disso, a ausência de dados detalhados sobre o estado da bateria dificulta a identificação de problemas e a tomada de decisões informadas sobre o gerenciamento de energia. A implementação de um BMSi acessível e integrado ao ROS é de grande importância para superar essas limitações.

Embora sistemas de gerenciamento de bateria inteligente já estejam disponíveis no mercado, este trabalho propõe uma alternativa mais econômica que se integra ao ROS. Através da integração do BMSi com o ROS, é possível implementar algoritmos de controle e gerenciamento de energia mais avançados. Isso permite que sistemas robóticos tomem decisões mais eficientes sobre o uso da energia, resultando em uma melhoria no desempenho geral. Esta solução tem o potencial de impactar significativamente diversas áreas, incluindo manufatura, saúde e transporte.

1.2 Objetivos da Pesquisa

Esta tese tem como objetivo principal desenvolver um Sistema de Gerenciamento de Baterias Inteligente (BMSi) que supere as limitações dos BMS atuais, sendo acessível, eficiente e de fácil implementação. O BMSi proposto devera ser integrado com o Sistema Operacional de Robôs (ROS), sendo capaz de fornecer dados em tempo real, como tensões (incluindo células individuais), corrente, capacidade, temperatura da bateria e dos MOSFETs, bem como detecções de falhas e conter todas as proteções necessárias para garantir o uso seguro de baterias.

Assim tem-se como objetivos específicos:

- Estabelecer os requisitos de bateria do robô Magni, como capacidade, tensão, corrente, dimensionando corretamente o projeto eletrônico e a bateria de Li-Íon para

suprir as suas necessidades, substituindo o seu sistema de bateria de chumbo-ácido.

- Dimensionar e desenvolver os circuitos necessários para o projeto, criando um esquemático e uma placa de circuito impresso para o BMSi.

- Desenvolver uma biblioteca em C++ para o circuito integrado (CI) BQ76942 da Texas Instruments se baseando nas instruções do Technical Reference Manual (TRM) [1] e código exemplo fornecido [2], criando uma abstração da sua comunicação, configuração e uso, facilitando a interação com o CI, tornando-o acessível e fácil de usar.

- Implementar algoritmos no microcontrolador ESP32 para criar um BMSi intuitivo capaz de comunicar os dados da bateria pelo uso do sistema ROS, realizar registro de dados, e fornecer uma interface homem-máquina, facilitando o monitoramento e a análise dos dados da bateria.

- Testar no robô Magni a sua nova bateria e BMSi, validando a performance, comunicação de dados, e eficácia do sistema posposto em uma ambiente de aplicação real, comparando com o sistema anterior.

Com esses objetivos, a pesquisa busca contribuir para o avanço das tecnologias de BMS, particularmente no contexto de aplicações robóticas, tornando-as mais acessíveis, eficientes e confiáveis.

1.3 Organização do Documento

A organização deste documento segue uma estrutura lógica e progressiva, começando com uma introdução no Capítulo 1 contextualizando o problema e estabelecendo os objetivos da pesquisa. Em seguida, é apresentado o estado da arte no Capítulo 2, que fornece uma visão geral dos BMS, baterias de Li-Íon, o robô Magni, o sistema ROS, o ESP32 e o monitor de bateria BQ76942.

A solução proposta é então detalhada no Capítulo 3, seguida por uma discussão no Capítulo 4 sobre o desenvolvimento e implementação do sistema, incluindo as especificações e requisitos do Magni, o dimensionamento do BMS e da bateria, as duas versões desenvolvidas do BMSi, os principais componentes utilizados, os circuitos desenvolvidos, a placa de circuito impresso (PCB), os algoritmos utilizados e uma análise econômica do projeto.

Os resultados e discussões são apresentados em seguida no Capítulo 5, fornecendo uma análise dos dados coletados e uma avaliação do desempenho do sistema. Finalmente, o Capítulo 6 do documento conclui com uma seção de conclusão e trabalho futuro, que resume os principais achados da pesquisa e sugere direções para pesquisas futuras.

Capítulo 2

Estado da Arte

Neste capítulo, será apresentado o estado da arte do presente documento, com o objetivo de contextualizar o leitor sobre o tema proposto e tecnologias utilizadas. Serão descritos os componentes utilizados no projeto, como circuitos integrados, protocolos, sistemas como ROS, entre outros, destacando as contribuições específicas do trabalho e identificando lacunas a serem preenchidas.

2.1 Sistema de Gerenciamento de Bateria (BMS)

O estado da arte atual dos sistemas de gerenciamento de bateria (BMS) reflete avanços significativos na eficiência e confiabilidade desses sistemas. Os BMS desempenham funções críticas, como monitorar as tensões e corrente da bateria, proteger e balancear as células, entre outras proteções, sendo essencial para proteger o sistema de danos e aumentar sua vida útil, especialmente em aplicações robóticas. O diagrama simplificado de um BMS é demostrado na Figura 2.1.

Os avanços recentes incluem o uso de algoritmos especiais para o monitoramento da bateria, levando em consideração as suas características, que mudam significativamente durante sua vida útil devido ao envelhecimento. Além disso, os BMS têm evoluído para integrar algoritmos de inteligência artificial e *machine learning*, permitindo a adaptação dinâmica do sistema às condições de operação em tempo real, maximizando o desempenho

e a durabilidade das baterias.

Uma tendência emergente é o desenvolvimento de BMS modulares, escaláveis e inteligentes, que facilitam a adaptação a diferentes aplicações, manutenção e substituição de componentes. Além disso, a busca por materiais de eletrodos mais avançados, como o uso de grafeno e nanotubos de carbono, visa aumentar a densidade de energia e a taxa de carga das baterias.

Apesar dos avanços no estado da arte dos BMS, sua implementação é limitada devido à predominância dessas tecnologias em produtos de alto custo e complexidade. Os sistemas disponíveis para o público em geral carecem das capacidades e comunicações oferecidas por esses avanços, assim este projeto contribui estabelecendo uma base acessível para futuramente a implementação de tais tecnologias e algorítimos.

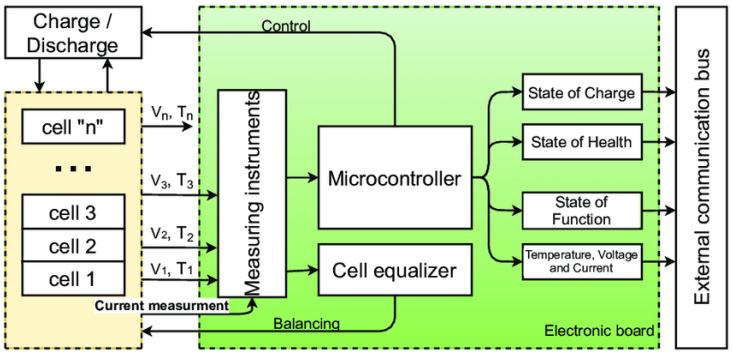

Figura 2.1: Diagrama de um BMS simplificado[3].

2.2 Baterias de Li-Íon

As baterias de íon de lítio são amplamente utilizadas em robótica devido à sua alta densidade de energia, densidade de potência e eficiência representando uma tecnologia promissora para o armazenamento e fornecimento de energia em diversas aplicações. Elas são compostas por uma ou mais células conectadas eletricamente em série e/ou paralelo para fornecer níveis específicos de voltagem e corrente. O estado da arte das baterias de Li-Íon abrange a modelagem matemática, algoritmos de otimização, sistemas de gerenciamento avançados e a sua seleção adequada para diferentes aplicações, refletindo o

contínuo avanço e interesse nessa tecnologia promissora.

A tese *Energy Management of Li-Po Batteries in the Mobile Robotics Domain*[4], de Chellal, descreve o desenvolvimento de um sistema de gerenciamento de baterias com estimação de estados utilizando filtro de Kalman, incluindo a sua proteção e o gerenciamento adequado em diversas aplicações. O trabalho também aborda a modelagem matemática da bateria, algoritmos de otimização, estratégias de balanceamento das células e a estimação precisa do estado da bateria.

A caracterização da bateria inclui a curva de carga/descarga, capacidade teórica e energia teórica. O estado de carga (SOC) pode ser determinado por diferentes métodos, incluindo dispositivos de laboratório desenvolvidos em centros de pesquisa em todo o mundo ou dispositivos já comercializados. Algoritmos podem ser aplicados para definir os principais estados da bateria, incluindo estado de carga (SOC), Estado de Saúde (SOH) e Estado de Função (SOF) que permitem o gerenciamento em tempo real das baterias.

2.3 Robô Magni

O robô Magni, demostrado na Figura 2.2, é uma plataforma móvel autônoma, pronta para ser integrada e implementada em projetos de robótica. Sendo uma desenvolvida pela Ubiquity Robotics [5] em junho de 2018 buscando disponibilizar uma plataforma acessível no mercado, construída a partir de componentes prontos. Sendo considerado umas das plataformas mais acessíveis atualmente, sendo uma opção atrativa para aplicações de robótica. Tendo capacidade de carga de até 100kg, utiliza o sistema operacional ROS para comunicação com periféricos e dispositivos secundários, facilitando a atualização e desenvolvimento de aplicações personalizadas [6].

Este robô é utilizado em pesquisas no campo de robótica pelo Laboratório Research Centre in Digitalization and Intelligent Robotics (CeDRI) [7] do Instituto Politécnico de Bragança (IPB), onde são desenvolvidas e implementadas novas tecnologias que contribuem para o avanço deste campo. Este robô é uma ferramenta fundamental para alunos

de mestrado e doutorandos, permitindo a aplicação prática de conceitos teóricos e o desenvolvimento de soluções inovadoras em áreas como inteligência artificial, reconhecimento de imagem e outras tecnologias de ponta. O CeDRI desempenha um papel crucial na formação de pesquisadores e na promoção do avanço tecnológico por meio da aplicação prática de conhecimentos em robótica e áreas relacionadas.

Figura 2.2: Ubiquity Robotics, Magni [6].

2.4 Robot Operating System (ROS)

O Sistema Operacional de Robô (ROS) é uma plataforma de código aberto amplamente utilizada no desenvolvimento de aplicativos de robótica. Fornecendo um conjunto de estruturas de software que facilitam a criação de sistemas robóticos avançados [8]. Embora o ROS não seja um sistema operacional no sentido tradicional, este oferece serviços para abstração de hardware, controle de dispositivos de baixo nível, passagem de mensagens entre processos e gerenciamento de pacotes. Sendo baseado em um sistema de processamento em nós, onde conjuntos de processos baseados em ROS são representados em uma arquitetura de grafo. Isso permite que os nós recebam, publiquem e multiplexem dados de sensores, controle, estado, planejamento, atuadores e outras mensagens.

O ROS oferece funcionalidades essenciais, como ferramentas para visualizar e gravar

dados, navegar facilmente nas estruturas de pacotes do ROS e criar *scripts* para automatizar processos de configuração complexos. A adição dessas ferramentas aumenta significativamente as capacidades dos sistemas que utilizam o ROS, simplificando e fornecendo soluções para diversos problemas comuns no desenvolvimento de robótica. Assim neste projeto, foi escolhido utilizar ROS para possibilitar a comunicação com o robô Magni, assim periodicamente enviando tópicos dentro de um nó contendo todas as medições realizadas pelo BMS, além de erros detectados e proteções acionadas.

2.5 ESP32

Os microcontroladores desempenham um papel crucial em uma variedade de aplicações, sendo utilizado desde dispositivos domésticos até sistemas de automação industrial. O ESP32 [9], desenvolvido pela Espressif Systems [10], é um microcontrolador de baixo custo e baixo consumo de energia, que ganhou popularidade devido à sua versatilidade e recursos avançados. Ele é baseado em dois núcleos de processador Xtensa 32-bit LX6, opera a uma frequência de até 240 MHz e possui 520 KB de RAM. Além disso, o ESP32 oferece conectividade Wi-Fi e Bluetooth integrada, tornando-o ideal para aplicações IoT (Internet das Coisas) [11].

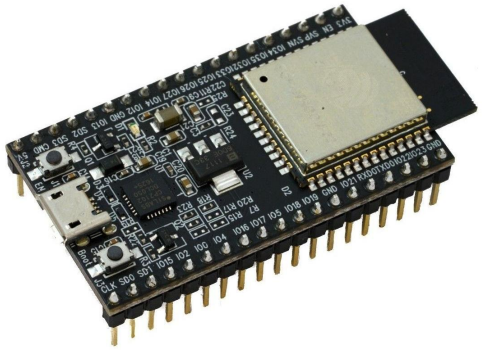

Figura 2.3: ESP32.

Para o desenvolvimento de software, o ESP32 é suportado pelo ambiente de desenvolvimento integrado (IDE) Arduino, bem como pelo *framework* de desenvolvimento Espressif IoT Development Framework (ESP-IDF) [12]. Permitindo que os desenvolvedores escolham a abordagem mais adequada às suas necessidades e experiência. Sendo amplamente utilizado em uma variedade de aplicações, como dispositivos de monitoramento remoto, sistemas de automação residencial, dispositivos *wearables* e projetos de IoT em geral. Sua capacidade de conectividade e baixo consumo de energia o tornam uma escolha popular para projetos que exigem comunicação sem fio e eficiência energética [11].

O ESP32 continua a evoluir com o lançamento de novas variantes e atualizações de *firmware*, ampliando ainda mais suas capacidades e desempenho. Além disso, a comunidade de desenvolvedores ativos contribui constantemente com bibliotecas e exemplos de código, enriquecendo o ecossistema de desenvolvimento em torno do ESP32. Tendo estes avanços em mente, o ESP32 foi escolhido para ser utilizado no projeto para realizar a interface, comunicação e algorítimos do BMS inteligente.

2.6 Monitor e Protetor de Bateria BQ76942

O circuito integrado (CI) BQ76942 [2], ilustrado na Figura 2.4 fabricado pela Texas Instruments [13], é um componente de baixo custo (aproximadamente 4 Euros) [14] que representa um avanço significativo na gestão de baterias. Ele desempenha um papel crucial em sistemas de armazenamento de energia, sistemas robóticos e outras aplicações que exigem monitoramento preciso, proteção de baterias e alta flexibilidade de uso, suportando comunicação bidirecional com um microcontrolador através de comandos diretos e subcomandos por interfaces de comunicação padrão, como I2C [15] e SPI [16], permitindo alta configurabilidade em sistemas mais amplos e complexos. Conforme mostrado na Figura 4.10, este CI é altamente integrado, simplificando o *hardware* dos circuitos externos do BMS entre eles *drivers* internos para os MOSFETs que controlam o chaveamento da bateria, conversores analógico digital para sensores de temperatura, medidor de corrente por resistor *shunt* [17] externo e *Coulomb Counter* [18], enquanto oferece recursos

avançados de monitoramento de células de bateria, balanceamento e todas as proteções necessárias para uma bateria de Li-Íon, como proteção contra sobrecarga, sobredescarga e curto-circuito.

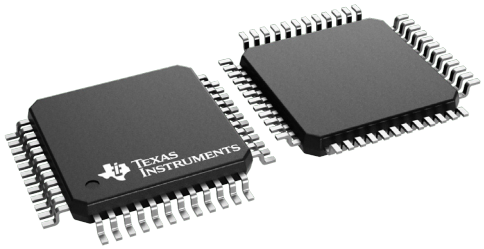

Figura 2.4: Encapsulamento do CI BQ76942 [2].

Devido à sua precisão, confiabilidade e baixo custo, o BQ76942 é amplamente utilizado em aplicações que exigem uma gestão eficiente e segura de baterias, como veículos elétricos, sistemas de armazenamento de energia residencial e industrial, dispositivos portáteis e equipamentos médicos. Sua capacidade de monitorar e proteger as células de bateria de forma precisa e confiável contribui significativamente para a segurança e desempenho desses sistemas. Com o avanço contínuo da tecnologia de baterias e a crescente demanda por soluções de armazenamento de energia mais eficientes, o BQ76942 tem evoluído para atender a essas demandas. A Texas Instruments continua a aprimorar o CI, lançando novas versões e atualizações de *firmware*, incorporando avanços na tecnologia de baterias.

Apesar de ser um CI altamente integrado e avançado para a gestão de baterias, sua implementação pode apresentar desafios específicos. Sua complexidade pode ser um obstáculo, exigindo um longo período de estudo e conhecimento técnico para configurar, calibrar, obter medições e utilizar o dispositivo corretamente. A maior dificuldade está na comunicação com este CI, devido ao seu protocolo de comunicação complexo, que utiliza comandos diretos e subcomandos, requerendo a leitura e total compreensão de extensos manuais técnicos, como *Datasheet* [19], Technical Reference Manual (TRM) [1], Software Development Guides [20], entre outros [21]–[23].

Portanto, este CI representa como o estado da arte de gestão de baterias tem avançado

significativamente, oferecendo uma combinação única de recursos avançados, confiabilidade e integração flexível, mantendo um baixo custo, e tendo uma baixa complexidade de implementação de *hardware* em troca de alta complexidade de *software*. Sua desvantagem reside nesta complexidade, o que limita seu uso em projetos com poucos recursos e tempo limitado.

Capítulo 3

Solução Proposta

A proposta de solução para o problema identificado consiste no desenvolvimento de um BMS inteligente (BMSi), adequado às necessidades do robô Magni. Este BMSi, que tem seu diagrama demostrado na Figura 3.1, será capaz de estabelecer comunicação com o ROS por meio de uma conexão USB, sendo utilizado com a nova bateria de Li-Íon do robô para ampliar sua eficiência e capacidade.

A motivação para esta solução reside na necessidade de um BMS que forneça ao robô Magni informações detalhadas sobre a bateria em uso, porém mantendo o projeto acessível e de fácil manuseio. Isso permitirá ao robô tomar decisões informadas, possibilitando um funcionamento mais eficiente e inteligente.

O CI BQ76942 foi selecionado devido ao seu custo-benefício e por possuir as funções necessárias para o funcionamento do BMSi, como medição, balanceamento e proteções. Além disso, o CI BQ76942 possui sistemas internos que simplificam o projeto eletrônico, como balanceamento interno, *driver* para MOSFETs, medidor de corrente e *Coulomb Counter*. O balanceamento interno elimina a necessidade de desenvolver um circuito de balanceamento externo, enquanto o driver para MOSFETs permite o uso de MOSFETs tipo N que são mais eficientes e econômicos em *high side switching*. O medidor de corrente do BQ76942 utiliza um resistor *shunt* externo, permitindo ao BMSi realizar medições de corrente e, ao utilizar o *Coulomb Counter*, determinar o estado de carga da bateria, ambas informações cruciais para o robô.

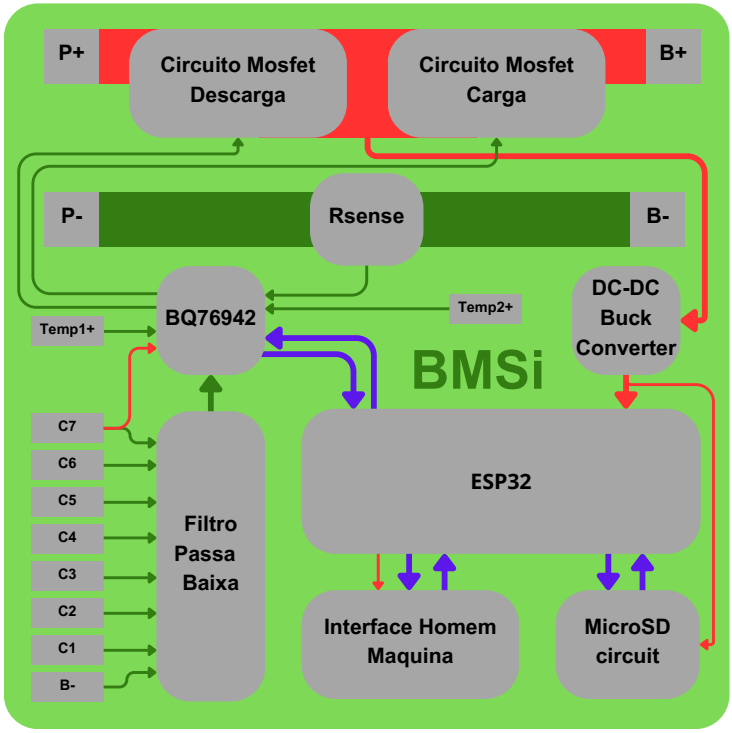

Figura 3.1: Diagrama simplificado do BMSiv2 proposto.

Para lidar com a complexidade da comunicação do CI BQ76942, que utiliza comandos e subcomandos específicos, e a necessidade de algoritmos complexos para configuração, calibração e uso, propõe-se a criação de uma biblioteca em C++ com documentação adequada, esta seguindo as instruções de implementação do TRM [1] e sendo desenvolvida a partir do código de exemplo de uso do BQ76942 fornecido pela Texas Instruments [2]. Assim criando uma camada de abstração para o uso deste CI, garantindo acessibilidade e facilidade de uso neste projeto e para futuros desenvolvedores que desejam utilizar este CI.

A implementação da solução será realizada utilizando um ESP32, escolhido por seu custo-benefício e ampla utilização. Este microcontrolador será responsável por configurar e obter os dados do CI BQ76942, processar esses dados e enviá-los para o robô Magni em forma de tópicos ROS. Além disso, o ESP32 será necessário para fornecer uma interface homem-máquina para a bateria e para realizar o registro de dados por meio de um microSD, essencial para analisar o uso e parâmetros da bateria, facilitando a obtenção de

dados que poderão ser utilizados em futuros projetos e algoritmos avançados para estimar o estado de carga e a saúde da bateria.

A implementação desta solução tem o potencial de atender a todos os objetivos do projeto, oferecendo alta segurança e eficiência, com a expectativa de aumentar a eficiência do robô Magni, garantindo uma bateria segura, confiável e duradoura. Isso resultará em um maior tempo operacional e uma redução no peso, contribuindo para o avanço dos sistemas de gestão de baterias, especialmente no contexto de aplicações robóticas.

Capítulo 4

Desenvolvimento e Implementação

Nesta seção, será detalhado o processo de desenvolvimento do projeto do BMS inteligente (BMSi), abordando as etapas de concepção, design e implementação com ênfase na programação de microcontroladores e desenvolvimento de circuitos eletrônicos, uma área de especialização relevante para este projeto. O objetivo é fornecer uma visão abrangente de como o projeto evoluiu desde a sua concepção inicial até a sua realização final. É importante discutir primeiramente a fase de concepção, onde as ideias iniciais foram geradas e os requisitos do projeto foram definidos. Em seguida, a fase de design, onde essas ideias foram transformadas em um plano concreto de ação. E por final a fase de implementação, onde o plano foi colocado em prática, detalhando os desafios encontrados durante a implementação e como eles foram superados.

4.1 Especificações e Requisitos do Magni

O robô Magni opera utilizando duas baterias de chumbo-ácido de $12V$ em série contendo um carregador no seu circuito para baterias deste tipo de química e tensão, estas baterias são armazenadas no seu compartimento interno de carga podendo suportar um peso de até $100kg$, tendo como dimensões no mínimo $205 \times 258mm$ e $184mm$ de altura (devido a tolerâncias de fabricação pode ser um pouco maior, porém não garantido) como

demostrado na Figura 4.1[1][6]. O manual do Magni sugere o uso das bateria de chumbo-ácido especificadas na Tabela 4.1, podendo ser selecionada de acordo com a necessidade do projeto.

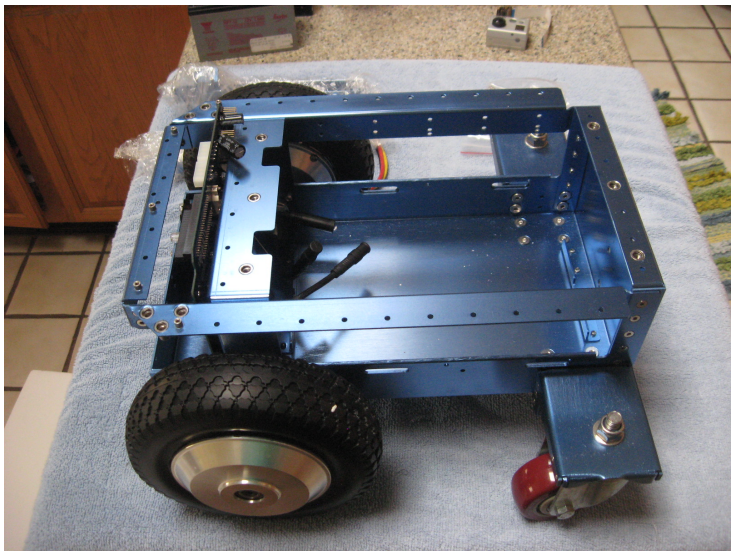

Figura 4.1: Compartimento interno do Magni [6].

Tabela 4.1: Baterias de chumbo-ácido e sua duração de uso no Magni [24].

Tamanho da Bateria	Capacidade (Ah)	Duração de Uso (h)	Notas
1250/1255	4 - 6	3 - 4	Usado quando a portabilidade do robô é importante - por exemplo, se você estiver viajando de avião com o robô.
1270	7 - 10	6 - 8	A bateria deste tamanho torna o robô ainda leve o suficiente para ser levantado.
12350	30-35	12 - 24	Recomendado apenas para quem precisa ter uma resistência extraordinária. Este tamanho de bateria torna o robô suficientemente pesado para que seja difícil para a maioria dos usuários levantá-lo.

Apesar do seu manual sugerir o uso das baterias especificadas na Tabela 4.1, também consta que o robô aceita qualquer química de bateria desde que respeite o limite máximo de $30V$ e que se mantenha acima de $22,5V$ [25]. Assim é possível converter o robô para utilizar bateria de Li-Íon, entretanto o carregador interno do Magni é apenas para baterias de chumbo-ácido se fazendo necessário o uso de um carregador externo que será

[1]Este peso máximo de $100kg$ considera o peso total do robô, assim não é apenas o peso suportado para a bateria em si.

dimensionado posteriormente. Neste manual também consta o consumo de corrente do robô bem como a corrente instantânea em casos de teste de estresse, estes demostrados na Tabela 4.2.

Tabela 4.2: Uso de corrente tipico do Magni [24].

Estado de Operação	**Corrente DC** (A)
Robô estacionário usando o Pi4 com 4GByte e em terreno plano com motor desligado.	0.4 - 0.45
Condução em superfície plana sem carga a cerca de 0,5 metros/seg.	0.8 - 0.9
Girando no lugar sem carga (quase o mesmo que dirigir devagar).	0.8
Estacionário em terreno plano com alimentação para os motores.	0.5 - 0.6
Estacionário em uma superfície plana, mas empurrando para baixo e para trás no robô, de modo que as rodas tenham que lutar para permanecer no mesmo lugar, mas ainda não estão escorregando.	1.2
Robô posicionado de forma que ele não possa se mover e com uma grande quantidade de torque aplicada em cada roda para que o controlador do motor tenha que lutar para manter as rodas firmes.	2.0 - 3.0
Corrente instantânea em casos de teste específicos de estresse (transiente).	$\cong$ 10

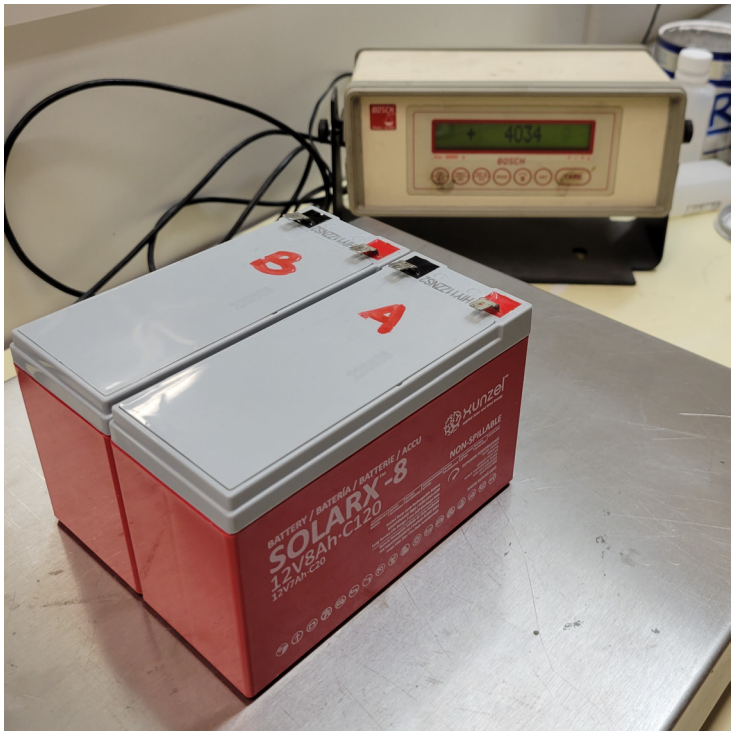

Figura 4.2: Peso das baterias de chumbo-ácido do Magni.

O Magni do laboratório CeDRI, anteriormente utilizava duas baterias de chumbo-ácido de $12V$ e $8Ah$ em série como demostrado na Figura 4.2, teoricamente deveria se

obter um tempo de uso de 6-7 horas, entretanto devido a situações especificas do Magni do laboratório como conter sistemas extras (LIDAR, e entre outros) e a bateria conter muito tempo de uso, havendo a possibilidade de não conter a capacidade nominal devido ao envelhecimento, obtém-se aproximadamente 1h de uso com o robô se movendo, isto será testado e validado posteriormente no capitulo 5.

4.2 Dimensionamento do BMSi e Bateria

Com base nos requisitos do Magni, foram determinados os parâmetros para o desenvolvimento do BMSi e da bateria de Li-Íon. Estes incluem: a tensão não deve ultrapassar $30V$ em nenhuma circunstância, deve se manter acima de $22.5V$, suportar correntes instantâneas de no mínimo $10A$ e uma corrente contínua de $3A$, e utilizar um carregador externo para baterias de Li-Íon.

Considerando que as baterias de Li-Íon são compostas de células com tensão nominal de $3,7V$, tensão máxima carregada de $4,2V$, e mínima descarregada de $3,3V$, para atender aos requisitos do Magni, será necessário utilizar uma bateria de Li-Íon com 7 células em série (7S). Isso resultará em uma tensão nominal de $25,9V$, tensão máxima carregada de $29,4V$ e mínima descarregada de $23,1V$, respeitando assim os limites impostos pelo robô.

Para dimensionar a quantidade de células em paralelo, é necessário calcular primeiramente a potência que o robô utiliza. Isso pode ser feito com os dados das tabelas 4.1 e 4.2, utilizando a Equação (4.1) para obter a Tabela 4.3 de potência nominal em Watts do robô. Os valores obtidos são consistentes com a Tabela 4.1.

Tendo como objetivo garantir uma autonomia mínima de 10 horas para o robô e com base na potência utilizada pelo Magni, que é de aproximadamente $29W$ quando em movimento com carga (conforme a Tabela 4.3), calcula-se no pior dos casos um consumo total de energia de $290Wh$. Isso é baseado na fórmula (4.2).

Para uma bateria de Li-Íon de 7S, tendo tensão nominal de $25,9V$, calcula-se a capacidade necessária para fornecer a energia requerida utilizando a fórmula (4.3), resultando em uma capacidade mínima necessária de $11,20Ah$. Tendo isto em mente foi adquirida

Tabela 4.3: Potência em casos tipico do Magni.

Estado de Operação	**Potência (W)**
Robô estacionário usando o Pi4 com 4GByte e em terreno plano com motor desligado.	9.6 - 10.8
Condução em superfície plana sem carga a cerca de 0,5 metros/seg.	12.2 - 21.6
Girando no lugar sem carga (quase o mesmo que dirigir devagar).	19.2
Estacionário em terreno plano com alimentação para os motores.	12 - 14.4
Estacionário em uma superfície plana, mas empurrando para baixo e para trás no robô, de modo que as rodas tenham que lutar para permanecer no mesmo lugar, mas ainda não estão escorregando.	28.8
Robô posicionado de forma que ele não possa se mover e com uma grande quantidade de torque aplicada em cada roda para que o controlador do motor tenha que lutar para manter as rodas firmes.	48 - 72

a bateria da Figura 4.3, uma bateria de $12Ah$ com 7 células em série e 5 células em paralelo (7S5P) na dimensões de 130 X 100 x 70 mm, poderia ser obtido uma bateria de maior capacidade, porém devido a restrições de mercado esta foi a de maior capacidade encontrada sendo 7S.

$$\text{Potência (W)} = \text{Carga (Ah)} \times \text{Tensão Nominal (V)} \tag{4.1}$$

$$\text{Energia (Wh)} = \text{Potência (W)} \times \text{Tempo (h)} \tag{4.2}$$

$$\text{Carga (Ah)} = \frac{\text{Energia(Wh)}}{\text{Tensão Nominal (V)}} \tag{4.3}$$

A bateria ilustrada na Figura 4.3 possui um BMS (Sistema de Gerenciamento de Bateria) no interior da sua isolação. Para acessar o BMS, é necessário remover a manga termo-retrátil da bateria, como pode ser visto na Figura 4.4 apos removido. Apesar de várias tentativas, não foi possível localizar o *Datasheet* deste modelo de BMS. No entanto, uma descrição do produto foi encontrada em lojas online [26], indicando que o BMS oferece apenas proteção de balanceamento, sobrecarga, sobredescarga e sobrecorrente, isto feito utilizando componentes de baixa qualidade e confiabilidade. Assim não apresentando uma operação segura e eficiente.

Figura 4.3: Bateria Li-Íon 7S5P.

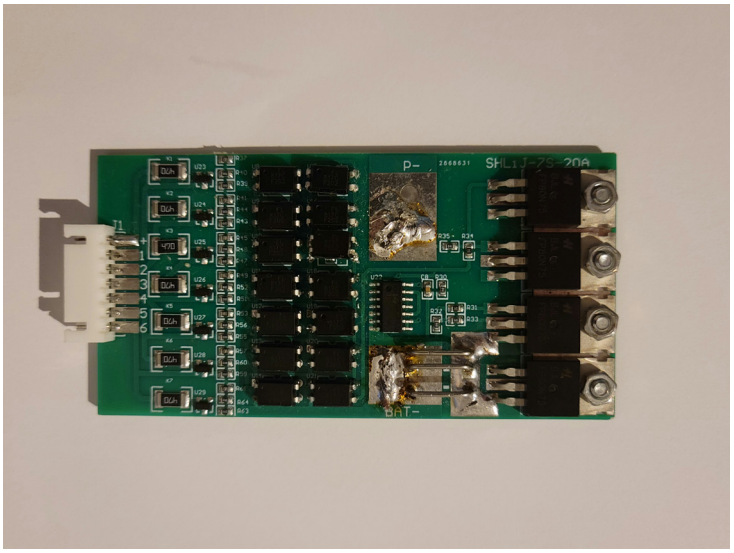

Figura 4.4: BMS originalmente instalado na bateria Li-Íon 7S5P.

Após a remoção do BMS originalmente instalado na bateria, foi preciso soldar novos cabos nas células para o seu uso no BMSi. Além disso, um sensor de temperatura foi inserido entre as células para possibilitar a medição da temperatura da bateria. Concluídos

esses passos, a bateria foi novamente isolada com uma manta termo-retrátil para garantir sua proteção. O resultado desse processo pode ser observado na Figura 4.5, onde a bateria preparada será utilizada no BMSi.

Figura 4.5: Bateria Li-Íon 7S5P preparada para uso no BMSi.

4.3 Primeira Versão do BMSi

Com base nas necessidades do Magni para o desenvolvimento do BMSi, a primeira versão que pode ser vista na Figura 4.6 foi concebida com uma abordagem menos integrada nos componentes. Esta versão visava a criação de uma PCB que incorporasse um monitor de células, um monitor de corrente, MOSFETs para chaveamento e um ESP32. Neste modelo, o ESP32 assumiria total controle e responsabilidade sobre todos os circuitos, implicando na necessidade de calcular a quantidade de carga da bateria com base nos dados fornecidos pelo CI monitor de corrente INA260 [27], esta tarefa se mostrou complexa.

Além disso, o ESP32 seria responsável pelos sistemas de segurança da bateria, tendo que detectar curto-circuito, sobre-tensão, sobrecorrente, temperatura, entre outros. Isso se deve ao fato de que o CI utilizado para monitorar a bateria, o LTC6804 [28], não possui internamente as proteções necessárias nem circuitos extras como driver para MOSFETs

e balanceamento das células. Isso exigiu a utilização de MOSFETs do tipo P, que são consideravelmente mais caros, e o desenvolvimento de um circuito para balanceamento das células. Considerando essas características, pode-se afirmar que esta solução é menos segura e eficiente.

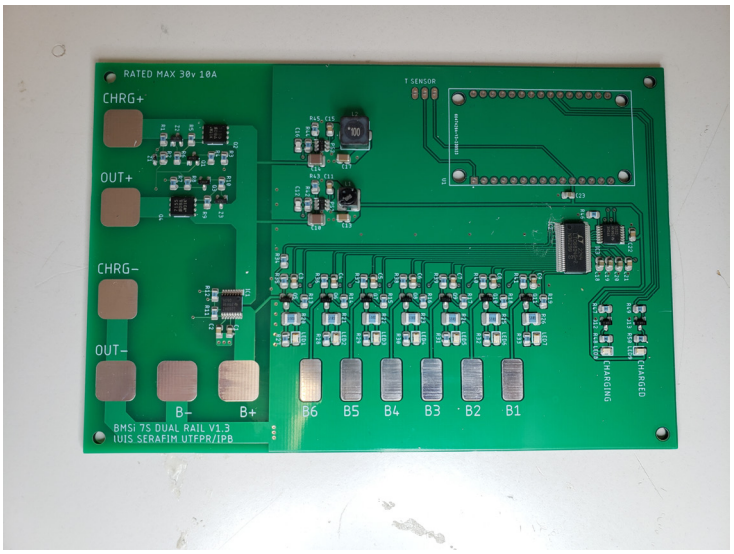

Figura 4.6: Primeira versão do BMSi.

Ao testar esta versão, ocorreu uma falha no circuito de balanceamento e no INA260 como pode ser visto na Figura 4.7, apesar de extensas análises sobre a falha desta versão, a causa exata ainda é desconhecida. No entanto, acredita-se que tenha sido devido a um erro de soldagem, que criou um curto-circuito entre o terminal da bateria e o GND no CI INA260. Este ponto de falha pode ser observado na Figura 4.7. Embora tenha havido esforços para testar os outros circuitos deste BMSi, todos foram danificados.

No entanto, devido ao alto custo dos componentes utilizados (aproximadamente 150 Euros), que é outra desvantagem deste protótipo, e à falta de capital do pesquisador, o desenvolvimento desta versão foi encerrado e o projeto prosseguiu com uma nova abordagem.

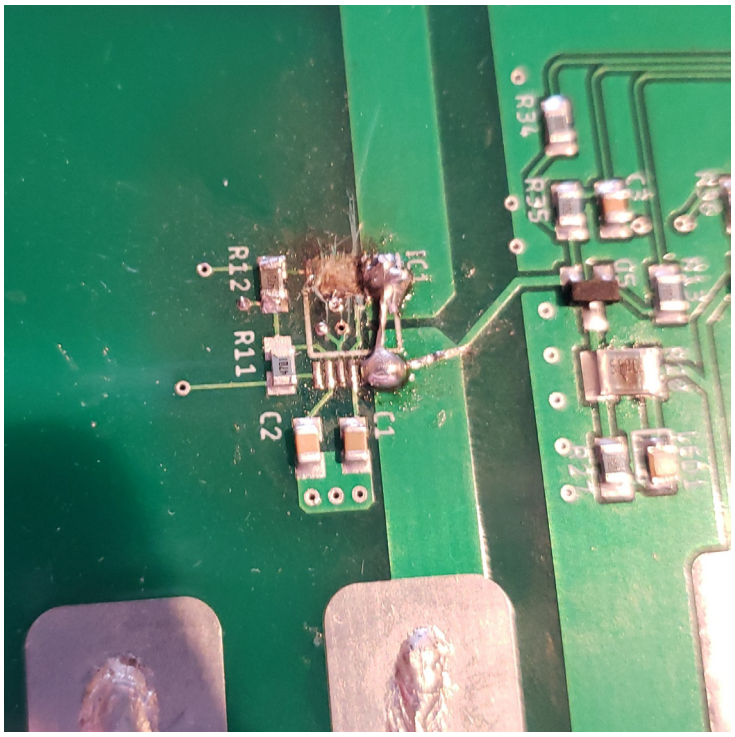

Figura 4.7: Ponto de falha da primeira versão do BMSi.

4.4 Segunda Versão do BMSi

No restante do desenvolvimento, será discutida a segunda versão do projeto BMSi (BMSiv2), que foi concebida com o objetivo de superar os desafios encontrados na primeira versão e, ao mesmo tempo, lidar com a limitação de recursos financeiros. A estratégia adotada para esta versão foi a utilização de componentes com maior relação custo-benefício, tornando o projeto mais acessível, porém capaz de atender a todos os requisitos e objetivos estabelecidos. Esta abordagem não apenas tornou o projeto mais eficiente comparado a primeira versão, mas também mais seguro.

As Figuras 4.8 e 4.9 apresentam o fluxograma do funcionamento do BMSiv2, ilustrando suas operações. É possível observar, em cinza, os comandos utilizados da biblioteca desenvolvida para o CI BQ76942. Isso demonstra como a utilização desta biblioteca que será abordada em detalhe na Seção 4.9, simplifica a sua implementação, sendo necessário apenas o uso dos métodos da classe (no fluxograma, o objeto criado é denominado BQ).

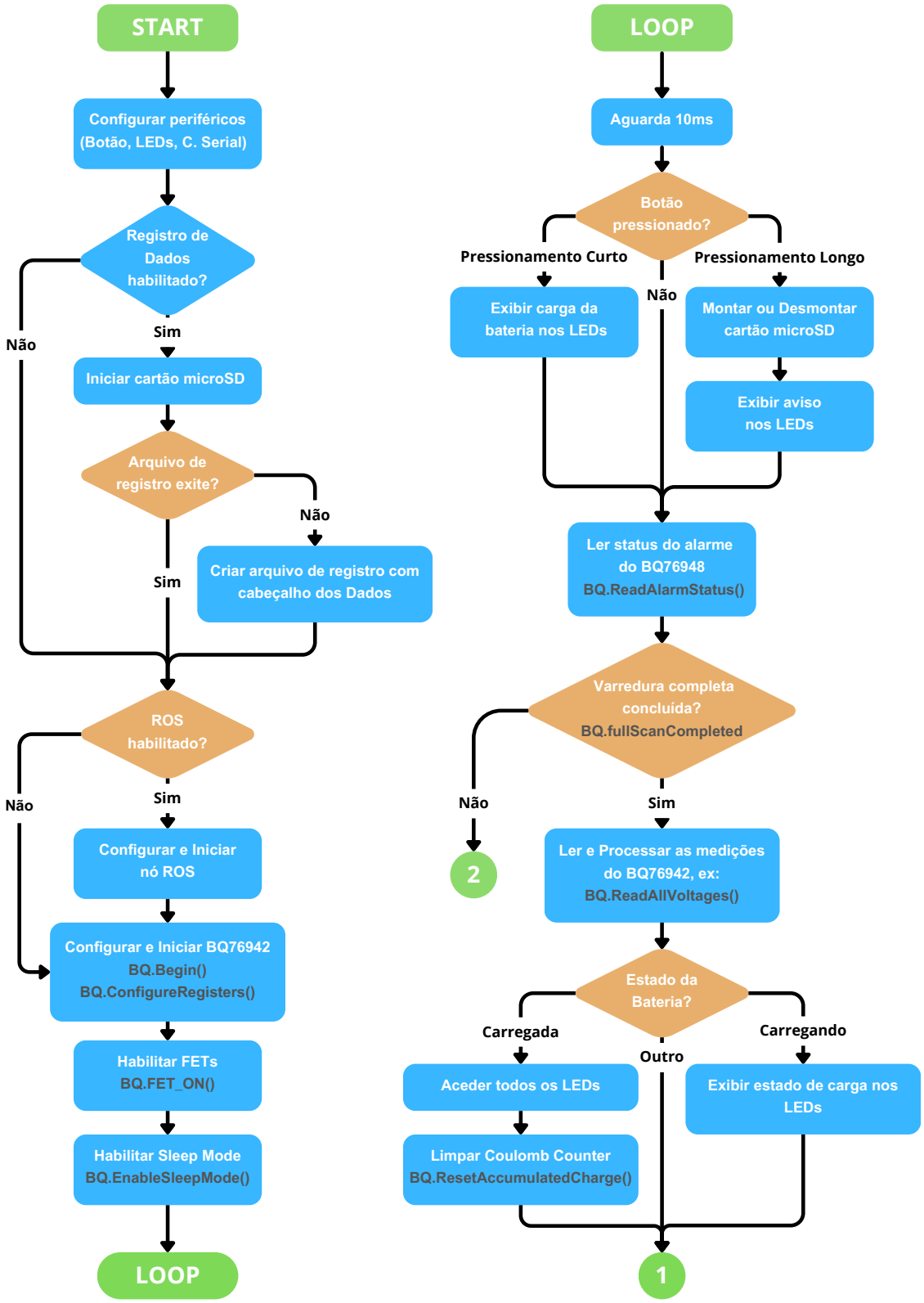

Figura 4.8: Fluxograma do funcionamento do BMSiv2 - Parte 1.

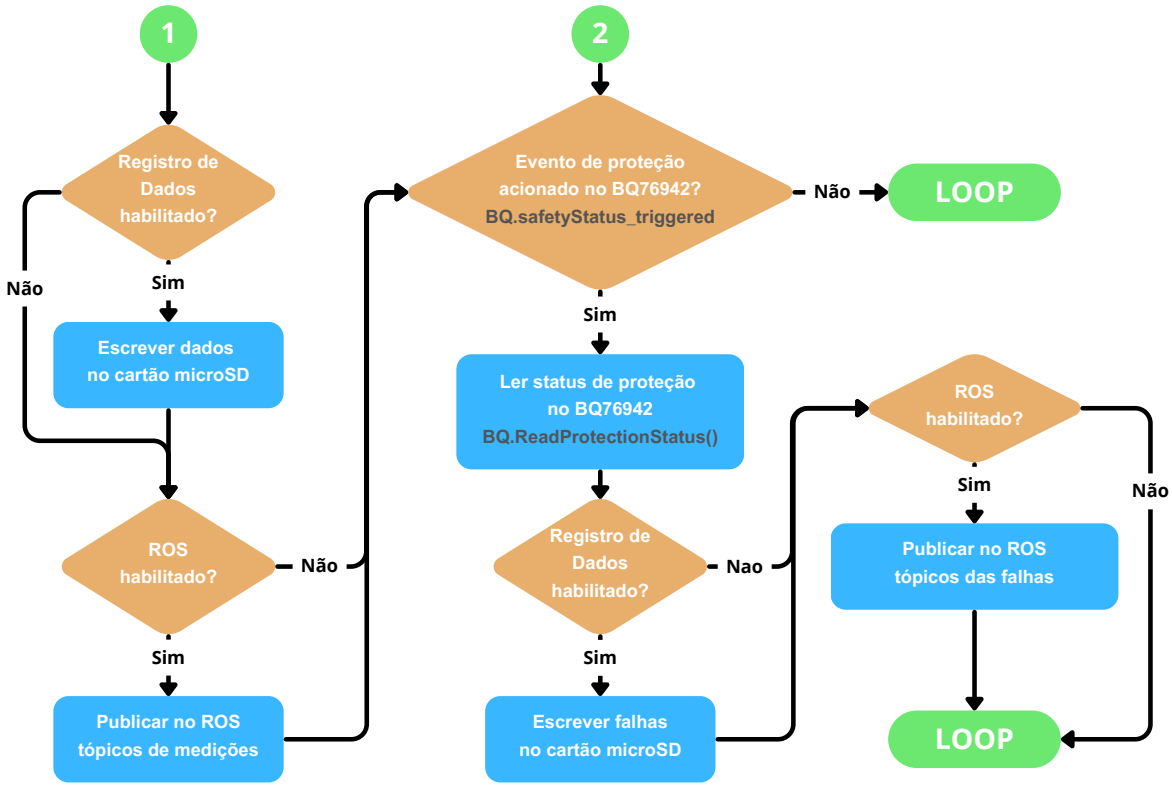

Figura 4.9: Fluxograma do funcionamento do BMSiv2 - Parte 2.

O BMSiv2 possui dois modos de operação: comunicação com ROS ou via terminal serial para *debugging*, que podem ser selecionados de acordo com a necessidade ao definir USE_ROS ou USE_TERMINAL no código do ESP32. No entanto, o fluxograma acima apresentou apenas a comunicação com ROS, por ser o método principal de operação, e o utilizado pelo robô Magni.

Tendo apresentados os Requisitos do robô Magni na seção 4.1, e considerado as características das baterias de Li-Íon para o seu uso com segurança o BMSiv2, a Tabela 4.4 contém os níveis de proteções e características de operação do BMSiv2. Estes valores podem ser configurados de acordo com a necessidade do sistema, mas para o Magni serão considerados o da tabela para o desenvolvimento do projeto.

Tabela 4.4: Características e proteções do BMSiv2.

Tensão Máxima do Sistema	$29.4V$
Tensão Miníma do Sistema	$23.1V$
Quantidade de Células em série do Sistema	7 Células
Proteção Sobrecorrente de Descarga (OCD)	$15A$ por $50ms$ $25A$ por $10ms$
Proteção Curto-Circuito de Descarga (SCD)	$50A$ por $30\mu s$
Proteção Sobrecorrente de Carga (OCC)	$10A$ por $50ms$
Proteção Sobretemperatura de Descarga (OTD)	$60°C$
Proteção Sobretemperatura de Carga (OTC)	$55°C$
Proteção Baixa temperatura de Descarga (UTD)	$-10°C$
Proteção Baixa temperatura de Carga (UTC)	$0°C$
Proteção Sobrecarga das Células	$4.2V$ por $240ms$
Histerese de Recuperação de Sobrecarga das Células	$100mV$
Proteção Sobredescarga das Células	$3.3V$ por $240ms$
Histerese de Recuperação de Sobredescarga das Células	$200mV$
Delta de Balanceamento das Células	$30mV$

A segunda versão do projeto, no entanto, apresentou seus próprios desafios. A necessidade de equilibrar custo, eficiência e segurança exigiu uma análise cuidadosa na seleção de componentes e na estratégia de implementação. Assim também serão discutidos os desafios enfrentados e as estratégias adotadas para superá-los.

4.5 Principais Componentes Utilizados

Nesta seção, serão abordados os componentes de maior complexidade incorporados no desenvolvimento do projeto. A ênfase nesses elementos específicos é motivada pelos desafios substanciais que representam em termos de implementação. Através de uma análise detalhada, busca-se esclarecer as características e qualidades que tornam esses componentes indispensáveis para a execução do projeto. Esta abordagem proporciona

uma compreensão mais aprofundada dos aspectos técnicos envolvidos e da importância desses componentes na estrutura geral do projeto

4.5.1 BQ76942

A escolha do CI BQ76942 para o projeto foi baseada em sua alta integração e baixo custo comparados a outros CIs do gênero, que simplificam a necessidade de circuitos externos no funcionamento do BMSiv2. Este componente possui sistemas internos altamente complexos e integrados, conforme ilustrado na Figura 4.10, o que torna sua comunicação um desafio.

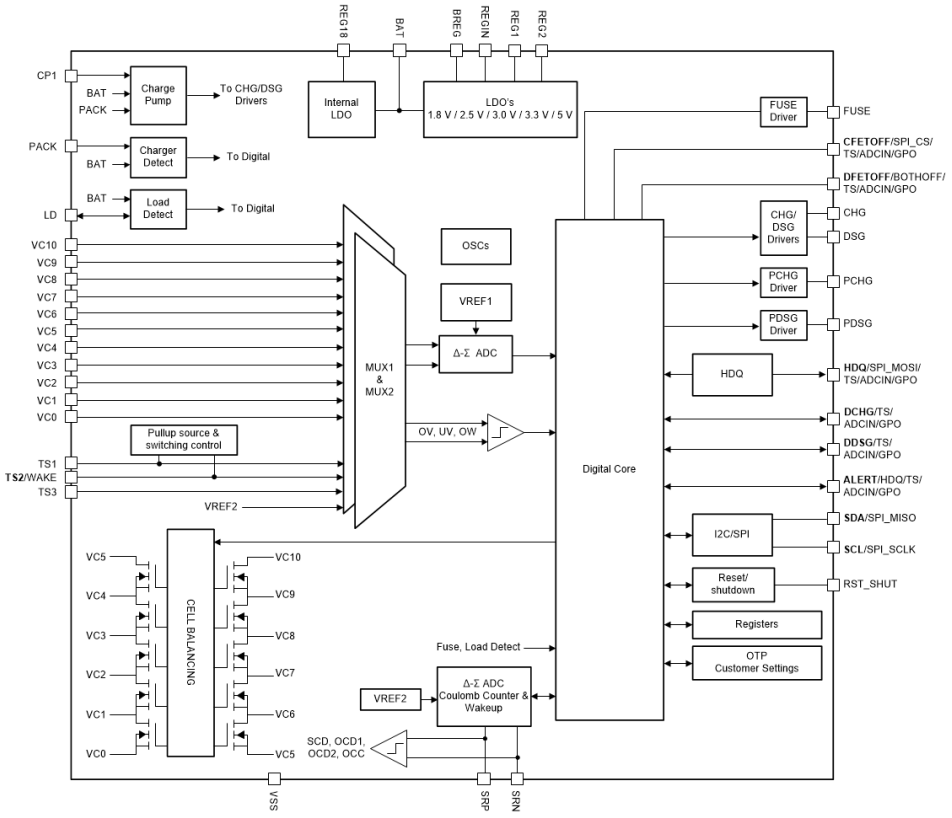

Figura 4.10: Diagrama de bloco do CI BQ76942 [2].

No entanto, suas características principais, conforme descrito em seu *Datasheet* [19], destacam-se como a opção ideal para este projeto, oferecendo uma solução eficiente e acessível para o gerenciamento de baterias. Entre estas estão:

- **Monitoramento de baterias de 3 a 10 células:** Capaz de monitorar a tensão de cada célula usando um circuito baseado em comparador, acionando um alerta ou falha quando a tensão de uma célula excede o limite de sobretensão (COV). O limite de COV é programável de $1.0V$ a $5.5V$ em passos de $50mV$.

- ***Charge pump* para MOSFETs tipo N em *high-side switching*:** Inclui uma bomba de carga integrada e drivers de *high side switching* para MOSFETs do tipo N para conduzir os MOSFETs de Carga (CHG) e Descarga (DSG).

- **Proteções configuráveis extensas:** Contém uma ampla gama de proteções para o gerenciamento de baterias, incluindo proteções individuais que podem ser habilitadas e configuradas.

- **Controle autônomo:** Neste modo pode detectar falhas de proteção e desativar autonomamente os MOSFETs, monitorar uma condição de recuperação e reativar autonomamente os MOSFETs, sem a necessidade de envolvimento do processador host.

- **Tolerâncias a conexão aleatória das células:** permite a conexão de células em qualquer ordem, o que proporciona flexibilidade no design do sistema de bateria.

- **Alta exatidão do *Coulomb Counter*:** O BQ76942 possui um contador de Coulomb de alta precisão com desvio menor que $1\mu V$, que é essencial para o monitoramento preciso da capacidade da bateria.

- **Comunicação I2C ou SPI:** Suporta comunicação através dos protocolos I2C ou SPI, permitindo a integração fácil com uma variedade de microcontroladores e outros dispositivos de *hardware*.

O BQ76942 oferece uma série de proteções primárias que são indispensáveis para garantir a segurança e eficiência do sistema. Estas proteções são fundamentais para prevenir potenciais danos e garantir um desempenho otimizado. A seguir, será detalhado

cada uma dessas proteções, explicando sua importância e como elas contribuem para o funcionamento geral do sistema.

- **Subtensão das células (CUV):** Esta proteção é acionada quando a tensão em qualquer célula da bateria cai abaixo de um limite pré-definido, indicando que a célula está sendo excessivamente descarregada.

- **Sobretensão das células (COV):** Esta proteção é acionada quando a tensão em qualquer célula da bateria excede um limite pré-definido, indicando que a célula está sendo excessivamente carregada.

- **Sobrecorrente em carga (OCC):** Esta proteção é acionada quando a corrente de carga excede um limite pré-definido, indicando que a bateria está sendo carregada a uma taxa que pode ser prejudicial.

- **Sobrecorrente em descarga com tres níveis de proteção (OCD):** Esta proteção é acionada quando a corrente de descarga excede um de três limites pré-definidos. A ação tomada depende de qual limite é excedido.

- **Curto circuito em descarga (SCD):** Esta proteção é acionada quando um curto-circuito é detectado durante a descarga, o que pode causar danos significativos à bateria e a outros componentes do sistema.

- **Baixa temperatura em carga (UTC) e baixa temperatura em descarga (UTD):** Estas proteções são acionadas quando a temperatura durante a carga ou descarga cai abaixo de um limite pré-definido, o que pode indicar condições que são prejudiciais para a bateria.

- **Sobreaquecimento em carga (OTC) e Sobreaquecimento em descarga (OTD):** Estas proteções são acionadas quando a temperatura durante a carga ou descarga excede um limite pré-definido, o que pode indicar condições que são prejudiciais para a bateria.

- **Sobreaquecimento dos MOSFETs (OTFET):** Esta proteção é acionada quando a temperatura dos MOSFETs excede um limite pré-definido, o que pode indicar que estes estão operando fora de suas especificações seguras.

O CI BQ76942 suporta comandos diretos e subcomandos para comunicação, seja por I2C ou SPI. Os comandos diretos são acessados enviando um comando de 7 bits para o endereço de comando 0x3E, que pode iniciar uma ação, escrever um valor no CI ou instruir o dispositivo a reportar algum dado ao *host*. Os subcomandos são comandos adicionais de 16 bits que são acessados indiretamente, escrevendo primeiro o endereço do subcomando nos endereços de comando 0x3E (byte inferior) e 0x3F (byte superior). Isso permite a transferência de blocos de dados [1], [19].

Os comandos diretos e subcomandos no BQ76942 são realizados escrevendo e lendo registradores do dispositivo. Para utilizar comandos diretos e subcomandos, é necessário seguir múltiplos passos de escrita e leitura, mantendo a mesma estrutura básica padrão em comunicações I2C, conforme ilustrado na Figura 4.11.

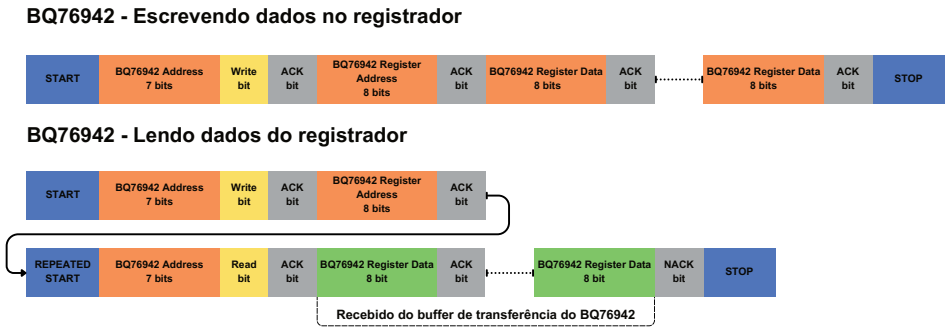

Figura 4.11: Leitura e escrita de registradores no BQ76942 em I2C.

Quando um subcomando é iniciado, o dispositivo assume inicialmente que uma leitura de dados pode ser necessária e preenche automaticamente um *buffer* de transferência de 32 bytes (endereço de comando 0x40 a 0x5F) com os dados existentes e escreve o *checksum* desses dados no endereço 0x60. Se o *host* pretende escrever dados no dispositivo, ele sobrescreve os novos dados no *buffer* de transferência, o *checksum* no endereço 0x60, e o comprimento dos dados no endereço 0x61. Assim que o endereço 0x61 é escrito,

o dispositivo verifica o *checksum* e, se estiver correto, transfere os dados do *buffer* de transferência para a memória do dispositivo. Alguns subcomandos são usados apenas para iniciar uma ação e não envolvem o envio ou recebimento de dados. Nesses casos, o *host* pode simplesmente escrever o subcomando nos endereços 0x3E e 0x3F, sem a necessidade de escrever o comprimento, *checksum* ou qualquer dado adicional [1], [19]. Este processo de comunicação utilizando subcomandos e comandos diretos será abordado com mais profundidade na seção 4.9.

A implementação padrão do BQ76942, ilustrada na Figura 4.12, foi adaptada para este projeto, com alterações que serão discutidas em detalhes na seção 4.6.

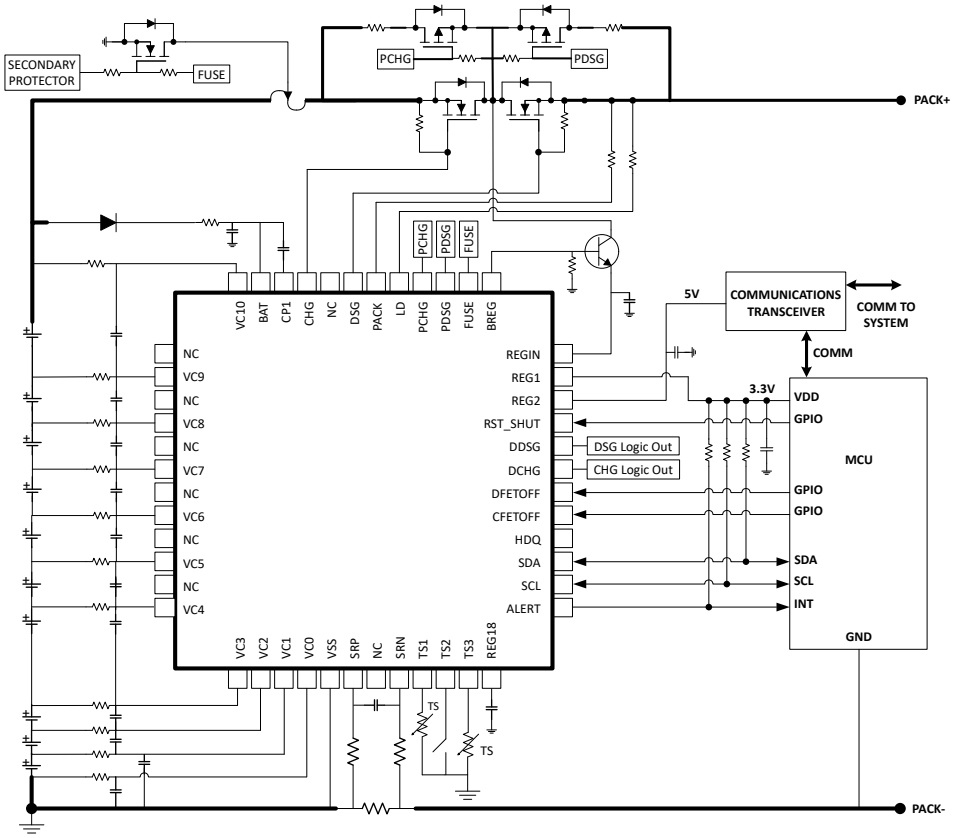

Figura 4.12: Implementação típica simplificada do BQ76942 [19].

O BQ76942 foi integrado ao sistema BMSiv2 com a finalidade de proteger e monitorar a bateria, além de controlar os MOSFETs de chaveamento, comunicando-se com o ESP32 por meio do protocolo I2C. Quando ocorre um evento no BQ76942, como a disponibilidade

de novas medições ou a ativação de proteções, o pino de Alerta do BQ76942 (em estado de alta impedância quando inativo) é puxado para o GND. Este pino é utilizado como interrupção externa do ESP32, indicando a ocorrência de um evento para ser lido por I2C e ser processado. Nesse contexto, todos os dados são recebidos e processados para serem enviados ao robô Magni.

As configurações principais necessárias no BQ76942 para o uso no BMSiv2 incluem a habilitação dos MOSFETs, a configuração do pino de Alerta para interrupção do ESP32, a instalação de um sensor de temperatura no pino TS1 do BQ76942 para medição da temperatura dos MOSFETs, e um sensor de temperatura no pino TS2 do BQ76942 para medição da temperatura da bateria. Além disso, é necessário configurar o sistema para o uso de 7 células, habilitar e configurar as proteções, ativar a realização das medições necessárias e ativar o balanceamento em estado de carga e em relaxamento da bateria.

4.5.2 MOSFETs

Para o BMSiv2, optou-se pelo uso de MOSFETs do tipo N para o chaveamento de carga e descarga. Essa escolha se deve às características desses dispositivos, como a baixa resistência de dreno-fonte ($RDS(on)$), que minimiza as perdas de potência durante o chaveamento e permite uma alta capacidade de corrente. Isso é essencial para lidar com as demandas de corrente das baterias de Li-Íon utilizadas no sistema. Os MOSFETs foram configurados em *back-to-back* com drenos conectados. Nessa configuração, dois MOSFETs são conectados em série, permitindo que controlem a corrente em ambas as direções (carga e descarga), o que é crucial para o funcionamento de uma bateria. Essa configuração foi escolhida por sua eficiência e confiabilidade.

Os MOSFETs foram utilizados para realizar o chaveamento do terminal positivo da bateria, um modo de operação conhecido como *high side switching*, demostrado a comparação com *low side switching* na Figura 4.13. Com essa configuração, o circuito alimentado mantém a mesma referência GND, pois não há interrupção nem queda de tensão no GND.

Isso é importante para sistemas que precisam se comunicar, como o robô Magni e o BMSiv2. Além disso, esse modo de operação oferece maior segurança, pois a estrutura e os sistemas do Magni estão conectados ao GND, tornando mais seguro desconectar o positivo da bateria do que o GND.

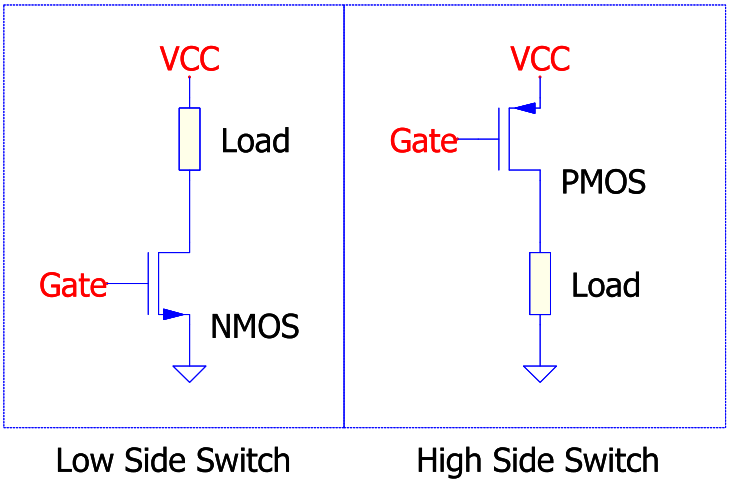

Figura 4.13: *High Side Switching* E *Low Side Switching.*

No entanto, essa operação é mais complexa de implementar, especialmente ao usar MOSFETs do tipo N, pois é necessária uma tensão porta-fonte (VGS) positiva para ativar o MOSFET. Isso é problemático, pois a tensão de porta precisa ser maior que a tensão máxima do sistema para entrar em condução. No entanto, devido ao CI BQ76942 possuir *drivers* integrados para MOSFETs do tipo N em *high side switching*, a implementação desse modo de operação se torna extremamente simples.

Assim conhecendo os requisitos do BMSiv2 e da bateria, utilizando um fator de segurança de 50%, é necessário utilizar MOSFETs do tipo N que suportem uma tensão de dreno-fonte de no mínimo $45V$, e uma corrente de dreno de $22A$. Foi decidido utilizar MOSFETs modernos pois possuem baixa resistência de dreno-fonte ($RDS(on)$) (existindo modelos abaixo de $10m\Omega$) para manter ao mínimo as perdas de potência e seu aquecimento. Não havendo uma maneira simples de escolher um MOSFET, isto foi feito realizando cálculos com diversos modelos até encontrar um que satisfaça as necessidade mantendo um baixo custo. Assim o modelo escolhido foi o da Figura 4.14 o IRFB3207

da International Rectifier, com as suas principais características obtidas no seu *Datasheet* [29] e destacadas abaixo:

- Tensão de ruptura de dreno-fonte máxima de $75V$;
- Tensão de porta-fonte máxima de $\pm 20V$;
- Corrente de drenagem contínua máxima a $25°C$ de $170A$;
- Corrente de fuga de Dreno-Fonte máxima de $20\mu A$;
- Resistência estática de Dreno-Fonte de $3.6m\Omega$ a $25°C$ e de $8.2m\Omega$ a $175°C$;
- Resistencia térmica de junção-ambiente máxima de $62°C/W$;
- Temperatura de operação da junção máxima de $175°C$;

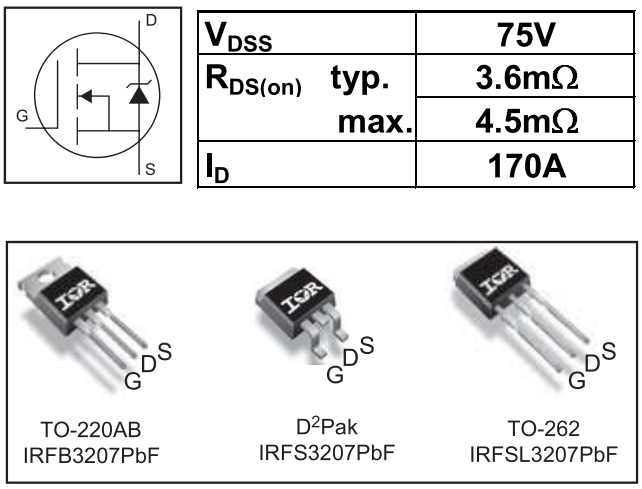

Figura 4.14: MOSFET IRFBP3207 encapsulamento TO-220 [29].

Assim foi calculado a dissipação de calor e perda de potência deste MOSFET operando a $15A$ para confirmar que atende os requisitos do projeto.

Para calcular a corrente máxima que o MOSFET suporta sem dissipador de calor, considerando uma temperatura ambiente média de $25°C$, conhecendo a temperatura máxima da junção ($Tjmax$) e a resistência térmica de junção-ambiente ($Rthja$) foi possível calcular a potência máxima que pode ser dissipada no MOSFET utilizando a Equação (4.4):

$$\begin{aligned} \text{Potência Dissipada} &= \frac{Tjmax - Ta}{Rthja} \\ \text{Potência Dissipada} &= \frac{175°C - 25°C}{62°C/W} \\ \text{Potência Dissipada} &= 2.42 \end{aligned} \tag{4.4}$$

Para calcular a corrente máxima que pode ser conduzida sem exceder essa potência, utiliza-se a Equação (4.5):

$$\begin{aligned} \text{Corrente} &= \sqrt{\frac{\text{Potência Dissipada}}{RDS(on)}} \\ \text{Corrente} &= \sqrt{\frac{2.42W}{8.2m\Omega}} \\ \text{Corrente} &= 17.18A \end{aligned} \tag{4.5}$$

Portanto, considerando que o BMSiv2 possui proteção de corrente em $15A$ e que o Magni utiliza no máximo $3A$, com exceções apenas em raras ocasiões de picos transientes, o MOSFET IRFB3207 atende aos requisitos do projeto, suportando a corrente máxima de $15A$ sem a necessidade de um dissipador de calor. Para estimar a temperatura de junção (Tj) do MOSFET em sua operação normal a $3A$, é possível utilizar a equação da lei de Ohms mencionadas anteriormente:

$$\begin{aligned} \text{Potência Dissipada} &= \text{Corrente}^2 \times \text{RDS(on)} \\ \text{Potência Dissipada} &= 3\text{A}^2 \times 3.6\text{m}\Omega \\ \text{Potência Dissipada} &= 32\text{mW} \end{aligned}$$

Para calcular a temperatura de junção (Tj) do MOSFET, utiliza-se novamente a Equação (4.4) arranjada para calcular Tj:

$$Tj = (\text{Potência Dissipada} \times Rthja) + Ta$$
$$Tj = (32mW \times 62°C/W) + 25°C$$
$$Tj = 26.98°C$$

Com base nesses cálculos, a temperatura de junção (Tj) do MOSFET em sua operação normal a $3A$ é de aproximadamente $26.98°C$, mostrando que não é necessário o uso de dissipadores no modelo de MOSFET utilizado.

4.5.3 AP63203

O ESP32 utilizado no BMSiv2 precisa ser alimentado com $3.3V$, porem devido a utilização do ESP32 em um *devkit*, o kit de desenvolvimento da Espressif [10] contém um conversor de tensão linear DC-DC que reduz a tensão de $5V$ para $3.3V$, dissipando a energia restante na forma de calor. No entanto, essa abordagem não é eficiente, especialmente em sistemas alimentados por bateria, como o BMSiv2. Além disso, a tensão máxima suportada por este conversor não é compatível com a tensão da bateria de $29.4V$.

Considerando esses fatores, é necessário o uso de um conversor de tensão DC-DC que seja compatível com o sistema utilizado. Portanto, decidiu-se utilizar a família AP63200, que é um conversor de tensão chaveado síncrono de alta eficiência e baixo custo, com MOSFETs integrados ao CI, facilitando a implementação e o dimensionamento.

Este CI suporta uma tensão máxima de $32V$ e uma corrente máxima de saída de $2A$. A família AP63200 inclui vários modelos, como o AP63200, que tem tensão totalmente ajustável, e o AP63203, que tem uma tensão fixa de 3.3V. Para simplificar o projeto, optou-se pelo uso do AP63203, que atende às necessidades do projeto, exigindo poucos componentes externos.

A Figura 4.15 demostra o circuito da implementação tipica do AP63203. Este circuito difere um pouco na implementação do BMSiv2, isto sera justificado na Seção 4.6.

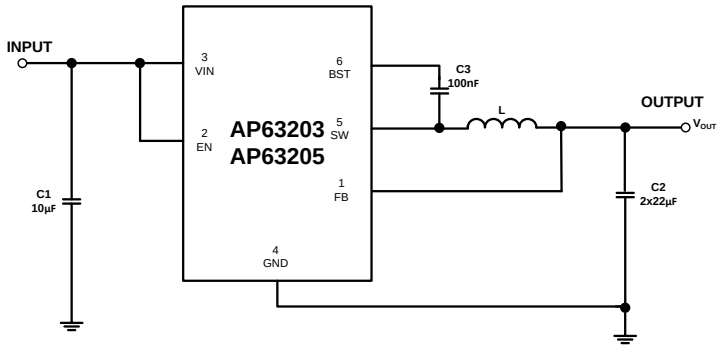

Figura 4.15: Implementação tipica do conversor de tensão DC-DC AP63203 [30].

4.6 Circuitos desenvolvidos para o BMSiv2

Nesta seção, será abordado os circuitos elaborados para o BMSiv2. Estes incluem o circuito de interface homem-máquina (IHM), os circuitos dos MOSFETs, os circuitos externos do BQ76942, o circuito do cartão microSD e o circuito de comunicações. Estes foram projetados para cumprir os requisitos do BMSiv2, com o objetivo de assegurar a segurança e otimizar a eficiência do sistema.

4.6.1 Circuitos de Interface Homem-Máquina (IHM)

A Interface Homem-Máquina (IHM) do circuito é uma parte crucial do BMSi. Permitindo que o usuário interaja com o sistema e obtenha informações importantes sobre o estado da bateria, como a carga atual e possíveis falhas.

O circuito de IHM é controlado por um ESP32, responsável por gerenciar a interação entre o usuário e o BMSi. Sendo a IHM composta por quatro LEDs de baixo consumo e corrente, fornecendo uma indicação visual do estado da bateria e do sistema. Por exemplo, indicando se a bateria está carregada, descarregada, em processo de carregamento ou se há alguma falha no sistema.

Um botão foi incluído na IHM para permitir a interação do usuário com o sistema. Este botão pode ser usado para exibir o estado da bateria caso seja pressionado brevemente, ou ejetar o cartão microSD caso pressionado por $5s$, garantindo a segurança dos dados

escritos.

A IHM também inclui um conector para um cartão microSD. Este cartão é usado para o registro de dados, permitindo que o sistema salve informações importantes sobre o estado e o desempenho da bateria ao longo do tempo, bem como falhas detectadas. Isso pode ser útil para a análise de desempenho e para a identificação e resolução de problemas. O circuito necessário para o cartão microSD é integrado ao circuito de IHM. Incluindo os componentes necessários para a comunicação entre o cartão e o ESP32.

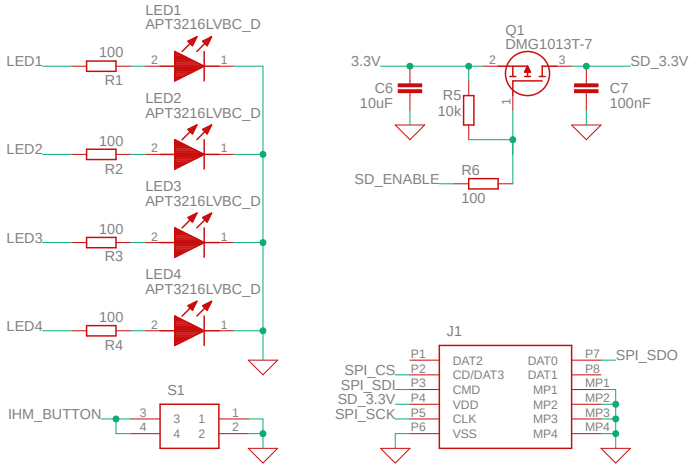

Figura 4.16: Circuito de IHM do BMSiv2.

4.6.2 Circuitos externos do BQ76942

Embora o CI BQ76942 seja altamente integrado, ainda necessita de circuitos auxiliares externos, como o circuito de filtragem e proteção da alimentação, circuito de filtragem e de balanceamento das células, circuito de sensoriamento de corrente, circuito dos termistores e de comunicação I2C. O desenvolvimento desses circuitos foi realizado seguindo o *Datasheet* do BQ76942 e documentos auxiliares encontrados na página do BQ76942 [1], [2], [19], [23].

O circuito desenvolvido para BQ76942 neste projeto difere da implementação típica demonstrada na Figura 4.12 no circuito dos MOSFETs, no circuito de sensor de corrente

e na não utilização de um circuito regulador de tensão externo do BQ76942. Esses elementos são normalmente utilizados caso sejam empregados os reguladores internos REG1 e REG2 do BQ76942 e para comunicação *push-pull*. No entanto, no caso deste projeto, os reguladores do BQ76942 não são utilizados, devido a utilização do modo de comunicação I2C em *open-drain*. O circuito de MOSFET difere pois não foi implementado o uso de MOSFETs tipo P para *soft-start* do BMSiv2, simplificando assim o circuito dos MOSFETs. Com essas diferenças em mente, o circuito completo do BQ76942 resultante é o da Figura 4.17. A seguir, será detalhado e justificado cada um desses circuitos.

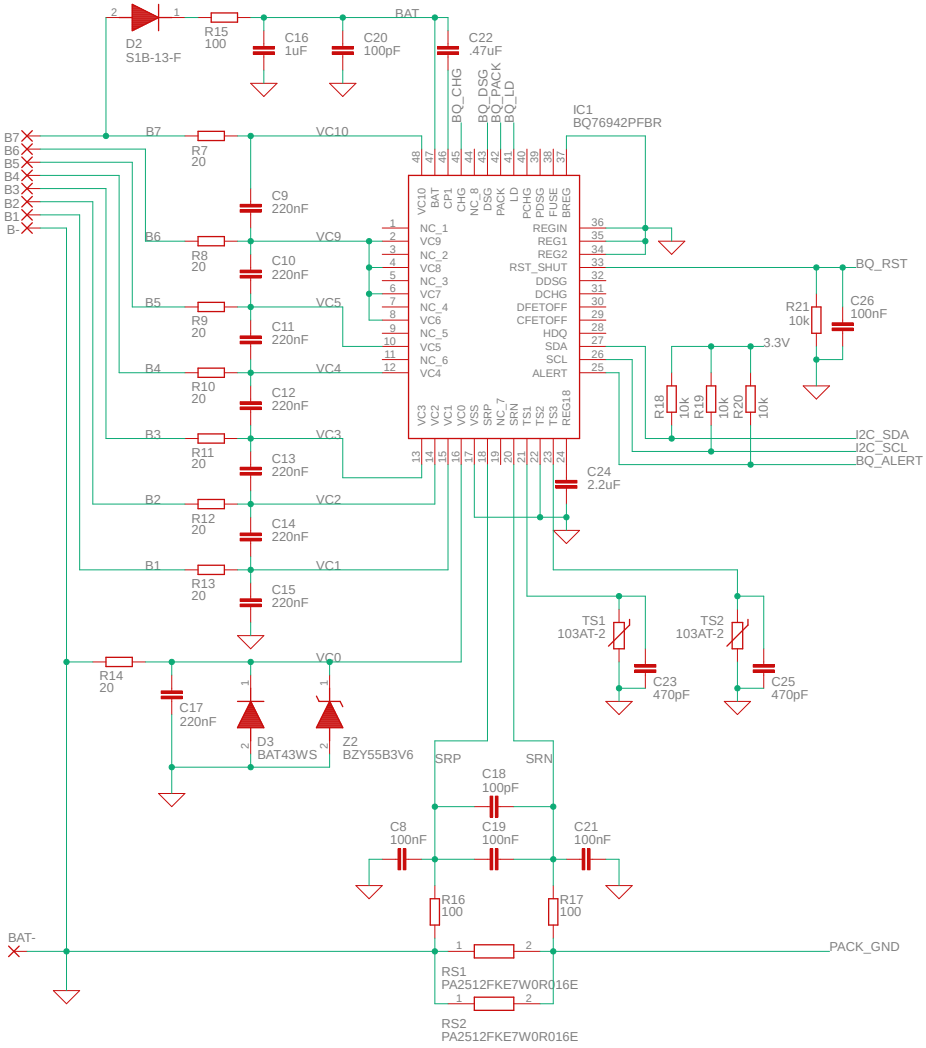

Figura 4.17: Circuito do BQ76942 do BMSiv2.

Alimentação do BQ76942

Para a alimentação do BQ76942 e do seu *charge pump*, é aconselhado pelo *Datasheet* o uso de um diodo em série com o terminal BAT do BQ76942 e de capacitores conectados do BAT ao GND (D2, R15, C16 e C20 da Figura 4.17, com valores recomendados pelo *Datasheet*). Isso garante que o dispositivo continue funcionando por um breve período em caso de curto-circuito, o que faria o terminal PACK+ e BAT cair para cerca de $0V$ [19]. O diodo impede que a tensão do terminal BAT caia, permitindo tempo suficiente para ativar a proteção contra curto-circuito e desativar o MOSFET de descarga.

A alimentação do *charge pump*, que é usado para ativar os MOSFETs, pode ser derivada do circuito usado para alimentar o BQ76942, compartilhando assim seus benefícios. No entanto, é necessário um capacitor (C22 da Figura 4.17) para gerar a tensão necessária para os MOSFETs, usando o valor recomendado pelo *Datasheet*.

Filtragem e Medição das Células

Para medir a tensão e realizar o balanceamento das células da bateria, é necessário utilizar resistores e capacitores externos, conforme ilustrado na Figura 4.18. Os capacitores e resistores atuam como filtros passa-baixa para a leitura da tensão, enquanto os resistores também funcionam como dissipadores de potência para o balanceamento. O dimensionamento deste circuito é sugerido pela nota de aplicação do balanceamento [23]. Devido à capacidade de balanceamento interno do BQ76942, este modo foi escolhido para este projeto, já que não há necessidade de grandes correntes de balanceamento. A corrente de $65mA$, sugerida na nota de aplicação, é suficiente. O resistor recomendado para o balanceamento interno de 20Ω foi utilizado, maximizando a corrente de balanceamento e mantendo a dissipação interna no BQ76942 dentro dos valores recomendados. Essa corrente é de $65mA$.

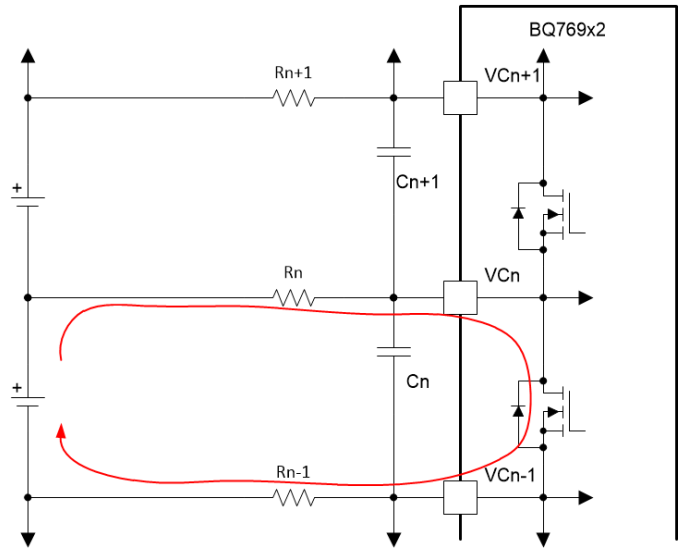

Figura 4.18: Circuito de balanceamento interno do BQ76942 [23].

Sensoriamento de Corrente

A medição de corrente é efetuada através da medição da tensão diferencial nos terminais SRP e SRN, estes terminais estão conectados a um resistor *shunt*. Contudo, devido à dissipação de potência neste resistor, é necessário o uso de resistências extremamente baixas, tipicamente em miliOhms ($m\Omega$). Segundo o *Datasheet* do BQ76942, a tensão máxima em situação normal de carga e descarga sobre esta resistência deve ser de $200mV$ para realizar as medições [19]. Considerando que o BMSiv2 deve ter uma corrente máxima de $15A$, pela lei de Ohm, calcula-se uma resistência de $8m\Omega$. Assim, a potência dissipada em $15A$ é de $1.8W$. Devido a preocupações com o sobreaquecimento do resistor *shunt*, optou-se por utilizar dois resistores de $16m\Omega$ em paralelo, resultando nos $8m\Omega$ necessários, dissipando em cada um $0.9W$. O resistor utilizado foi o PA2512FKE7W0R16E, um resistor *shunt* de precisão de $16m\Omega$ com 1% de tolerância e potência máxima de $2W$, com um coeficiente de temperatura de $50PPM/°C$, variando pouco a sua resistência com a temperatura.

O *Datasheet* do BQ76942 também recomenda a utilização de resistores de 100Ω em série com os seus terminais SRP e SRN, e capacitores de $100nF$ e $100pF$ em paralelo para filtragem diferencial da tensão sobre o resistor *shunt*. Além disso, sugere capacitores de

$100nF$ em cada terminal SRP e SRP conectado ao GND para filtragem adicional [19]. O *Datasheet* indica também que as trilhas na PCB do resistor *shunt* até os terminais SRP e SRN devem ser de menor comprimento possível e totalmente simétricas. Com todas essas considerações de desenvolvimento em mente foi obtido o circuito para medição da corrente demostrado Figura 4.17.

Sensoriamento de Temperatura

O CI BQ76942 é capaz de monitorar a temperatura através da leitura de termistores externos, que são sensores de temperatura. Esses termistores podem ser configurados para monitorar a temperatura da bateria, dos MOSFETs ou qualquer outra configuração personalizada para o sistema. Conforme indicado no *Datasheet*, é recomendado o uso de capacitores conectados do terminal do termistor ao GND para compensar a necessidade de fios longos comumente associada aos sensores de temperatura. Usando a resistência interna de *pull-up* do BQ76942 de $18k\Omega$ ou $180k\Omega$, o capacitor deve ser $< 4nF$ ou $< 400pF$, respectivamente.

Para a medição da temperatura da bateria e do MOSFET, optou-se por utilizar dois termistores NTC 103AT-2 de $10k\Omega$, devido à sua precisão e baixo custo [31]. Como esses são termistores de $10k\Omega$, foram utilizados capacitores de $470pF$ e resistência de *pull-up* interna do BQ76942 de $18k\Omega$, conforme indicado pelo *Datasheet* [19]. Esses termistores estão conectados aos terminais TS1 e TS3 do BQ76942.

Circuito de comunicação

A comunicação entre o BQ76942 e o ESP32 é estabelecida através dos protocolos de comunicação serial I2C ou SPI. O I2C, por sua simplicidade, é frequentemente a escolha preferida. O BQ76942 suporta dois modos de comunicação: *push-pull*, onde o sinal alterna entre GND e $3.3V$, e *open-drain*, onde o sinal está conectado ao GND ou em um estado de alta impedância.

O modo *open-drain* é comumente utilizado em comunicações I2C devido à sua simplicidade. No caso do BQ76942, essa configuração é particularmente vantajosa, pois o uso do

modo *push-pull* exigiria circuitos externos de regulador de tensão para que o BQ76942 pudesse utilizar seus reguladores internos de tensão REG1 e REG2. No entanto, ao utilizar o modo *open-drain*, não é necessário o uso desses reguladores de tensão.

Por essas razões, optou-se pela comunicação I2C com *open-drain*. Para manter a comunicação em nível lógico alto no estado de alta impedância, é necessário o uso de resistores de *pull-up* conectados a $3.3V$. Assim, resistores de $10k\Omega$ são necessários e devem ser conectados aos pinos SDA, SCL e ALERT do BQ76942, conforme pode ser observado na Figura 4.17.

Circuitos MOSFET do BQ76942

O circuito para os MOSFETs foi desenvolvido com base nas recomendações do *Datasheet* do BQ76942 [19] e na nota de aplicação sobre MOSFET com o BQ76942 [22]. Essas referências foram fundamentais para a elaboração do circuito na segunda versão do Sistema Inteligente de Gerenciamento de Bateria (BMSi). Resultando no circuito para operação e proteção dos MOSFETs da Figura 4.19.

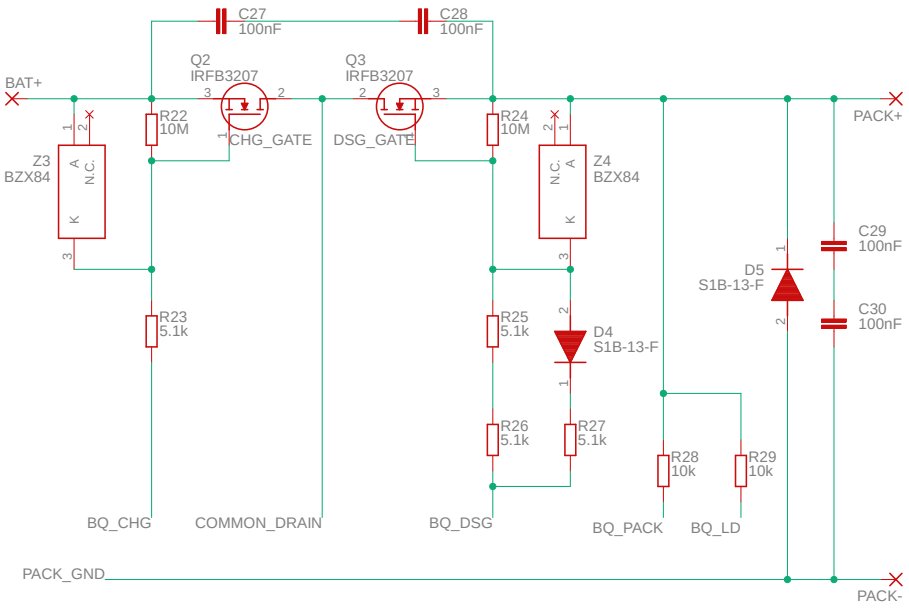

Figura 4.19: Circuito dos MOSFETs de chaveamento do BMSiv2.

Os capacitores C27 e C28, conectados entre as fontes dos MOSFETs de carga e descarga, têm a função de estabilizar a tensão. Eles atuam como filtros de ruído, reduzindo a interferência eletromagnética (EMI) e a oscilação na tensão. Isso ajuda a proteger o BMSi e os MOSFETs contra transientes de tensão que podem ocorrer durante mudanças bruscas de estados dos MOSFETs. Já os capacitores C29 e C30 atuam como capacitores de desacoplamento e filtragem, eliminando possíveis ruídos de alta frequência presentes no carregador ou no Magni.

O diodo D5 está configurado em antiparalelo, permitindo a passagem de corrente quando a tensão no PACK- é maior que a tensão no PACK+. Isso é relevante em cargas indutivas, como motores, onde o diodo em antiparalelo fornece um caminho para a corrente quando esta é interrompida abruptamente. Essa configuração ajuda a dissipar a energia armazenada no campo magnético do indutor e evita picos de tensão prejudiciais que poderiam danificar outros componentes do circuito.

Os terminais PACK são utilizados para monitorar a tensão total na saída do BMSi, onde a carga a ser alimentada é conectada. Isso permite que o BMS monitore a tensão total do pacote para gerenciamento de carga e descarga. O terminal LD, por sua vez, é usado para detectar a presença de uma carga conectada ao BMSi. Ele é utilizado principalmente para a recuperação automática em caso de curto-circuito, detectando a desconexão da carga curto-circuitada e reabilitando a utilização da bateria.

O terminal *Common-Drain* é utilizado para alimentar os circuitos de Interface Homem-Máquina (IHM) e ESP32. Ao utilizar este terminal, ao contrário do terminal BAT+, é possível alimentar o circuito pela bateria ou pelo carregador caso conectado, conduzindo pelos diodos internos dos MOSFETs. Assim, após carregar a bateria, esses circuitos continuam sendo alimentados pelo carregador, permitindo que as células da bateria atinjam totalmente o estado de relaxamento após serem carregadas. Isso é essencial para estimativas precisas do estado de carga da bateria.

Os diodos Zener, conectados entre a porta e a fonte dos MOSFETs, são necessários para limitar a tensão na porta do MOSFET a um nível seguro. Isso é importante porque a tensão porta-fonte máxima do MOSFET utilizado é limitada a um máximo de $20V$,

um valor comum em muitos MOSFETs. O valor de tensão de ruptura do diodo Zener é escolhido para ser um pouco menor que esta tensão, neste caso, escolhido $16V$. Os resistores são usados para limitar a corrente que flui para o gate do MOSFET, que é essencialmente um capacitor. Quando a tensão na porta muda, uma corrente flui para carregar ou descarregar esse capacitor. Se essa corrente for muito alta, pode danificar o MOSFET ou o circuito de controle. O valor desses resistores foi recomendado pela nota de aplicação sobre MOSFETs do BQ76942 [22], limitando a corrente para um nível seguro.

O diodo D4, no circuito de gate do MOSFET de descarga, é utilizado para fornecer uma via de descarga rápida para a carga armazenada no gate do MOSFET. Isso permite que, quando o MOSFET está sendo desligado, a carga da porta seja rapidamente drenada pela resistência de $5k\Omega$, desligando o MOSFET mais rapidamente. Por outro lado, quando o MOSFET está sendo ligado, o diodo D4 está reversamente polarizado, forçando a corrente pelos dois resistores em série de $5k\Omega$. Isso permite que a velocidade de ligar e desligar o MOSFET seja configurada independentemente, minimizando a dissipação de energia [22].

4.6.3 Circuitos do ESP32

O núcleo de todas as comunicações e algoritmos necessários para o BMSiv2 é o microcontrolador ESP32. Este é alimentado diretamente com $3.3V$, provenientes de um conversor Buck DC-DC, que eficientemente reduz a tensão da bateria para o nível adequado ao ESP32. Isso evita o uso do conversor de tensão interno do ESP32, que é menos eficiente e limitado a uma tensão máxima de $5V$ [11].

O ESP32 estabelece comunicação com três dispositivos distintos: o BQ76942, utilizado para o gerenciamento de bateria, o cartão microSD, utilizado para armazenamento de dados e o robô Magni, o dispositivo final para onde são enviados os dados do sistema. O ESP32 também é utilizado para o funcionamento da IHM do BMSiv2. A conexões realizadas do ESP32 para os circuitos do BMSiv2 pode ser vistas na Figura 4.20.

A comunicação para a transferência de dados com o BQ76942 é realizada através do protocolo I2C. O terminal BQ_ALERT para gerar uma interrupção no ESP32 quando

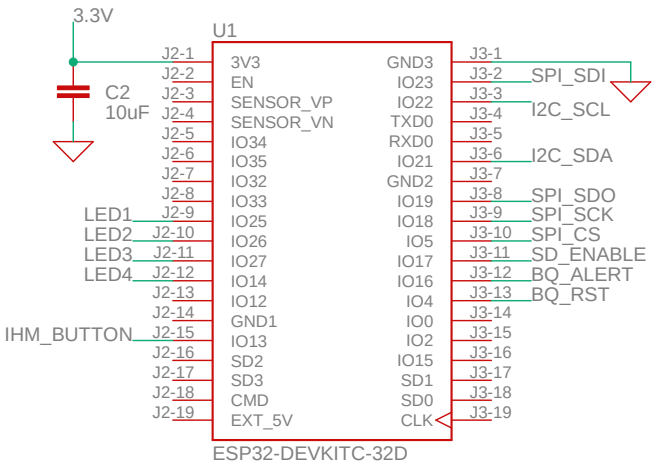

Figura 4.20: Circuito do ESP32 do BMSiv2.

uma nova leitura de dados está disponível ou quando ocorre um evento específico no BQ76942, como uma condição de sobrecarga ou sobredescarga da bateria. O terminal BQ_RST para reiniciar o BQ76942 quando necessário, permitindo que o dispositivo retorne ao seu estado inicial e reinicie suas operações, útil em situações em que é preciso reconfigurar o BQ76942 ou recuperá-lo de possíveis falhas [19].

A comunicação para a transferência de dados no cartão microSD é feita através do protocolo SPI, devido à sua simplicidade de implementação para operação de um cartão microSD. A alternativa seria o uso do protocolo Secure Digital Input Output (SDIO), um protocolo proprietário com maior complexidade de implementação e consumo de energia, limitando a acessibilidade do projeto. O terminal SD_ENABLE ativa ou desativa a alimentação do cartão microSD por meio de um MOSFET, possibilitando a economia de energia não mantendo o cartão microSD sempre ligado. O cartão microSD é utilizado para o registro de dados das medições provenientes do monitoramento da bateria pelo BQ76942, possibilitando a análise desses dados e do desempenho da bateria, bem como o registro das falhas ocorridas. Com o uso de um cartão microSD, é possível implementar futuramente algoritmos inteligentes para determinação do SOC, podendo ser armazenado no cartão microSD diversos dados sobre a bateria como SOH, capacidade inicial, tabelas de tensão por estado e carga, entre outros.

A Interface Homem-Máquina (IHM) foi projetada para funcionar com LEDs de baixo consumo e corrente, permitindo que sejam alimentados diretamente pelo ESP32. Caso contrário, devido às limitações de corrente nas saídas digitais do ESP32, seria necessário o uso de transistores para o chaveamento dos LEDs, aumentando o custo e reduzindo a eficiência do sistema.

O botão da IHM está conectado diretamente ao ESP32 e ao GND, assim quando o botão é pressionado, o sinal do terminal IHM_BUTTON é puxado ao GND, para que isto funcione corretamente, é necessário que o ESP32 utilize um resistor de *pull-up* interno para o botão [11]. Esta configuração foi escolhida devido à sua simplicidade, pois não necessita de alimentação externa.

Também foi implementado um algoritmo para detecção de aperto curto e longo do botão, possibilitando uma variedade de ações a partir do mesmo botão. No BMSi, o aperto curto do botão é utilizado para exibir nos LEDs a capacidade da bateria, e o aperto longo para ejetar o cartão microSD, garantindo a segurança dos dados contidos nele.

A interação com o robô Magni é estabelecida através de uma conexão USB, empregando a biblioteca ROSserial. Esta biblioteca, licenciada sob a BSD, facilita a integração de dispositivos personalizados em projetos que utilizam o ROS, como é o caso do BMSi com o robô Magni [32]. Dessa forma, o ESP32 é capaz de transmitir ao robô dados relativos à bateria por meio de tópicos, transformando o projeto em um Sistema de Gerenciamento de Bateria Inteligente (BMSi).

O ESP32 possui capacidades integradas de WiFi e Bluetooth. Essas funcionalidades permitem a transmissão sem fio de dados para outros dispositivos, como *smartphones* ou computadores. Sendo possível transmitir os dados coletados sobre a bateria para um dispositivo remoto. No futuro, há a possibilidade de explorar esta capacidade para implementar uma interface de *dashboard*. A interface gráfica permitiria a visualização rápida e fácil das informações da bateria, apresentando os dados em gráficos ou tabelas para facilitar a compreensão do estado atual da bateria.

4.6.4 Circuito do conversor de tensão DC-DC

A Figura 4.15 na Seção 4.5 apresenta o circuito típico do AP63203. Tendo isto como principio a Figura 4.21 ilustra o circuito empregado no BMSiv2. A principal distinção entre os dois reside no uso de um diodo Zener de $3.3V$ no BMSiv2. Apesar da segurança proporcionada pelo CI AP63203, o diodo Zener foi adicionado para lidar com o pior cenário possível, oferecendo uma margem de segurança adicional. Outra diferença é a inclusão de um diodo $D1$ no Vin do AP63203. Este diodo foi adicionado para evitar que, em caso de curto-circuito na saída do BMS, a entrada do regulador de tensão seja levada a valores negativos de tensão, o que poderia danificar o componente.

Os valores dos componentes utilizados foram sugeridos pelo datasheet do AP63203 na seção de informações de aplicação [30]. Isso assegura o funcionamento adequado do circuito e fornece uma tensão estável de $3.3V$ para o ESP32 do BMSiv2

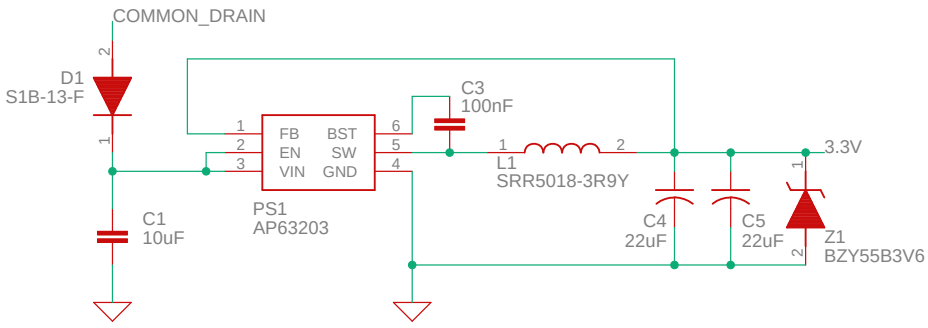

Figura 4.21: Circuito do conversor de tensão DC-DC do BMSiv2. [30].

4.7 Placa de Circuito Impresso (PCB)

A Placa de Circuito Impresso (PCB) foi projetada para satisfazer os requisitos do projeto, e acomodar todos os componentes e circuitos necessários para o funcionamento do BMSiv2. A PCB foi projetada utilizando o software Eagle da Autodesk, um software de Computer-Aided Design (CAD) para design de PCB, levando em consideração as

restrições de espaço e a necessidade de acomodar todos os componentes de forma eficiente e segura. Devido ao tamanho da bateria utilizada no BMSiv2, o desenvolvimento da PCB tem a restrição de tamanho de $130 \times 100mm$ (parte superior e superfície de maior área da bateria), assim sendo necessário o desenvolvimento de uma PCB compacta. A PCB desenvolvida foi a da Figura 4.22 com dimensões de $75 \times 100mm$.

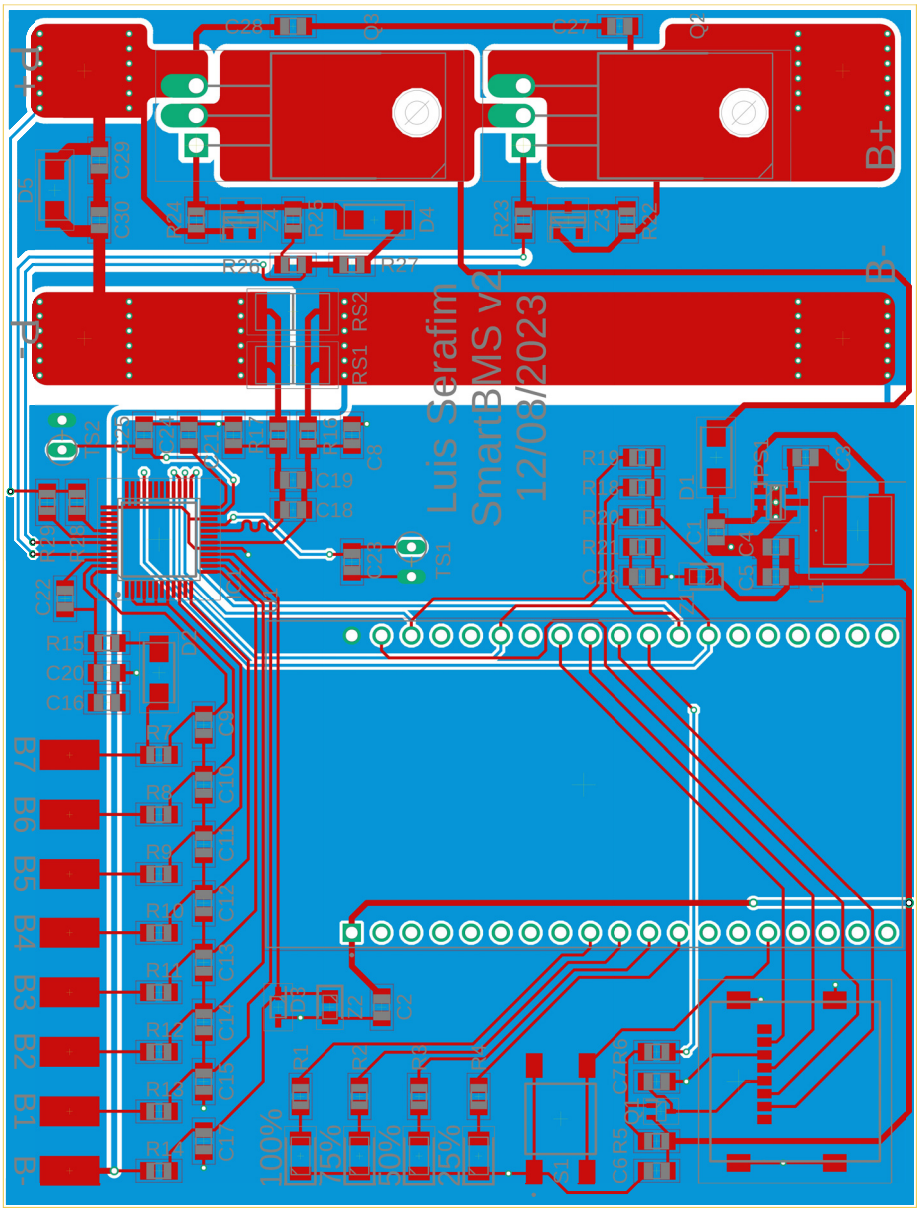

Figura 4.22: PCB do BMSiv2.

Para o desenvolvimento de uma PCB compacta, optou-se pela utilização de componentes de Montagem em Superfície (SMD). Esses componentes foram escolhidos devido às suas dimensões reduzidas, o que contribui para a miniaturização do projeto. Além disso, os componentes SMD possuem uma ampla disponibilidade de componentes modernos e são economicamente vantajosos. A montagem da placa foi facilitada pelo uso desses componentes, uma vez que todos são montados na mesma superfície, eliminando a necessidade de solda na parte inferior da PCB.

Os componentes SMD, como resistores e capacitores, foram selecionados no tamanho de SMD 0805 para este projeto. A escolha deste tamanho foi motivada pela facilidade de soldagem manual, eliminando a necessidade de equipamentos especializados. Isso facilita o processo de montagem e manutenção, tornando o projeto acessível.

Os resistores SMD utilizados no projeto são de precisão, com uma tolerância de 1% e um baixo coeficiente de temperatura de $100PPM/°C$. Isso significa que o valor da resistência desses componentes varia minimamente em função da mudança de temperaturas, garantindo uma performance consistente e confiável. Resistores de precisão são ideais para aplicações que requerem uma alta precisão na resistência, pois eles mantêm seu valor de resistência dentro de uma faixa estreita, mesmo quando expostos a variações de temperatura.

Os capacitores SMD necessários para o projeto foram escolhidos com dielétrico X7R, que apresenta uma tolerância de 10% e uma tensão máxima de $50V$. A escolha do dielétrico X7R se deve à sua alta relação de capacitância por volume e à sua tolerância de variação de capacitância em relação à temperatura relativamente baixa, o que garante uma performance estável em uma ampla faixa de temperaturas. Além disso, a tensão máxima de $50V$ oferece uma proteção robusta contra descargas eletrostáticas (ESD), melhorando a confiabilidade do sistema.

Como o BMSiv2 deve ser capaz de fornecer correntes de até $15A$, foram necessário realizar cálculos para determinar a largura das trilhas que conduziriam esta corrente. Para reduzir a largura necessária em uma placa com restrições de dimensão, foram utilizados a camada inferior e superior para conduzir esta corrente, assim reduzindo pela metade a

largura da trilha necessária em cada camada, portando os cálculos foram realizados para uma corrente de $7.5A$.

Os dimensionamento da trilha de potência da PCB foram realizados conforme a norma internacionalmente aceita IPC-2221 [33] para o dimensionamento correto das trilhas em um circuito impresso, garantindo que a corrente determinada possa ser conduzida sem causar superaquecimento ou danos à placa. Esta norma foi desenvolvida pelo Institute of Printed Circuits (IPC), uma organização global que estabelece padrões para a indústria de eletrônica.

A largura da trilha foi calculada usando a Equação 4.6 fornecida pela norma IPC-2221, calculada com base na corrente, no aumento de temperatura e na espessura do cobre utilizado. A norma IPC2221 também recomenda um aumento máximo de temperatura de $40°C$ na trilha, assim com isto em mente foi desejado um aumento máximo de $10°C$, abaixo do especificado pela normal e completamente seguro ao toque. A PCB utilizada foi a amplamente disponível de $1oz/ft^2$ de cobre com material dielétrico FR4 (resina epóxi retardante de chamas reforçada com fibra de vidro) devido ao seu baixo custo e segurança. Com isto em mente utilizando a Equação 4.6 foi calculado uma área necessária de $262.46mils^2$ para atender os requisitos.

$$
\begin{aligned}
A &= \left(\frac{I}{k\Delta T^{0.44}}\right)^{1/0.725} \\
A &= \left(\frac{7.5A}{0.048 \times 10°C^{0.44}}\right)^{1/0.725} \\
A &= 262.46mils^2
\end{aligned}
\tag{4.6}
$$

Onde:

- A: Área da seção transversal em $mils^2$;
- I: Corrente em amperes;
- ΔT: Aumento de temperatura em °C;

- k: Constante da localização da trilha (0.048 para cobre externo e 0.024 para cobre interno, ambos assumindo $1oz/ft^2$ de peso de cobre);

Assim a largura da trilha pode ser calculada utilizando a Equação 4.7 com base na espessura da camada de cobre da PCB, neste caso de $1oz/ft^2$. Resultando no valor de $190.46mils$.

$$\begin{aligned} L &= \frac{A}{1.378E} \\ L &= \frac{A}{1.378E} \\ L &= 190.46mils \end{aligned} \tag{4.7}$$

Onde:

- A: Área da seção transversal em $mils^2$;
- L: Largura da trilha em $mils$;
- E: espessura do cobre em oz/ft^2;

Utilizando a Equação 4.8, resulta em uma largura necessária de $4.84mm$ para a trilha de PCB conduzir $7.5A$ com um aumento de temperatura de $10°C$, porém para ter uma margem de segurança foi decidido utilizar um fator de segurança arbitrário de 65%, resultando em uma espessura de $8mm$. Assim foi utilizada essa espessura na camada superior e inferior para as trilhas de potência do BMSi, sendo possível conduzir $15A$ com uma boa margem de segurança.

$$\begin{aligned} Lmm &= Lmils \times 0.0254 \\ Lmm &= 190.46 \times 0.0254 \\ Lmm &= 4.84mm \end{aligned} \tag{4.8}$$

4.8 Estudo Financeiro

Para a realização deste projeto, os componentes foram adquiridos através da Mouser Electronics [34], uma distribuidora de componentes eletrônicos de nível global. A decisão de escolher a Mouser foi baseada em critérios econômicos, incluindo a disponibilidade imediata de produtos em estoque e a política de não exigir quantidades mínimas para a compra de componentes. Esses fatores foram considerados vantajosos para o projeto, pois permitiram a aquisição de componentes em quantidades adequadas para o escopo do projeto, mantendo o baixo custo.

A Tabela 4.5 apresenta os custos dos componentes adquiridos para a execução deste projeto, totalizando 30.04 €. O custo da bateria não esta incluído na Tabela devido ser escolhida de acordo com a necessidade do sistema, entretanto a bateria 7S5P 12Ah utilizada no Magni teve um custo de 200 €.

Tabela 4.5: Custos de componentes eletrônicos do BMSiv2

Componentes	Custo Unit.	Unid.	Preço	Componentes	Custo Unit.	Unid.	Preço
Resistores precisão SMD	0.03 €	29	0.74 €	ESP32-DEVKITC-32D	9.11 €	1	9.11 €
Resistor Shunt 16 mOhm	0.57 €	2	1.13 €	Monitor Bateria - BQ76942	4.44 €	1	4.44 €
Capacitores cerâmicos SMD	0.06 €	30	1.82 €	MOSFET N - IRFB3207	1.64 €	2	3.28 €
Indutor 3.9uH SMD	0.35 €	1	0.35 €	MOSFET P - DMG1013T	0.27 €	1	0.27 €
Termistor 103AT-2	0.97 €	2	1.93 €	Buck DC/DC - AP63203	0.80 €	1	0.80 €
Diodos SMD	0.22 €	9	1.94 €	Soquete microSD	1.04 €	1	1.04 €
LEDs SMD	0.33 €	4	1.30 €	Botão Táctil SMD	0.29 €	1	0.29 €
Fabricação PCB	1.59 €	1	1.59 €				
						Total	30.04 €

É relevante destacar que o maior custo do BMSiv2 é atribuído ao uso do ESP32, escolhido para facilitar a implementação do projeto. Vale ressaltar também que o valor deste projeto corresponde a uma única unidade do BMSiv2. Embora o custo já seja baixo em comparação com outros sistemas de gerenciamento de baterias inteligentes disponíveis no mercado, a produção em larga escala deste sistema poderia resultar em economias de escala [35], reduzindo ainda mais o custo por unidade e tornando o sistema mais acessível.

4.9 Biblioteca desenvolvida para BQ76942

Os algoritmos empregados neste projeto foram desenvolvidos para assegurar uma comunicação eficaz entre o BQ76942, o BMSi e o Magni, além de processar e analisar os dados da bateria. Estes algoritmos foram implementados no microcontrolador ESP32, utilizando a linguagem de programação C++ e a biblioteca desenvolvida especificamente para o CI BQ76942.

As funções desempenhadas pelos algoritmos incluem:

- Inicialização e configuração do BQ76942, que envolve a configuração e ativação das proteções.
- Leitura e processamento dos dados da bateria, que abrange as tensões das células, corrente, temperatura e estado das proteções.
- Transmissão dos dados da bateria para o Magni por meio de tópicos ROS, permitindo que o robô tome decisões informadas sobre a utilização da bateria.
- Implementação da interface homem-máquina para monitorar e analisar os dados da bateria, facilitando a manutenção e a resolução de problemas do sistema.

Com a implementação desses algoritmos, o BMSiv2 é capaz de fornecer ao Magni informações detalhadas e precisas sobre a bateria em uso, permitindo um funcionamento mais eficiente e inteligente do robô.

A biblioteca para o BQ76942, programada em C++, foi desenvolvida utilizando as instruções de programação do Technical Reference Manual (TRM) [1] e analisando o código exemplo fornecido [2], tendo como objetivo facilitar a interação com o chip BQ76942, abstraindo a complexidade do código e fornecer métodos para simplificar a sua implementação. Assim superando os desafios mencionados no Capítulo 4.5.1, permitindo a leitura e escrita de registros, execução de comandos e subcomandos, e manipulação de variáveis de medição e calibração.

A biblioteca define a classe BQ76942, que abstrai a funcionalidade do CI. Esta classe inclui métodos privados para a leitura e escrita de registros via I2C, e métodos públicos para executar comandos e subcomandos, configurar registros, ler e limpar status de alarme, controlar FETs, e realizar medições. A comunicação I2C é realizada através da biblioteca Wire.h do Arduino.

A classe BQ76942, implementada em C++, é usada para manipular o chip BQ76942, um monitor de bateria de célula múltipla. A classe contém métodos privados que são usados internamente para realizar operações de baixo nível, como leitura e escrita de registros e manipulação de bits. Esses métodos não são expostos ao usuário da biblioteca, mas são essenciais para o funcionamento dos métodos públicos, esses metodos são:

- **Checksum(*pointer, length)**: Retorna o *checksum* calculado de um *array* de bytes;
- **GetBit(data, bit)**: Retorna o valor do bit especificado;
- **I2C_WriteRegister(register_address, *register_data, data_bytes)**: Escreve dados em um registrador do BQ76942 utilizando comunicação I2C;
- **I2C_ReadRegister(register_address, *register_data, data_bytes)**: Lê dados de um registrador do BQ76942 utilizando comunicação I2C;

Os métodos públicos simplificam a configuração e o uso do BQ76942. Alguns dos métodos públicos incluem:

- **BQ76942(SDA_Pin, SCL_Pin, Alert_Pin, RST_Pin)**: Construtor que inicializa o objeto BQ76942 com os pinos SDA, SCL, Alert e RST especificados;
- **void SetRegister(register_address, register_data, data_bytes)**: Método para configurar um registradores do BQ76942;
- **void CommandOnly_Subcommand(command)**: Método para enviar um comando direto do tipo subcomando ao BQ76942;

- **void Subcommand(command_address, command_data, command_type)**: Método para executar um subcomando no BQ76942;
- **void DirectCommand(command_address, command_data, command_type)**: Método para enviar um comando direto ao BQ76942;
- **uint16_t ReadDirectCommand(directCommand)**: Método para ler um comando direto do BQ76942;
- **void Begin()**: Inicia a comunicação com BQ76942;
- **void ConfigureRegisters()**: Configura os registradores do BQ76942;
- **void ResetRegisters()**: Reinicia os registradores do BQ76942;
- **void Reset()**: Reinicia o BQ76942;
- **void DisableSleepMode()**: Desativa o modo de suspensão do BQ76942;
- **void EnableSleepMode()**: Ativa o modo de suspensão do BQ76942;
- **void ReadAlarmStatus()**: Lê o status do alarme do BQ76942;
- **void ClearAlarmStatus()**: Limpa o status do alarme do BQ76942;
- **void ReadProtectionStatus()**: Lê o status de proteção do BQ76942;
- **void ReadPermanentFailureStatus()**: Lê o status de falha permanente do BQ76942;
- **void ClearFullScanbit()**: Limpa o bit de leitura completa do BQ76942;
- **void ClearSafetyStatus()**: Limpa o status de segurança do BQ76942;
- **void ReadEventCUV()**: Lê o *snapshot* da bateria no evento CUV;
- **void ReadEventCOV()**: Lê o *snapshot* da bateria no evento COV;
- **void ReadActiveCellBalancing()**: Lê os balanceamentos ativos de células da bateria;

- **void EnableFET()**: Habilita o driver de MOSFET do BQ76942;
- **void FET_ON()**: Ativa os MOSFETs;
- **void FET_OFF()**: Desativa os MOSFETs;
- **void ReadFETStatus()**: Lê o status do MOSFET;
- **void ReadInstantCurrent()**: Lê a corrente instantânea da bateria;
- **void ReadMovingAverageCurrent()**: Lê a corrente média móvel da bateria.
- **void ReadAllVoltages()**: Lê todas as tensões do BQ76942;
- **void ReadMinMaxCellVoltage()**: Lê a tensão mínima e máxima das células do BQ76942;
- **void ReadAllTemperatureSensors()**: Lê todos os sensores de temperatura do BQ76942;
- **void ReadCalibration1()**: Lê a calibração 1 do BQ76942;
- **void ReadCalibrationCell()**: Lê a calibração das células do BQ76942;
- **void ReadAccumulatedCharge()**: Lê a carga acumulada do BQ76942;
- **void ResetAccumulatedCharge()**: Reinicia a carga acumulada do BQ76942;
- **void ReadCellTotalBalancingTime()**: Lê o tempo total de balanceamento das células do BQ76942;

Checksum

O método Checksum da classe BQ76942 no Algoritmo A.3 é utilizado para calcular a soma de verificação de um *array* de bytes. O método recebe dois parâmetros:

- **pointer**, que é um ponteiro para o *array* de bytes
- **length**, que é o comprimento do *array* de bytes.

E então o método realiza as seguintes etapas:

1. Inicializa a variável *checksum* como 0.
2. Percorre o *array* de bytes.
3. Adiciona cada byte ao valor atual de *checksum*.
4. Retorna o complemento do *checksum*, garantindo que o resultado seja de um byte.

O cálculo do *checksum* é uma técnica comum usada para verificar a integridade dos dados. Neste caso, o método percorre cada byte no *array*, adiciona seu valor ao *checksum* atual e, finalmente, retorna o complemento do *checksum*. O operador '~' é usado para calcular o complemento, e o resultado é comparado com a *bitmask* 0xFF para garantir que seja de um byte.

Esse método é útil para verificar a integridade dos dados transmitidos entre o microcontrolador e o dispositivo BQ76942. Se o *checksum* calculado não corresponder ao *checksum* recebido, isso indica que os dados podem ter sido corrompidos durante a transmissão.

WriteReg

O método I2C_WriteRegister da classe BQ76942 no Algoritmo A.1 é utilizado para escrever dados no registrador especificado do dispositivo BQ76942 através da comunicação I2C. O método recebe três parâmetros:

- **register_address** é o endereço do registrador no dispositivo BQ76942 onde os dados serão escritos.
- **register_data** é um ponteiro para os dados que serão escritos no registrador.
- **data_bytes** é o número de bytes que serão escritos no registrador.

O método realiza as seguintes etapas:

1. Inicia uma transmissão I2C para o endereço do dispositivo BQ76942 com o método de I2C **beginTransmission**(*BQ*76942*_address*).
2. Envia o endereço do registrador para o qual os dados serão escritos com o método de I2C **write**(*register_address*).
3. Escreve os dados no registrador especificado utilizando o metodo da biblioteca I2C **write**(*register_data*, *data_bytes*). Nesta etapa, *register_data* é um ponteiro para os dados a serem escritos, e *data_bytes* especifica o número de bytes a serem escritos.
4. Termina a transmissão e retorna o status da transmissão com o método de I2C **endTransmission**().

Esse método é utilizada subsequentemente por outros métodos para configurar e controlar o dispositivo BQ76942, permitindo escrever valores em seus registros para ajustar seu comportamento e configurações

ReadReg

O método I2C_ReadRegister da classe BQ76942 no Algoritmo A.2 é utilizado para ler dados do registrador especificado do dispositivo BQ76942 através da comunicação I2C. O método recebe três parâmetros:

- **register_address** é o endereço do registrador no dispositivo BQ76942 de onde os dados serão lidos.
- **register_data** é um ponteiro para o local onde os dados lidos serão armazenados.
- **data_bytes** é o número de bytes que serão lidos do registrador.

O método realiza as seguintes etapas:

1. Inicia uma transmissão I2C para o endereço do dispositivo BQ76942 com o método de I2C **beginTransmission(***BQ*76942*_address*).
2. Envia o endereço do registrador para o qual os dados serão lidos com o método de I2C **write(***register_address*).
3. Termina a transmissão mas mantém a conexão I2C ativa com o método de I2C **endTransmission(false)**. (o parâmetro 'false' indica isso).
4. Solicita a quantidade de bytes *data_bytes* do BQ76942.
5. Verifica se o número de bytes disponíveis para leitura é igual ao número de bytes a ser lido. Se não for, retorna 1 indicando erro.
6. Lê a quantidade de bytes *data_bytes* e armazena no local apontado por *register_data*.
7. Termina a transmissão e retorna o status da transmissão com o método de I2C **endTransmission()** e retorna o status da transmissão (0 se bem-sucedida).

Esse método é utilizada subsequentemente por outros métodos para configurar e controlar o dispositivo BQ76942, permitindo ler valores de seus registradores para monitorar seu comportamento e configurações.

DirectCommand

O método DirectCommand da classe BQ76942 no Algoritmo A.4 é utilizado para enviar comandos diretos ao dispositivo BQ76942. O método recebe três parâmetros:

- O **command_addres** é o endereço do comando a ser enviado.
- O **command_data** são os dados a serem enviados para o comando.
- O **command_type** é o tipo de comando, que pode ser de leitura (R) ou de escrita (W).

O método realiza as seguintes etapas:

1. Inicializa um *array* de 2 bytes TX_Data para armazenar os dados a serem transmitidos.
2. Prepara os dados a serem transmitidos, onde o byte menos significativo e o byte mais significativo do *command_data* são obtidos e armazenados no formato *little endian* em TX_Data.
3. Se o tipo de comando for de leitura (R), o método realiza a leitura do comando direto. Os dados lidos são armazenados em $RX\_DirectCommandBuffer$, uma variável global. Em seguida, o método aguarda $2ms$ para garantir que a leitura seja concluída.
4. Se o tipo de comando for de escrita (W), o método realiza a escrita do comando direto armazenado em TX_Data. Em seguida, o método atrasa a execução do programa em $2ms$ para permitir que o comando seja processado.

Este método é utilizado para enviar comandos diretos ao dispositivo BQ76942, permitindo a leitura e escrita de dados em seus registradores. Isso é útil para configurar e controlar o dispositivo, ajustando seu comportamento e configurações de acordo com as necessidades do usuário.

ReadDirectCommand

O método ReadDirectCommand da classe BQ76942 no Algoritmo A.5 é utilizado para ler comandos diretos do dispositivo BQ76942. O método recebe um parâmetro directCommand, que é o comando direto a ser lido, e retorna o resultado da leitura.

O método realiza as seguintes etapas:

1. Chama o método **DirectCommand(***directCommand***, 0x00, R)** para enviar o comando direto para leitura. O parâmetro R indica que é uma operação de leitura.
2. Calcula o resultado da leitura como a soma do segundo byte do *buffer* de comando direto ($RX\_DirectCommandBuffer[1]$) multiplicado por 256 e o primeiro byte do *buffer* ($RX\_DirectCommandBuffer[0]$).

Esse método é utilizado para ler informações do dispositivo BQ76942, como tensões, correntes e temperaturas, e pode ser usada em conjunto com outras funções da biblioteca para monitorar e controlar o dispositivo.

Subcommand

O método Subcommand da classe BQ76942 no Algoritmo A.6 é utilizado para executar subcomandos no dispositivo BQ76942. O método recebe três parâmetros:

- O **command_address** é o endereço do comando a ser executado.
- O **command_data** são os dados a serem escritos no comando.
- O **command_type** é o tipo de comando a ser executado, podendo ser de leitura (R), escrita de 1 byte (W) ou escrita de 2 bytes (W2).

O método realiza as seguintes etapas:

1. Inicializa os arrays $TX\_Checksum$ e TX_Data para armazenar o *checksum* e os dados a serem transmitidos.

2. Prepara os dados a serem transmitidos no formato *little endian*.

3. Se o tipo de comando for de leitura (R), o método escreve o endereço do comando no registrador de subcomandos e lê os dados do *buffer* de transferência.

4. Se o tipo de comando for de escrita de 1 byte (W), o método prepara os dados a serem transmitidos, escreve os dados no registrador de subcomandos, calcula o *checksum* e o escreve no registrador de *checksum* do BQ76942. Em seguida, o método atrasa a execução do programa em $1ms$ para permitir que o comando seja processado.

5. Se o tipo de comando for de escrita de 2 bytes (W2), o método prepara os dados a serem transmitidos, escreve os dados no registrador de subcomandos, calcula o *checksum* e o escreve no registrador de *checksum* do BQ76942. Em seguida, o método atrasa a execução do programa em $1ms$ para garantir que a escrita seja concluída.

O método Subcommand é utilizado para executar comandos no dispositivo BQ76942, permitindo ler e escrever valores em seus registradores para ajustar seu comportamento e configurações.

ComandOnlySubcommand

O método CommandOnly_Subcommand da classe BQ76942 no Algoritmo A.7 é utilizado para enviar subcomandos de comando ao dispositivo BQ76942. A função recebe um parâmetro *command*, que é o comando a ser enviado.

O método realiza as seguintes etapas:

1. Inicializa um *array* de 2 bytes TX_Data para armazenar os dados a serem transmitidos.

2. Converte o comando de 16 bits para o formato *little endian* e armazena no *array* TX_Data.

3. Escrever os dados no endereço do subcomando BQ76942.

4. Atrasa a execução do programa em $2ms$ para permitir que o comando seja processado.

Este método é utilizado para enviar subcomandos de comandos ao dispositivo BQ76942, permitindo executar operações específicas, como habilitar ou desabilitar FETs, ativar ou desativar o modo de dormir, entre outras funções.

SetRegister

O método SetRegister no Algoritmo A.8 é responsável por configurar os registradores do dispositivo BQ76942. Este recebe três parâmetros:

- **register_address** é o endereço do registrador a ser configurado.
- **register_data** são os dados a serem escritos no registrador.
- **data_bytes** é o número de bytes a serem escritos.

O método realiza as seguintes etapas:

1. Inicializa dois *arrays*, $TX\_Checksum$ e TX_Data, para armazenar o *checksum* a ser transmitido e os dados a serem transmitidos, respectivamente.
2. Preenche os dois primeiros bytes de TX_Data com o endereço do registrador em formato *little endian*.
3. Preenche o terceiro byte de TX_Data com o primeiro byte dos dados $register\_data$ do registrador.
4. Verifica o número de bytes de dados a serem escritos no registrador. Dependendo do valor de $data\_bytes$, o método executa diferentes ações:
    - Se $data\_bytes$ for 1, o método escreve o endereço do registrador e o byte dos dados no BQ76942, calcula o *checksum* dos dados transmitidos, define o primeiro byte do *array checksum* com o calculado, define o segundo byte do

checksum como 0x05 (soma do tamanho do endereço, dados e do *checksum*), e o escreve no BQ76942.

- Se *data_bytes* for 2, o método preenche o quarto byte de TX_Data com o segundo byte dos dados do registrador, escreve o endereço do registrador e os dois bytes dos dados no BQ76942, calcula o *checksum* dos dados transmitidos, define o primeiro byte do *checksum* com o calculado, define o segundo byte do *checksum* como 0x06 (soma do tamanho do endereço, dados e do *checksum*), e o escreve no BQ76942.
- Se *data_bytes* for 4, o método preenche o quarto, quinto e sexto bytes de TX_Data com os três últimos bytes dos dados do registrador, escreve o endereço do registrador e os quatro bytes dos dados no BQ76942, calcula o *checksum* dos dados transmitidos, define o primeiro byte do *checksum* com o calculado, define o segundo byte do *checksum* como 0x08 (soma do tamanho do endereço, dados e do *checksum*), e o escreve no BQ76942.

Em cada caso, o método aguarda 2ms após escrever os dados e o *checksum* no BQ76942. Isso é feito para garantir que o dispositivo tenha tempo suficiente para processar os dados recebidos antes de receber mais dados. Este método é utilizado para configurar os registradores do BQ76942.

Capítulo 5

Resultados e Discussões

5.1 Placa de Circuito Impresso (PCB)

Após o desenvolvimento dos circuitos e da PCB do BMSiv2, esta foi enviada para uma empresa especializada para sua confecção. O resultado é a placa mostrada na Figura 5.1. Posteriormente, os componentes foram soldados na placa, conforme pode ser observado na Figura 5.2.

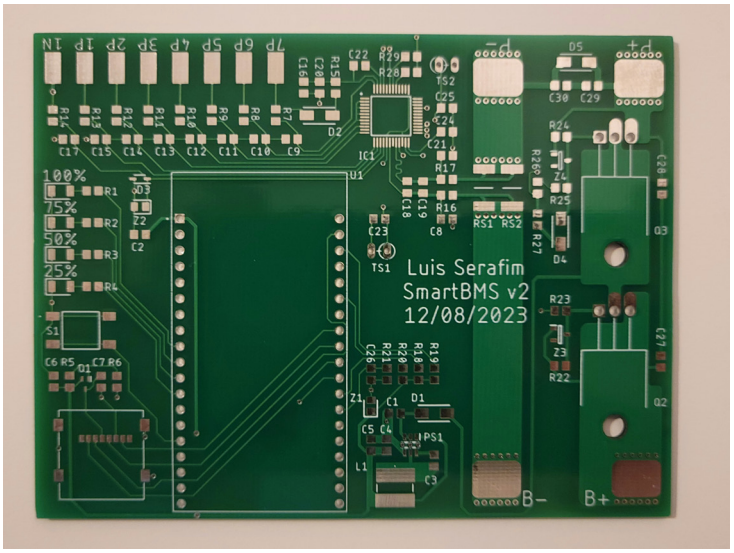

Figura 5.1: PCB do BMSiv2.

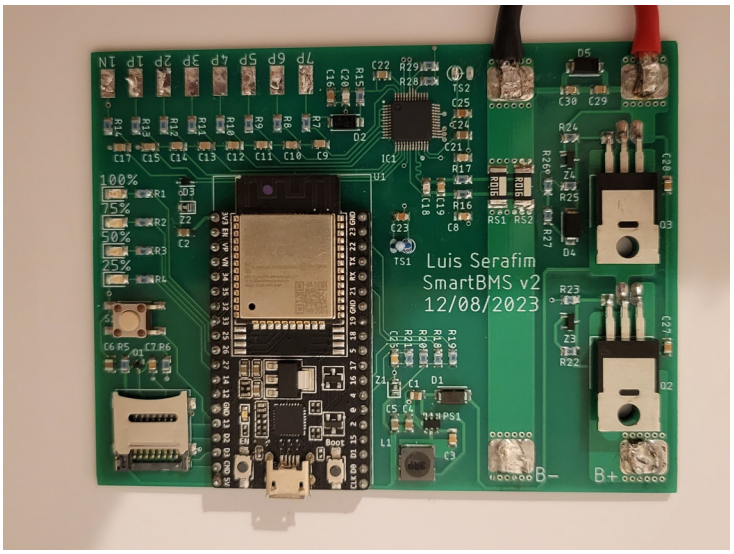

Figura 5.2: PCB do BMSiv2 com componentes soldados.

5.2 Testes Preliminares

Após a soldagem da PCB do BMSiv2, iniciaram-se os testes preliminares. Nesta fase, foram avaliadas a comunicação e a configuração do BQ76942 com o ESP32. No entanto, para garantir a segurança durante esta etapa de teste, não foi utilizada a bateria final do projeto. Em vez disso, optou-se por uma bateria contendo 7 células em série e 1 em paralelo (7S1P), como pode ser visto na Figura 5.3. Esta bateria foi escolhida por sua potência ser consideravelmente mais segura para a realização dos testes, embora mantenha a mesma tensão que a bateria 7S5P.

Nesta fase, foram testados a programação do ESP32 e a biblioteca desenvolvida para o BQ76942. As medições da bateria foram obtidas através do terminal serial, conforme mostrado na Figura 5.4(a). Está figura contém as medições disponíveis, demonstrando a eficácia da comunicação entre o ESP32 e o BQ76942.

- CFET state: Estado de ativação do MOSFETs de carga.
- DFET state: Estado de ativação do MOSFETs de descarga.
- CC: Carga do *Coulomb Counter* em mAh.

- Pck v: Tensão no terminal de carga do BMSi em mv.
- Sck v: Tensão no terminal de bateria do BMSi em mV.
- LD v: Tensão de detecção de carga do BMSi em mV.
- I: Corrente instantânea do BMSi em mA.
- Max CV: Tensão da célula com maior tensão da bateria em mV.
- Min CV: Tensão da célula com menor tensão da bateria em mV.
- VC10: Tensão do terminal de célula 10 do BMSi (conectada a célula 7) em mV.
- VC9: Tensão do terminal de célula 9 do BMSi (não utilizado) em mV.
- VC8: Tensão do terminal de célula 8 do BMSi (não utilizado) em mV.
- VC7: Tensão do terminal de célula 7 do BMSi (não utilizado) em mV.
- VC6: Tensão do terminal de célula 6 do BMSi (conectada a célula 6) em mV.
- VC5: Tensão do terminal de célula 5 do BMSi (conectada a célula 5) em mV.
- VC4: Tensão do terminal de célula 4 do BMSi (conectada a célula 4) em mV.
- VC3: Tensão do terminal de célula 3 do BMSi (conectada a célula 3) em mV.
- VC2: Tensão do terminal de célula 2 do BMSi (conectada a célula 2) em mV.
- VC1: Tensão do terminal de célula 1 do BMSi (conectada a célula 1) em mV.
- TS1: Temperatura no sensor TS1 em $°C$.
- TS2: Temperatura no sensor TS2 em $°C$.

Após a verificação do funcionamento das medições do BQ76942, realizou-se um teste com uma resistência de carga de $15W$ 68Ω para medir a corrente do BMSiv2. Na Figura 5.4(b), observou-se uma corrente de $-364mA$. O valor é negativo porque a corrente está

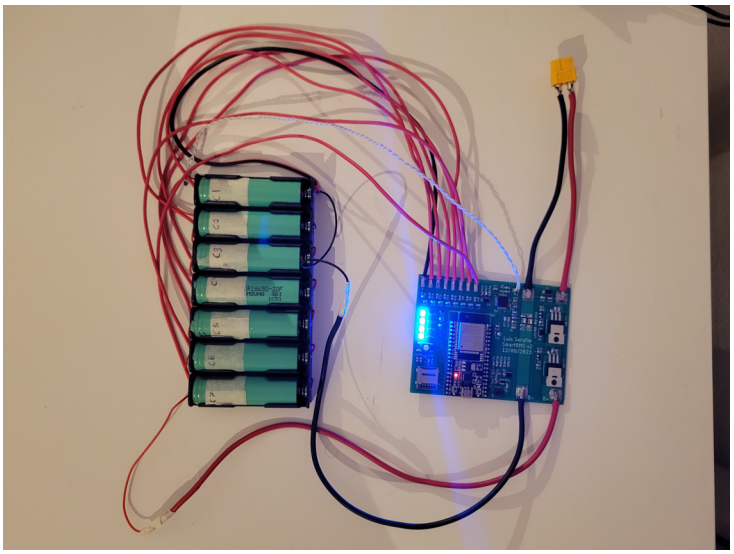

Figura 5.3: Teste prelimiar do BMSiv2 com bateria 7S1P.

saindo da bateria. Ao comparar com a medição do multímetro, observou-se uma corrente de $365mA$. Embora o resultado tenha sido satisfatório, ele pode ser aprimorado com uma futura calibração. Isso é importante, pois a precisão das medições de tensão e corrente é crucial para o monitoramento eficaz da bateria e para garantir a segurança do sistema.

5.3 Testes do BMSiv2 com a Bateria 7S5P

Após a realização dos testes preliminares que confirmaram a segurança do BMSiv2, este foi soldado à bateria 7S5P já preparada, conforme demonstrado na Figura 4.5. Os fios da bateria foram conectados ao BMSiv2, com os fios de sensoriamento positivo das células aos terminais P1 a P7 e o sensoriamento GND ao N1. Os fios de potência positivo e negativo da bateria foram conectados aos terminais B+ e B-, respectivamente. O sensor de temperatura do MOSFET foi conectado ao TS1 e o sensor de temperatura da bateria ao TS2. Após a realização das conexões da bateria, um conector DC para carregamento da bateria e um conector XT60 para o Magni foram soldados aos terminais P+ e P-. O resultado final pode ser observado na Figura 5.5.

Após a realização de medições e testes iniciais na nova bateria, uma manta termo-retrátil transparente foi aplicada, isolando a bateria e o BMSiv2. Este procedimento

```
Measurements:
CFET state: 0
DFET state: 1
CC: -61
Pck V: 24221
Stk V: 25127
LD V: 24407
I: -2
Max CV: 3607
Min CV: 3605
VC10: 3607
VC9: 0
VC8: 0
VC7: 0
VC6: 3607
VC5: 3606
VC4: 3605
VC3: 3605
VC2: 3605
VC1: 3606

TS1: 29.55
TS3: 24.15
```

(a) Teste sem carga

```
Measurements:
CFET state: 1
DFET state: 1
CC: -61
Pck V: 24734
Stk V: 25065
LD V: 24913
I: -364
Max CV: 3601
Min CV: 3598
VC10: 3600
VC9: 0
VC8: 0
VC7: 0
VC6: 3599
VC5: 3599
VC4: 3598
VC3: 3598
VC2: 3598
VC1: 3599

TS1: 29.45
TS3: 24.15
```

(b) Teste com carga de 68Ω

Figura 5.4: Terminal serial do teste do BMSiv2 com a bateria 7S1P.

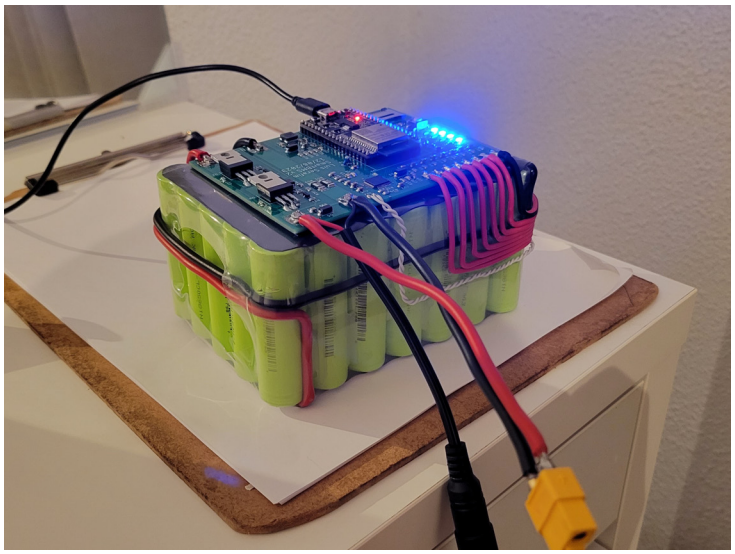

Figura 5.5: Conexão do BMSiv2 a bateria de Li-Íon 7S5P.

protege contra toques acidentais, mas foi feito um recorte na manta para permitir o acesso ao cabo USB e ao cartão microSD posteriormente. O resultado final do BMSiv2

acoplado à bateria 7S5P Li-Íon pode ser visto na Figura 5.6. Além disso, a Figura 5.7 mostra a etiqueta que foi colocada dentro da manta termo-retrátil, contendo informações sobre o BMSiv2 e sua bateria.

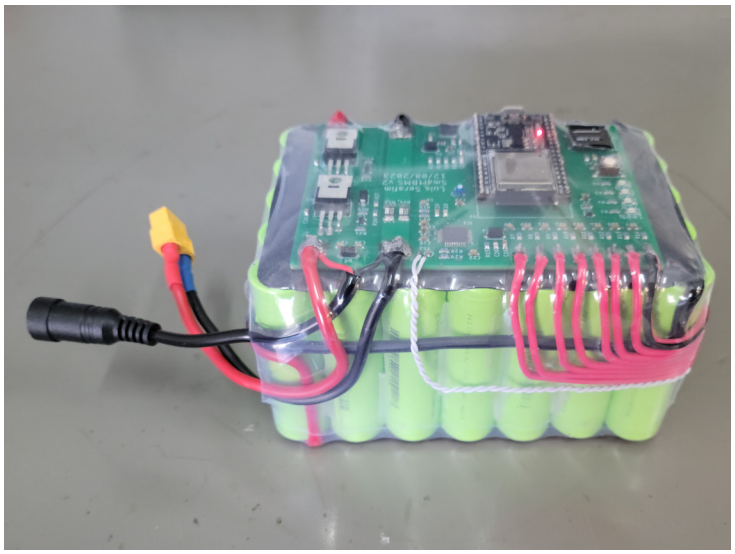

Figura 5.6: BMSiv2 finalizado.

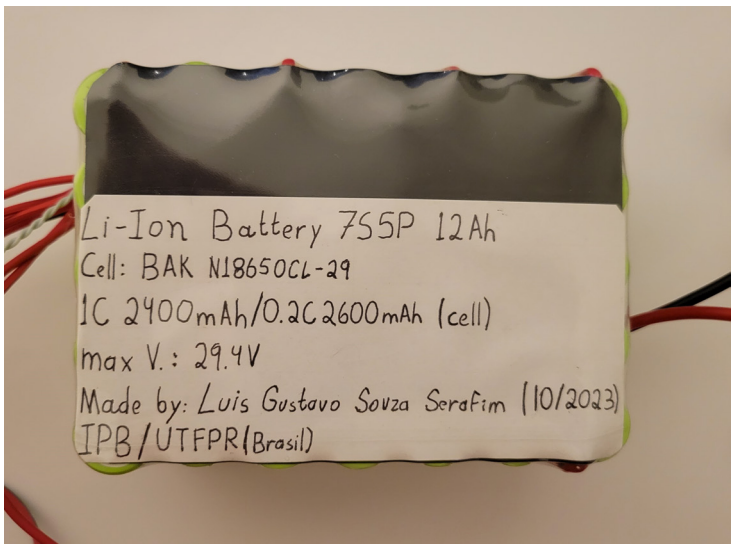

Figura 5.7: Etiqueta com informações do BMSiv2.

A bateria foi carregada utilizando um carregador de $5A$ específico para baterias de Li-Íon de 7S, conforme pode ser observado na Figura 5.8. A Figura 5.9 apresenta as medições obtidas pelo terminal serial durante os últimos instantes de carregamento, quando

a bateria já estava próxima de sua carga completa.

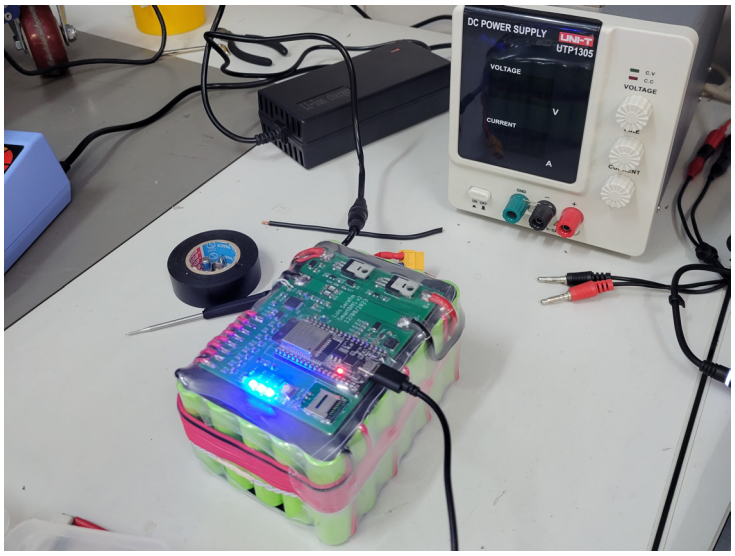

Figura 5.8: Bateria 7S5P sendo carregada.

```
Measurements:
CFET state: ON
DFET state: ON
Acc. Charge: 151 mAh
Package V: 28407 mV
Stack V: 28783 mV
LoadDetect V: 28624 mV
I: 1219 mA
Max Cell Voltage: 4132 mV
Min Cell Voltage: 4129 mV
Cell 7 Voltage: 4129 mV, Balancing OFF
Cell 6 Voltage: 4132 mV, Balancing OFF
Cell 5 Voltage: 4131 mV, Balancing OFF
Cell 4 Voltage: 4130 mV, Balancing OFF
Cell 3 Voltage: 4129 mV, Balancing OFF
Cell 2 Voltage: 4130 mV, Balancing OFF
Cell 1 Voltage: 4130 mV, Balancing OFF
Temp S1 (FETs): 27.05 °C
Temp S2 (cell): 26.35 °C
```

Figura 5.9: Dados obtidos do carregamento do BMSiv2 no terminal serial.

5.4 Teste de corrente do BMSiv2

Após a finalização da nova bateria, foi realizado um teste com uma carga resistiva de 3Ω para verificar a capacidade do BMSiv2 de suportar altas correntes por um período

prolongado. Este teste foi conduzido durante 30 minutos e resultou em uma corrente de $8.44A$, conforme pode ser observado na Figura 5.10. Os valores obtidos no terminal serial foram utilizados para calibrar a medição de corrente, aprimorando assim sua precisão.

Após os 30 minutos de teste, tanto os MOSFETs quanto a trilha de potência da PCB estavam em temperatura ambiente. Isso confirma a capacidade do BMSiv2 de conduzir a corrente desejada sem a necessidade de dissipadores de calor nos MOSFETs, comprovando o correto dimensionamento do projeto.

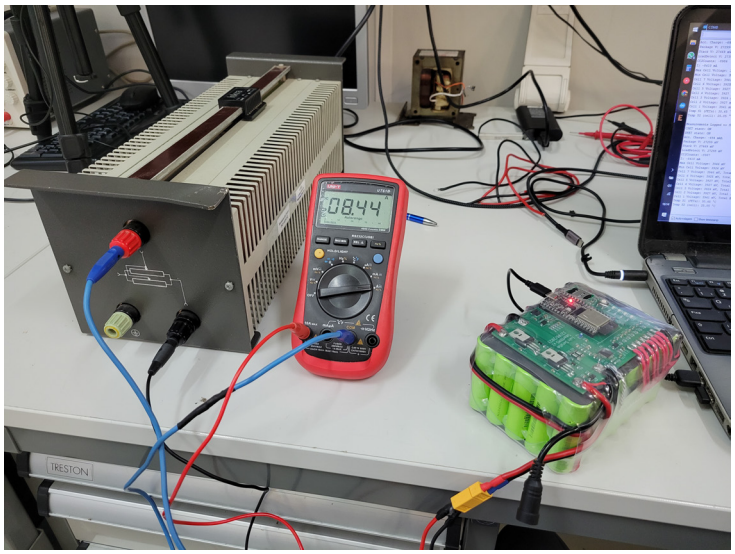

Figura 5.10: Teste de estresse do BMSiv2 (carga de 3Ω).

5.5 Teste de capacidade da bateria de Li-Íon 7S5P com o BMSiv2

Após a confirmação do funcionamento do BMSiv2, foi realizado um teste para medir a capacidade total da bateria de 7S5P. É importante notar que as baterias nem sempre possuem a capacidade que afirmam ter. Idealmente, este teste deveria ser realizado com uma corrente constante. No entanto, devido à indisponibilidade de um equipamento que pudesse fornecer uma corrente constante, o teste foi realizado utilizando uma carga resistiva de 8.3Ω, esta carga e o BMSiv2 podem ser vistos na Figura 5.11.

Figura 5.11: Teste de capacidade da bateria de Li-Íon com o BMSiv2 (carga de 8.3Ω).

Para iniciar este teste, é necessário que a bateria esteja totalmente carregada. No entanto, devido ao fato de que o carregador de bateria de Li-Íon carrega apenas até $29V$, não foi possível realizar o teste com a capacidade máxima da bateria de $29.4V$. Isso pode ser observado no Gráfico 5.12.

Os resultados do teste de descarga da bateria de Li-Íon 7S5P foram coletados através do recurso de registro de dados por cartão microSD do BMSiv2. O tempo necessário para descarregar completamente a bateria foi de 4h10min, com uma corrente média de $3.18A$, conforme ilustrado no Gráfico 5.13. A multiplicação desses valores resulta em uma capacidade de $13.26Ah$. No entanto, o valor mais preciso, obtido pelo *Coulomb Counter* e apresentado no Gráfico 5.14, é de $13.40Ah$. Isso confirma que a bateria, que supostamente teria uma capacidade de $12Ah$, na verdade possui uma capacidade maior de $13.40Ah$. Uma possível explicação para essa diferença pode devido o fabricante não ter especificado até qual tensão a capacidade anunciada foi medida.

É importante discutir a variação brusca de tensão observada no Gráfico 5.12 no início e no final do teste de descarga. Essa variação é atribuída à resistência interna das células. Quando a corrente de $3.18A$ foi conduzida, ocorreu uma queda de tensão nos terminais da bateria. Quando a condução dessa corrente foi interrompida, a tensão se normalizou após

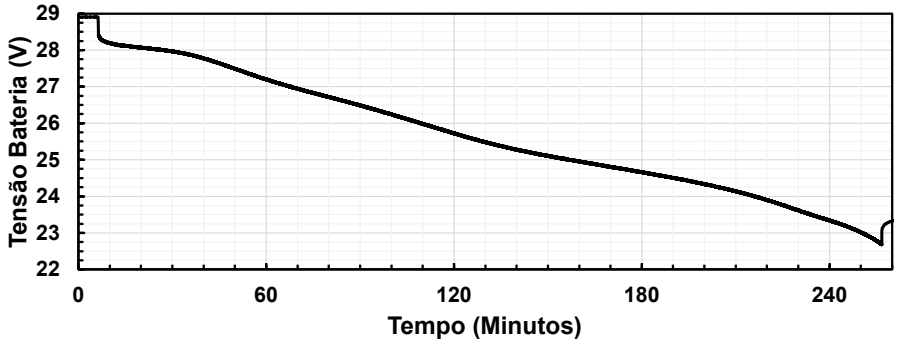

Figura 5.12: Tensão do BMSiv2, descarregando (carga de 8.3Ω).

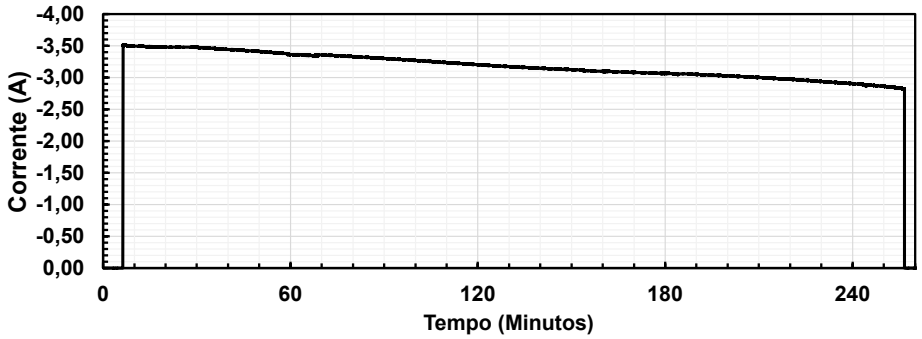

Figura 5.13: Corrente do BMSiv2, descarregando (carga resistiva de 8.3Ω).

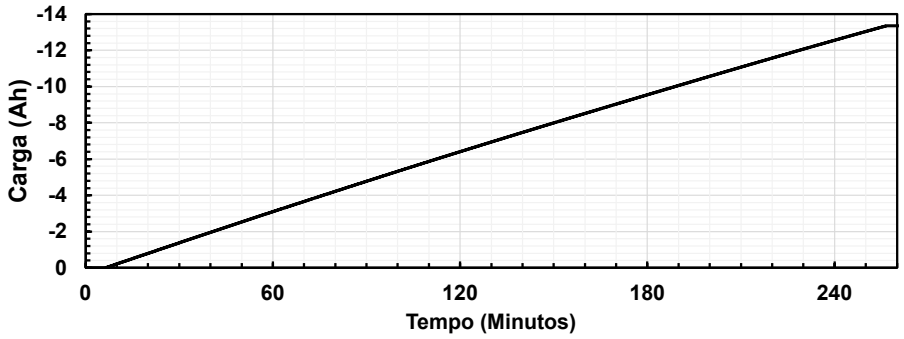

Figura 5.14: Carga do BMSiv2, descarregando (carga de 8.3Ω).

algum tempo [3]. Isso ilustra a impossibilidade de determinar o Estado de Carga (SOC) da bateria apenas pela medição da tensão durante os estados de carga ou descarga. É necessário que a bateria relaxe para que a tensão possa ser usada como uma estimativa do SOC. Essa estimativa pode ser aprimorada com o uso de algoritmos avançados de SOC.

5.6 Teste da comunicação ROS do BMSiv2

Após a conclusão dos testes de *hardware* do BMSiv2, avaliou-se a capacidade de comunicar os dados da bateria ao sistema ROS. Para simplificar este teste, considerando que o pesquisador não estava familiarizado com os sistemas do Magni, decidiu-se realizar os testes em uma máquina virtual Ubuntu com o sistema ROS Noetic instalado. Este é o mesmo sistema presente no robô Magni, garantindo que os resultados obtidos seriam equivalentes aos do robô.

Com o BMSiv2 transmitindo dados da bateria para os tópicos e após a inicialização do sistema ROS, o comando "rosrun rosserial_python serial_node.py _port:=/dev/ttyUSB0 _baud:=112500"foi inserido no terminal do Ubuntu. Este comando executa o nó serial do pacote rosserial_python no ROS, responsável por estabelecer a comunicação entre o Magni e o BMSiv2 através de uma conexão serial. No comando, "_port:=/dev/ttyUSB0"indica a porta serial a ser utilizada (neste caso, USB 0) e "_baud:=112500"estabelece a taxa de transmissão em 112500 bauds, a mesma utilizada pelo BMSiv2.

Após a conexão com o BMSiv2, inseriu-se o comando "rosnode info /serial_node"para obter informações detalhadas sobre o nó no ROS. Ao executar esse comando, foi possível visualizar as informações sobre os tópicos publicados. Assim, na Figura 5.15, é possível ver todas as informações que o BMSiv2 disponibiliza ao sistema ROS.

Para ler o valor de algum tópico do BMSiv2, utiliza-se o comando "rostopic echo **TÓPICO**", onde **TÓPICO** pode ser substituído pelo tópico de interesse. Para demonstrar esta funcionalidade, leu-se o tópico da tensão da bateria, como observado na Figura 5.16.

Isso demonstra a capacidade do BMSiV2 de se comunicar com sistemas ROS, disponibilizando a este todos os dados sobre a bateria, permitindo que sistemas tenham total conhecimento sobre as informações da bateria.

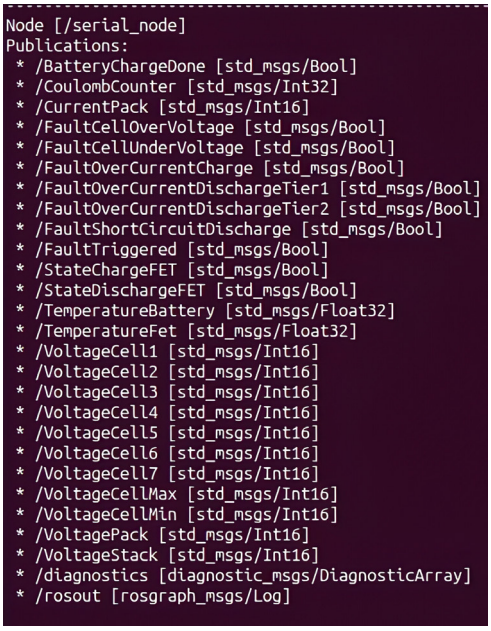

Figura 5.15: Informação no ROS sobre os tópicos do BMSiv2.

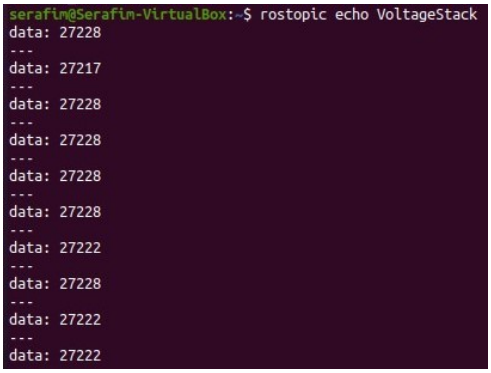

Figura 5.16: Leitura no ROS do tópico de tensão da bateria do BMSiv2.

5.7 Testes no Magni

O BMSiv2 foi instalado no dispositivo Magni para a realização de uma série de testes. A Figura 5.17 ilustra essa instalação. Os testes realizados tinham como objetivo avaliar o desempenho do Magni em condições de uso diário no laboratório, bem como determinar a duração da nova bateria instalada no dispositivo. Além disso, foi realizado um teste comparativo com a bateria de chumbo anteriormente utilizada no Magni. A finalidade desses testes era comparar a performance entre as duas baterias, a fim de avaliar qual

delas oferece melhor desempenho e eficiência energética.

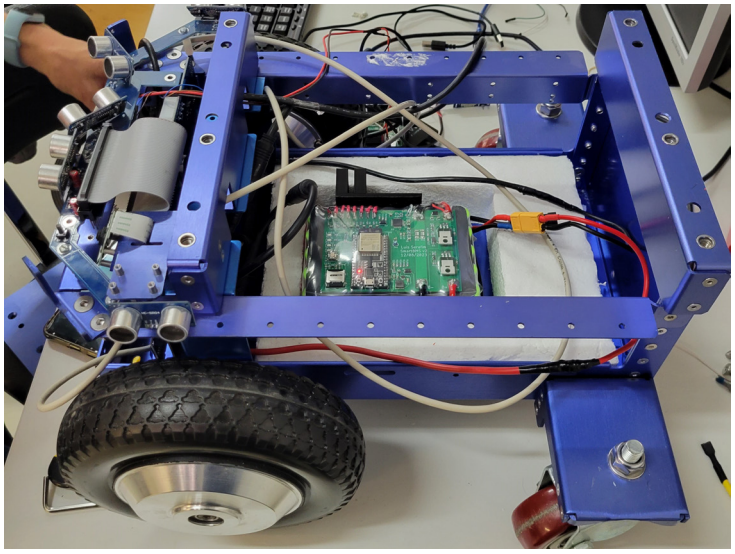

Figura 5.17: BMSiv2 instalado no Magni.

5.7.1 Uso tipico do Magni

Com o projeto concluído e o Magni operando com a nova bateria, observou-se o seu desempenho durante 8 dias no laboratório. Durante este período, o Magni foi utilizado pelos pesquisadores do laboratório em suas respectivas pesquisas. Este avanço é significativo, considerando que a bateria anterior só conseguia manter o Magni operacional por 3 horas, o que limitava consideravelmente as possibilidades de uso do robô.

A análise do Gráfico 5.19 revela que, durante o período de 8 dias, o Magni foi utilizado quatro vezes e carregado preventivamente uma vez. Este gráfico também ilustra a corrente de carga do carregador utilizado, que é de aproximadamente $5A$. Além disso, é possível notar que o uso de corrente pelo Magni se aproxima dos valores apresentados na Tabela 4.2.

A carga da bateria apresentada no Gráfico 5.20 ultrapassa $0Ah$ durante o carregamento, pois o *Coulomb Counter* foi reiniciado sem que a bateria estivesse completamente carregada. Ao finalizar o carregamento, o BMSiv2 reiniciou novamente o *Coulomb Counter* para reduzir possíveis desvios. É possível observar também que, enquanto o Magni

não está em uso, a carga se mantém constante, entretanto, no Gráfico 5.18, a tensão diminui gradualmente devido ao consumo de $300mW$ do ESP32. Essa situação evidencia uma desvantagem do uso exclusivo do *Coulomb Counter* para estimar o SOC. Portanto, seria necessário empregar algoritmos que considerem a tensão da bateria em relaxamento para aprimorar essa estimativa.

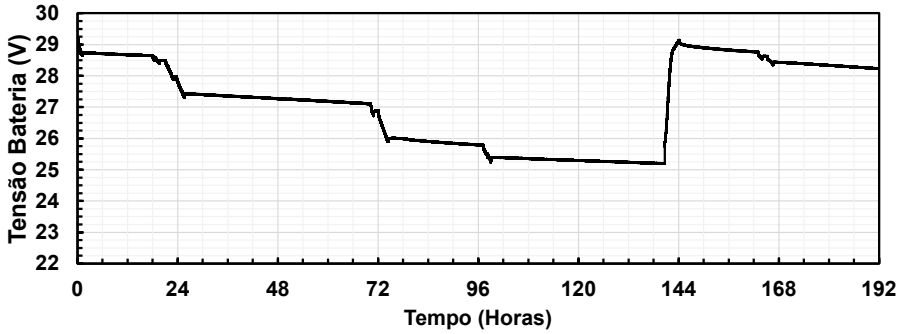

Figura 5.18: Tensão do BMSiv2, bateria de Li-Íon em uso cotidiano no Magni.

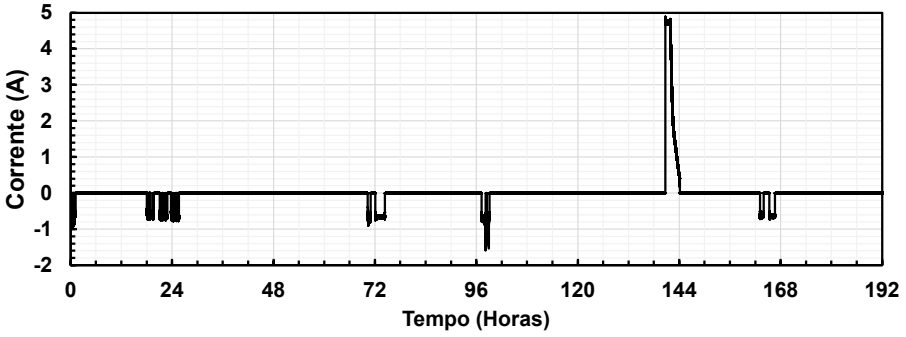

Figura 5.19: Corrente do BMSiv2, bateria de Li-Íon em uso cotidiano no Magni.

5.7.2 Teste de autonomia da bateria de Li-Íon 7S5P no Magni

Este teste foi conduzido para avaliar o tempo que a bateria de Li-Íon 7S5P poderia alimentar o Magni em uso contínuo de seus motores e sistemas. O teste começou com a bateria totalmente carregada. Durante o experimento, o Magni movia seus pneus em direções opostas, mantendo-se estático no lugar, mas girando. A corrente utilizada seria

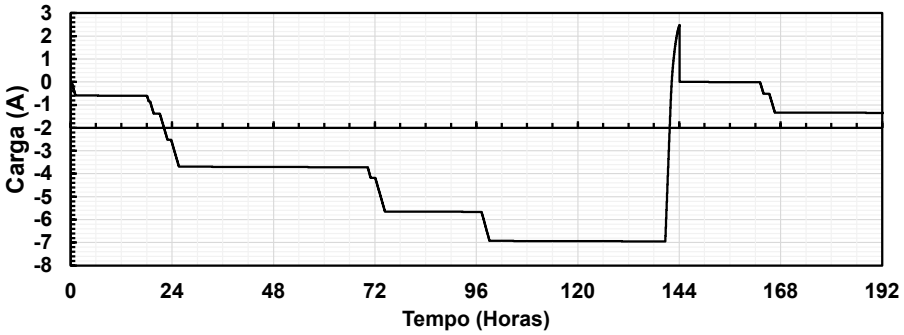

Figura 5.20: Carga do BMSiv2, bateria de Li-Íon em uso cotidiano no Magni.

equivalente àquela se o Magni estivesse se movendo para frente com carga nos motores. Este procedimento foi adotado para garantir a simplicidade e a repetibilidade do teste, com o objetivo de compará-lo com a bateria de chumbo originalmente usada no Magni. Na Figura 5.21, onde o teste já estava em andamento por 8 horas, é possível observar as marcas dos pneus do Magni no chão, resultado da constante repetição do movimento.

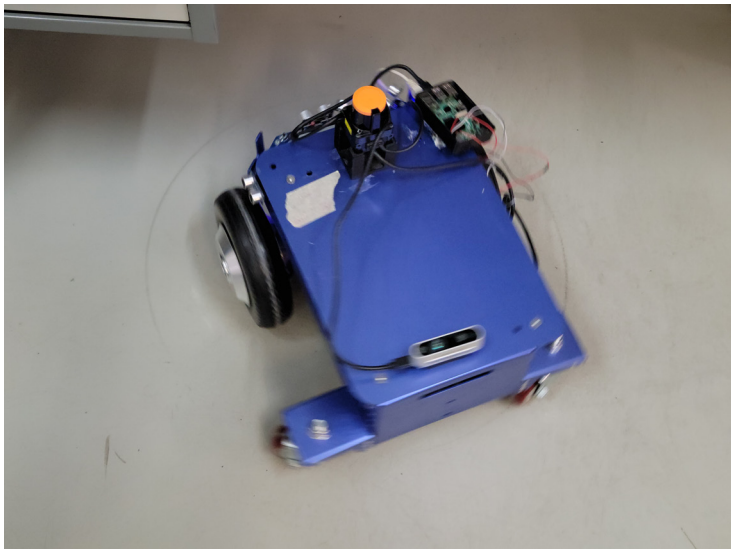

Figura 5.21: Teste de autonomia da bateria de Li-Íon do BMSiv2 no Magni.

A ativação da proteção de baixa tensão das células (CUV) sinalizou que a bateria estava totalmente descarregada, marcando assim a conclusão do teste. A corrente média utilizada pelo Magni durante o teste, conforme ilustrado no Gráfico 5.23, foi de $878mA$, apresentando valores semelhantes aos da Tabela 4.2. Além disso, foi observado que o

Magni alcançou uma autonomia de 14h37min, um resultado significativamente superior ao esperado.

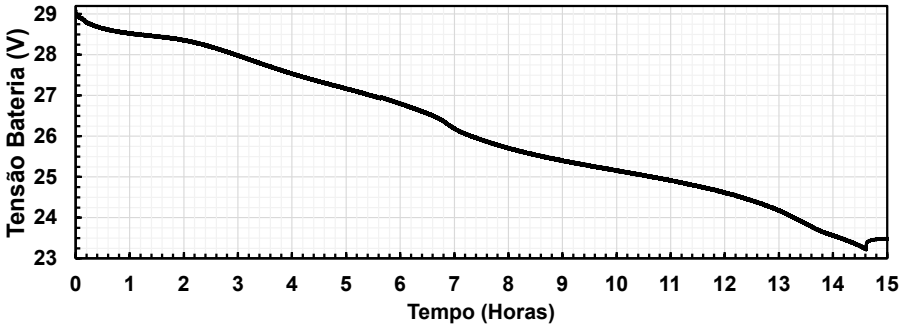

Figura 5.22: Tensão do BMSiv2, teste autonomia do Magni.

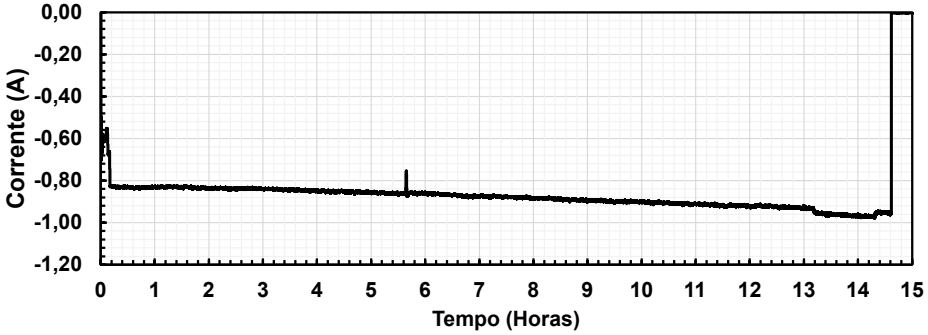

Figura 5.23: Corrente do BMSiv2, teste autonomia do Magni.

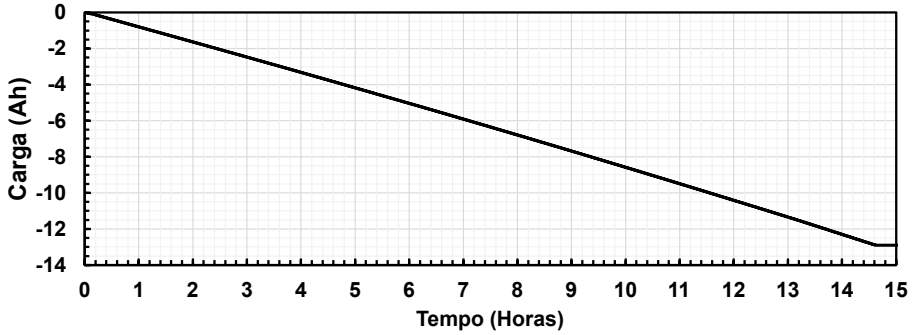

Figura 5.24: Carga do BMSiv2, teste autonomia do Magni.

5.7.3 Teste de descarga da bateria de chumbo no Magni

Para comparar o desempenho da bateria de Li-Íon do BMSiv2, foi necessário realizar o mesmo teste com a bateria de chumbo. Assim, a bateria de chumbo foi reinstalada no Magni. No entanto, devido à ausência de um BMS inteligente nesta bateria, foi possível obter apenas as medições de tensão realizadas pelo próprio Magni, que também ficou responsável pelo registro dos dados.

Os resultados deste teste são apresentados no Gráfico 5.25. A bateria começou com carga total de $25V$ e foi descarregada até $22V$, que é a tensão mínima para baterias de chumbo. A autonomia observada com esta bateria foi de 1h08min, conforme estimado anteriormente pelos pesquisadores do laboratório. No entanto, assumindo que a corrente de descarga foi semelhante à do teste da bateria de Li-Íon, aproximadamente $900mA$, a autonomia do robô com esta bateria foi extremamente baixa. Mesmo para uma bateria de chumbo, a autonomia deveria ter sido próxima àquela apresentada no manual do Magni para baterias [24], conforme pode ser visto na Tabela 4.1. Portanto, é possível inferir que esta bateria não possui a capacidade que afirma ter.

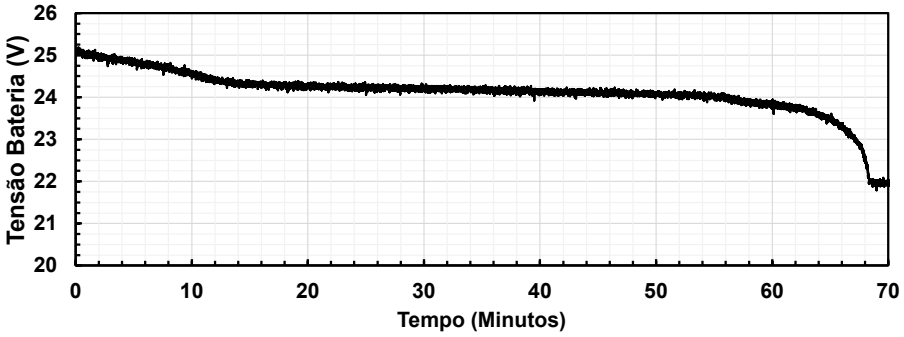

Figura 5.25: Teste de descarga da bateria de chumbo no Magni.

Com base nos testes realizados, é possível concluir que a autonomia do Magni melhorou significativamente com o uso da nova bateria de Li-Íon. Isso possibilita a realização de tarefas mais longas.

Capítulo 6

Conclusão e Trabalho Futuro

Este trabalho, que teve como objetivo desenvolver um Sistema de Gerenciamento de Bateria Inteligente (BMSi) para superar as limitações dos BMSs disponíveis, alcançou todos os objetivos propostos. O BMSi permite uma operação inteligente de dispositivos, como o robô Magni, fornecendo informações completas sobre a bateria, mantendo-se acessível e de fácil implementação. Foi adquirida uma bateria de Li-Íon que atende às necessidades de autonomia do robô Magni, dimensionaram-se corretamente os circuitos necessários, e desenvolveu-se o *hardware* do BMSi com uma interface IHM, capacidade de *datalogging* e comunicação com sistemas ROS. Além disso, foi criada uma biblioteca intuitiva para o CI BQ796942 e realizado a integração o BMSi com o sistema ROS do Magni, garantindo uma autonomia de 15h37min com uso contínuo de todos os sistemas e motores, esta autonomia pode ser ainda maior em usos cotidianos do robô.

Durante o desenvolvimento do trabalho, foram conduzidos diversos testes para validar o funcionamento, precisão e segurança do BMSi. Entre esses testes, estimou-se e comparou-se o desempenho deste sistema com o anterior, além de testes para demonstrar a alta configurabilidade e capacidade de transmissão de dados sobre a bateria que o BMSi possui.

Uma contribuição significativa deste trabalho foi o desenvolvimento de uma biblioteca em C++ para o CI monitor de baterias BQ76942, facilitando seu uso devido à alta abstração que está biblioteca fornece aos usuários. Isso garante a acessibilidade deste

CI para futuros desenvolvedores.

Os resultados obtidos nesta pesquisa superaram as expectativas iniciais sobre o projeto. Além disso, disponibilizou-se ao laboratório CeDRI [7] uma bateria de Li-Íon com o Sistema de Gerenciamento de Bateria Inteligente desenvolvido, abrindo portas para futuras pesquisas e projetos que serão desenvolvidos com o robô Magni, aproveitando as capacidades que este sistema fornece.

6.1 Trabalho Futuro

Para futuras investigações, propõe-se o desenvolvimento de algoritmos inteligentes para o cálculo do Estado de Carga (SOC) a serem implementados no Sistema de Gerenciamento de Bateria Inteligente (BMSi). Isso aprimorará a precisão da estimativa de carga. Adicionalmente, é crucial considerar melhorias e otimizações nas funcionalidades deste projeto. Entre elas, destaca-se o desenvolvimento da versão 3 do BMSi, que suportará de 3 a 10 células, permitindo sua aplicação em sistemas de 12V a 42V, aumentando assim a flexibilidade de uso.

Outra proposta é a criação de uma interface gráfica, aproveitando as capacidades de comunicação sem fio do ESP32, utilizado no BMSi. Essa implementação poderia ser complementada com armazenamento em nuvem para os dados do BMSi, expandindo as possibilidades de aplicação deste dispositivo.

Com estas propostas, espera-se contribuir para o avanço das tecnologias de BMSi, tornando seu uso mais acessível e confiável, especialmente no contexto de aplicações robóticas.

Bibliografia

[1] Texas Instruments, *BQ76942 Technical Reference Manual (Rev. B)*, Acesso em: 18 de outubro 2023, Texas Instruments, 2022. URL: https://www.ti.com/lit/pdf/sluuby1.

[2] *BQ76942 product details*, Acesso em: 18 de outubro 2023. URL: https://www.ti.com/product/BQ76942.

[3] A. A. Chellal, J. Gonçalves, J. Lima, V. Pinto e H. Megnafi, «Design of an embedded energy management system for Li–Po batteries based on a DCC-EKF approach for use in mobile robots,» *Machines*, vol. 9, n.º 12, p. 313, 2021.

[4] A. A. Chellal, «Energy management of Li-Po batteries in the mobile robotics domain,» tese de doutoramento, Instituto Politecnico de Braganca (Portugal), 2021.

[5] *Ubiquity Robotics*, Acesso em: 15 de outubro 2023. URL: https://www.ubiquityrobotics.com/.

[6] Ubiquity Robotics, *Magni Robot Store*, Acesso em: 15 de outubro 2023. URL: https://ubiquityrobotics.com/products-magni.

[7] *CeDRI*, Acesso em: 18 de outubro 2023. URL: https://cedri.ipb.pt/.

[8] *ROS Wiki*, Acesso em: 18 de outubro 2023. URL: https://www.ros.org/.

[9] *esp32 devkitc*, Acesso em: 18 de outubro 2023. URL: https://www.espressif.com/en/products/devkits/esp32-devkitc.

[10] *Espressif*, Acesso em: 18 de outubro 2023. URL: https://www.espressif.com/.

[11] *ESP323 datasheet*, Acesso em: 18 de outubro 2023. URL: https://www.espressif.com/sites/default/files/documentation/esp32-s3-wroom-2_datasheet_en.pdf.

[12] *Espressif IoT Development Framework (ESP-IDF)*, Acesso em: 18 de outubro 2023. URL: https://www.espressif.com/en/products/sdks/esp-idf.

[13] *Texas Instruments*, Acesso em: 18 de outubro 2023. URL: https://www.ti.com/.

[14] *BQ76942 mouser store*, Acesso em: 18 de outubro 2023. URL: https://pt.mouser.com/ProductDetail/Texas-Instruments/BQ7694201PFBR?qs=iLbezkQI%252BsjbsTRVrFvjUA%3D%3D.

[15] J. Mankar, C. Darode, K. Trivedi, M. Kanoje e P. Shahare, «Review of I2C protocol,» *International Journal of Research in Advent Technology*, vol. 2, n.º 1, 2014.

[16] S. Choudhury, G. Singh e R. Mehra, «Design and verification serial peripheral interface (SPI) protocol for low power applications,» *International Journal of Innovative Research in Science, Engineering and Tecgnology*, pp. 16 750–16 758, 2014.

[17] M. Spang e N. Hofstoetter, «Evaluation of current measurement accuracy for a power module with integrated shunt resistors,» em *PCIM Europe 2017; International Exhibition and Conference for Power Electronics, Intelligent Motion, Renewable Energy and Energy Management*, VDE, 2017, pp. 1–8.

[18] K. Movassagh, A. Raihan, B. Balasingam e K. Pattipati, «A critical look at coulomb counting approach for state of charge estimation in batteries,» *Energies*, vol. 14, n.º 14, p. 4074, 2021.

[19] Texas Instruments, *BQ76942 3-Series to 10-Series High Accuracy Battery Monitor and Protector for Li-Ion, Li- Polymer, and LiFePO4 Battery Packs datasheet (Rev. B)*, Acesso em: 18 de outubro 2023, Texas Instruments, 2021. URL: https://www.ti.com/lit/gpn/bq76942.

[20] Texas Instruments, *BQ769x2 Software Development Guide (Rev. B)*, Acesso em: 18 de outubro 2023, Texas Instruments, 2021. URL: https://www.ti.com/lit/pdf/sluaa11.

[21] Texas Instruments, *BQ769x2 Calibration and OTP Programming Guide (Rev. A)*, Acesso em: 18 de outubro 2023, Texas Instruments, 2021. URL: https://www.ti.com/lit/pdf/sluaa32.

[22] Texas Instruments, *Multiple FETs with the BQ76952, BQ76942 Battery Monitors (Rev. A)*, Acesso em: 18 de outubro 2023, Texas Instruments, 2021. URL: https://www.ti.com/lit/pdf/sluaa09.

[23] Texas Instruments, *Cell Balancing With BQ769x2 Battery Monitors (Rev. A)*, Acesso em: 18 de outubro 2023, Texas Instruments, 2022. URL: https://www.ti.com/lit/pdf/sluaa81.

[24] Ubiquity Robotics, *Magni Batteries Overview*, Acesso em: 15 de outubro 2023. URL: https://learn.ubiquityrobotics.com/noetic_overview_batteries.

[25] Ubiquity Robotics, *Magni Requeriments*, Acesso em: 15 de outubro 2023. URL: https://ubiquityrobotics.github.io/learn_legacy_archive/need_to_know#batteries.

[26] *BMS SHL J-7S-20A para bateria de lítio loja*, Acesso em: 18 de outubro 2023. URL: https://pt.aliexpress.com/i/32947712152.html?gatewayAdapt=glo2bra.

[27] *INA260 Precision Digital Current and Power Monitor With Low-Drift, Precision Integrated Shunt Datasheet (Rev. C)*, Acesso em: 18 de outubro 2023. URL: https://www.ti.com/lit/gpn/ina260.

[28] *LTC6804-1/LTC6804-2: Multicell Battery Monitors Datasheet*, Acesso em: 18 de outubro 2023. URL: https://www.analog.com/media/en/technical-documentation/data-sheets/680412fc.pdf.

[29] *IRFB3207 75V Single N-Channel Power MOSFET Datasheet*, Acesso em: 18 de outubro 2023. URL: https://www.infineon.com/dgdl/Infineon-IRFS3207-DataSheet-v01_01-EN.pdf?fileId=5546d462533600a401535636708e215e.

[30] *Ap63203 3.8V-32V, 2A Low Iq Synchronous Buck with Enhanced EMI Reduction Datasheet*, Acesso em: 18 de outubro 2023. URL: https://www.diodes.com/assets/Datasheets/AP63200-AP63201-AP63203-AP63205.pdf.

[31] *Termistor NTC 103AT-2 10kohm 1% Datasheet*, Acesso em: 18 de outubro 2023. URL: https://pt.mouser.com/datasheet/2/362/semitec_atthermistor-1202913.pdf.

[32] *ROSserial Wiki*, Acesso em: 18 de outubro 2023. URL: https://wiki.ros.org/rosserial_arduino.

[33] IPC, *IPC-2221A - Generic Standard on Printed Board Design*, Institute of Printed Circuits, 1998.

[34] *Mouser Electronics Store*, Acesso em: 18 de outubro 2023. URL: https://pt.mouser.com/.

[35] G. J. Stigler, «The economies of scale,» *The Journal of Law and Economics*, vol. 1, pp. 54–71, 1958.

Apêndice A

Biblioteca BQ76942

Algoritmo A.1: Método WriteRegister da classe BQ76942 em $C++$

```
1  // Função para configurar um registro do BQ76942
2  // Parâmetros:
3  // - register_address: o endereço do registro a ser configurado
4  // - register_data: os dados a serem escritos no registro
5  // - data_bytes: o número de bytes a serem escritos
6  uint8_t BQ76942::I2C_WriteRegister(uint8_t register_address, uint8_t *
       register_data, uint8_t data_bytes)
7  {
8    // Inicia a transmissão I2C para o endereço do dispositivo BQ76942
9    I2C.beginTransmission(BQ76942_address);
10
11   // Envia o endereço do registro para o qual os dados serão escritos
12   I2C.write(register_address);
13
14   // Escreve os dados no registro especificado
15   // register_data é um ponteiro para os dados a serem escritos
16   // data_bytes especifica o número de bytes a serem escritos
17   I2C.write(register_data, data_bytes);
18
19   // Termina a transmissão e retorna o status da transmissão (0 se bem-
       sucedida)
```

```
20    return I2C.endTransmission();
21  }
```

Algoritmo A.2: Método ReadRegister da classe BQ76942 em $C++$

```
1  // Função para ler um registro do BQ76942 via I2C
2  // Parâmetros:
3  // - register_address: o endereço do registro a ser lido
4  // - register_data: ponteiro para o local onde os dados lidos serão
       armazenados
5  // - data_bytes: o número de bytes a serem lidos
6  // Retorna: 0 se a leitura foi bem-sucedida, 1 se o número de bytes
       disponíveis for diferente do esperado
7
8  uint8_t BQ76942::I2C_ReadRegister(uint8_t register_address, uint8_t *
       register_data, uint8_t data_bytes)
9  {
10   // Inicia a transmissão I2C para o endereço do dispositivo BQ76942
11   I2C.beginTransmission(BQ76942_address);
12
13   // Envia o endereço do registro para o qual os dados serão lidos
14   I2C.write(register_address);
15
16   // Termina a transmissão, mas mantém a conexão I2C ativa (o parâmetro
       'false' indica isso)
17   I2C.endTransmission(false);
18
19   // Solicita 'data_bytes' bytes do BQ76942
20   I2C.requestFrom(BQ76942_address, data_bytes);
21
22   // Verifica se o número de bytes disponíveis para leitura é igual ao n
       úmero de bytes a ser lido
23   if (I2C.available() != data_bytes)
24     return 1; // Se não for, retorna 1 indicando erro
25
```

```
26    // Lê 'data_bytes' bytes e armazena no local apontado por '
       register_data'
27    for (uint8_t i = 0; i < data_bytes; i++)
28    {
29      register_data[i] = I2C.read();
30    }
31
32    // Termina a transmissão e retorna o status da transmissão (0 se bem-
       sucedida)
33    return I2C.endTransmission();
34  }
```

Algoritmo A.3: Método CheckSum da classe BQ76942 em $C++$

```
1   // Função para calcular o checksum de um array de bytes
2   // Parâmetros
3   // - pointer um ponteiro para o array de bytes
4   // - length o comprimento do array de bytes
5   // Retorna o checksum calculado como um byte de 8 bits
6
7   uint8_t BQ76942::Checksum(uint8_t *pointer, uint8_t length)
8   {
9     // Inicializa o checksum como 0
10    uint8_t checksum = 0;
11
12    // Percorre o array de bytes
13    for (uint8_t i = 0; i < length; i++)
14      // Adiciona cada byte ao valor atual de checksum
15      checksum += pointer[i];
16
17    // Retorna o complemento de checksum, garantindo que o resultado seja
       um byte de 8 bits
18    return (0xff & ~checksum);
19  }
```

Algoritmo A.4: Método DirectCommand da classe BQ76942 em $C++$

```
1  // Função para enviar um comando direto ao BQ76942
2  // Parâmetros:
3  // - command_address: o endereço do comando a ser enviado
4  // - command_data: os dados a serem enviados para o comando
5  // - command_type: o tipo de comando, que pode ser de leitura (R) ou de
       escrita (W)
6
7  void BQ76942::DirectCommand(uint8_t command_address, uint16_t
       command_data, uint8_t command_type)
8  {
9    // Consulte o Manual de Referência Técnica para obter uma lista
       completa de Comandos Diretos
10   // command_type: R = leitura, W = escrita
11
12   // Inicializa um array de 2 bytes para armazenar os dados a serem
       transmitidos
13   uint8_t TX_Data[2] = {0x00, 0x00};
14
15   // Preparando os dados a serem transmitidos (TX_Data) no formato
       little endian
16   TX_Data[0] = command_data & 0xff;         // Obtém o byte menos
       significativo do command_data
17   TX_Data[1] = (command_data >> 8) & 0xff; // Obtém o byte mais
       significativo do command_data
18
19   // Se o tipo de comando for de leitura (R)
20   if (command_type == R)
21   {
22     // Realiza a leitura do comando direto
23     // RX_DirectCommandBuffer é uma variável global que armazenará os
       dados lidos
24     (void)I2C_ReadRegister(command_address, RX_DirectCommandBuffer, 2);
```

```
25      delayMicroseconds(2000); // Aguarda 2ms para garantir que a leitura
        seja concluída
26    }
27    // Se o tipo de comando for de escrita (W)
28    else if (command_type == W)
29    {
30      // Realiza a escrita do comando direto
31      // Os dados a serem escritos estão em TX_Data
32      (void)I2C_WriteRegister(command_address, TX_Data, 2);
33
34      // Atrasa a execução do programa em 2000 microssegundos para
        permitir que o comando seja processado
35      delayMicroseconds(2000);
36    }
37  }
```

Algoritmo A.5: Método ReadDirectCommand da classe BQ76942 em $C++$

```
1  // Função para ler um comando direto do BQ76942
2  // Parâmetros:
3  // - directCommand: o comando direto a ser lido
4  // Retorna: o resultado da leitura.
5
6  uint16_t BQ76942::ReadDirectCommand(uint8_t directCommand)
7  {
8    // Envia o comando direto para leitura
9    DirectCommand(directCommand, 0x00, R);
10
11   // Retorna resultado calculado como a soma
12   // do segundo byte do buffer de comando direto (RX_DirectCommandBuffer
       [1])
13   // multiplicado por 256 e o primeiro byte do buffer (
       RX_DirectCommandBuffer[0]).
14   return (RX_DirectCommandBuffer[1] * 256 + RX_DirectCommandBuffer[0]);
15 }
```

Algoritmo A.6: Método Subcommand da classe BQ76942 em $C++$

```
1  // Função para executar um subcomando no BQ76942
2  // Parâmetros:
3  // - command_address: o endereço do comando a ser executado
4  // - command_data: os dados a serem escritos no comando
5  // - command_type: o tipo de comando a ser executado (R = leitura, W =
       escrita de 1 byte, W2 = escrita de 2 bytes)
6  void BQ76942::Subcommand(uint16_t command_address, uint16_t command_data
       , uint8_t command_type)
7  {
8    // Inicializa os arrays para armazenar o checksum e os dados a serem
       transmitidos
9    uint8_t TX_Checksum[2] = {0x00, 0x00};
10   uint8_t TX_Data[4] = {0x00, 0x00, 0x00, 0x00};
11
12   // Prepara os dados a serem transmitidos no formato little endian
13   TX_Data[0] = command_address & 0xff;
14   TX_Data[1] = (command_address >> 8) & 0xff;
15
16   // Se o tipo de comando for de leitura (R), realiza a leitura dos
       dados
17   if (command_type == R)
18   {
19     // Escreve o endereço do comando no registro de subcomandos e lê os
       dados do buffer de transferência
20     (void)I2C_WriteRegister(BQ76942_subcommand_address, TX_Data, 2);
21     delayMicroseconds(2000);
22     (void)I2C_ReadRegister(BQ76942_transferBuffer_address,
       RX_32BytesBuffer, 32);
23   }
24   // Se o tipo de comando for de escrita de 1 byte (W), realiza a
       escrita dos dados
25   else if (command_type == W)
26   {
```

```
27      // Prepara os dados a serem transmitidos no formato little endian
28      TX_Data[2] = command_data & 0xff;
29      // Escreve os dados no registro de subcomandos e calcula o checksum
30      (void)I2C_WriteRegister(BQ76942_subcommand_address, TX_Data, 3);
31      delayMicroseconds(1000);
32      TX_Checksum[0] = Checksum(TX_Data, 3);
33      TX_Checksum[1] = 0x05; // comprimento combinado do endereço dos
      registros, dados, checksum e comprimento
34      // Escreve o checksum no registro de checksum
35      (void)I2C_WriteRegister(BQ76942_checksum_address, TX_Checksum, 2);
36      // Atrasa a execução do programa em 1000 microssegundos para
      permitir que o comando seja processado
37      delayMicroseconds(1000);
38    }
39    // Se o tipo de comando for de escrita de 2 bytes (W2), realiza a
       escrita dos dados
40    else if (command_type == W2)
41    {
42      // Prepara os dados a serem transmitidos no formato little endian
43      TX_Data[2] = command_data & 0xff;
44      TX_Data[3] = (command_data >> 8) & 0xff;
45      // Escreve os dados no registro de subcomandos e calcula o checksum
46      (void)I2C_WriteRegister(BQ76942_subcommand_address, TX_Data, 4);
47      delayMicroseconds(1000);
48      TX_Checksum[0] = Checksum(TX_Data, 4);
49      TX_Checksum[1] = 0x06; // comprimento combinado do endereço dos
      registros e dos dados do comando
50      // Escreve o checksum no registro de checksum
51      (void)I2C_WriteRegister(BQ76942_checksum_address, TX_Checksum, 2);
52      // Aguarda 1ms para garantir que a escrita seja concluída
53      delayMicroseconds(1000);
54    }
55  }
```

Algoritmo A.7: Método CommandOnlySubcomand da classe BQ76942 em $C++$

```
1  // Função para enviar um subcomandos de comando ao BQ76942
2  // Parâmetros:
3  // - command: o comando a ser enviado
4
5  void BQ76942::CommandOnly_Subcommand(uint16_t command)
6  {
7    // Inicializa um array de 2 bytes para armazenar os dados a serem
       transmitidos
8    uint8_t TX_Data[2] = {0x00, 0x00};
9
10   // Converte o comando de 16 bits para o formato little endian
11   // O operador & é usado para obter os 8 bits inferiores do comando
12   TX_Data[0] = command & 0xff;
13
14   // O operador >> é usado para deslocar os bits do comando 8 posições
       para a direita,
15   // efetivamente obtendo os 8 bits superiores do comando
16   TX_Data[1] = (command >> 8) & 0xff;
17
18   // Chama a função I2C_WriteRegister para escrever os dados no endereço
       do subcomando BQ76942
19   // Ignora o valor de retorno da função com o operador void
20   (void)I2C_WriteRegister(BQ76942_subcommand_address, TX_Data, 2);
21
22   // Atrasa a execução do programa em 2000 microssegundos para permitir
       que o comando seja processado
23   delayMicroseconds(2000);
24 }
```

Algoritmo A.8: Método SetRegister da classe BQ76942 em $C++$

```
1  // Função para configurar um registro do BQ76942
2  // Parâmetros:
3  // - register_address: o endereço do registro a ser configurado
4  // - register_data: os dados a serem escritos no registro
5  // - data_bytes: o numero de bytes a serem escritos
6
7  void BQ76942::SetRegister(uint16_t register_address, uint32_t
       register_data, uint8_t data_bytes)
8  {
9    // Inicializa um array de 2 bytes para armazenar o checksum a ser
       transmitido
10   uint8_t TX_Checksum[2] = {0x00, 0x00};
11
12   // Inicializa um array de 6 bytes para armazenar os dados a serem
       transmitidos
13   uint8_t TX_Data[6] = {0x00, 0x00, 0x00, 0x00, 0x00, 0x00};
14
15   // Preenche os dois primeiros bytes de TX_Data com o endereço do
       registrador em formato little endian
16   TX_Data[0] = register_address & 0xff;
17   TX_Data[1] = (register_address >> 8) & 0xff;
18
19   // Preenche o terceiro byte de TX_Data com o primeiro byte dos dados
       do registrador
20   TX_Data[2] = register_data & 0xff;
21
22   // Verifica o número de bytes de dados a serem escritos no registrador
23   switch (data_bytes)
24   {
25   case 1: // Se o tamanho dos dados for 1 byte
26     // Escreve o endereço do registrador e o dado no registrador de
       subcomando do BQ76942
27     (void)I2C_WriteRegister(BQ76942_subcommand_address, TX_Data, 3);
```

```
28      delayMicroseconds(2000); // Aguarda 2ms
29
30      // Define o primeiro byte do TX_Checksum com o checksum calculado
    dos dados transmitidos
31      TX_Checksum[0] = Checksum(TX_Data, 3);
32
33      // Define o segundo byte do checksum como o tamanho somado do endere
    ço do registrador,
34      // dos dados do registrador e do próprio checksum
35      TX_Checksum[1] = 0x05;
36
37      // Escreve o TX_Checksum ao registrador de checksum do BQ76942
38      (void)I2C_WriteRegister(BQ76942_checksum_address, TX_Checksum, 2);
39      delayMicroseconds(2000); // Aguarda 2ms
40      break;
41
42    case 2: // Se o tamanho dos dados for 2 bytes
43      // Preenche o quarto byte de TX_Data com o segundo byte dos dados a
    ser escrito
44      TX_Data[3] = (register_data >> 8) & 0xff;
45
46      // Escreve o endereço do registrador e os dois bytes de dado no
    registrador de subcomando do BQ76942
47      (void)I2C_WriteRegister(BQ76942_subcommand_address, TX_Data, 4);
48      delayMicroseconds(2000); // Aguarda 2ms
49
50      // Define o primeiro byte do TX_Checksum com o checksum calculado
    dos dados transmitidos
51      TX_Checksum[0] = Checksum(TX_Data, 4);
52
53      // Define o segundo byte do checksum como o tamanho somado do endere
    ço do registrador,
54      // dos dados do registrador e do próprio checksum
55      TX_Checksum[1] = 0x06;
56
```

```
57      // Escreve o TX_Checksum ao registrador de checksum do BQ76942
58      (void)I2C_WriteRegister(BQ76942_checksum_address, TX_Checksum, 2);
59      delayMicroseconds(2000); // Aguarda 2ms
60      break;
61
62    case 4: // Se o tamanho dos dados for 4 bytes (usado apenas para
       CCGain e Capacity Gain)
63      // Preenche o quarto, quinto e sexto bytes de TX_Data com os três ú
       ltimos bytes dos dados
64      // do registrador
65      TX_Data[3] = (register_data >> 8) & 0xff;
66      TX_Data[4] = (register_data >> 16) & 0xff;
67      TX_Data[5] = (register_data >> 24) & 0xff;
68
69      // Escreve o endereço do registrador e os quatro bytes de dado no
       registrador de subcomando do BQ76942
70      (void)I2C_WriteRegister(BQ76942_subcommand_address, TX_Data, 6);
71      delayMicroseconds(2000); // Aguarda 2ms
72
73      // Define o primeiro byte do TX_Checksum com o checksum calculado
       dos dados transmitidos
74      TX_Checksum[0] = Checksum(TX_Data, 6);
75
76      // Define o segundo byte do checksum como o tamanho somado do endere
       ço do registrador,
77      // dos dados do registrador e do próprio checksum
78      TX_Checksum[1] = 0x08;
79
80      // Escreve o TX_Checksum ao registrador de checksum do BQ76942
81      (void)I2C_WriteRegister(BQ76942_checksum_address, TX_Checksum, 2);
82      delayMicroseconds(2000); // Aguarda 2ms
83      break;
84    }
85  }
```