UNIVERSIDADE FEDERAL DA PARAÍBA
CENTRO DE ENERGIAS ALTERNATIVAS E RENOVÁVEIS
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA

Dissertação de Mestrado

Desenvolvimento de um Sistema de Gerenciamento de Baterias em Nuvem

Allan Alex de França

JOÃO PESSOA
28 de maio de 2025

ALLAN ALEX DE FRANÇA

DESENVOLVIMENTO DE UM SISTEMA DE GERENCIAMENTO DE BATERIAS EM NUVEM

Dissertação apresentada ao Programa de Pós-Graduação em Engenharia Elétrica - PPGEE, da Universidade Federal da Paraíba - UFPB, como requisito parcial para a obtenção do título de Mestre em Engenharia Elétrica.

Orientador: Prof. Dr. Euler Cássio Tavares de Macêdo

JOÃO PESSOA

28 de maio de 2025

Catalogação na publicação
Seção de Catalogação e Classificação

F814d França, Allan Alex de.
Desenvolvimento de um sistema de gerenciamento de baterias em nuvem / Allan Alex de França. - João Pessoa, 2025.
77 f. : il.

Orientação: Euler Cássio Tavares de Macêdo.
Dissertação (Mestrado) - UFPB/CEAR.

1. Baterias - sistema de gerenciamento. 2. Internet das Coisas. 3. Armazenamento de energia. I. Macêdo, Euler Cássio Tavares de. II. Título.

UFPB/BC CDU 621.352(043)

Elaborado por RUSTON SAMMEVILLE ALEXANDRE MARQUES DA SILVA - CRB-15/0386

ALLAN ALEX DE FRANÇA

Banca Examinadora

Prof. Dr. Euler Cássio Tavares de Macêdo
Orientador

Prof. Dr. Cícero da Rocha Souto
Avaliador Interno, DEE UFPB

Prof. Dr.Ivanovitch Medeiros dantas da Silva
Avaliador Externo, UFRN

Dissertação aprovada em 28 de Fevereiro de 2025

JOÃO PESSOA

28 de fevereiro de 2025

Ao bom DEUS, que em tudo é perfeito e agradável e que tanto me dá forças para continuar minha caminhada perseverante na fé; agradeço a Ele por conceder a oportunidade de completar mais uma etapa de minha vida e de estar cada dia mais renovado e feliz. Aos familiares, em especial, aos meus pais Paulo Nogueira e Maria da Paz França e minha irmã Allen França, que sempre estiveram e estarão presentes na minha vida, mostrando-me e guiando-me nos caminhos corretos deste mundo; a eles sou eternamente grato, e se cheguei até aqui foi graças ao cuidado incondicional deles por mim. E a minha amada esposa companheira Elãne França, que comigo compartilhou e compartilha tantas vitórias e derrotas sempre com serenidade e sabedoria que só à ela pertencem.

AGRADECIMENTOS

A Deus por ter me dado ânimo, ímpeto e saúde para continuar na caminhada em direção ao conhecimento.

Incondicionalmente aos meus familiares, que sempre estiveram me apoiando, dando condições viáveis à execução deste e de tantos outros trabalhos.

A Universidade Federal da Paraíba, especificamente ao Programa de Pós-Graduação em Engenharia Elétrica do Centro de Ciências Alternativas e Renováveis, que me proporcionou acesso à equipamentos, estruturas e servidores competentes, fundamentais no desenvolvimento desse trabalho.

Ao orientador e professor doutor Euler Cássio Tavares De Macêdo, e ao professor doutor Juan Moises Mauricio Villanueva, que guiaram-me nos caminhos científicos acerca deste trabalho e proporcionaram sua realização com excelência.

Ao Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPQ) por ter facilitado o acesso aos créditos de utilização da plataforma Amazon Web Services (AWS), por meio sa chamada CNPq/AWS N°64/2022 -Acesso aos Serviços de Computação em Nuvem da AWS Cloud Credits for Research.

A empresa Quântica Brasil, na pessoa no gerente operacional Adriano Silva, que incentivou e facilitou o curso das atividades realizadas durante o programa de pós graduação.

O amor é paciente, o amor é bondoso.
Não inveja, não se vangloria, não se orgulha.
Não maltrata, não procura seus interesses, não se ira facilmente, não guarda rancor.
O amor não se alegra com a injustiça, mas se alegra com a verdade.
Tudo sofre, tudo crê, tudo espera, tudo suporta.
(1 Coríntios 13:4-7)

RESUMO

As baterias são vitais em muitas aplicações modernas, desde dispositivos eletrônicos portáteis até veículos elétricos e sistemas de armazenamento de energia. Seu funcionamento correto é essencial para garantir a segurança, o desempenho otimizado, a eficiência energética e a redução de custos ao longo do tempo. Para isso, o Sistema de Gerenciamento de Bateria (*BMS - Battery Management System*) desempenha um papel fundamental, realizando monitoramento das células, estimativa do estado, balanceamento de carga, controle térmico e atividades de controle de carga e descarga. No entanto, a baixa capacidade de processamento e armazenamento de dados desses dispositivos é uma limitação significativa. Para resolver esse problema, uma nova abordagem de *hardware* e *software* foi desenvolvida e utilizada em sistemas de gerenciamento, o BMS em nuvem ou *cloud* BMS. Nessa nova arquitetura, a disponibilidade de capacidade de processamento e armazenamento de dados cresce exponencialmente. Este trabalho apresenta o desenvolvimento de uma solução de *hardware* e *software* para um sistema de gerenciamento de bateria baseado em nuvem, utilizando o módulo IoT ESP32 e a plataforma AWS IoT. O sistema projetado visa superar as limitações dos sistemas tradicionais ao aproveitar a computação em nuvem para proporcionar um maior poder de processamento e capacidades de armazenamento de dados. A solução proposta inclui um sistema formado por uma placa BMS especialmente projetada, *software* embarcado a esta placa que permite o monitoramento, estimativa de estado, balanceamento de carga, controle térmico e controle de carga/descarga, um *dashboard* hospedado num servidor remoto e uma aplicação python executando na nuvem um algoritmo de rede neural para estimação do estado de carga da bateria. Para validação desse sistema, um ensaio foi realizado inicialmente conectando à entrada do BMS projetado um *pack* de 48V formado por 13 células de íons de lítio e à saída uma carga resistiva de aproximadamente 50 Ω para gerar consumo de corrente. O monitoramento da tensão do *pack*, tensão das células individuais, corrente do *pack*, temperatura das células e temperatura das FETS de controle durante o ensaio é realizado por meio do *dashboard* e a estimativa do estado de carga do *pack* é realizada por dois métodos distintos, o coulomb counting (SoC local) e por uma rede neural previamente treinada (SoC remoto). Ao final do ensaio a comparação entre os resultados dos dois métodos de estimação retornou um valor MSE de 7,82 com um MAPE de 6,57% e um índice MAE de 2,33, indicando que a estimação remota do SoC por rede neural se aproxima com boa exatidão ao valor do SoC local.

Palavras-chave: cloud BMS, internet das coisas, sistema de baterias.

ABSTRACT

Batteries are vital in many modern applications, from portable electronic devices to electric vehicles and energy storage systems. Their correct operation is essential to ensure safety, optimized performance, energy efficiency, and cost reduction over time. To achieve this, the BMS (Battery Management System) plays a fundamental role, performing cell monitoring, state estimation, charge balancing, thermal management, and charge/discharge control activities. However, the low processing and data storage capacity of these devices is a significant limitation. To address this problem, a new hardware and software approach has been developed and utilized in management systems, the cloud BMS. In this new architecture, the availability of processing and data storage capacity grows exponentially. This work presents the development of a hardware and software solution for a cloud-based battery management system, using the ESP32 IoT module and the AWS IoT platform. The designed system aims to overcome the limitations of traditional BMS by leveraging cloud computing for greater processing power and data storage capabilities. The proposed solution includes a system composed of a specially designed BMS (Battery Management System) board, embedded software that enables monitoring, state estimation, charge balancing, thermal control, and charge/discharge management, a dashboard hosted on a remote server, and a Python application running in the cloud that executes a neural network algorithm for battery state-of-charge (SoC) estimation. For system validation, an initial test was conducted by connecting a 48V battery pack, consisting of 13 lithium-ion cells, to the input of the designed BMS, while a resistive load of approximately 50 Ω was connected to the output to generate current consumption. The monitoring of pack voltage, individual cell voltage, pack current, cell temperature, and the temperature of the control FETs during the test is carried out via the dashboard. The estimation of the pack's state of charge (SoC) is performed using two distinct methods: Coulomb counting (local SoC) and a previously trained neural network (remote SoC). At the end of the test, the comparison between the results of the two estimation methods returned an MSE value of 7.82, a MAPE of 6.57%, and an MAE index of 2.33, indicating that the remote SoC estimation by the neural network closely approximates the local SoC value with good accuracy.

Keywords: Cloud BMS. Internet of Things. Battery System.

LISTA DE ILUSTRAÇÕES

LISTA DE QUADROS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

API	*Application Programming Interface*
BESS	*Battery Energy Storage System*
BMS	*Battery Management System*
CDN	*Content Delivery Network*
GPIO	*General Purpose Input Output*
IoT	*Internet of Things*
LDO	*Low Dropout Regulator*
LiFePO4	Fosfato de ferro e lítio
MAE	*Mean Absolut Error*
MOSFET	*Metal Oxide Semiconductor Field Effect Transistor*
MSE	*Mean Squared Error*
NIST	*National Institute of Standards and Technology*
NTC	*Negative Temperature Coefficient*
PTC	*Positive Temperature Coefficient*
RISC	*Reduced Instruction Set Computer*
SDK	*Software Development Kit*
SoC	*System on Chip*
SOC	*State of Charge*
SOH	*State of Health*
SQL	*Structured Query Language*
TQFP	*Thin Quad Flat Package*
Wi-Fi	*Wireless Fidelity*

LISTA DE SÍMBOLOS

mΩ	mili-Ohm
kΩ	quilo-Ohm
mA	mili-Ampère
μA	micro-Ampère
MHz	mega-Hertz

SUMÁRIO

1 INTRODUÇÃO

As baterias recarregáveis têm desempenhado um papel fundamental na revolução tecnológica das últimas décadas, impulsionando o desenvolvimento de dispositivos eletrônicos portáteis, veículos elétricos e sistemas de armazenamento de energia (DING et al., 2019; OPITZ et al., 2017; MIAO et al., 2019). Sua ampla aplicação em diversas áreas tem sido resultado do contínuo aprimoramento das tecnologias empregadas nesses dispositivos, visando o aumento da capacidade de armazenamento, a melhoria da eficiência e a garantia de maior segurança operacional.

Existem vários tipos de baterias recarregáveis disponíveis no mercado e dentre as principais estão: baterias de chumbo-ácido, Níquel-Hidreto Metálico (NiMH), Níquel-Cádmio (NiCd), Polímero de Lítio (LiPo) e Íon de Lítio (Li-ion). A escolha do melhor tipo depende das necessidades específicas da aplicação, por exemplo, as baterias de Níquel-Cádmio possuem densidade energética entre 60 e 150 Wh/L e possuem longos ciclos de vida, enquanto que as de chumbo ácido possuem baixa densidade energética (entre 50 e 80 Wh/L) mas possuem uma alta capacidade de corrente. Dentre esses tipos a tecnologia que se destaca por sua alta densidade energética (entre 200 e 400 Wh/L), baixa taxa de auto descarga, ciclo de vida longo (próximo de 10.000 ciclos), peso leve e alta densidade de potência (entre 1.500 e 10.000 W/L), são as baterias de íons de Lítio (Li-ion) (IRENA, 2017). Porém essas baterias possuem desvantagens como sensibilidade a altas temperaturas, risco de explosão ou incêndio se danificada ou carregada inadequadamente e custo mais alto.

Com a crescente utilização das baterias de íons de lítio em diversos equipamentos e sistemas, se faz necessário a sua correta operação e gerenciamento para se garantir a eficiência, segurança e confiabilidade (HU et al., 2019). O dispositivo responsável por esse gerenciamento é o BMS (*Battery Management System*), que combina recursos de hardware e software para desempenhar atividades de monitoramento, controle e proteção das baterias. Além dessas atividades, o BMS também realiza a estimação de estados da bateria, balanceamento de carga, controle térmico e controle de carga e descarga (TRAN et al., 2022).

Os principais estados da bateria que devem ser monitorados e estimados são o Estado de Carga (*State of Charge*, SOC), o Estado de Saúde (*State of Health*, SOH) e o Estado de Energia (*State of Energy*, SOE). Cada um desses estados fornece informações cruciais sobre a condição e a capacidade da bateria. O SOC representa a quantidade de energia disponível na bateria em relação à sua capacidade total, o SOH indica a condição geral da bateria em relação as especificações de fábrica, refletindo a capacidade máxima atual e a eficiência de carregamento e o SOE refere-se à quantidade de energia utilizável

disponível na bateria em relação à sua capacidade total de energia.

Devido as reações químicas internas das baterias serem complexas e não lineares, seu comportamento varia com o tempo e o uso, incluindo efeitos como histerese (diferença na tensão durante carga e descarga) e taxa de auto descarga. Se torna bastante difícil a obtenção dos estados da bateria de forma precisa e direta, sendo necessário a estimação desses parâmetros. Diversas técnicas de estimação foram desenvolvidas ao longo dos anos e estão disponíveis na bibliografia, dentre elas têm-se a Contagem de Coulomb (*Coulomb Counting*), Tensão de Terminal da bateria, Modelo de Circuito Equivalente, Análise de Impedância Espectral, Resistência Interna, filtros de Kalman e Redes Neurais (HU et al., 2019). Cada método de estimação possui suas limitações e dificuldades inerentes. Por exemplo, na técnica de estimação por Modelo de Circuito Equivalente, modelos mais precisos são mais complexos e exigem maior poder computacional; nos filtros de Kalman, a implementação em sistemas embarcados com recursos limitados pode ser computacionalmente impraticável e as técnicas avançadas que utilizam Redes Neurais muitas vezes necessitam de grandes quantidades de dados de treinamento para fornecer estimativas precisas.

Com o crescente desenvolvimento de algoritmos de estimação de estados cada vez mais eficientes e complexos e que exigem um poder computacional relativamente alto e grandes quantidade de armazenamento de dados, é constante a busca por opções de implementação de BMS que permitam uma boa capacidade de processamento, armazenamento e escalabilidade. Para contornar esses obstáculos, uma nova arquitetura de *software* e *hardware* foi proposta e têm sido desenvolvida nos BMS, utilizando os conceitos de *Internet of Things* (IoT) e *cloud computing*, chamada de "*cloud* BMS". Com essa nova abordagem, o poder de processamento disponível para executar os algoritmos de estimação, predição e diagnósticos, cresce a uma taxa exponencial, bem como a capacidade de armazenamento dos dados históricos de funcionamento e de escalabilidade.

1.1 OBJETIVO GERAL

Este trabalho tem por objetivo geral o projeto e construção de um sistema composto por *hardware* e *software* que permita o monitoramento e controle de baterias de íons de lítio aplicando os recursos de computação em nuvem (*cloud computing*) e internet das coisas.

1.2 OBJETIVO ESPECÍFICO

Como objetivo específico, têm-se a construção de uma plataforma em nuvem capaz de executar algoritmos avançados de estimação de estados das baterias e predição de falhas, agrupados em uma única placa, bem como a exibição por meio de um painel *dashboard*

das medições de grandezas relacionadas às baterias, como tensão, corrente e temperatura.

1.3 CONTRIBUIÇÕES DO TRABALHO

Pelo estudo e análise dos trabalhos já publicados a respeito do tema dessa dissertação, observa-se que há um esforço maior por parte dos autores no desenvolvimento de estruturas de *software* para análise dos dados das baterias, utilização de técnicas modernas na estimação de estados e predição de falhas, ficando a estrutura de *hardware* adaptada, composta por mais de um dispositivo de *hardware* para criação do *cloud* BMS.

Este trabalho se destaca pela proposta de criação de um sistema completo de *hardware* e *software* dedicados a operação dos dados das células de baterias e estimação de estados usando uma placa unificada. O *hardware* se caracteriza por atender a uma série de requisitos como eficiência energética, precisão de medição, acessibilidade remota sem fio e segurança operacional, tudo isso embarcado em uma única placa compacta. Já o *software* permitirá alta escalabilidade do sistema e adequação às mais variáveis situações de utilização das baterias.

1.4 ORGANIZAÇÃO DO TRABALHO

Além deste capítulo introdutório, este trabalho de dissertação contém mais cinco capítulos. No segundo capítulo, são apresentados trabalhos relacionados ao tema de pesquisa que norteiam o entendimento do leitor.

No terceiro capítulo, é apresentada a fundamentação teórica sobre o tema de gerenciamento de baterias, definições e conceitos da computação em nuvem, plataforma *Amazon Web Services* e seu serviço de internet das coisas e a descrição do sistema de gerenciamento de baterias baseado em nuvem.

No quarto capítulo apresenta-se a metodologia e os materiais utilizados para o desenvolvimento do trabalho proposto.

No quinto capítulo, os resultados são apresentados, em especial os que tratam da implementação de um circuito impresso e da plataforma de gerenciamento desenvolvida, assim como, a descrição detalhada sobre cada um destes.

O sexto capítulo consiste na conclusão do trabalho, no qual é realizada uma comparação dos resultados obtidos com os esperados ao traçar os objetivos. Também são discutidas as futuras aplicações e utilidades deste projeto, bem como, propostas de trabalhos futuros.

2 REVISÃO BIBLIOGRÁFICA

Neste capítulo é apresentada a revisão bibliográfica baseada em alguns dos principais artigos publicados a respeito do tema de *cloud* BMS que vão desde sistemas de armazenamento de baterias em larga escala, até testes em conjuntos de baterias reconfiguráveis em aplicações de mobilidade e estacionárias.

2.1 TRABALHOS RELACIONADOS

Muitos estudos relacionados à soluções em nuvem para monitoramento, controle e operação de dados em células de bateria foram realizados ao longo dos anos.

Com ênfase em aplicações de sistemas de armazenamento em larga escala utilizando baterias de íons de lítio (KIM et al., 2018) apresentam uma proposta de plataforma baseada em nuvem para o monitoramento da condição das baterias e diagnóstico de falhas. Esse sistema é formado basicamente por dois componentes: o primeiro é o módulo de gerenciamento sem fio, no qual são realizadas as tarefas de aquisição dos sinais das baterias a dada frequência de amostragem, armazenamento local temporário desses dados e transmissão ao servidor em nuvem (nesse trabalho foram utilizadas três placas Raspberry Pi para simular dados de trinta células de bateria para validação dos resultados); o segundo componente é a plataforma em nuvem de gerenciamento da bateria, composta pelo armazenamento em massa, algoritmos e ferramentas de *software* para análise e visualização dos dados. Esse segundo componente foi desenvolvido no ambiente do *Google Cloud* e foram implementados nele um algoritmo baseado em filtro híbrido (uma combinação do filtro de Kalman estendido com um filtro de estrutura variável suave[1]) para o monitoramento das condições das células, e um algoritmo de diagnóstico de falhas baseado em detecção de *outliers*. Após análises de custo computacional em um ambiente de simulação da plataforma proposta, os autores verificaram sua validade em escalabilidade, velocidade e custo benefício.

Em um trabalho mais recente, (WEIHAN et al., 2020) propõem a construção de um gêmeo digital de baterias de chumbo ácido e íons de lítio baseado em um BMS implementado em nuvem. Para tal, as informações das baterias, tais como temperatura, tensão e corrente, são obtidas por um chip monitor de baterias multi célula (LTC 6804G-2) e um microcontrolador ARM de 32 bits (LPC 11C14F/301). Esse, por conseguinte se comunica com um computador *Raspberry* Pi que é responsável pela transmissão dos dados à nuvem a partir dos protocolos MQTT e TCP/IP. Nessa aplicação, além da capacidade

[1] *Smooth Variable Structure Filter* (SVSF), é um filtro de estimação robusto baseado em conceitos de controle em modo deslizante. Ele é projetado para fornecer uma estimativa confiável do estado de sistemas dinâmicos, mesmo na presença de incertezas e ruídos.

de armazenamento e de visualização de dados em tempo real das baterias, os autores propõem a estimação dos estados de carga e de saúde por meio da implementação em nuvem de algoritmos baseados em modelo de circuito equivalente. Para a obtenção do SOC, é implementado o filtro H-infinito adaptativo estendido, e para o SOH é utilizado o algoritmo de otimização por enxame de partículas. Ao final de testes em diferentes condições de operação das baterias, foram obtidos resultados de estimação de tensão e SOC com erros na ordem de 0,01V e 0,06%, respectivamente.

Uma arquitetura para analisar e armazenar medições de sistemas estacionários e móveis de baterias reconfiguráveis foi desenvolvida por Karnehm et al. (2023). Essa arquitetura é composta por três componentes: unidade estacionária, unidade móvel e unidade em nuvem. Na unidade estacionária são realizadas medições de corrente e tensão de um conjunto de 12 células de baterias emuladas por fontes de alimentação *Rohde & Schwarz*. Esse conjunto de células alimenta um conversor híbrido multinível em cascata que por sua vez fornece energia a dois resistores de carga de 10 Ω por 250 W em série. Um computador *Raspberry* Pi 4 é utilizado para controlar o conversor e enviar os dados das medições para o servidor em nuvem. Nessa unidade o objetivo é analisar a eficiência do conversor na conversão de potência DC das células para o AC. Na unidade móvel, um carrinho de praia movido a eletricidade é utilizado como veículo de teste. Nele está contido um sistema trifásico de baterias reconfiguráveis conectado a um motor síncrono trifásico de ímã permanente. Os módulos de bateria são operados por uma unidade de controle central e esta se comunica com uma placa *Raspberry* Pi 4 para conexão com o servidor e transferência de dados de tensão, corrente e temperatura das células. Para comunicação com o servidor foram implementadas três soluções: uma conexão WLAN em bancada de testes dentro do laboratório, um roteador 5G para conexão fora do laboratório e um *buffer* local em contingência, caso nenhuma das conexões anteriores ocorram. As unidades estacionária e móvel da arquitetura proposta se conectam a uma infraestrutura em nuvem da *Amazon Web Services* (AWS) por meio do serviço *Kinesis*. Após a conexão os dados são armazenados, processados e visualizados com o auxílio de serviços fornecidos pela AWS. Como resultado do trabalho, os autores desenvolveram um aplicativo *Web* para análise em tempo real que exibe a eficiência do conversor em formato de gráfico de barra, a variação de tensão do conversor em formato de gráfico de série histórica e a corrente das 12 células também em formato de gráfico de série histórica.

Uma plataforma de monitoramento de baterias que integra Kubernetes e tecnologia *cloud-edge* para monitoramento eficiente da temperatura de grandes quantidades de baterias é apresentada por (CHEN et al., 2024), o KBMP. A plataforma oferece alertas antecipados de fuga térmica em até 30 minutos, reduz a latência de transmissão de dados em até 20% e utiliza o algoritmo K-Means para detecção de falhas por fuga térmica. A arquitetura inclui quatro camadas principais: recepção de dados, armazenamento, análise e visualização. A recepção de dados coleta os dados das baterias usando protocolos IoT como MQTT. Nessa

etapa, é utilizada uma placa Raspberry pi como MQTT broker. Para o armazenamento é utilizado um mini computador Linux executando banco de dados InfluxDB para armazenar séries temporais. E para a análise e visualização, é utilizado outro mini computador Linux executando o algoritmo K-Means para detecção de fuga térmica e a plataforma Grafana como ferramenta de *dashboard*. Além disso, existe ainda outro mini computador Linux executando o Kubernet e orquestrando todo o restante da plataforma. Ao final, dados reais de fuga de baterias são transmitidos ao MQTT broker de forma simulada e utilizados para testar a performance da plataforma KBMP.

Em (WU et al., 2021) é proposto um método de estimativa do estado de saúde (SOH) de baterias de íon-lítio baseado numa abordagem nuvem-para-borda. Os dados de tensão e corrente durante o processo de carregamento da bateria são extraídos para modelar o processo de degradação e representar o SOH. Em seguida, as informações obtidas são transferidas para a plataforma remota na nuvem por uma unidade de telecomunicação, onde um regressor *Random Forest* é implementado para a estimativa do SOH. Os dados operacionais da bateria são medidos e extraídos por um BMS composto por um microcontrolador de 16-bit da NXP e um monitor de baterias. Um processo de seleção em três etapas de características de corrente e tensão foi proposto nesse trabalho, reduzindo o ruído e otimizando a estimativa do SOH. Com a coordenação entre a plataforma em nuvem e o BMS, o SOH pôde ser estimado com precisão. Experimentos com diferentes tipos de baterias de íon-lítio foram realizados para verificar o método proposto.

No trabalho de (WANG et al., 2022), é explorado o uso de gêmeos digitais em colaboração com uma arquitetura em camadas (nuvem-borda-terminal) para gerenciamento inteligente de baterias, replicando seu comportamento e integrando dados coletados por sensores e algoritmos avançados na nuvem. A arquitetura é composta por quatro camadas: camada de computação de borda, acesso aos dados, armazenamento e análise de dados e camada de aplicação. A camada de computação de borda contém o servidor de computação de borda que executa os algoritmos de estimativa do BMS e a estratégia de controle com base nos dados coletados, além de retornar os comandos de controle da nuvem para o BMS. A camada de acesso aos dados tem como principal função fornecer um canal bidirecional entre a camada de computação de borda e a nuvem. Já na camada de armazenamento e análise de dados, todos os dados operacionais das baterias são armazenados de forma uniforme, e podem ser utilizadas em tarefas como relatórios, visualização, análise e aprendizado de máquina. Por fim, a camada de aplicação é composta por módulos de aplicação, como visualização de dados, exibição de relatórios, aviso de falhas, entre outros. Para obter resultados dessa arquitetura, um ambiente de laboratório foi montado, formado por um sistema de teste de bateria, uma cabine de temperatura, um computador de mesa, um módulo BMS *master*, um módulo BMS *slave*, um *pack* de bateria, um módulo transmissor 5G, uma fonte de tensão DC e algumas antenas. Os testes resultaram que a maioria dos erros de previsão do modelo de gêmeo digital é inferior a 30 mV, com o MAE

e o RMSE sendo de 19 mV e 27 mV, respectivamente, alcançando portanto alta precisão do modelo com o método proposto.

Na Tabela 1 estão reunidas as informações dos *hardwares* utilizados nos trabalhos relacionados citados nesse capítulo, bem como os serviços da nuvem e a existência de painel *dashboard* implementados em cada um deles, e na Tabela 2 apresenta-se o comparativo dos recursos disponíveis nos *hardwares* de processamento Raspberry Pi 4 (RASPBERRY PI, 2019) e chip LPC11C14F/301 (NXP, 2016), utilizados pelos autores, com a placa ESP32-C3-MINI-1 (ESPRESSIF SYSTEMS, 2022).

Pela análise da Tabela 1 é possível notar que em nenhum desses trabalhos foi utilizada uma solução completa de *cloud BMS*, com hardware dedicado, contendo funções de processamento de dados, monitoramento de bateria, controle de carga e descarga, controle térmico e acesso remoto, bem como a integração com plataformas na nuvem em conjunto com *dashboard* customizável e escalável. E observando as características dos hardwares da Tabela 2, e realizando a interseção das características de conectividade remota, consumo de energia e custo, observa-se que a plataforma ESP32 C3-MINI-1 se destaca das outras por reunir as vantagens de baixo custo, conectividade Wi-Fi nativa, consumo reduzido e fácil integração com IoT (MQTT, HTTP). Os preços mencionados na Tabela 2 foram estimados com base em informações disponíveis em distribuidores e varejistas de componentes eletrônicos, tais como Digikey (DIGIKEY, 2024), PiHut (THE PIHUT, 2024) e Mouser (MOUSER ELETRONICS, 2024).

Tabela 1 – Tabela comparativa das soluções de *cloud* BMS.

Solução	***Hardware***	**Serviços**	***Painel***
(KIM et al., 2018)	Placa *Raspberry* Pi. (Dados simulados)	*Google Cloud (Google App Engine, Google Sheets* API)	
(WEIHAN et al., 2020)	Placa *Raspberry* Pi; chip LTC 6804G-2; microcontrolador ARM LPC 11C14F/301).	*Data logger* e banco de dados hospedado na Alemanha	✓
(KARNEHM et al., 2023)	Placa *Raspberry* Pi 4 com CAN-shield. (Dados simulados)	AWS (Kinesis, S3, Glue, Athena, Amplify, Grafana, Lambda) *React(Web App)*	✓
(CHEN et al., 2024)	Placa *Raspberry* Pi 4B; 3x mini computador Linux. (Dados simulados)	Kubernetes, EMQX, InfluxDB e Grafana, Hospedagem Local	✓
(WU et al., 2021)	BMS (Microcontrolador NXP e monitor de baterias); Unidade remota de telecomunicação; Computador (3.7 GHz (CPU), 1.6 GHz (GPU), 16 GB RAM, e 500 GB SSD).		
(WANG et al., 2022)	Computador; Módulo BMS master; Módulo BMS slave; Módulo transmissor 5G;		
(FRANÇA, A., 2025)	PCB confeccionada contendo o chip ESP32-C3-MINI-1 e o chip BQ76952	AWS IoT Core; AWS TimeStream; Sagemaker Jupyter; Thingsboard.	✓

Fonte: Produzido pelo autor.

Tabela 2 – Tabela comparativa dos *hardwares* de processamento.

Característica	**ESP32-C3-MINI-1**	**Raspberry Pi 4**	***ARM LPC11C14F/301***
Processamento	160 MHz, 32-bit RISC-V	1.5 GHz, Quad-core ARM Cortex-A	50 MHz, Cortex-M0
Conectividade	Wi-Fi 4, Bluetooth 5	Wi-Fi, Ethernet, Bluetooth	CAN/UART
Protocolos IoT	MQTT, HTTP, HTTPS	MQTT, HTTP, HTTPS, WebSockets	
Consumo	∼100 mA	>500 mA	∼10 mA
Memória	400 KB RAM, 4 MB Flash	2 GB – 8 GB LPDDR4 + MicroSD	8 KB SRAM + 32 KB Flash
Custo	∼ $3 a $5	∼ $35 a $70	∼ $2 a $5

Fonte: Produzido pelo autor.

3 FUNDAMENTAÇÃO TEÓRICA

Este capítulo fornece as bases teóricas e conceituais necessárias ao entendimento do trabalho desenvolvido. Nele são abordados os temas de sistema de gerenciamento de baterias, definições e conceitos da computação em nuvem, plataforma *Amazon Web Services* e seu serviço de internet das coisas e a descrição do sistema de gerenciamento de baterias baseado em nuvem. Ao final é apresentado o tema de redes neurais artificias.

3.1 SISTEMA DE GERENCIAMENTO DE BATERIAS

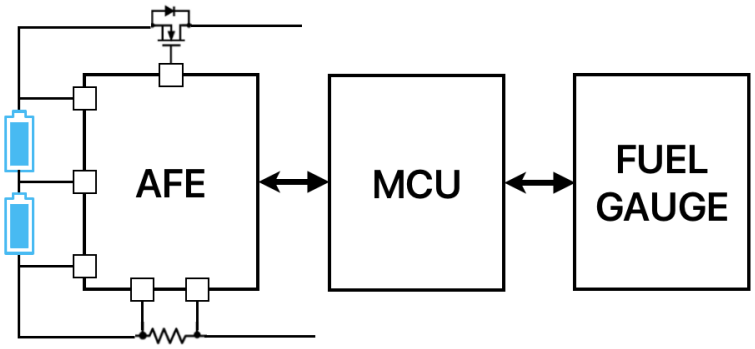
Um Sistema de Gerenciamento de Baterias, do inglês *Battery Management System* (BMS) é um sistema eletrônico integrado responsável por supervisionar, gerenciar e proteger baterias recarregáveis, geralmente compostas por células individuais conectadas em série e/ou paralelo (MPS, 2022). Suas funções principais incluem:

- **Monitoramento:** Monitora continuamente a tensão, corrente, temperatura e estado de cada célula da bateria;
- **Balanceamento:** Realiza o equilíbrio de carga para garantir que todas as células num *pack* de bateria mantenham níveis de carga uniformes, prevenindo sobrecargas ou descargas desiguais;
- **Gerenciamento de Carga e Descarga:** Controla os processos de carga e descarga da bateria de acordo com os parâmetros de operação especificados;
- **Detecção de Falhas:** Identifica e diagnostica potenciais falhas no sistema, como desequilíbrios de carga, curtos-circuitos ou falhas de células individuais;
- **Proteção contra Sobrecarga e Descarga:** Implementa circuitos de proteção para interromper a carga ou descarga da bateria caso os limites de tensão ou corrente sejam excedidos;
- **Proteção Térmica:** Monitora a temperatura da bateria e ativa dispositivos de segurança para evitar superaquecimento;
- **Estimação de estados:** Realiza estimativas de estados de saúde e de carga das células para indicar por exemplo, os níveis percentuais de carga e vida útil restantes da bateria.

Apresenta-se na Figura 1 a arquitetura principal de um BMS de baixa e média tensões. Essa arquitetura é composta basicamente por três dispositivos: uma interface

analógica (AFE - *front-end analog interface*), um microcontrolador principal (MCU - *microcontroller*) e um dispositivo para cálculos e exibição de informações (*Fuel Gauge*)[1].

Figura 1 – Arquitetura geral de um BMS.



Fonte: Produzido pelo autor.

A interface ou dispositivo AFE geralmente é um circuito integrado responsável pela aquisição da tensão, corrente e temperatura das células e disponibilização ao MCU e o *Fuel Gauge*, além de atuar nos dispositivos de proteção que seccionam a bateria caso haja alguma falha de operação. Com base nas leituras de seu ADC embarcado e em parâmetros configurados na sua memória, o AFE é capaz de detectar falhas e atuar em chaves (geralmente MOSFETS) dedicados para proteção primária do sistema.

A obtenção da corrente no AFE se dá principalmente por meio do método *shunt*, no qual uma resistência de valor baixo e de baixa tolerância (em torno de 1%) é inserida em série com o fluxo de corrente da bateria, o que gera uma queda de tensão nesse resistor, proporcional à intensidade da corrente circulante. O conversor analógico/digital (A/D) de alta exatidão geralmente presente no AFE realiza a aquisição da tensão no resistor e a converte para corrente calculada. O método *shunt* possui a vantagem de ser simples e de resposta linear, além de ter a possibilidade de alcançar elevadas larguras de banda (MARKUS et al., 2018).

A medição das tensões das células da bateria e a tensão total do módulo de baterias (*pack*)[2] é realizada no AFE por meio de seu conversor AD dedicado e eventual circuito passivo de filtragem, como redes RC. Já a temperatura é obtida principalmente com o emprego de sensores NTC e PTC, nos quais a tensão nesses sensores é digitalizada pelo conversor A/D e assim, a temperatura é calculada.

No geral, são utilizados nos BMS, MOSFETS de proteção e controle, que geralmente são de canal N, pois possuem uma resistência interna em modo de condução menor que

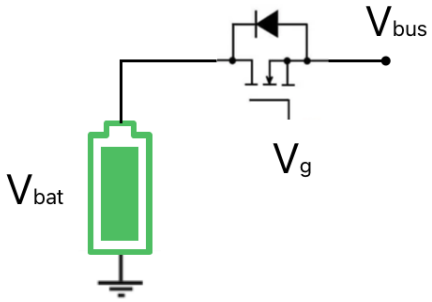
[1] Em algumas implementações o dispositivo *Fuel Gauge* pode ser embarcado junto com o MCU ou pode ser outro microcontrolador dedicado.

[2] Em algumas configurações de BMS a tensão do módulo de baterias (*pack*) é dividida por resistores para não se ultrapassar o limite máximo de tensão do conversor A/D implementado no AFE.

os de canal P, e além disso, eles podem ser projetados para seccionar o terminal positivo ou o negativo da célula de bateria. É mais recomendada a utilização dos MOSFETS no seccionamento do terminal positivo da bateria, conforme apresentado na Figura 2, pois dessa forma a conexão do terra (GND) para o restante do circuito permanece sempre constante, o que evita flutuações e falhas de comunicação com o MCU por falta referência ao GND. Por outro lado, utilizar o MOSFET no terminal positivo da bateria requer que a tensão de porta (V_{GATE}) seja maior que à tensão do módulo a para correta operação, isso implica na implementação de circuitos elevadores de tensão para as portas dos transistores.

A atuação do AFE nos MOSFETS se dá baseada na detecção de condições fora dos limites configurados e salvos na sua memória. Essas condições envolvem sobretensão (OV) e sub tensão (UV) das células, sobrecorrente (OC) e curto circuito (SC), sobretemperatura (OT) e sub temperatura (UT), que são continuamente monitoradas para garantir a proteção do sistema.

Figura 2 – Seccionamento do terminal positivo da bateria.



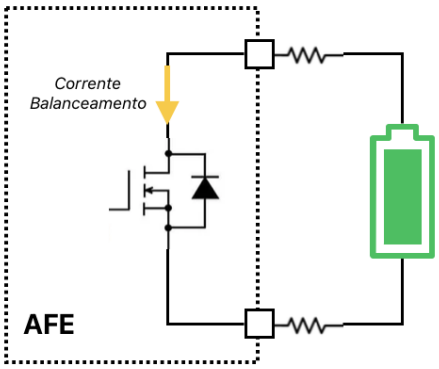
Fonte: Produzido pelo Autor.

Outro recurso disponível no AFE e que colabora para um melhor desempenho das das células dentro do módulo é o balanceamento de carga passivo. Esse procedimento consiste no descarregamento das células com maior tensão dentro do módulo até que todas estejam equalizadas. O descarregamento no AFE é realizado a partir de circuitos de chaves internas, como ilustrado na figura 3, e células consecutivas não podem ser balanceadas ao mesmo tempo devido a aspectos construtivos do chip, o que resulta num tempo maior para equalização de todas as células.

O MCU apresentado na arquitetura do BMS é responsável pela comunicação com o AFE e o *Fuel Gauge*, configuração dos parâmetros do AFE, proteção secundária e comunicação com outros dispositivos externos ao BMS.

O *Fuel Gauge* a depender da arquitetura, pode ser um circuito integrado dedicado ou pode estar embarcado no programa do MCU, mas em ambas situações sua função é de estimação de parâmetros como estado de carga (SOC) e estado de saúde (SOH) das células, baseado nas leituras de corrente, tensão e temperatura.

Figura 3 – Balanceamento interno de célula no AFE.



Fonte: Produzido pelo autor.

3.1.1 Estimação de parâmetros

Uma das principais funções de um BMS é a estimativa de parâmetros chave necessários à correta operação da bateria. Um dos parâmetros fundamentais utilizados no gerenciamento das baterias é o Estado de Carga, do inglês *State of Charge*-SOC. Com ele, é possível obter a capacidade de energia remanescente e/ou tempo de uso da bateria, bem como o controle de carregamento e descarregamento pode ser exercido com maior eficiência. Devido ao comportamento eletroquímico dinâmico das baterias, não é possível realizar a medição direta do SOC a partir de sensores. Porém, com a medição da tensão, corrente, temperatura e resistência interna, é possível se estimar o estado de carga com boa precisão (ZUOLU et al., 2021).

3.1.1.1 Estado de Carga (SOC)

O SOC é definido como o percentual da capacidade restante de carga em relação a capacidade máxima total da bateria (KAILONG et al., 2019). Em termos matemáticos, pode ser definido como

$$\mathrm{SOC(t)} = \frac{C_r}{C_m} \times 100\% \tag{3.1}$$

Em que C_r é a capacidade remanescente e C_m a capacidade total da bateria. Um valor de 0% do SOC indica que a bateria esta completamente descarregada, enquanto que 100% indica o contrário. Em aplicações práticas, as baterias geralmente operam em faixas de SOC de 20% a 80% para se evitar sobrecarregamentos e descarregamentos profundos (ZUOLU et al., 2021).

Devido a relativa facilidade de medição direta de corrente drenada e fornecida às baterias nos BMS, um dos métodos de estimação do SOC que possuem uma implementação simples e de baixa complexidade computacional é o *Coulomb Counting* (MUHAMMAD et

al., 2019). Nesse método, com a medição e integração da corrente instantânea de carga e descarga da bateria, é possível obter o SOC por meio da Equação 3.2:

$$\text{SOC(t)} = \text{SOC}(t_0) + \frac{\eta}{C_m}\int_{t_0}^{t} I_{bat}(t)dt, \tag{3.2}$$

Sendo $SOC(t_0)$ o estado de carga inicial no instante t_0, C_m a capacidade total da bateria, η a eficiência da bateria que é a razão da capacidade de descarga pela capacidade de carga num mesmo ciclo, e $I_{bat}(t)$ é a corrente da bateria no instante t, considerada negativa na descarga e positiva na carga. A forma discreta da Equação 3.2 pode ser:

$$\text{SOC [k]} = \text{SOC [k-1]} + \frac{\eta}{C_m} \cdot \Delta T \cdot I[k], \tag{3.3}$$

sendo ΔT o período de amostragem e I [k] a corrente medida de carga/descarga.

A definição desse método, obtida a partir da Equação 3.2, se torna altamente precisa quando se considera que toda carga armazenada na bateria pode ser obtida sob quaisquer condições de operação e a qualquer instante (BERGVELD, 2001). Porém, algumas limitações impostas pela construção e complexidade de reações químicas nas baterias impedem que este método obtenha a mesma precisão em situações diversas. Fatores que influenciam no desempenho do método *Coulomb Counting* são:

- **Eficiência de carregamento**: Reações colaterais ocorrem ao final do carregamento, resultando na falta de armazenamento da carga total na bateria. A eficiência de carregamento depende do *SOC*, corrente e temperatura;
- **Eficiência de descarregamento**: Apenas parte da carga disponível na bateria pode ser recuperada. No geral, menos carga pode ser obtida da bateria em baixas temperaturas e/ou em altas correntes de descarga, bem como o envelhecimento pode causar o efeito de elevação da resistência interna da bateria;
- **Auto-Descarga**: Toda bateria perde gradualmente um certo nível de carga com o passar do tempo, o que se torna aparente quando não se é utilizada por um longo período;
- **Perda de capacidade**: Naturalmente, a capacidade máxima de carga em Ampère-hora (Ah) da bateria decresce com o envelhecimento. Alguns fatores contribuem na aceleração dessa perda, por exemplo a operação incorreta de carregamento e descarregamento e a alta contagem de ciclos de carga e descarga;
- **Efeitos de armazenamento**: Alguns tipos de bateria podem apresentar desvios temporários em suas características após longos períodos de armazenamento aliado à variações de temperatura.

Além disso, este método é altamente dependente da medição de corrente, sendo assim, por ser um processo integrativo, o acúmulo de erros ao longo do tempo pode afetar significativamente a precisão da estimação, e além disso, a determinação do SOC inicial é o seu principal desafio de implementação.

3.2 COMPUTAÇÃO EM NUVEM

Computação em nuvem é um novo modelo operacional e conjunto de tecnologias para o gerenciamento de recursos de computação compartilhados (CSA, 2017). Permite o acesso remoto a serviços de armazenamento, processamento e gestão de dados a partir da internet. Essa abordagem oferece uma série de vantagens em termos de flexibilidade, escalabilidade, e economia de custos, transformando a maneira como empresas e indivíduos utilizam e gerenciam recursos computacionais.

O movimento de computação em nuvem é motivado pela ideia de que o processamento e armazenamento de dados podem ser realizados mais eficientemente em grandes sistemas de computação e armazenamento, acessíveis via internet (MARINESCU, 2018). Algumas características essenciais da computação em nuvem são:

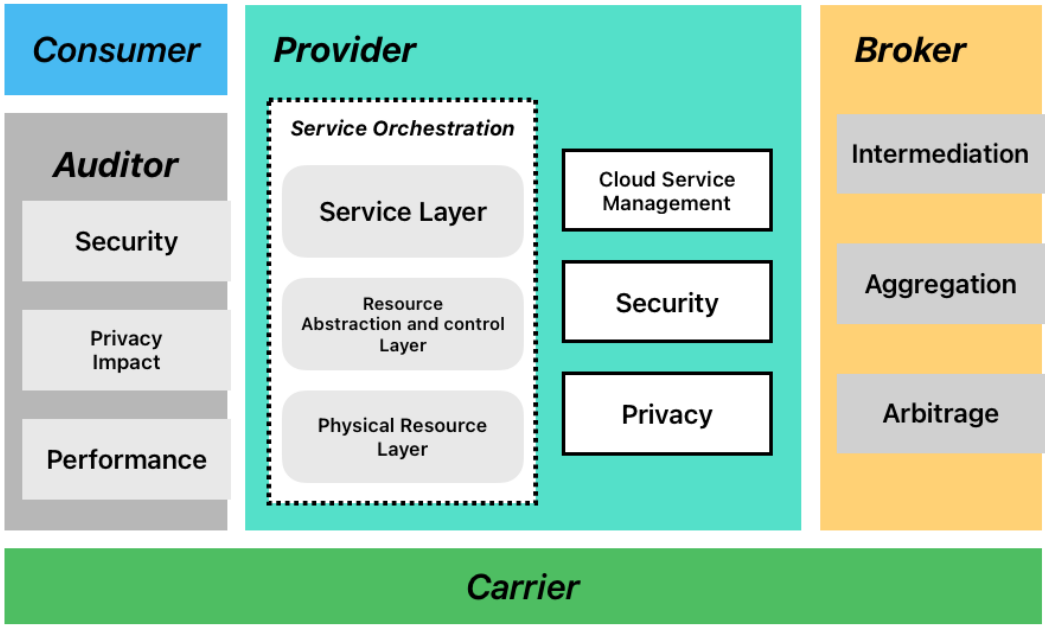
- **Agrupamento de recursos**: É a característica fundamental da computação em nuvem, consiste na abstração de recursos de infraestrutura e sua alocação à diferentes consumidores;
- **Elasticidade**: Um provedor de serviços em nuvem tem a capacidade de suportar diferentes cargas de trabalho e de fornecer recursos de infraestrutura que se adéquam as diversas aplicações;
- **Acesso amplo**: Todos os recursos da nuvem estão disponíveis na rede, sem a necessidade de acesso físico direto a infraestrutura;
- **Serviços sob demanda**: Os consumidores da nuvem gerenciam a utilização dos seus próprios serviços, sem a necessidade de terceiros;
- **Serviços mensuráveis**: Todo recurso utilizado pode ser medido e cobrado proporcionalmente ao uso.

3.2.1 Arquitetura da computação em nuvem

Segundo a arquitetura de referência da computação em nuvem proposta pelo (NIST, 2011), os agentes que a compõem são: o consumidor (*Consumer*), o auditor, o provedor (*Provider*), o *broker* e a transportadora (*Carrier*), ilustrados na Figura 4. Cada ator nessa arquitetura é uma entidade que participa de uma transação ou processo, além de realizar tarefas na computação em nuvem. O consumidor é a parte mais interessada nos serviços

em nuvem. Ele pode representar uma pessoa ou organização que mantém uma relação de negócios com o provedor e utiliza seus serviços. O provedor é a entidade responsável por disponibilizar os serviços às partes interessadas; é quem gerencia a infraestrutura de computação necessária para atender os serviços, executa os softwares da nuvem e os torna acessíveis ao consumidor a partir da rede. O auditor é a parte do sistema que realiza análises independentes de controles dos serviços em nuvem, com o objetivo de gerar relatórios de desempenho, controles de segurança e de privacidade, por exemplo.

Figura 4 – Modelo de referência da Computação em Nuvem.



Fonte: Produzido pelo autor.

O *broker* gerencia o uso, a performance, e a entrega dos serviços na nuvem aos consumidores, além de intermediar a relação entre o provedor e consumidor. Já a transportadora atua como um intermediário realizando a conexão e o transporte de dados e serviços entre o consumidor e o provedor. Essa conexão pode ser realizada por meio de redes cabeadas de internet ou até mesmo por serviços de telecomunicações.

A computação em nuvem é frequentemente categorizada em três principais modelos de serviço:

- **Infraestrutura como Serviço (*Infrastructure as a Service* - IaaS)**: Nesse modelo os recursos de computação fundamentais, tais como, processamento, armazenamento e rede, estão disponíveis ao consumidor para implementação de seu próprio software, desde sistemas operacionais a aplicações. Alguns exemplos de serviços nesse modelo são: *Backup* e recuperação de dados, armazenamento em massa escalável e CDN[3] para sistemas web;

[3] *Content Delivery Network - CDN* (Rede de Distribuição de Conteúdo). Trata-se de uma rede de

- **Plataforma como Serviço (*Platform as a Service* - PaaS)**: Disponibiliza plataformas para desenvolvimento de aplicações utilizando linguagens de programação e ferramentas nativas do provedor. Por esse modelo, o consumidor não tem a liberdade de configuração ou controle dos recursos de infraestrutura de computação. Alguns exemplos são: plataforma para criação de *dashboards* e análise de dados, serviços de armazenamento non-SQL e plataformas escalável para ciclos de teste de aplicações;
- **Software como Serviço (*Software as a Service* - SaaS)**: Apenas executa aplicações desenvolvidas e fornecidas pelo provedor de serviços. Essas aplicações podem ser acessadas por um navegador web, um aplicativo para dispositivos móveis ou um *software* cliente. Aplicativo de e-mail, gerenciador de documentos e redes sociais são exemplos de implementações nesse modelo.

O futuro da computação em nuvem promete avanços contínuos em termos de integração com tecnologias emergentes. A adoção crescente de inteligência artificial, *machine learning* e internet das coisas (IoT) está expandindo as possibilidades da nuvem, permitindo soluções mais inteligentes e conectadas (TSURUOKA, 2016). Além disso, a computação em borda (*Edge Computing*) está emergindo como uma extensão da nuvem, trazendo recursos computacionais mais próximos dos dispositivos finais para reduzir a latência e aumentar a eficiência.

3.2.2 *Amazon Web Services* (AWS)

A *Amazon Web Services* é uma plataforma de serviços web na nuvem que oferece soluções de computação, armazenamento, e rede em diferentes camadas de abstração. Nos últimos anos, emergiu como uma das plataformas de computação em nuvem mais influentes e dominantes do mundo, liderando o mercado global de infraestrutura em nuvem em 2024 com cerca de 31% de participação (THE REGISTER, 2024).

Fundado pela Amazon em 2006, o AWS começou como uma solução interna para gerenciar sua própria infraestrutura de tecnologia. Desde então, evoluiu para se tornar um serviço globalmente acessível que revolucionou a forma como empresas de todos os tamanhos operam digitalmente. O consumidor pode optar por utilizar *data centers* distribuídos pelos Estados Unidos, Europa, Ásia e América do Sul.

O termo *web service* indica que os serviços disponíveis pela Amazon em sua plataforma são acessíveis e controlados por meio de uma API[4], a partir de sua interface *web* por quatro diferentes maneiras: utilizando um console de gerenciamento desenvolvido para navegadores web; por meio de uma interface de linha de comando; por meio de

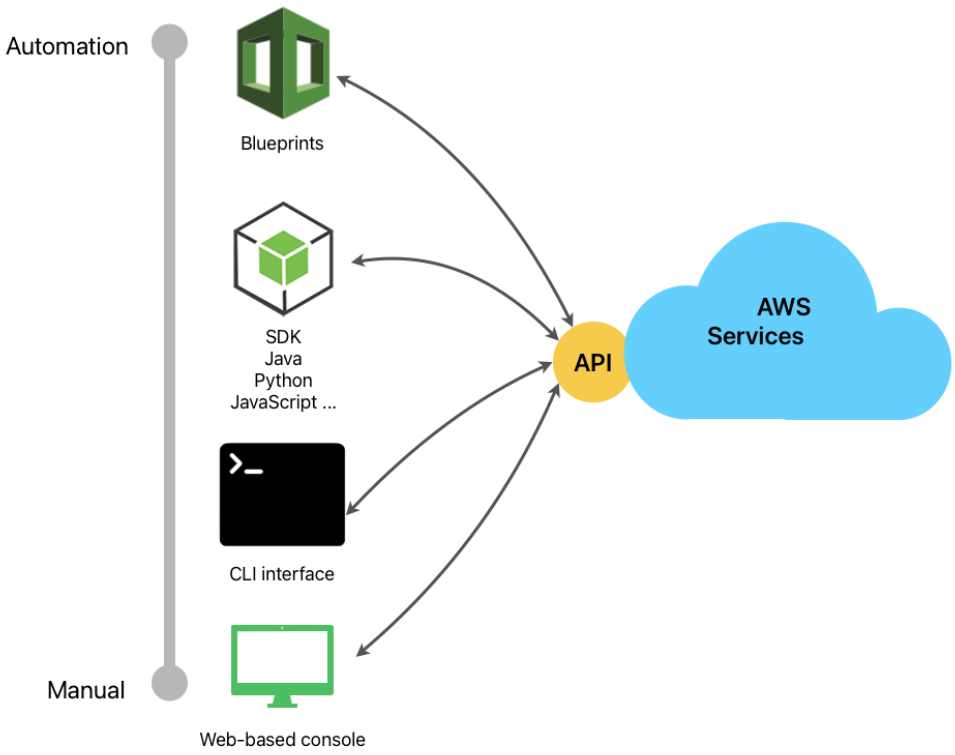
servidores distribuídos geograficamente que trabalham juntos para fornecer conteúdo da internet de forma rápida e eficiente aos usuários finais.

[4] *Application Programming Interface - API* (Interface de Programação de Aplicações). É um conjunto de regras e definições que permite que diferentes *softwares* se comuniquem entre si.

descrições no formato de *blueprints*; com SDKs desenvolvidos para linguagens como Java, Python, PHP, entre outras (WITTIG; WITTIG, 2016).

Ilustra-se na Figura 5 as maneiras de acesso à API da AWS e o grau de automatização delas.

Figura 5 – Ferramentas para interação com a API da AWS.



Fonte: Produzido pelo autor.

Mostra-se na Tabela 3 algumas soluções disponíveis na plataforma da AWS relacionadas a computação, armazenamento, banco de dados, redes, análise e aprendizado de máquina.

A *Amazon Web Services* revolucionou a maneira como usuários de todos os portes concebem e implementam suas infraestruturas tecnológicas. Com uma oferta diversificada de serviços e uma plataforma que continua a inovar e expandir, a AWS se estabeleceu como uma força dominante na computação em nuvem, ajudando empresas a serem mais ágeis, inovadoras e eficientes. Algumas de suas vantagens são: **Alta escalabilidade** - A AWS permite que consumidores escalem seus recursos de computação e armazenamento conforme necessário, seja verticalmente ou horizontalmente, para atender à demanda variável; **Flexibilidade** - Com uma ampla gama de serviços e ferramentas, a AWS oferece flexibilidade para escolher as tecnologias e arquiteturas que melhor se adaptam às necessidades específicas de negócios e aplicações; **Custo Efetivo** - O modelo de pagamento conforme o uso da AWS significa que os usuários só pagam pelos recursos que realmente

Tabela 3 – Exemplos de serviços oferecidos pela AWS.

Serviço	**Descrição**
EC2 (*Elastic Compute Cloud*)	Serviço que fornece capacidade de computação escalável na nuvem. Os usuários podem lançar instâncias de servidores virtuais conforme necessário, pagando apenas pelo tempo de uso.
Lambda	Serviço de computação sem servidor que permite executar código em resposta a eventos e alocar automaticamente os recursos de computação necessários.
S3 (*Simple Storage Service*)	Serviço de armazenamento de objetos que oferece escalabilidade, alta disponibilidade e segurança para armazenar e recuperar qualquer quantidade de dados de qualquer lugar na web.
RDS (*Relational Database Service*)	Serviço gerenciado de banco de dados relacional que suporta vários mecanismos de banco de dados, incluindo MySQL, PostgreSQL, MariaDB, Oracle e SQL Server.
VPC (*Virtual Private Cloud*)	Permite provisionar uma seção logicamente isolada da nuvem AWS onde é possível lançar recursos AWS em uma rede virtual definida pelo usuário.
SageMaker	Plataforma que permite construir, treinar e implantar modelos de aprendizado de máquina em escala.

Fonte: Produzida pelo autor.

utilizam, sem a necessidade de investimentos iniciais pesados em hardware; **Segurança** - A AWS proporciona uma infraestrutura segura com robustos mecanismos de proteção de dados, conformidade com diversas normas regulatórias e uma arquitetura resiliente a falhas; **Alta disponibilidade** - Garantia de ininterruptibilidade dos serviços, a medida que implementa técnicas e recursos avançados de redundância de operação.

3.2.2.1 *AWS IoT*

A *AWS IoT* é uma plataforma que fornece serviços em nuvem que permitem conectar dispositivos de internet das coisas (IoT) a outros dispositivos e a outros serviços do ecossistema da AWS (AWS IOT, 2024). A plataforma pode ser acessada por meio de:

- **SDKs para dispositivos**: Dispositivos de *hardware* IoT que executam SDKs

desenvolvidos para as linguagens C, C++, Python, Javascript, Java, Android e iOS, e que enviam e recebem mensagens por meio dos protocolos MQTT, MQTT sobre o Websocket seguro, HTTPS e LoraWAN;

- **LoraWAN**: Uma API especificamente desenvolvida para dispositivos *Long Range* WAN (LoraWAN);
- **Interface de comando de linha**: Comandos específicos numa aplicação que pode executar em dispositivos Windows, MacOS e linux;
- **HTTP e HTTPS**: Uma API dedicada para requisições HTTP e HTTPS;
- **SDKs da AWS**: Utilização de APIs para linguagens suportadas pela AWS como Go, Kotlin, PHP, entre outras;
- **Console**: Um console que executa em navegadores Web, no qual é possível criar, configurar e gerenciar os dispositivos IoT, certificados de segurança, regras, políticas, entre outros.

Fornecendo suporte desde os dispositivos de borda que interagem com o mundo físico até a manipulação dos dados gerados por esses dispositivos, o AWS IoT disponibiliza uma arquitetura de serviços de software, controle e operação dos dados, gerando uma solução completa de IoT. Apresenta-se na Figura 6 a representação dos serviços disponíveis dessa arquitetura IoT AWS, as quais estão distribuídos em três categorias gerais: Dados, Controle e software dos dispositivos.

Na base da arquitetura está o software dos dispositivos de borda que irão interagir com os serviços da nuvem. Para isso, a AWS disponibiliza SDKs em diversas linguagens que podem ser implementados em várias plataformas de *hardware*, tais como Raspberry Pi, Arduino, ESP32, LoraWAN, e mais. Como solução proprietária, a AWS disponibiliza o **Greengrass** que é um recurso de *software* que executa no sistema operacional FreeRTOS em dispositivos de borda. Esse recurso transforma os dispositivos em extensões da nuvem AWS IoT, sendo possível coletar e analisar localmente dados gerados por equipamentos IoT, gerar predições baseadas em aprendizado de máquina, reagir autonomamente a disparos de eventos e se conectar aos serviços da nuvem.

Os dados gerados pelos equipamentos podem ser manipulados na plataforma AWS IoT pelos seguintes serviços disponíveis:

- **Kinesis**: Com o Kinesis é possível armazenar, criptografar e indexar na nuvem grandes fluxos de dados como por exemplo transmissões de video em tempo real, arquivos de áudio, grandes volumes de dados de sensores a altas taxas de amostragem, etc;

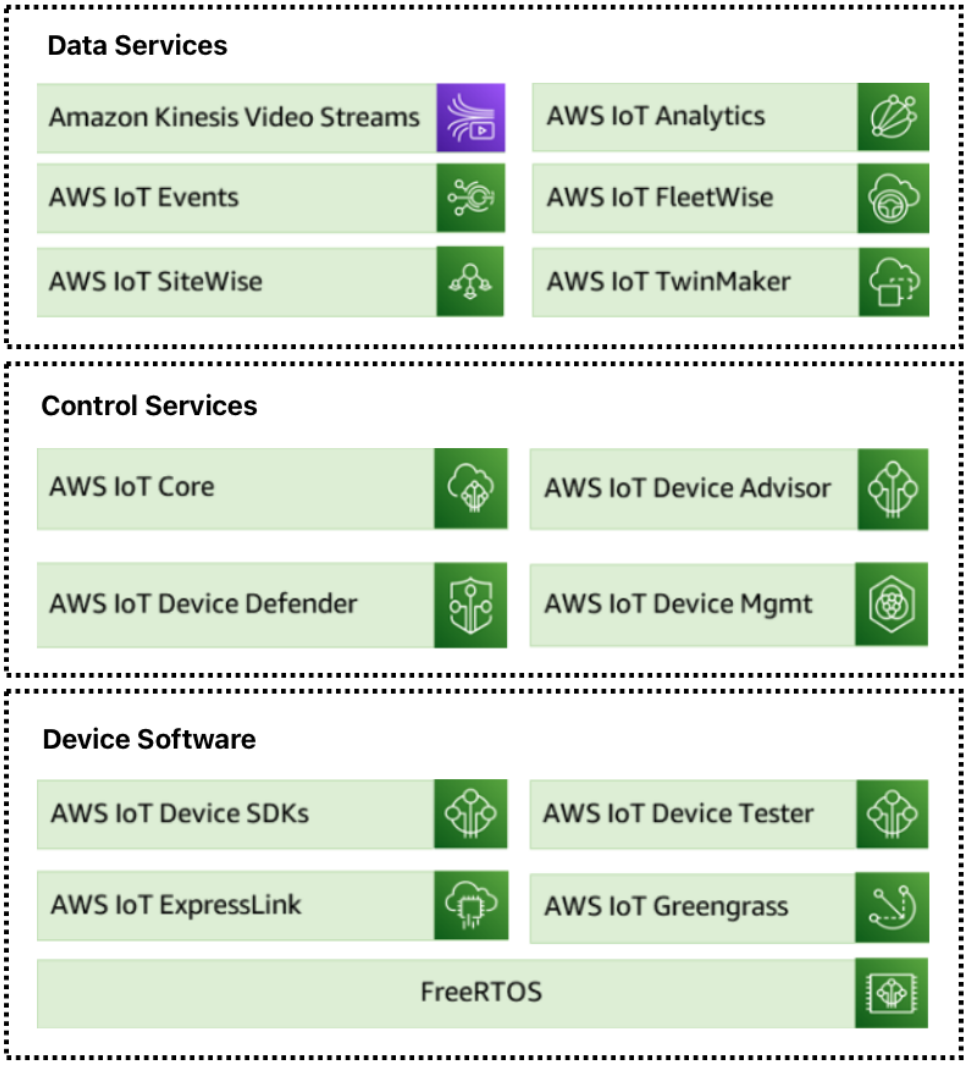
- ***Analytics***: Possibilita executar análises sofisticadas em grandes volumes de dados gerados pelos dispositivos IoT. Este serviço automatiza etapas de filtragem e eventuais transformações e enriquecimento dos dados para análise (AWS IOT ANALYTICS, 2024);
- ***Events***: Serviço responsável por detectar e responder eventos gerados por sensores ou por outras aplicações, com base na detecção de padrões nos dados gerados pelos dispositivos;
- ***FleetWise***: Serviço gerenciado que facilita a coleta, transformação e transferência de dados de veículos conectados para a nuvem. Este serviço permite a captura e análise de dados de veículos em tempo real para melhorar a segurança, a manutenção preditiva, a eficiência operacional e o desenvolvimento de novos serviços e funcionalidades;
- ***SiteWise***: Realiza a coleta, armazenamento, organização e monitoramento de dados industriais em larga escala. Este serviço é projetado para auxiliar as empresas a obter melhor uso dos dados de seus equipamentos industriais e sistemas de produção;
- ***TwinMaker***: Permite a criação e o uso de gêmeos digitais de sistemas físicos, auxiliando na modelagem, monitoramento e otimização de ativos físicos e processos em tempo real.

Os dispositivos corretamente configurados com a API da *AWS IoT* podem interagir com outras aplicações da nuvem e com outros dispositivos por meio do **IoT Core**, um serviço que oferece uma infraestrutura robusta para gerenciar, processar e interagir com dados gerados por dispositivos conectados, além de fornecer uma comunicação segura e bidirecional com outros serviços da AWS por meio dos protocolos MQTT, HTTPS e LoraWAN.

Apresenta-se na Figura 7 o diagrama de blocos com os principais serviços disponíveis no *IoT Core* e as suas relações de comunicação e execução. A integração dos dispositivos IoT com os outros serviços da nuvem inicia-se na conexão com o **broker**, um serviço que permite que dispositivos e aplicativos publiquem mensagens em tópicos e assinem tópicos para receber mensagens. Isso facilita a comunicação em tempo real entre dispositivos e entre dispositivos e aplicativos na nuvem. As mensagens são enviadas ao *broker* por meio dos protocolos MQTT, HTTP, ou MQTT sobre o *Websocket*.

As mensagens publicadas no *broker* são distribuídas por ele a outros dispositivos que possuem assinaturas em tópicos e aos serviços de **Device Shadows** e **Rule Engine**. O *Device Shadows* é um serviço que oferece uma representação virtual do estado de um dispositivo IoT. Isso permite que aplicações e dispositivos interajam e sincronizem estados, mesmo quando os dispositivos estão desligados. A informação do estado atual dos dispositivos é armazenada em arquivos no formato JSON. Já o *Rule Engine* é um

Figura 6 – Arquitetura da plataforma IoT da AWS.

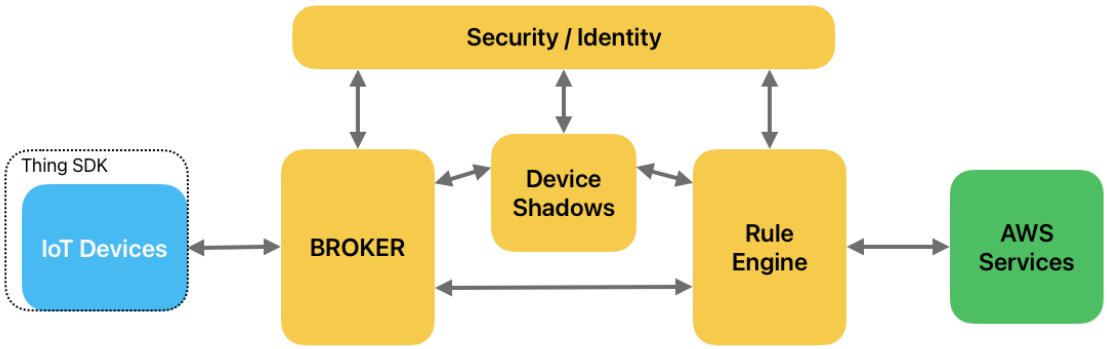


Fonte: Produzido pelo autor.

serviço que processa as mensagens enviadas pelo *broker* dos dispositivos conectados e executa ações baseadas nessas mensagens. Ele facilita a criação de regras que transformam e roteiam dados de dispositivos para outros serviços da AWS ou para sistemas externos.

O serviço de Segurança e identidade do *IoT Core* garante uma conexão segura dos dispositivos IoT à nuvem. Ele suporta autenticação mútua e criptografia ponta a ponta para garantir que os dados transferidos sejam protegidos. A autenticação dos dispositivos é realizada por meio de certificado X.509 e o gerenciamento das políticas de acesso utiliza o IAM (*Identity and Access Management*) para controlar o acesso a recursos IoT. As políticas definem permissões detalhadas para ações que dispositivos e usuários podem executar.

Figura 7 – Serviços executados no AWS IoT Core.



Fonte: Produzido pelo autor.

3.3 BMS EM NUVEM

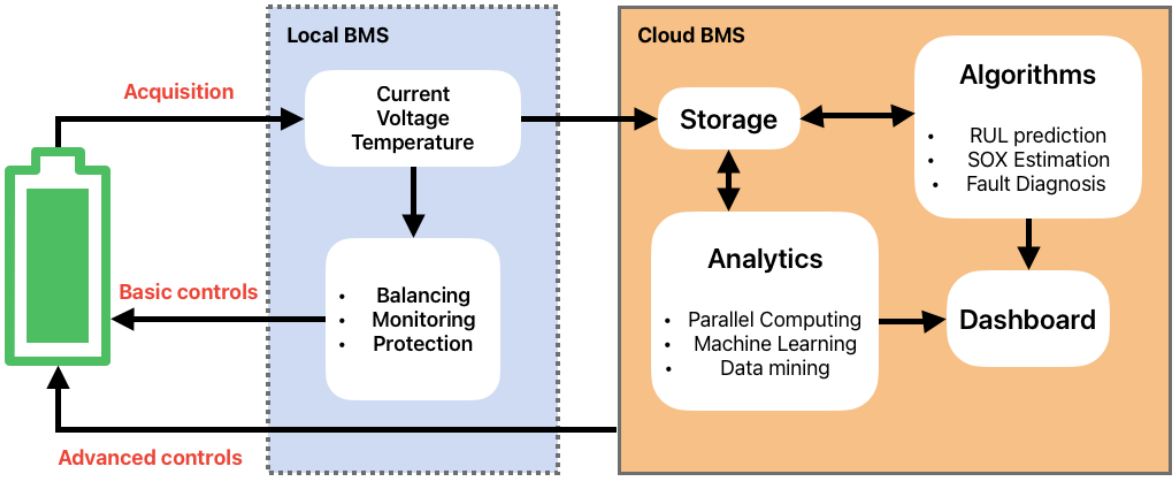
Como visto na seção 3.1, uma das principais funções do BMS é a estimação de parâmetros da bateria, tais como o SOC e o SOH, para a correta operação das mesmas, fornecendo informações acerca do nível de carga e nível de envelhecimento. Devido a facilidade de implementação e baixo esforço computacional, algoritmos como o *Coulomb Counting* são embarcados nos BMS tradicionais, porém devido a falta de precisão por eventuais acúmulos de erros de medição, esses métodos têm dado espaço a algoritmos mais complexos e precisos que requerem maiores poderes computacional e capacidade de armazenamento.

Para contornar essas limitações o BMS tradicional pode integrar as tecnologias de internet das coisas e computação em nuvem na sua arquitetura (WEIHAN et al., 2020), com isso, passa a disponibilizar de alto poder de processamento, armazenamento de dados em larga escala e alta confiabilidade de sistema. Em termos de *hardware*, o BMS em nuvem pode reduzir alguns componentes de computação local, resultando em dispositivos mais compactos e de menor custo(TRAN et al., 2022).

Apresenta-se na Figura 8 uma implementação de um BMS em nuvem. Essa arquitetura é formada por três componentes: o BMS local, responsável pelas tarefas de balanceamento e proteção primária, além da aquisição da tensão, corrente e temperatura da(s) bateria(s); um componente de *software* implementado em nuvem que realiza o armazenamento dos dados, a análise desses dados, execução de algoritmos avançados de estimação de estados, diagnóstico de falhas e predições, além de fornecer visualização dessas informações em tempo real; e um componente de interligação entre os outros dois, o *gateway*, que fornece os dados obtidos pelo BMS local para a nuvem.

Pela análise da arquitetura apresentada do BMS em nuvem, é possível notar a adição de alguns componentes se comparado a arquitetura de BMS tradicional. Tais componentes são fundamentais à operação em nuvem, sendo eles:

Figura 8 – Arquitetura de cloud BMS.



Fonte: Produzido pelo autor.

- **Componente IoT**: Necessário para comunicação e transmissão de dados das baterias. Necessita de conexão estável e confiável com a internet, como por exemplo o 5G. Alguns protocolos utilizados para comunicação com a nuvem são o TCP/IP e o MQTT;
- **Infraestrutura de nuvem**: Normalmente a infraestrutura de nuvem consiste de um *data logger* e um banco de dados. O *data logger* armazena uma grande quantidade de informações do BMS e as transfere ao banco de dados na nuvem. Este por sua vez armazena os dados de forma estruturada e os disponibiliza aos serviços de análise e predição;
- **API da nuvem**: Funciona como uma comunicação entre o banco de dados e as ferramentas de análise de informações e algoritmos implementados. Pode ser desenvolvida com linguagens como Python e Matlab;
- **Interface do usuário**: Necessária para que o usuário do sistema visualize o estado e operação das baterias em tempo real, assim como o acompanhamento de dados históricos de falhas e alarmes.

Enquanto um BMS clássico oferece funcionalidades básicas de gerenciamento de bateria localmente, um BMS baseado em nuvem oferece maior acessibilidade, escalabilidade e recursos avançados de análise e otimização, tornando-o uma opção mais eficiente e versátil para o gerenciamento de baterias em diversas aplicações. Porém, existem algumas desvantagem nesse sistema, como por exemplo a dependência de uma conexão rápida, estável e confiável com a internet, o que limita o desempenho do dispositivo, e os custos de operação e manutenção da plataforma na nuvem, a depender da disponibilidade e

uso dos serviços de análise e algoritmos, como também da frequência e quantidade de armazenamento realizado.

3.4 REDES NEURAIS ARTIFICIAIS

Nesta seção são apresentadas as definições básicas de uma rede neural artificial, suas principais arquiteturas, as principais funções de ativação, os métodos que são utilizados para o seu treino com os algoritmos de aprendizagem e algumas métricas de avaliação de desempenho dos modelos treinados.

Uma rede neural artificial nada mais é que um sistema de processamento de informações com características baseadas em redes neurais biológicas (FAUSETT, 1994). Por meio da generalização de modelos matemáticos da cognição humana, as redes neurais artificiais foram desenvolvidas baseadas nas seguintes definições:

- O processamento da informação ocorre em elementos simples denominados neurônios;
- Sinais são transmitidos entre neurônios por meio de linhas de comunicação;
- Cada linha de comunicação possui um peso associado que multiplica o sinal transmitido;
- Cada neurônio aplica uma função de ativação à sua entrada para determinar o seu sinal de saída.

Portanto, uma rede neural artificial é caracterizada pelo seu padrão de conexão entre os neurônios (arquitetura), a forma como ajusta seus pesos entre as conexões (algoritmo de aprendizagem) e função de ativação que utiliza. Com auxílio da Figura 9 é possível exemplificar uma rede simples contendo três informações de entrada, um neurônio de camada oculta e dois neurônios de camada de saída.

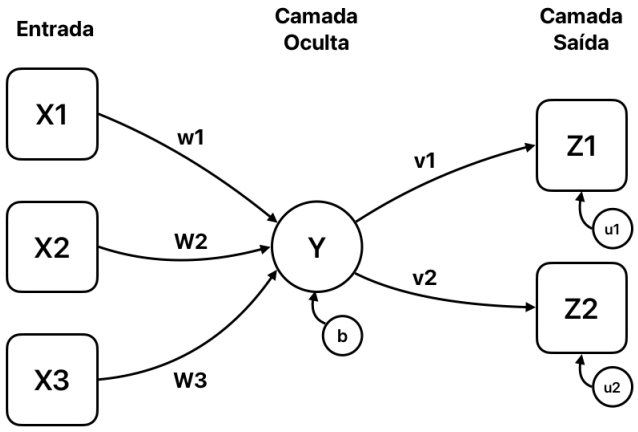
As saídas dessa rede são dadas por:

$$Z_1 = f(Y) \times v_1 + u_1 = f([X_1 \times w1 + X_2 \times w2 + X_3 \times w3] + b) \times v_1 + u_1 \tag{3.4}$$

$$Z_2 = f(Y) \times v_2 + u_2 = f([X_1 \times w1 + X_2 \times w2 + X_3 \times w3] + b) \times v_2 + u_2 \tag{3.5}$$

Em que $f(Y)$ é alguma função de ativação (podendo ser não linear), que opera sobre a saída Y.

Figura 9 – Exemplo de uma rede neural artificial.



Fonte: Produzido pelo autor.

3.4.1 Arquiteturas de redes neurais

As arquiteturas de redes neurais variam de acordo com a estrutura das conexões e o tipo de processamento realizado. A escolha da arquitetura depende da complexidade do problema e da natureza dos dados. As principais Redes Neurais são: Perceptron Multicamadas, Convolucionais e Recorrentes.

- **Perceptron Multicamadas (MLP - *Multi-Layer Perceptron*)**: Nessa arquitetura, a rede é composta por camadas de entrada, ocultas e de saída que utilizam conexões diretas (*feedforward*), onde os dados se propagam apenas em uma direção. O aprendizado ocorre de forma supervisionada por meio do algoritmo de retropropagação do erro (*Backpropagation*). É amplamente utilizada em problemas de classificação, regressão e aprendizado de padrões complexos;

- **Redes Neurais Convolucionais (CNNs - *Convolutional Neural Networks*)**: São estruturadas em camadas convolucionais que extraem características dos dados como bordas, texturas, formas; camadas de subamostragem (*pooling*) que reduzem a dimensão dos mapas de características, mantendo as informações mais importantes dos dados de entrada; e as camadas totalmente conectadas (*Fully Connected*) que conectam os neurônios da última camada convolucional a uma camada densa, semelhante a uma MLP. São altamente eficientes em reconhecimento de padrões em imagens e visão computacional (LECUN et al., 1998);

- **Redes Neurais Recorrentes (RNNs - *Recurrent Neural Networks*)**: Possuem uma estrutura diferenciada das redes neurais anteriores, pois incluem conexões recorrentes que permitem o processamento de dados sequenciais. São formadas por camada de entrada, camada oculta recorrente e camada de saída. A camada de entrada recebe os dados sequenciais dos quais cada elemento da sequência é processado

em um instante de tempo t. Já na camada oculta recorrente, cada neurônio recebe a entrada do instante t e o estado oculto anterior que contém a memória do passado, para que a partir disso um novo estado oculto possa ser calculado e transmitido para o próximo instante de tempo. Na camada de saída é produzida a saída desejada, que pode ser um valor único (previsão futura) ou uma sequência completa. Essas redes são muito utilizadas no processamento de linguagem natural (tradução automática, geração de texto, *chatbots*), reconhecimento de fala e áudio e análise de vídeo.

3.4.2 Funções de ativação

As funções de ativação definem como os neurônios processam os sinais recebidos. São funções matemáticas utilizadas para introduzir não-linearidade nos modelos. Elas contribuem no aprendizado de padrões complexos, transformando a soma ponderada das entradas dos neurônios em uma saída significativa. As principais são: Sigmoide, ReLU, Tanh e a Softmax.

A função Sigmoide transforma um valor real em um intervalo entre 0 e 1 o que a torna útil para problemas de probabilidade e classificação binária. Ela permite o uso do gradiente descendente para treinamento de redes neurais por ser totalmente diferenciável. Entretanto, quando os valores de entrada são muito grandes ou muito pequenos, a derivada da sigmoide se aproxima de zero, o que dificulta o aprendizado durante o treinamento da rede neural (*vanishing gradient*).

A função Tanh (Tangente hiperbólica) tem seu comportamento semelhante a Sigmoide, com a diferença que sua saída retorna valores entre -1 e 1, tornando-a mais útil para representar dados simétricos. Assim como na sigmoide, é totalmente diferenciável, podendo ser utilizada em gradiente descendente para otimização. Porém também possui a limitação do desaparecimento do gradiente quando os valores de entrada são muito grandes ou muito pequenos.

A função ReLU (*Rectified Linear Unit*) possui uma ampla utilização em redes neurais devido a sua simplicidade e eficiência computacional (NAIR; HILTON, 2010). Ela possui a forma de uma reta com inclinação 1 para valores de entrada positivos e zero para valores negativos. Ao contrário da sigmoide e da tangente hiperbólica, a ReLU não tem limites superiores, o que ajuda a mitigar o problema da desaparecimento do gradiente em redes profundas. No entanto, uma limitação conhecida é que, se os pesos da rede forem muito grandes ou muito pequenos, a função pode gerar valores negativos constantes para alguns neurônios, fazendo com que esses neurônios percam a capacidade de aprendizado. Essa situação é conhecida como o problema do "neurônio morto", onde a saída do neurônio é sempre igual a zero.

Apresenta-se na Tabela 4 um resumo comparativo das características e limitações das principais funções de ativação utilizadas nas redes neurais.

Tabela 4 – Tabela comparativa das principais funções de ativação.

Função	**Equação**	**Aplicação**	***Limitações***
Sigmoide	$\sigma(x) = \frac{1}{1+\exp^{-x}}$ Saída: intervalo 0 a 1	Classificação binária e modelagem de probabilidades	Desaparecimento do gradiente
Tanh	$\tanh(x) = \frac{\exp^{x} - \exp^{-x}}{\exp^{x} + \exp^{-x}}$ Saída: intervalo -1 a 1	Modelagem simétrica	Desaparecimento do gradiente
ReLU	$ReLU(x) = max(0, x)$ Saída: $0, x \leq 0$; $x, x > 0$	Modelos de regressão e classificação	Neurônio morto

Fonte: Produzido pelo autor.

3.4.3 Treinamento do modelo

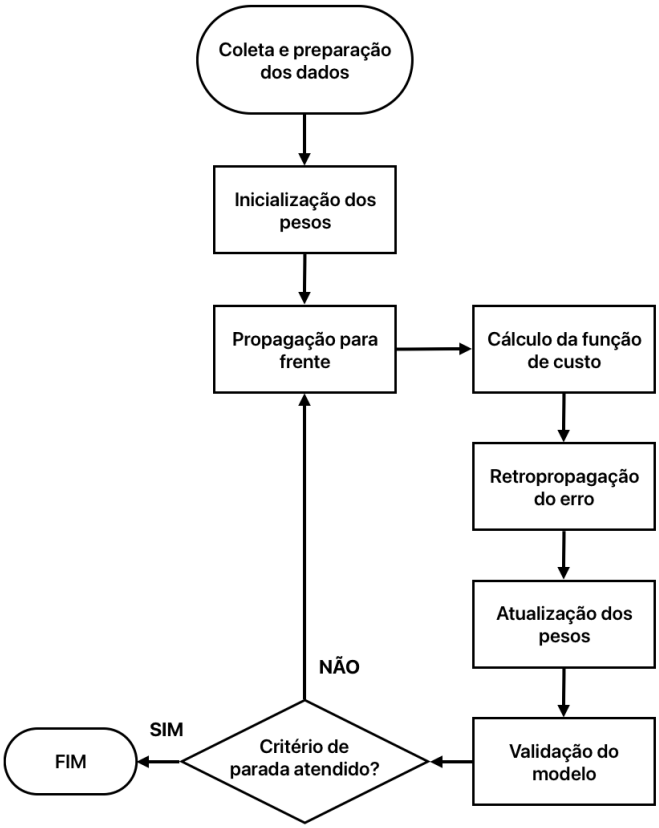
O treinamento de uma rede neural é um processo iterativo de ajuste dos pesos das conexões entre neurônios para que a rede consiga realizar uma determinada tarefa com precisão (HAYKIN, 2009). Esse procedimento pode ser divididos em algumas etapas que vão desde a coleta e preparação dos dados para treino até a validação do modelo por algum critério. As atividades do fluxograma da Figura 10 que compõem o processo de treino da rede são:

- **Coleta e preparação dos dados:** Nessa etapa, os dados para treino podem ser obtidos por diversas fontes tais como bancos de dados, web, sensores, etc. Em seguida esses dados podem ser submetidos a subetapas de pré-processamento como remoção de dados inconsistentes ou incompletos, normalização/padronização dos valores, interpolações, remoção de *outliers*, entre outras. Com os dados já tratados, é realizada a separação do conjunto para treino, validação e teste (por exemplo: 70%-20%-10%). Também nessa etapa é definida a arquitetura da rede a ser treinada, determinação do número de camadas e neurônios, escolha das funções de ativação, definição das conexões entre os neurônios e configuração dos hiperparâmetros tais como taxa de aprendizado, número de épocas de treinamento e algoritmo de otimização, por exemplo;

- **Inicialização dos pesos:** Antes do treinamento começar, os pesos e os *bias* das conexões da rede são inicializados. A maneira como são iniciados pode impactar significativamente o desempenho do aprendizado. Uma estratégia comum utilizada nessa etapa é a inicialização aleatória na qual os pesos são inicializados com valores pequenos e aleatórios para evitar simetria entre os neurônios;

- **Propagação para frente (*Forward Propagation*):** O início do treinamento

Figura 10 – Etapas de treino de uma rede neural.



Fonte: Produzido pelo autor.

propriamente dito começa nessa etapa. Neste estágio, os dados de entrada passam pela rede neural, camada por camada, até produzir uma saída. Cada neurônio recebe os sinais de entrada, multiplica pelos pesos e adiciona um *bias*. Esse valor é aplicado a uma função de ativação, que adiciona não-linearidade. Após percorrer todas as camadas, a rede gera uma saída, que é comparada com o valor verdadeiro esperado;

- **Cálculo da função de custo:** A função de custo mede a diferença entre a predição gerada pela rede e o valor real esperado. Uma função quadrática muito utilizada nessa etapa é o erro quadrático médio (MSE - *Mean Squared Error*). O principal objetivo do treinamento da rede é a minimização dessa função;

- **Retropropagação do erro:** A retropropagação é o processo de cálculo dos gradientes da função de custo em relação a cada peso da rede. O erro é obtido a partir da derivada da função de custo em relação à saída da rede. Utilizando a regra da cadeia, os gradientes são propagados para cada camada anterior, ajustando os pesos da rede;

- **Atualização dos pesos:** Nessa etapa os pesos da rede são ajustados para minimizar a função de custo selecionada para o treino. A regra básica para atualização dos pesos é dada pela Equação 3.6:

$$W^l = W^l + \eta \frac{\partial L}{\partial W^l}; \tag{3.6}$$

sendo l a camada, η a taxa de aprendizado e $\frac{\partial L}{\partial W^l}$ é o gradiente dos pesos.

Algoritmos como o Adam (*Adaptive Moment Estimation*), que é um método eficiente computacionalmente, requer um baixo consumo de memória, invariante ao redimensionamento diagonal dos gradientes e bem adequado para problemas grandes em termos de dados/parâmetros (KINGMA; BA, 2015), são utilizados para ajustar dinamicamente a taxa de aprendizado para cada peso da rede.

- **Validação do modelo:** Após os ajustes e atualização dos pesos, o modelo é avaliado para verificar sua capacidade de generalização. O conjunto separado de dados para teste é usado para medir o desempenho e ajustar os hiperparâmetros caso necessário. Nessa etapa, métricas de avaliação são utilizadas como a acurácia que mede a proporção de acertos e a precisão e *recall*;
- **Critério de parada:** Se algum critério predefinido de parada como número máximo de épocas de treino ou um valor mínimo de função de custo for atendido, o treinamento da rede é encerrado.

Uma única passagem pelos dados não é suficiente para que a rede aprenda corretamente. O treinamento é repetido por várias épocas, nas quais a cada época, a rede ajusta seus pesos para reduzir a perda e o desempenho melhora progressivamente até atingir um limiar.

Com o modelo treinado, utilizando bibliotecas como TensorFlow (GOOGLE, 2015), pode ser realizada a exportação do modelo em formato .h5, onde este refere-se a arquivos no formato HDF5 (*Hierarchical Data Format version* 5), um formato amplamente utilizado para armazenar e organizar grandes quantidades de dados científicos e numéricos de forma eficiente. Ou até mesmo, pode ser integrado em API's, aplicativos móveis ou em sistemas embarcados.

3.4.4 Métricas de avaliação do modelo

Métricas de avaliação de modelos são medidas quantitativas usadas para avaliar o desempenho do modelo em relação a um conjunto de dados de validação ou teste. Essas métricas ajudam a entender quão bem o modelo está generalizando e onde ele pode precisar de melhorias (HASTIE; TIBSHIRANI; FRIEDMAN, 2008).

- **MSE:** O *Mean Squared Error* (MSE), ou Erro Quadrático Médio, é uma métrica usada para avaliar a qualidade de modelos de regressão. Ele mede a média dos erros ao quadrado entre os valores reais e os valores previstos pelo modelo. Este

índice diferencia melhor modelos porque penaliza erros maiores devido a elevação ao quadrado (sensível a *outliers*). É muito utilizado em problemas de regressão e otimização de redes neurais. Sua expressão matemática está apresentada com a Equação 3.7:

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \widehat{y}_i)^2; \tag{3.7}$$

sendo y_i os valores reais observados, $\widehat{y}_i$ os valores estimados pelo modelo e n o número total de amostras.

A interpretação do resultado que essa métrica retorna é de que quanto menor for seu valor, melhor será a precisão do modelo. Porém essa interpretação se torna difícil na medida que este índice não está na mesma escala dos dados originais, mas sim na unidade ao quadrado desses dados.

- **MAE:** O *Mean Absolute Error* (MAE), ou Erro Médio Absoluto, é outra métrica usada para medir o desempenho de modelos de regressão. Ele calcula a média das diferenças absolutas entre os valores reais e os valores previstos pelo modelo, porém é menos sensível a *outliers* comparado ao MSE pois não eleva ao quadrado os dados. É definido matematicamente pela Equação 3.8

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|y_i - \widehat{y}_i|; \tag{3.8}$$

sendo y_i os valores reais observados, $\widehat{y}_i$ os valores estimados pelo modelo e n o número total de amostras.

Assim como o MSE, quanto menor for o resultado MAE, melhor será a precisão do modelo. Já sua interpretação se torna mais direta pois representa o erro médio na mesma unidade dos dados originais.

- **MAPE:** O *Mean Absolute Percentage Error* (MAPE), ou Erro Percentual Absoluto Médio, é uma métrica usada para avaliar a precisão de modelos de previsão. Ele expressa o erro médio em termos percentuais, tornando a interpretação mais intuitiva do que métricas como MSE ou MAE. É definido pela Equação 3.9:

$$MAPE = \frac{1}{n}\sum_{i=1}^{n}\left|\frac{y_i - \widehat{y}_i}{y_i}\right| \times 100; \tag{3.9}$$

sendo y_i os valores reais observados, $\widehat{y}_i$ os valores estimados pelo modelo e n o numero total de amostras.

Percebe-se pela Equação 3.9 que o uso dessa métrica não se aplica em amostras que contenham valores reais $y_i = 0$, pois causaria uma divisão indefinida.

A interpretação do valor MAPE é comumente associada a faixas subjetivas de confiabilidade, sendo um modelo que alcance um MAPE $< 10\%$ considerado excelente, $10\% \leq$ MAPE $< 20\%$ considerado bom, $20\% \leq$ MAPE $< 50\%$ modelo razoável e um MAPE $\geq 50\%$ é considerado um modelo ruim ou inadequado.

4 MATERIAIS E MÉTODOS

O desenvolvimento deste trabalho foi dividido em duas etapas: A primeira consistiu na elaboração da arquitetura de *hardware* e o respectivo desenvolvimento da placa de circuito impresso de um sistema de gerenciamento de baterias (BMS) responsável pela aquisição, manipulação e transmissão das medições das grandezas relacionadas às baterias, e na segunda etapa, foram implementados dois sistemas de *software*, um compreendido pelo *firmware* embarcado na placa do BMS e o outro que consiste na configuração dos serviços que são executados na nuvem.

4.1 ARQUITETURA DE HARDWARE

Para se alcançar os objetivos propostos nesse trabalho, algumas premissas de hardware foram estabelecidas para a concepção da placa eletrônica do BMS, sendo elas:

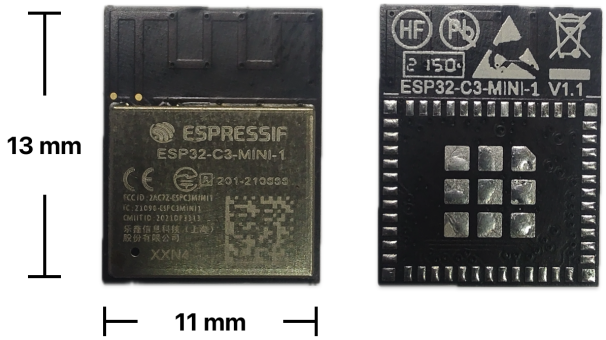
1. **Baixo consumo de bateria**: Com o objetivo de garantir uma maior autonomia das baterias utilizadas na aplicação a que se destinam, a eletrônica embarcada deve apresentar baixo consumo de corrente;
2. **Baixo erro de medição**: O sistema deve apresentar um erro de até 1% na medição de corrente, tensão e temperatura das células das baterias;
3. **Acesso remoto**: A placa deve disponibilizar uma interface de comunicação sem fio para que possa ser configurada e/ou ser capaz de se comunicar remotamente com um servidor;
4. **Segurança**: Por esta aplicação envolver a utilização de baterias com substâncias potencialmente inflamáveis, a correta operação das mesmas deve ser garantida, por meio de funcionalidades que em caso de falhas permitam a proteção contra sobrecorrentes e sobretensões.

Visando atender os requisitos 1, 2 e 4, foi escolhido o circuito integrado BQ76952 fabricado pela Texas *Instruments* (TEXAS INSTRUMENTS, 2020) para realizar as atividade de aquisição dos sinais das baterias, controle de carga e descarga e proteção contra falhas. Este chip é um monitor de baterias de íons de lítio, polímero de lítio e $LiFePO_4$ de 3 a 16 células em série de alta precisão. Está disponível em um encapsulamento de 48 pinos do tipo TQFP e apresenta dois conversores analógico/digital (ADC's) independentes de 32 bits de resolução cada, que garante uma alta precisão de medição de tensão das células, tipicamente menor que 10 mV. Possui um conjunto de proteções relacionadas a

tensão, corrente, temperatura e diagnósticos internos, e seu consumo em modo normal de operação é cerca de 286 μA. O *layout* da pinagem desse chip pode ser visto no anexo A.

Com base na comparação apresentada na Tabela 2, visando garantir o requisito 3 e focando também no menor consumo de energia possível, foi escolhido um módulo Wi-Fi/Bluetooth desenvolvido pela Espressif, o ESP32-C3-MINI-1 (ESPRESSIF SYSTEMS, 2022). Esse módulo é composto por um chip microcontrolador, um cristal de 40MHz, uma antena *on-board* além de alguns resistores e capacitores, tudo isso encapsulado numa estrutura de 11x13 mm, como pode visto na Figura 11 e cujo *layout* da pinagem está apresentado no anexo A.

Figura 11 – Encapsulamento do módulo ESP32-C3-MINI-1



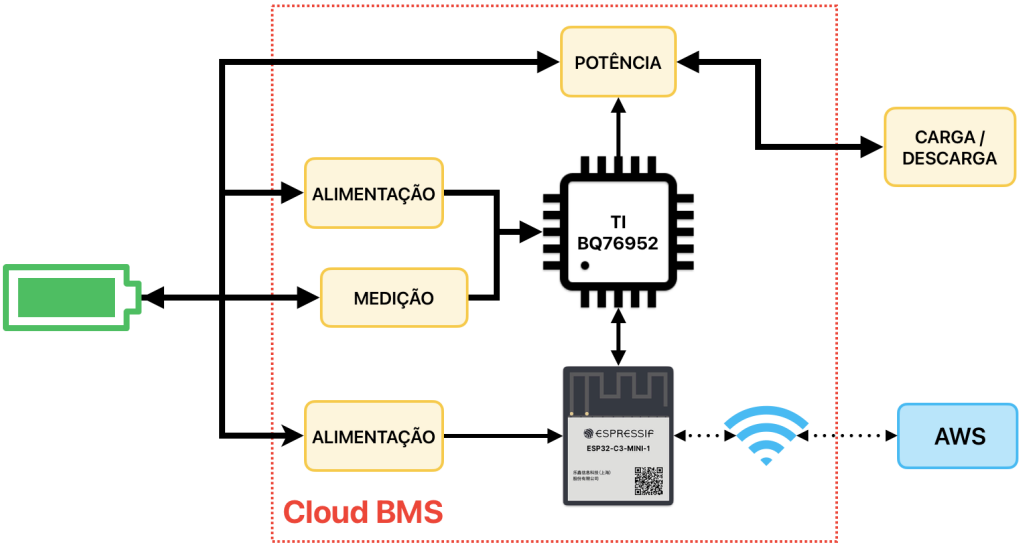
Fonte: Produzida pelo autor.

O chip encapsulado no módulo ESP32-C3-MINI-1 é um *System on Chip* (SoC) fabricado pela Espressif Systems, de código ESP32-C3FH4 (ESPRESSIF SYSTEMS, 2024). Esse chip possui 33 pinos numa estrutura QFN, e integra uma solução completa de Wi-Fi/Bluetooth a partir de uma série de circuitos internos, tais como, RF balun, chaveador de antena, amplificador de potência, amplificador de baixo ruído de recepção, filtros e módulos de gerenciamento de energia. O ESP32-C3FH4 também integra um processador de único núcleo de 32-Bits com arquitetura RISC-V e frequência de até 160 MHz, capaz de interagir com sensores e atuadores a partir de 22 pinos GPIO. Em modo normal de operação esse chip consome cerca de 20 mA de corrente, chegando até 350 mA de pico com o circuito de Rádio frequência ativo e em modo de transmissão.

4.1.1 Diagrama de blocos da arquitetura proposta

Na Figura 12 apresenta-se o diagrama de blocos da arquitetura de hardware proposta para a placa BMS. Nesse diagrama as setas indicam o sentido do fluxo de energia e a área tracejada em vermelho delimita os componentes e recursos projetados. A bateria representa um conjunto de 3 a 16 células em série, que são os limites inferior e superior para o funcionamento do sistema.

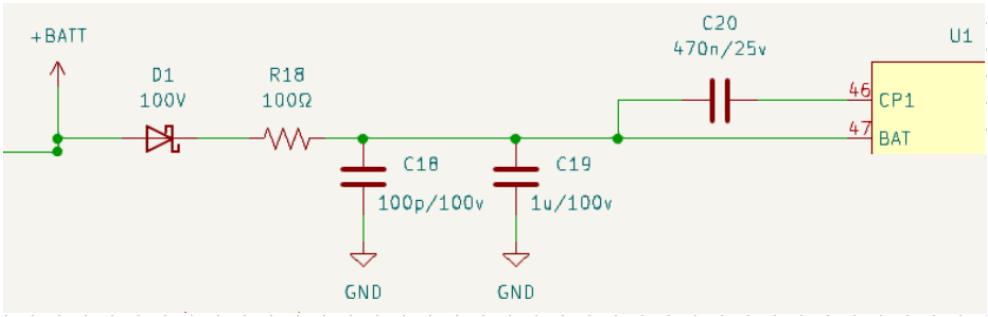
Figura 12 – Diagrama de blocos da arquitetura de hardware.



Fonte: Produzida pelo autor.

O bloco "Alimentação"que interliga as baterias ao chip BQ76952, é formado por um circuito simples composto por diodo de proteção, e filtro passa baixas de primeira ordem, conforme diagrama esquemático apresentado na Figura 13. Sua principal função é de fornecer energia ao chip BQ76952 por meio do seu pino 47 (BAT). O diodo é projetado para suportar 100V de condução direta pois ao se utilizar o limite máximo de 16 células, a tensão do pack (+BATT) pode atingir cerca de 67 Volts máximos. Já o filtro passa baixas tem a função de suprimir variações da tensão do Pack acima de cerca de 1,6 KHz e de garantir corrente estável ao chip em caso de rápidos surtos ocasionados por falhas no sistema.

Figura 13 – Circuito de alimentação do chip BQ76952.



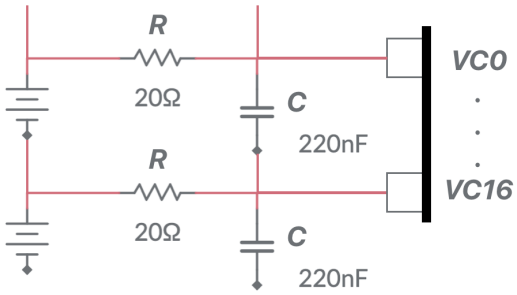
Fonte: Produzido pelo autor.

Já no bloco "Medição"estão compreendidos os circuitos de aquisição das tensões das baterias, medição de corrente e de temperatura das células e dos MOSFETs. Para a medição das tensões, é utilizado um acoplamento das células aos pinos do chip BQ76952

(VC0 a VC16) por meio de resistor limitador de corrente para o caso de ocorrência de balanceamento ativo de carga, além de capacitores de desacoplamento, conforme representado graficamente na Figura 14a.

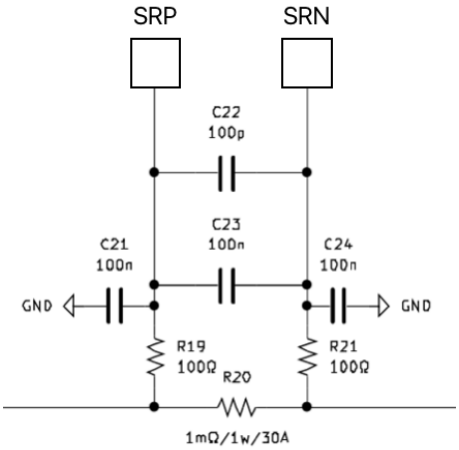
Figura 14 – Esquemático dos circuitos de medição de corrente e tensão.

(a) Medição de tensão das células.



Fonte: Produzido pelo autor.

(b) Medição de corrente.



Fonte: Produzido pelo autor.

Para medição da corrente, foi utilizado o circuito conforme representado na Figura 14b com o emprego de um resistor *shunt* cerâmico de 1 mΩ[1] (1W) conectado aos pinos SRP (18) e SRN (20) do chip. Entre os terminais do resistor e dos pinos SRP e SRN é utilizado um filtro passa baixas de primeira ordem com frequência de corte em torno de 16KHz para estabilização maior da tensão nesses pinos. A polaridade da queda de tensão presente no resistor *shunt* determina o sentido da corrente e sua magnitude é calculada com o emprego da Equação 4.1.

$$I = (CC\_ADC - \frac{Board\_Offset}{CC\_Offset\_Samples}) \cdot CC\_Gain; \tag{4.1}$$

sendo CC_ADC o valor de 32-bit da tensão medida no *shunt*, obtido do conversor AD dedicado. $Board\_Offset$ é um valor de ajuste acessível e configurável na memória do chip, onde seu valor padrão é 0. $CC\_Offset\_Samples$ é um fator de escala do parâmetro $Board\_Offset$, também acessível e configurável na memória do chip, seu valor padrão é de 64 e CC_Gain é um fator de ganho que depende diretamente do valor do resistor *shunt* na proporção de:

$$CC\_Gain = \frac{7.4768}{shunt\_em\_m\Omega} \tag{4.2}$$

[1] Segundo o datasheet do BQ76952, a faixa de medição da queda de tensão no *shunt* é de ± 200mV, o que limita a corrente do sistema em ± 200 A se for utilizado um resistor de 1 mΩ.

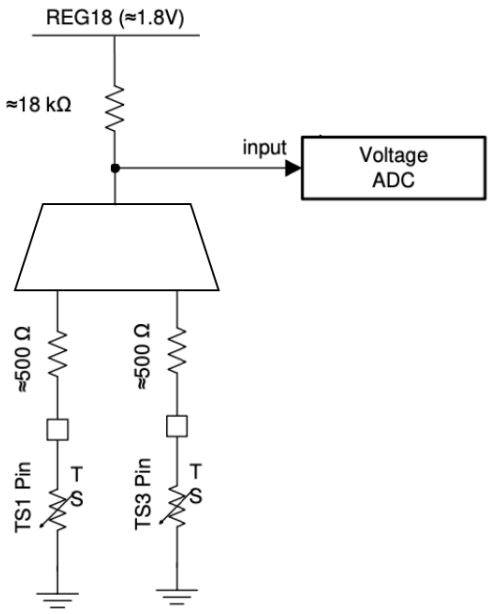
O chip BQ76952 oferece esses parâmetros de configuração da corrente para eventuais calibrações em linha de produção, afim de corrigir desvios e tolerâncias do dispositivo.

Para medição das temperaturas, são conectados termistores de 10 kΩ dos pinos TS1 (21) e TS3 (23) do BQ76952 ao terra do circuito. Nesses pinos exitem resistores internos de *pullup* que dividem a tensão de um LDO interno de 1,8 V com os termistores, conforme representado na Figura 15. O cálculo da temperatura é realizado pelo BQ76952 a partir dos valores de resistência calculados da tensão obtida desses pinos, e aplicados ao modelo da relação resistência-temperatura da equação 4.3.

$$R_T = 10000 \cdot e^{3435(\frac{1}{T} - \frac{1}{298.15})} \tag{4.3}$$

onde R_T é a resistência do termistor na temperatura T (Kelvin).

Figura 15 – Circuito para medição da temperatura.



Fonte: Produzido pelo autor.

No bloco "Alimentação"que interliga as baterias com o módulo ESP32, está presente o circuito integrado MP9486A (MONOLITHIC POWER, 2017), que é um regulador chaveado do tipo *step-down* capaz de fornecer até 1 A de corrente constante com uma tensão de entrada na faixa de 4,5 V a 100 V. Esse circuito é responsável pela regulação da tensão em cerca de 3,3 V necessária para alimentar o módulo ESP32-C3-MINI-1.

O bloco "Potência"contém os componentes responsáveis pela liberação do fluxo de corrente tanto para carga quanto para descarga das baterias. Nele estão presentes quatro MOSFET's de potência de canal N que chaveiam o terminal positivo do módulo de baterias, sendo dois para controle de carga e os outros dois para a descarga. Além desses

Tabela 5 – Pinos dedicados para comunicação entre o chip BQ76952 e o módulo ESP32-C3-MINI-1.

Pino: BQ76952 / ESP32-C3-MINI-1	**Função**
25 / IO6	**ALERT**: Utilizado como fonte de interrupção para o ESP32. Esse pino pode ser configurado no BQ76952 para ir a nível lógico baixo se algumas condições de alarme ocorrerem durante a operação, como por exemplo um curto circuito for detectado.
26 / IO5	**SCL**: Pino de sincronismo do protocolo I2C.
27 / IO4	**SDA**: Pino de dados do protocolo I2C.
29 / IO18	**CFETOFF**: Utilizado para desabilitar diretamente o driver dos MOSFETs de carga de maneira mais imediata, sem a necessidade de comandos I2C. Ativo em nível lógico alto.
30 / IO19	**DFETOFF**: Utilizado para desabilitar diretamente o driver dos MOSFETs de descarga de maneira mais imediata, sem a necessidade de comandos I2C. Ativo em nível lógico alto.
33 / IO7	**RST_SHUT**: Se esse pino é atuado em nível logico alto por menos de 1s, uma rotina de reinicialização no BQ76952 é inciada, porém se o tempo de atuação for maior que 1s, o chip entra em modo de desligamento, onde irá desativar todos os seus sistemas periféricos.

Fonte: Produzida pelo autor.

MOSFETs, exitem outros dois de canal P que atuam como atenuadores de corrente em situação de pré-carga e pré-descarga do módulo de baterias.

A comunicação entre o chip BQ76952 e o módulo ESP32-C3-MINI-1 se dá por meio de pinos com funções específicas. Apresenta-se na Tabela 5 os pinos reservados para comunicação entre os dois componentes e suas principais funções de operação.

4.1.2 Configuração do chip BQ76952

Para que o chip BQ76952 execute corretamente suas funções e se adéque as mais diversas situações de utilização das baterias, se faz necessário que seus registradores sejam configurados de acordo. Para tal, o chip disponibiliza uma interface escrava de comunicação I2C[2] e protocolo específico para acesso aos dados.

Existem três maneiras diferentes de acessar e configurar os registradores do chip: por comandos diretos, subcomandos ou a partir do acesso direto à memória. Os comandos diretos são acessados por meio de um endereço de 7 bits e são capazes de disparar alguma

[2] Por padrão essa versão de chip vem configurada com 400 KHz de velocidade de comunicação e endereços 0x10 para escrita e 0x11 para leitura.

ação, escrever dados na memória ou retornar valores da memória requisitados pelo ESP32. Já os subcomandos são comandos adicionais que são acessados indiretamente por endereços de 16 bits, utilizando endereços de 7 bits de comando. O algoritmo apresentado no Quadro 1 representa como se pode realizar um procedimento de leitura de dados a partir de um subcomando, como exemplo, o subcomando "DASTATUS5"de endereço 0x0075 será utilizado. Esse subcomando retorna 32 byte de dados contendo informações como: máxima tensão de célula, temperatura das células, valor da corrente, entre outras. O acesso direto à memoria se dá de forma semelhante aos subcomandos com a diferença de que os endereços de 16 bits acessados são dos próprios registradores e não de subcomandos. O algoritmo representado no quadro 2 permite demostrar o procedimento correto para escrita direta nos registradores do chip, como exemplo, é gravado na posição de memória 0x91C8 (*Board_Offset*) o valor 0x12C. A operação de *checksum* verificada nesse algoritmo, consiste num somatório de 8 bits e numa operação de inversão bit a bit desse somatório, conforme Equação 4.4.

$$checksum = BWI[addr\_LSB + addr\_MSB + data\_LSB + data\_MSB] \tag{4.4}$$

Em que *addr_LSB* corresponde ao byte menos significativo da posição de memória a ser acessada, assim como o *addr_MSB* corresponde ao byte mais significativo da posição de memória a ser acessada, *data_LSB* corresponde ao byte menos significativo do dado a ser gravado e *data_MSB* corresponde ao byte mais significativo do dado a ser gravado. *BWI* é a operação de inversão bit a bit que deve ser aplicada ao somatório e que resulta em um número de 8 bits que deve ser armazenado na posição de memória 0x60.

Quadro 1 – Leitura de dados por um subcomando.

1. Escrever o byte menos significativo do subcomando (0x75 nesse exemplo) no endereço de comando 0x3E;
2. Escrever o byte mais significativo do subcomando (0x00 nesse exemplo) no endereço de comando 0x3F;
3. Realizar a leitura dos dados nos endereços 0x3E e 0x3F, e enquanto ela for diferente do endereço do subcomando (0x0075 nesse exemplo), repetir;
4. Realizar a leitura do comprimento do buffer de resposta no endereço 0x61 (36 nesse exemplo);
5. Realizar a leitura do buffer de resposta, iniciando do endereço 0x40 até o comprimento do buffer - 4 (32 nesse exemplo).

Quadro 2 – Escrita de dados na memória do chip.

1. Escrever o byte menos significativo da memória (0xC8 nesse exemplo) no endereço de comando 0x3E;
2. Escrever o byte mais significativo da memória (0x91 nesse exemplo) no endereço de comando 0x3F;
3. Escrever no buffer de transferência o valor desejado para memória, iniciando do endereço 0x40.(Nesse exemplo será escrito 0x2C no endereço 0x40 e 0x1 no endereço 0x41). O limite de escrita por bloco é de 32 bytes;
4. Escrever o valor de checksum (0x79 nesse exemplo) no endereço 0x60, e o comprimento dos dados no endereço 0x61 (0x06 nesse exemplo, 2 bytes de memoria + 1 byte correspondendo ao endereço 0x60 + 1 byte correspondendo ao endereço 0x61 + 2 bytes de dados).

4.2 ARQUITETURA DE SOFTWARE

Para esta aplicação foram desenvolvidos dois sistemas de software distintos e que são executados em instâncias independentes.

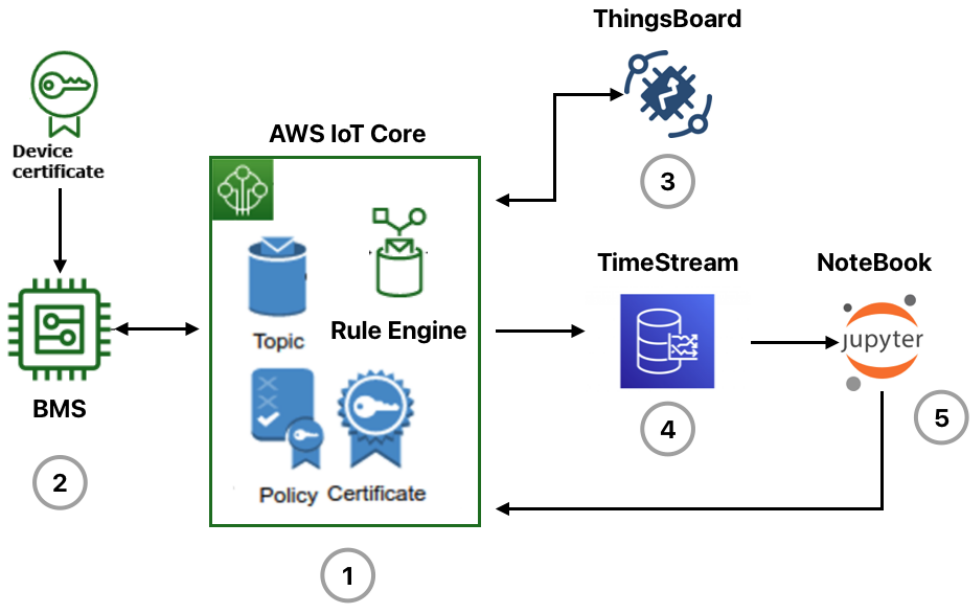
O primeiro sistema é o próprio *firmware* embarcado no módulo ESP32-C3-MINI-1 que divide-se em duas partes, uma é uma aplicação de servidor *web* utilizando *sockets* e a outra são as funções padrão de configuração do módulo, transmissão remota dos dados e API de comunicação e configuração do chip BQ76952. O *firmware* foi desenvolvido em linguagem C++ no ambiente Arduino IDE utilizando a API desenvolvida pela Espressif (ESPRESSIF, 2024) e sua execução segue a sequência de atividades do diagrama contido no apêndice A. Os dados de telemetria transmitidos ao serviço de nuvem (AWS) são encapsulados no protocolo MQTT e correspondem à tensão do banco de baterias, tensões individuais das células, corrente, temperatura das células, temperatura dos FETs e o SoC local.

Para o segundo sistema foi utilizada a plataforma da Amazon (AWS-IoT) (AMAZON WEB SERVICES, 2024), especificamente os serviços de IoT Core. Foi criado, no ambiente do AWS, um dispositivo IoT com certificados de segurança dedicados e os dados transmitidos de telemetria são direcionados e armazenados numa tabela do serviço Timestream que pode ser acessada por comandos SQL. Na Figura 16 é possível observar um diagrama no qual são apresentados os serviços utilizados na nuvem e o fluxo de dados entre eles.

Nesse diagrama estão separadas em etapas a configuração dos sistemas e serviços utilizados na nuvem, desde a criação dos dados das baterias até o seu armazenamento, manipulação e exibição em gráficos, sendo essas etapas definidas como segue:

1. **IoT Core**: Inicialmente é criada uma instância virtual do dispositivo BMS, dotada de

Figura 16 – Diagrama dos serviços utilizados na nuvem.



Fonte: Produzido pelo autor.

certificados X.509 e políticas de acesso associadas a esse certificado. Essas políticas compreendem as permissões para que o dispositivo possa se conectar ao *broker* MQTT; possa se inscrever e publicar em tópicos e possa receber comandos remotos do servidor. Nessa etapa também é criada uma regra de roteamento no serviço **Rule Engine**.

2. **BMS**: Os dados de tensões, corrente e temperaturas gerados pelo dispositivo BMS são publicados no tópico "esp32_bms/pub"no *IoT Core*. O dispositivo BMS deve conter em sua memória um certificado X.509, criado previamente na plataforma IoT, para que a conexão com o servidor seja permitida. Aqui também o dispositivo se inscreve no tópico "esp32_bms/sub"para receber as atualizações de estado da bateria estimadas remotamente;

3. **ThingsBoard**: A visualização dos dados das baterias a partir de gráficos e *dashboard* é feita por meio do ThingsBoard, uma plataforma IoT de código aberto capaz de integrar serviços de armazenamento, processamento e visualização de dados. (THINGSBOARD, 2024)

4. **Timestream**: Todos os dados publicados no tópico "esp32_bms/pub", são estruturados e armazenados numa tabela do serviço **Timestream** para que estejam disponíveis para futuras consultas;

5. **Jupyter**: Com a tabela criada, é possível realizar o processamento dos dados das baterias fazendo uso do *Jupyter notebook*, uma solução de código aberto que

utiliza linguagens de *script* como Python para executar exploração de dados e análises avançadas. É possível aplicar análises mais complexas utilizando métodos de aprendizado de máquina e modelos de regressão para previsão nos dados, por exemplo. Os notebooks Jupyter são hospedados no serviço Amazon SageMaker;

Uma vez que os dados das baterias estão disponíveis na tabela do Timestream, um código em linguagem Python que executa em uma instância de notebook do Jupyter, realiza a requisição desses dados por meio de chamada SQL e implementa uma rede neural do tipo *feedforward*, previamente treinada, para estimar o SoC do *pack* de baterias. Em seguida, esta estimativa é publicada no tópico "esp32_bms/sub", no IoT Core, para que o dispositivo BMS tenha acesso.

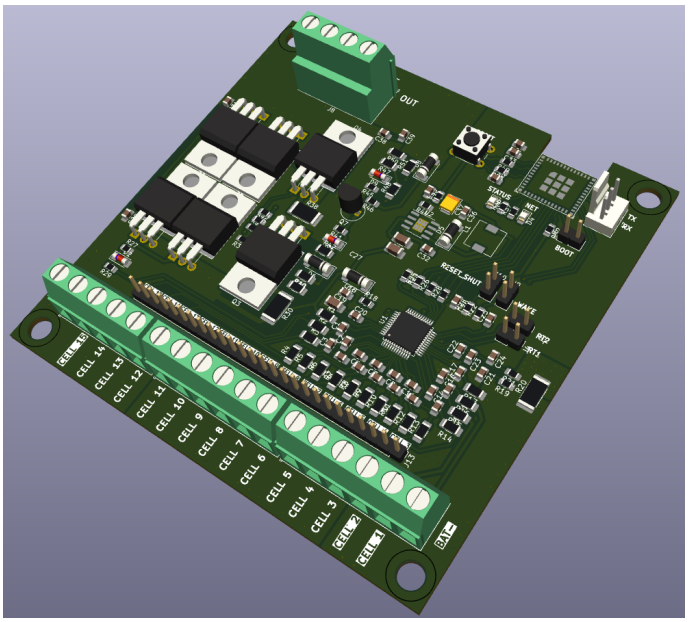
5 RESULTADOS

Neste capítulo são apresentados os resultados obtidos a partir da implementação de *software* e *hardware* descritas no capítulos anteriores.

5.1 *HARDWARE*

Com base nos requisitos de *hardware* apresentados, a placa BMS foi projetada com o auxílio do software de código aberto KiCad EDA e sua representação gráfica em 3D pode ser vista na Figura 17. A placa foi projetada no padrão dupla face com dimensões de 100 mm x 100 mm.

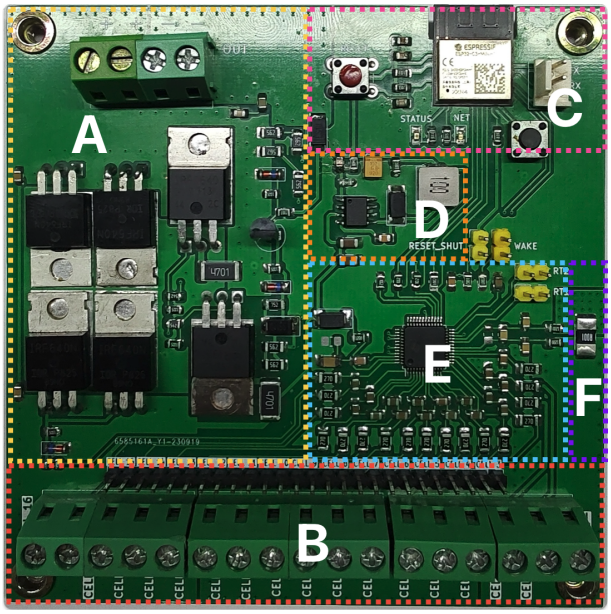
Figura 17 – Representação 3D da placa projetada.



Fonte: Produzido pelo autor.

Na Figura 18 apresenta-se a camada superior da placa já confeccionada em material FR4 de 1,6 mm de espessura, além da disposição dos componentes devidamente soldados. A partir da fotografia é possível visualizar o conector de saída/entrada de carga e circuito de proteção e controle de carga e descarga, formado por MOSFETs de potência de referência IRF840 (região A), os conectores pra as células (região B), o módulo ESP32-C3-MINI-1 e seus componentes auxiliares (região C), o chip MP9486A e seus componentes auxiliares (região D), o chip BQ76952 com seus componentes auxiliares (região E), e o resistor *Shunt* de medição de corrente (região F). Na Tabela 6 apresenta-se as principais especificações do projeto dessa placa.

Figura 18 – Vista superior da placa confeccionada.



Fonte: Produzido pelo autor.

Tabela 6 – Especificações da placa projetada.

Item	**Especificação**
Nº de células	3 a 16 em série
Tipo de células	Íons de lítio, Polímero de lítio e LiFePO4
Corrente CC	8A, 25ºC / 5.1A, 100ºC
Medições	Tensão (células, pack), Corrente do pack (shunt), Temperatura (FET e pack, termistor)

Fonte: Produzido pelo autor.

A esta placa é conectado um *pack* de baterias com tensão nominal de 48V e capacidade de 2950mAh, formado por 13 células de íons de lítio fabricadas pela Samsung, modelo INR18650-30Q, cujas especificações de fábrica constam na Tabela 7. Na Figura 19 é possível observar a integração do *pack* com a placa BMS desenvolvida nesse trabalho.

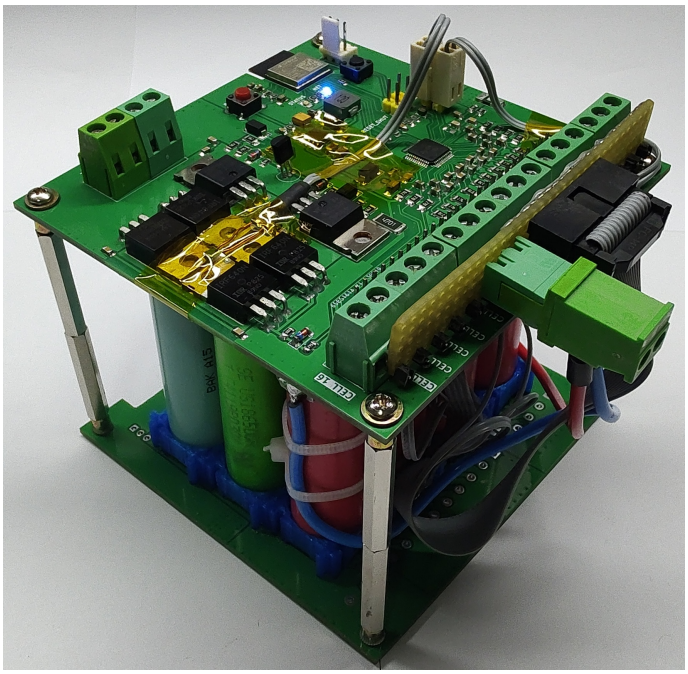
A configuração de alguns parâmetros de operação e controle da placa é realizada por meio de seu endereço de servidor *WebSocket* local. Ao acessar esse endereço a partir de um navegador, a página apresentada na Figura 20 é exibida e o usuário tem a oportunidade de acompanhar em tempo real as informações de estimativa de carga, tensão total das células, valores máximo e mínimo do conjunto de células, a corrente que está circulando no sistema, que pode ser positiva (carga) ou negativa (descarga) e as temperaturas interna do chip BQ76952, do conjunto de células e dos MOSFETs de controle. Nessa página, também estão disponíveis as configurações de habilitação das células que estão efetivamente conectadas

Tabela 7 – Especificações de fábrica do modelo INR18650-30Q.

Item	**Especificação**
Capacidade mínima de descarga	2950mAh
Tensão nominal/ Tensao de corte	3,6V/2,5V
Carga padrão	CCCV, 1.50A, 4.20 ± 0.05 V, 150mA corte
Carga máxima	CCCV, 4A, 4.20 ± 0.05 V, 100mA corte
Tempo de carga	Padrão : 180min / 150mA corte Máxima: 70min (at 25℃) / 100mA corte

Fonte: Produzido pelo autor.

Figura 19 – BMS conectado a um *pack* de baterias de 48V.



Fonte: Produzido pelo autor.

ao sistema[1] (CELL1 a CELL16) e habilitação do balanceamento automático de carga das células. Estão disponíveis também a visualização em tempo real de alarmes presentes no sistema, sendo eles:

SCD : Alarme de curto circuito no descarregamento;

OCD : Alarme de sobrecorrente no descarregamento;

OCC : Alarme de sobrecorrente no carregamento;

COV : Alarme de sobretensão das células;

CUV : Alarme de subtensão das células;

OTF : Alarme de sobretemperatura dos MOSFETs;

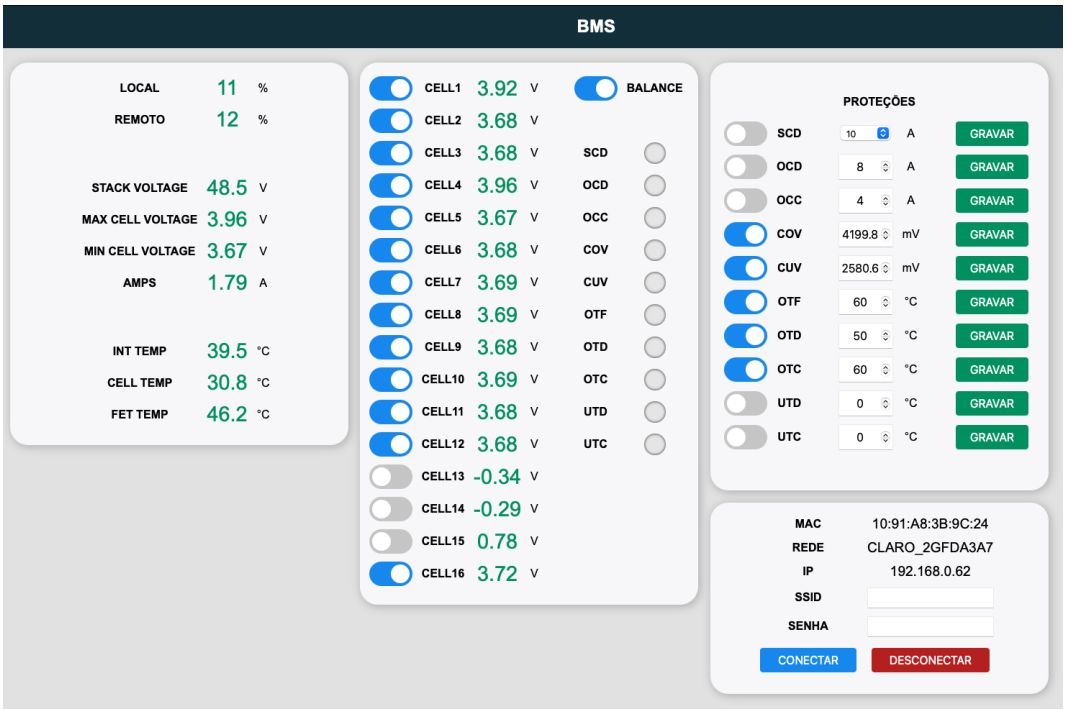
OTD : Alarme de sobretemperatura das células no descarregamento;

OTC : Alarme de sobretemperatura das células no carregamento;

UTD : Alarme de sub temperatura das células no descarregamento;

UTC : Alarme de subtemperatura das células no carregamento.

Figura 20 – Vista da interface de acesso ao BMS.



Fonte: Produzido pelo autor.

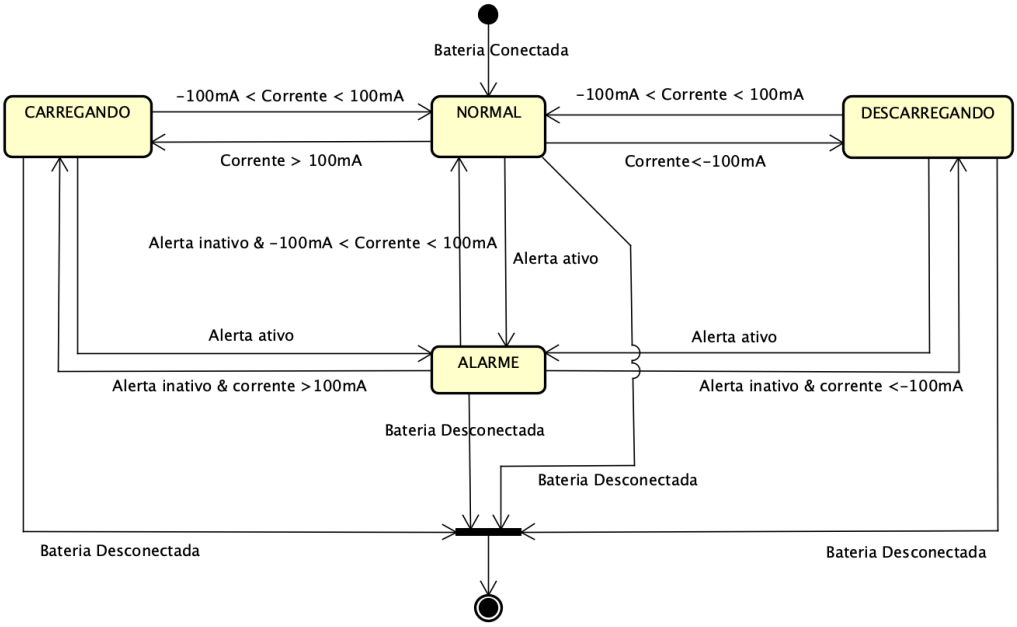
[1] Isso refletirá no acionamento dos alarmes de sub tensão e sobretensão das células.

Cada uma das proteções associadas a estes alarmes pode ser habilitada ou desabilitada, bem como seus limites também podem ser configurados conforme necessidade de cada aplicação. A última configuração possível na página está relacionada ao acesso remoto do sistema a partir da nuvem, por meio da inserção das credenciais corretas da rede WiFi a ser utilizada pelo módulo para conexão ao servidor MQTT da AWS.

A placa BMS projetada apresenta em sua execução os possíveis estados mostrados na Figura 21, nos quais as atividades executadas em cada um podem ser vistas na Tabela 8. A ação "Alerta ativo"que leva ao estado de alarme, refere-se ao disparo de qualquer uma das proteções já mencionadas. Uma vez nesse estado, para que retorne aos outros estados é necessário que o motivo que gerou a atuação da proteção seja cessado, por exemplo, se houve a ativação da proteção COV, o sistema permanecerá em alarme enquanto a tensão de uma ou mais células ativas permanecer acima do limite máximo.

A ação de monitoramento, constante na Tabela 8, refere-se a medição a cada cinco segundos dos parâmetros de corrente, tensão e temperatura, e a ação CC, é a contagem de coulomb (*Coulomb Counting*), disponível no chip BQ76952.

Figura 21 – Diagrama estados da placa BMS.



Fonte: Produzido pelo autor.

5.2 SERVIÇOS DA NUVEM

Dentro do console de gerenciamento do AWS IoT Core (acessível via navegador) é criada uma instância virtual do dispositivo BMS construído. Essa instância possui certificados X.509 e políticas de acesso associadas a esses certificados. Esses certificados

Tabela 8 – Ações executadas nos estados.

Estado	**Ações**
Normal	Monitoramento ativo; FETS ativos; Websocket ativo; Proteções desativadas; CC ativo; publicações periódicas na nuvem; Led de status acesso contínuo.
Carregando	Monitoramento ativo; FETS ativos; Websocket ativo; Proteções de carregamento ativadas; CC ativo; publicações periódicas na nuvem; Led de status acesso pulsando suave a cada segundo.
Descarregando	Monitoramento ativo; FETS ativos; Websocket ativo; Proteções de descarregamento ativadas; CC ativo; publicações periódicas na nuvem; Led de status acesso contínuo.
Alarme	Monitoramento ativo; FETS desativados; Websocket ativo; Proteções de carregamento e descarregamento ativadas; CC desativado; publicações periódicas na nuvem; Led de status acesso pulsando sólido a cada 500 ms.

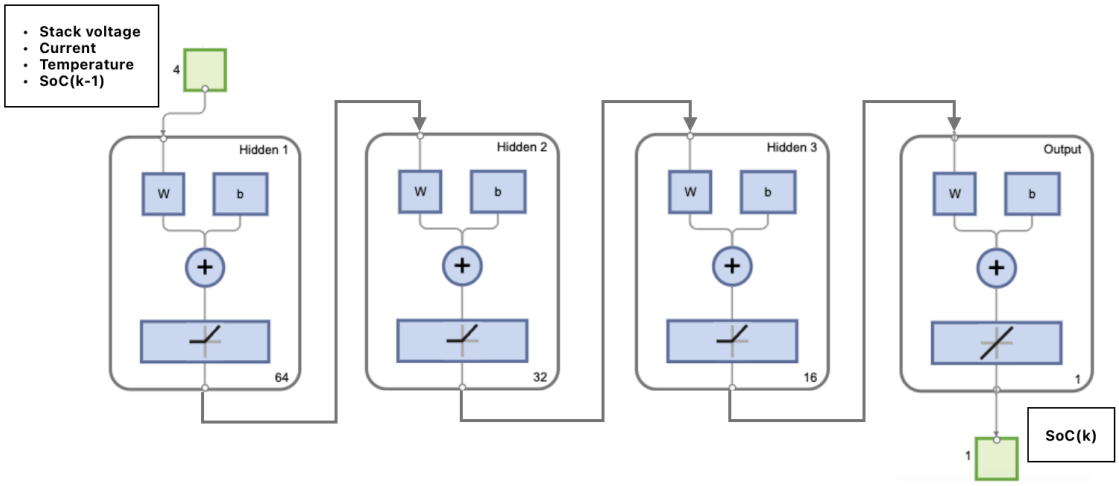
Fonte: Produzido pelo autor.

são embarcados no *firmware* do dispositivo afim de que este possa se conectar ao *broker* MQTT, possa se inscrever e publicar em tópicos e possa receber comandos remotos do servidor. Ainda no console, é criada uma regra de roteamento de mensagem. Nessa regra é executada a chamada SQL: *SELECT * FROM 'esp32_bms/pub'*, para que toda mensagem que for publicada no tópico "esp32_bms/pub"seja encaminha e armazenada numa tabela do serviço AWS Timestream, sendo possível assim, armazenar os dados das baterias para posterior consulta.

Em seguida, uma instância de notebook Jupyter (composta por 2 núcleos de CPU Intel Skylake E5 2686 v5, 4GiB de memória RAM e 5GiB de armazenamento) é criada na plataforma Sagemaker. Essa instância executa uma aplicação em Python responsável por estimar o SoC do *pack* por meio de um algoritmo de rede neural artificial (disponível no apêndice B). Esta rede é do tipo *feedforward*, formada por três camadas ocultas e uma de saída. A primeira camada oculta possui 64 neurônios, a segunda 32, a terceira 16 e a camada de saída possui apenas um neurônio. A função de ativação das camadas ocultas é do tipo linear retificada ("ReLU") e da camada de saída é do tipo linear. A rede possui quatro parâmetros como entrada, sendo eles: tensão do *pack*, corrente, temperatura e o SoC anterior; o parâmetro de saída é o SoC atual. Na Figura 22 apresenta-se o diagrama da rede utilizada.

Para o treino da rede, foi utilizado um conjunto de dados próprio formado por 4636 pontos obtidos por três ensaios de carga e dois de descarga do *pack* (Figura 23). Nesses ensaios foram registrados os valores de corrente de carga e descarga, tensão do *pack*, temperatura do *pack* e a capacidade (SoC), obtida por meio do *Coulomb Counting* próprio

Figura 22 – Diagrama da rede neural utilizada.



Fonte: Produzido pelo autor.

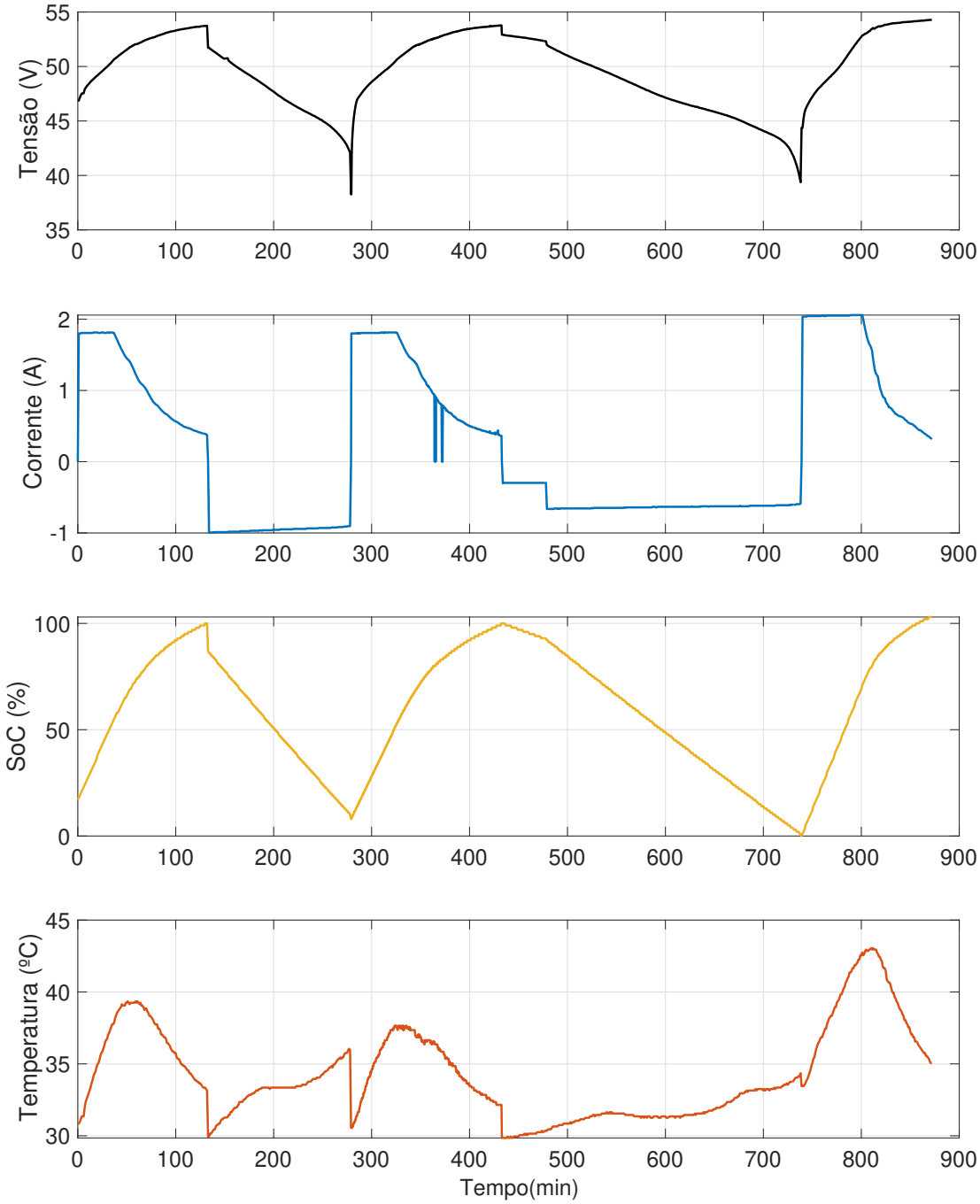
do chip BQ76952. Desses 4636 pontos, utilizou-se 80% de forma aleatória para o treino e 20% para teste de desempenho da rede. A função de otimização escolhida foi o Adam. A função de custo selecionada foi o MSE (*Mean Squared Error*) e foram escolhidas 30 épocas para o treino, porém como pode ser visto no gráfico da Figura 24, observa-se que a partir da época 18 o valor já converge para estabilidade, não resultando em grandes ganhos de performance ao longo de mais épocas. Após o treino, a rede foi submetida aos 20% inéditos do total de 4636 pontos para teste, resultando num índice MAE de aproximadamente 0,48 e um MSE de aproximadamente 0,32 que indicam uma boa acuracidade na estimação.

Uma vez treinada a rede é encapsulada num arquivo de formato ".h5"que é então armazenado num *bucket* do serviço S3 do AWS para que o algoritmo em execução na nuvem possa carregá-lo. O algorítmo do apêndice B consulta os dados mais recentes na tabela do Timestream, atualizados pelo BMS, a partir de uma chamada SQL; as variáveis corrente, tensão, temperatura e SoC anteriores são atualizadas e uma nova predição do SoC atual é realizada por meio da rede neural; esse valor é publicado no tópico "esp32_bms/sub"do AWS IoT Core e todo o processo se repete a cada 60 segundos.

5.2.1 Dashboard

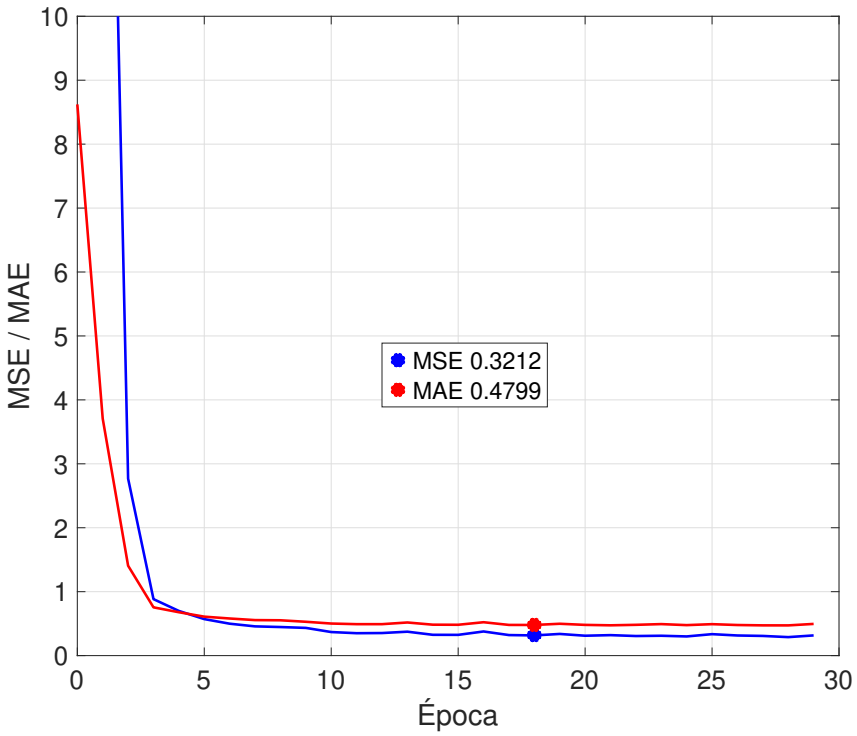
Na Figura 25 apresenta-se o painel de monitoramento do sistema de gerenciamento de bateria, hospedado no servidor *thingsboard*. Esse painel exibe informações importantes sobre o estado do sistema, como a tensão total do *pack* (*Stack*), a corrente de operação que está sendo consumida ou fornecida pelo sistema (*Current*), a temperatura média das células (*Cell Temp*) e dos transistores de potência (*FET Temp*). É possível observar também no painel o indicador de Estado de Carga obtido por *coulomb counting* (*Local SOC*) e estimado por rede neural na nuvem (*remote SOC*), bem como as tensões individuais

Figura 23 – Dados para treino da rede.



Fonte: Produzido pelo autor.

Figura 24 – Função de custo MSE e índice MAE ao longo das épocas de treino.



Fonte: Produzido pelo autor.

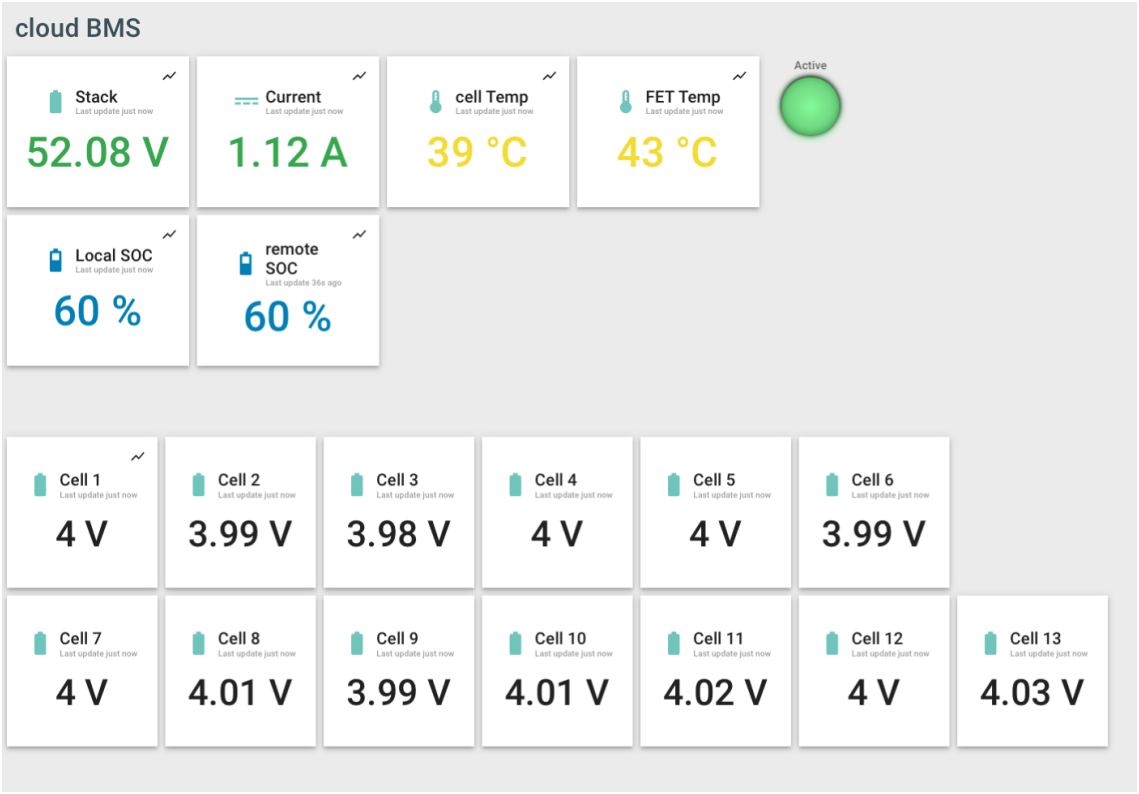
das 13 células. O indicador no canto superior direito sinaliza que o sistema está ativo e funcionando normalmente.

A integração da plataforma AWS IoT com o thingsboard ocorre na medida que este último se comporta como um dispositivo IoT e se inscreve nos tópicos "esp32_bms/pub"e "esp32_bms/sub"para receber as atualizações postadas pelo BMS e receber atualizações da estimativa de SoC da rede neural, respectivamente, lançando mão do recurso "*AWS IoT integration*", disponível na plataforma *Thingsboard*, bastando apenas adicionar os certificados de segurança e a chave de acesso previamente criados no AWS IoT.

5.3 ENSAIO DE AVALIAÇÃO DO SISTEMA

Com a finalidade de verificação do desempenho em aplicação estacionária, o sistema foi submetido a um ensaio de avaliação de carregamento e descarregamento do pack, conforme apresentado na Figura 26a. Inicialmente o *pack* foi carregado até um valor de aproximadamente 54V e logo em seguida, uma carga resistiva de aproximadamente 50 Ω foi conectada à saída, que gerou um consumo de aproximadamente 1A de corrente. Em intervalos de aproximadamente 20 minutos, a carga foi retirada e reinserida, para verificação da dinâmica do sistema em situações de variação de carga. O ensaio é encerrado quando a tensão do *pack* atinge aproximadamente 44V. Em todo esse processo a capacidade

Figura 25 – Página inical do *Dashboard*.



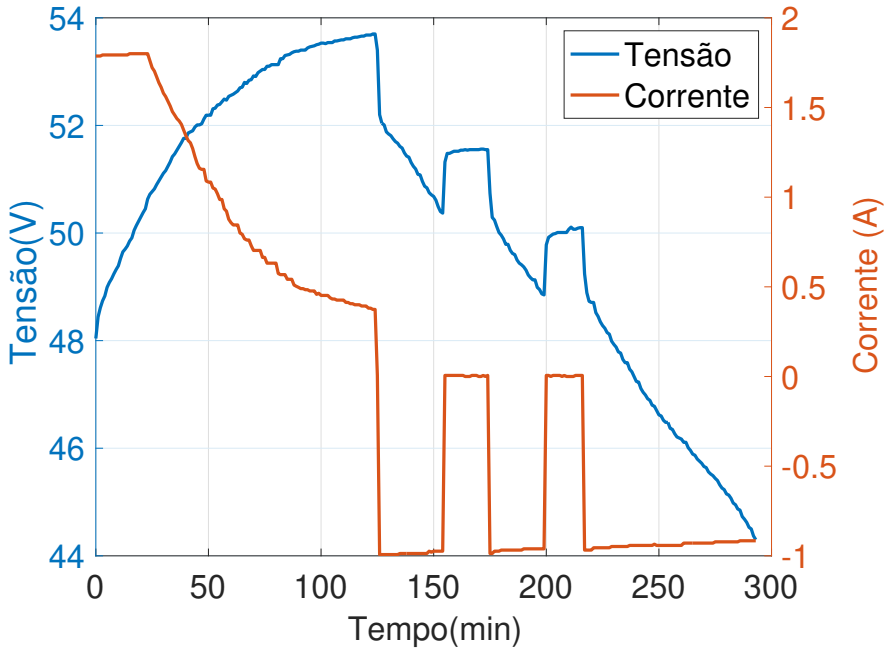
Fonte: Produzido pelo autor.

do *pack* é estimada de duas maneiras distintas, uma diretamente na placa BMS (*SoC local*) pelo método de *Coulomb Counting*, disponível no chip BQ76952, e a outra por meio da rede neural implementada (SoC Remoto) já treinada e executando na nuvem, com o intuito de comparação entre esses dois resultados. Na Figura 29b apresenta-se o resultado de estimação do SOC com a comparação entre os dois métodos de estimação durante o ensaio. Ao final do teste essa comparação retornou um valor MSE de 7,82 com um MAPE de 6,57% e um índice MAE de 2,33, indicando que a estimação remota do SoC por rede neural se aproxima com boa precisão ao valor do SOC estimado localmente pela placa.

Durante todo o ensaio, as medidas de corrente, tensão, temperatura e SOC, foram publicadas no servidor MQTT do AWS IoT, armazenadas no servidor *Thingsboard* e na tabela do *Timestream* e disponibilizadas no *Dashboard*, como pode ser visto na Figura 25.

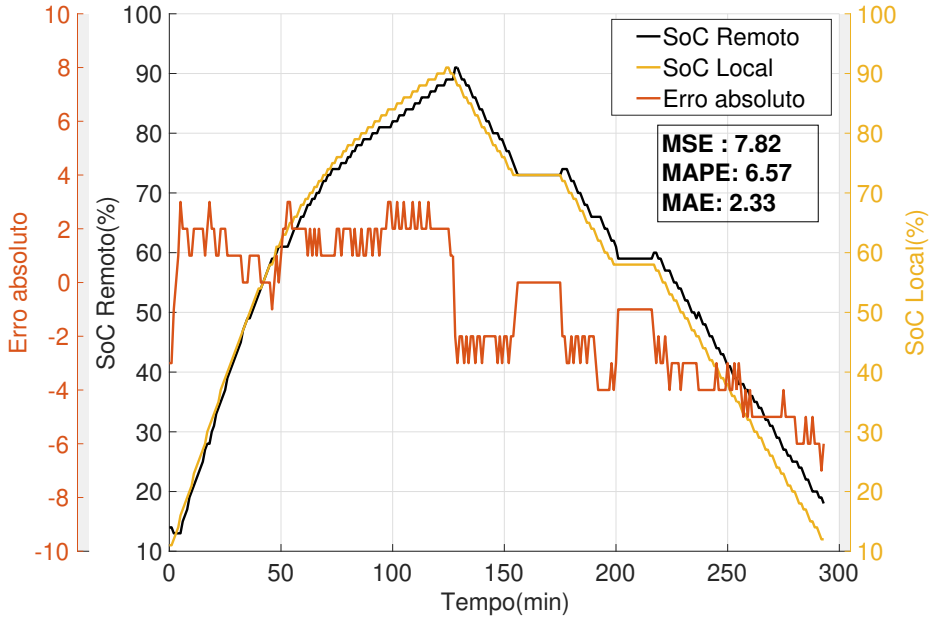
Figura 26 – Resultado do ensaio de avaliação.

(a) Medição de tensão e corrente do *pack*.



Fonte: Produzido pelo autor.

(b) Estimativa de carga.



Fonte: Produzido pelo autor.

6 CONCLUSÕES

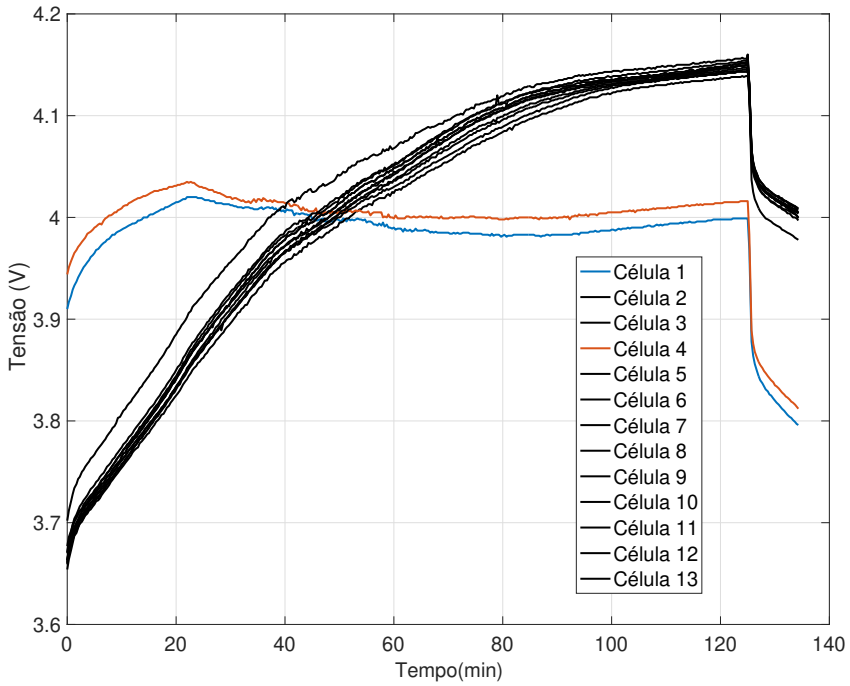
Neste trabalho, foi desenvolvida uma plataforma que integra *hardware e software* que compõem um sistema de gerenciamento de bateria baseado em nuvem, utilizando o módulo IoT ESP32 e a plataforma em nuvem AWS IoT. O sistema projetado visou superar as limitações dos Sistemas de Gerenciamento de Baterias (BMS) tradicionais ao aproveitar a computação em nuvem para maior poder de processamento e capacidades de armazenamento de dados. A solução proposta incluiu uma placa BMS especialmente projetada e software associado que permite monitoramento em tempo real, estimativa de estado, balanceamento de carga, controle térmico e controle de carga/descarga.

Diante dos resultados obtidos, a construção do *hardware* do Sistema de Gerenciamento de Baterias baseado em nuvem se mostrou bastante satisfatória na medida que este atende aos requisitos propostos, apresentando baixo consumo de bateria em modo normal de operação, precisão de medição, graças a utilização do conversador A/D de 32 bits do chip BQ76952, possibilidade de acesso remoto por meio do módulo ESP32-C3-MINI-1 e disponibilidade de recursos de segurança. Vale salientar que a forma como os componentes eletrônicos foram dispostos na placa, permitiu que ela se tornasse relativamente compacta, medindo 100 mm x 100 mm, que para aplicações nas quais existe pouca disponibilidade de espaço físico, como nos veículos elétricos, em especial em *scooters* e motos elétricas, essa característica é uma grande vantagem.

É importante salientar que apesar de o modelo de rede utilizado na estimação do SOC no ensaio de avaliação apresentar um desempenho aceitável, com um MAPE menor que 10%, pode-se alcançar resultados ainda mais precisos fornecendo mais pontos de treino para a rede, em diversas condições de operação, com cenários variados de temperatura e perfis de carga bem como em mais ciclos de carga e descarga. Além disso, o desempenho do sistema foi afetado nas dinâmicas de carregamento e descarregamento devido a uma discrepância observada no comportamento de duas células do *pack*, célula 1 e 4, como pode ser observado na Figura 27, as quais essas destoam bastante na resposta dinâmica comparando com as outras unidades.

A plataforma construída nesse trabalho permite que estudos acerca de algoritmos de estimação de estados de baterias (limitado de 3 a 16 células por placa) possam ser implementados e validados em testes práticos, e graças a acessibilidade e portabilidade do *hardware*, os estudos acerca do comportamento dinâmico das baterias sob condições variáveis de carga em diversas situações práticas pode ser realizado com auxilio do armazenamento dos dados das baterias e processamento na nuvem.

Figura 27 – Tensão das células.



Fonte: Produzido pelo autor.

6.1 TRABALHOS FUTUROS

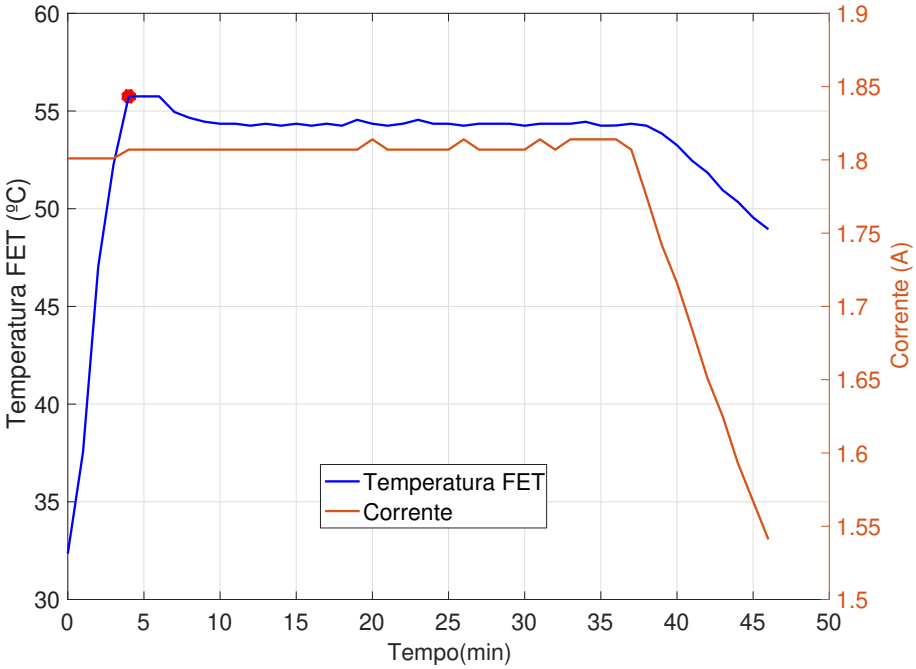
Para se aproveitar todo o poder de processamento e versatilidade dos recursos da nuvem, seria interessante a implementação de outros métodos de estimação de estados da bateria utilizando algoritmos avançados como os baseados em modelos de circuito equivalente de ordens elevadas ou até mesmo os que utilizam filtros adaptativos, como o Kalman estendido ou o filtro de partículas. Também é possível a utilização de métodos baseados em inteligência artificial, como os algoritmos genéticos, enxame de partículas, lógica Fuzzy, entre outros.

Outro ponto a ser explorado é a obtenção de resultados práticos em aplicação móvel, utilizando redes 5G, e em aplicação estacionária com perfis variados de carga, estabelecendo métricas de avaliação de desempenho do sistema como latência média, taxa de pacotes perdidos, eficiência e precisão de estimação, velocidade de convergência, estabilidade, custo computacional e custos associados aos serviços da nuvem.

Durante os testes realizados na placa, observou-se que a dissipação de potência nos MOSFET's de controle é um tópico de alta relevância na construção desse dispositivo, sendo necessário o projeto e implementação de um sistema de arrefecimento eficiente para mitigação de eventuais falhas nos FETS. Como demostração, em um dos ensaios de carregamento realizados na placa, a temperatura dos FETS foi acompanhada e verificou-se

que ao serem percorridos por uma corrente de aproximadamente 2A, a temperatura se eleva rapidamente, partindo de 32 ºC e atingindo cerca de 55 ºC em menos de cinco minutos, como pode ser visto na Figura 28. A partir desse instante (ponto vermelho no gráfico) é acionado um sistema de arrefecimento formado por dissipador de cobre e ventilação forçada, que mantém a temperatura aproximadamente constante, comprovando a eficiência desse sistema ao longo do restante do ensaio.

Figura 28 – Temperatura dos FETS.



Fonte: Produzido pelo autor.

REFERÊNCIAS

AMAZON WEB SERVICES. *AWS IoT Core Documentation*. 2024. Disponível em: <https://docs.aws.amazon.com/iot/>. Acesso em: Mai. 2024.

AWS IOT. *AWS IoT Core - Developer Guide*. 2024. Disponível em: <https://docs.aws.amazon.com/pdfs/iot/latest/developerguide/iot-dg.pdf#iot-gs>. Acesso em: Jun. 2024.

AWS IOT ANALYTICS. *AWS IoT Analytics - User Guide*. 2024. Disponível em: <https://docs.aws.amazon.com/pdfs/iotanalytics/latest/userguide/iotanalytics-ug.pdf>. Acesso em: Jun. 2024.

BERGVELD, H. J. *Battery Management Systems: Design by Modelling*. Tese (Doutorado) — Eindhoven University of Technology, 2001.

CHEN, Q. et al. Kbmp: Kubernetes-orchestrated iot online battery monitoring platform. *IEEE INTERNET OF THINGS JOURNAL*, IEEE, v. 11, n. 14, Julho 2024.

CSA. *Security Guidance for Critical Areas of Focus in Cloud Computing v4.0*. 2017. Disponível em: <https://cloudsecurityalliance.org/artifacts/security-guidance-v4>. Acesso em: Jun. 2024.

DIGIKEY. 2024. Disponível em: <https://www.digikey.com.br/en>. Acesso em: Dez. 2024.

DING, Y. et al. Automotive li-ion batteries: Current status and future perspectives. *Electrochemical Energy Reviews*, Springer, v. 2, p. 1–28, Jan 2019.

ESPRESSIF. *ESP32 Arduino Core's documentation*. 2024. Disponível em: <https://docs.espressif.com/projects/arduino-esp32/en/latest/index.html>. Acesso em: Mai. 2024.

ESPRESSIF SYSTEMS. *ESP32-C3-MINI-1 & MINI-1U Datasheet v1.3*. 2022. Disponível em: <https://www.espressif.com/sites/default/files/documentation/esp32-c3-mini-1_datasheet_en.pdf>. Acesso em: Mai. 2024.

ESPRESSIF SYSTEMS. *ESP32-C3 Series Datasheet v1.7*. 2024. Disponível em: <https://www.espressif.com/sites/default/files/documentation/esp32-c3_datasheet_en.pdf>. Acesso em: Mai. 2024.

FAUSETT, L. *Fundamentals of Neural Networks: Arquitectures, Algorithms ans applications*. New Jersey: Prentice Hall, 1994.

GOOGLE. *An end-to-end platform for machine learning*. 2015. Disponível em: <https://www.tensorflow.org>. Acesso em: Dez. 2024.

HASTIE, T.; TIBSHIRANI, R.; FRIEDMAN, J. *The Elements of Statistical Learning Data Mining, Inference, and Prediction*. 2. ed. [S.l.]: Springer, 2008.

HAYKIN, S. *Neural Networks and Learning Machines*. 3. ed. New Jersey: Prentice Hall, 2009.

HU, X. et al. State estimation for advanced battery management: Key challenges and future trends. *Renewable and Sustainable Energy Reviews*, Elsevier, v. 114, 2019.

IRENA. *Electricity Storage And Renewables: Costs And Markets To 2030*. 2017. Disponível em: <https://www.irena.org/publications/2017/Oct/Electricity-storage-and-renewables-costs-and-markets>. Acesso em: Jun. 2024.

KAILONG, L. et al. A brief review on key technologies in the battery management system of electric vehicles. *Frontiers of Mechanical Engineering*, Springer, v. 14, p. 47–64, 2019.

KARNEHM, D. et al. Introduction of a cloud computing architecture for the condition monitoring of a reconfigurable battery system for electric vehicles. *6th Conference on Cloud and Internet of Things*, IEEE, 2023.

KIM, T. et al. Cloud-based battery condition monitoring and fault diagnosis platform for large-scale lithium-ion battery energy storage systems. *Energies*, MDPI, 2018.

KINGMA, D. P.; BA, J. L. Adam: A method for stochastic optimization. *ICLR*, ICLR, 2015.

LECUN, Y. et al. Gradient based learning applied to document recognition. *Proceedings of IEEE*, IEEE, 1998.

MARINESCU, D. C. *Cloud Computing Theory and Practice*. 2. ed. Massachusetts: Morgan Kaufmann, 2018.

MARKUS, L. et al. Battery management system hardware concepts: An overview. *Applied Sciences*, MDPI, v. 8, Mar 2018.

MIAO, Y. et al. Current li-ion battery technologies in electric vehicles and opportunities for advancements. *Energies*, MDPI, v. 68, 2019.

MONOLITHIC POWER. *MP9486A – 100V INPUT, 3.5A, SWITCHING CURRENT LIMIT STEP-DOWN CONVERTER*. 2017. Disponível em: <https://www.monolithicpower.com/en/documentview/productdocument/index/version/2/document_type/Datasheet/lang/en/sku/MP9486AGN-Z/document_id/3749>. Acesso em: Mai. 2024.

MOUSER ELETRONICS. 2024. Disponível em: <https://www.mouser.com>. Acesso em: Dez. 2024.

MPS. *How to Design a Battery Management System (BMS)*. 2022. Disponível em: <https://www.monolithicpower.com/how-to-design-a-battery-management-system-bms>. Acesso em: Jun. 2024.

MUHAMMAD, U. A. et al. Towards a smarter battery management system for electric vehicle applications: A critical review of lithium-ion battery state of charge estimation. *Energies*, MDPI, v. 12, Jan 2019.

NAIR, V.; HILTON, G. E. Rectified linear units improve restricted boltzmann machines. *Proceedings of the 27th International Conference on Machine Learning*, IEEE, 2010.

NIST. *NIST Cloud Computing Reference Architecture*. 2011. Disponível em: <https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication500-292.pdf>. Acesso em: Jun. 2024.

NXP. *LPC11Cx2/Cx4 32-bit ARM Cortex-M0 microcontroller; 16/32 kB flash, 8 kB SRAM; C_CAN*. 2016. Disponível em: <https://www.nxp.com/docs/en/data-sheet/LPC11CX2_CX4.pdf>. Acesso em: Dez. 2024.

OPITZ, A. et al. Can li-ion batteries be the panacea for automotive applications? *Renewable and Sustainable Energy Reviews*, Elsevier, v. 68, p. 685–692, 2017.

RASPBERRY PI. *Raspberry Pi 4*. 2019. Disponível em: <https://www.raspberrypi.com/products/raspberry-pi-4-model-b/>. Acesso em: Dez. 2024.

TEXAS INSTRUMENTS. *BQ76952 3-Series to 16-Series High Accuracy Battery Monitor and Protector for Li-Ion, Li-Polymer, and LiFePO4 Battery Packs*. 2020. Disponível em: <https://www.ti.com/lit/ds/symlink/bq76952.pdf?ts=1714587221829&ref_url=https%253A%252F%252Fwww.google.com%252F>. Acesso em: Mai. 2024.

THE PIHUT. 2024. Disponível em: <https://thepihut.com>. Acesso em: Dez. 2024.

THE REGISTER. *AWS hits $100B revenue run rate, expands margins, delivers most of Amazon's profit*. 2024. Disponível em: <https://www.theregister.com/2024/05/01/amazon_q1_2024/>. Acesso em: Jun. 2024.

THINGSBOARD. *ThingsBoard Open-source IoT Platform*. 2024. Disponível em: <https://thingsboard.io>. Acesso em: Dez. 2024.

TRAN, M. K. et al. Concept review of a cloud-based smart battery management system for lithium-ion batteries: Feasibility, logistics, and functionality. *Batteries*, MDPI, Fev 2022.

TSURUOKA, Y. Cloud computing - current status and future directions. *Journal of Information Processing*, Information Processing Society of Japan, v. 24, n. 2, p. 183–194, 2016.

WANG, Y. et al. Digital twin and cloud-side-end collaboration for intelligent battery management system. *Journal of Manufacturing Systems*, Elsevier, v. 62, 2022.

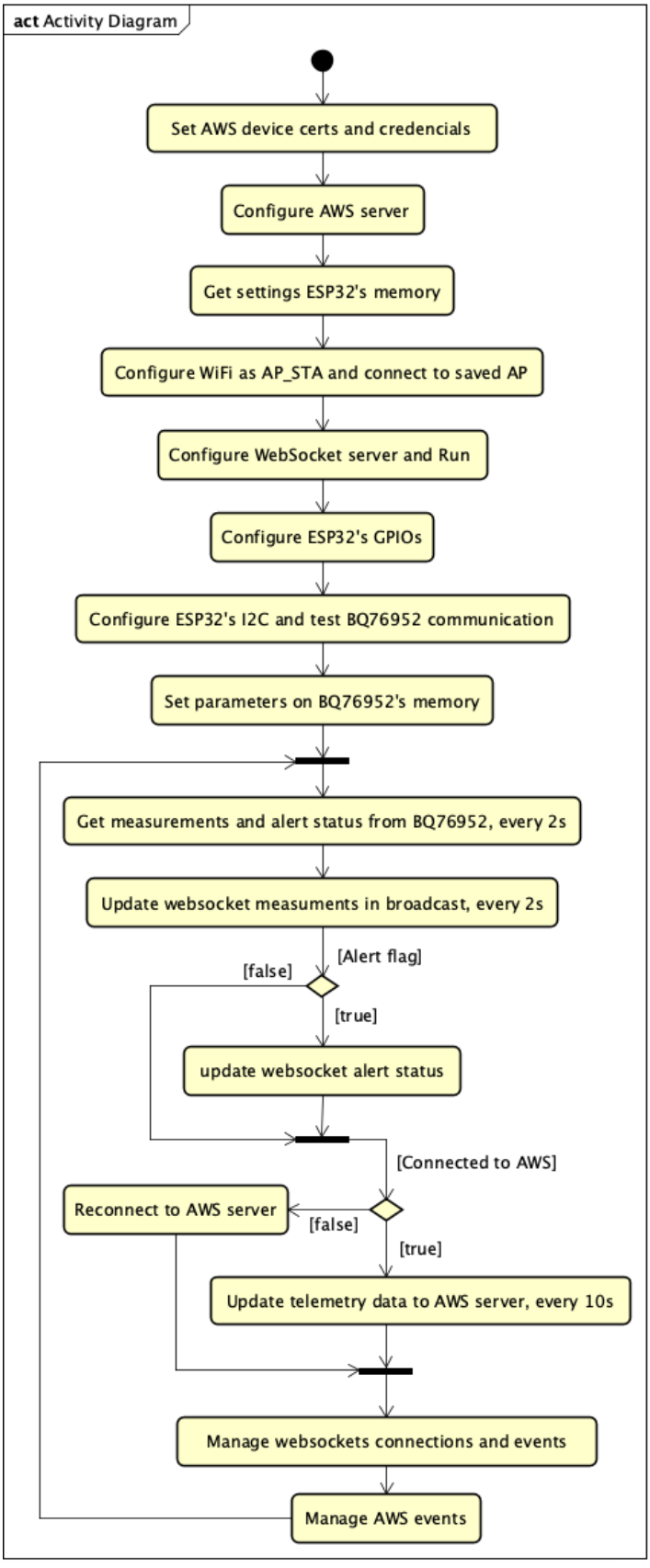
WEIHAN, L. et al. Digital twin for battery systems: Cloud battery management system with online state-of-charge and state-of-health estimation. *Journal of Energy Storage*, ELSEVIER, v. 30, 2020.

WITTIG, A.; WITTIG, M. *Amazon Web Services in Action*. Nova York: Manning Publications Co., 2016.

WU, J. et al. Cloud-to-edge based state of health estimation method for lithium-ion battery in distributed energy storage system. *Journal of Energy Storage*, Elsevier, v. 41, 2021.

ZUOLU, W. et al. A review on online state of charge and state of health estimation for lithium-ion batteries in electric vehicles. *Energy Reports*, Elsevier, n. 7, Ago 2021.

APÊNDICE A – DIAGRAMA DE ATIVIDADES DO MÓDULO



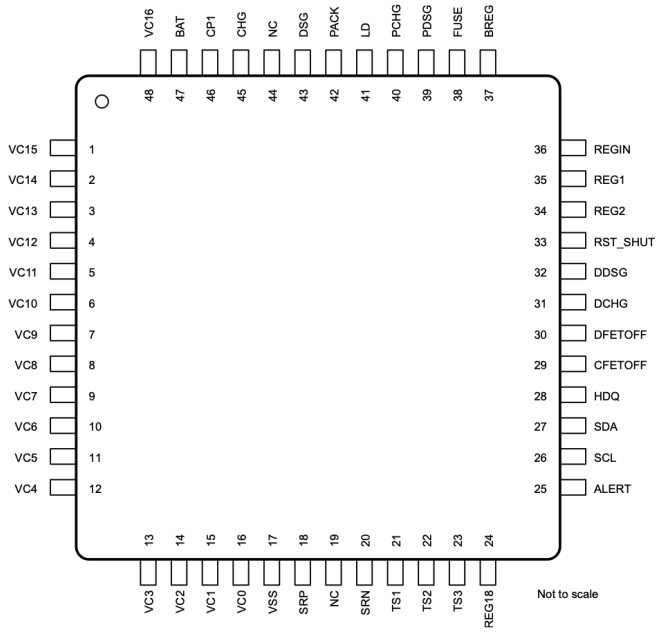
APÊNDICE B – ALGORITMO PARA ESTIMAÇÃO DO SOC NA NUVEM

```
import boto3
import numpy as np
import pandas as pd
import awswrangler as wr
import time
import joblib
import tensorflow as tf
from tensorflow.keras.models import load_model
from keras.src.legacy.saving import legacy_h5_format

iot_client=boto3.client('iot-data')
topic = "esp32_bms/sub"

timestream_client = boto3.client('timestream-query')

query = """
SELECT * FROM "esp32_bmsDB"."esp32_bmsTable" WHERE time between
ago(15m) and now() ORDER BY time DESC LIMIT 1 """

s3 = boto3.client('s3')
s3.download_file('esp32-bms-bucket', 'soc_model.h5', 'soc_model.h5')

RNA_model = legacy_h5_format.load_model_from_hdf5("soc_model.h5",
custom_objects={'mse': 'mse'})

def predict_soc(model, current, voltage, temperature, soc_previous):
    input_data = np.array([[current, voltage, temperature,
    soc_previous]])
    predicted_soc = model.predict(input_data,verbose=0)
    return predicted_soc[0][0]

soc_pred = [0,0]
result = wr.timestream.query(sql=query)
```

```
stack = float(result.at[0,'stack'])
current = float(result.at[0,'current'])
temp = float(result.at[0,'celltemp'])
soc = float(result.at[0,'soc'])
soc_pred[1] = soc

while 1:

    result = wr.timestream.query(sql=query)

    stack = float(result.at[0,'stack'])
    current = float(result.at[0,'current'])
    temp = float(result.at[0,'celltemp'])
    soc = float(result.at[0,'soc'])

    soc_pred[0] = predict_soc(RNA_model,current,stack,temp,
    soc_pred[1])
    soc_pred[1] = soc_pred[0]

    iot_response = "{\"soc_aws\":" + str(soc_pred[0]) + "}"
    iot_client.publish(topic=topic, payload=iot_response)

    time.sleep(60)
```

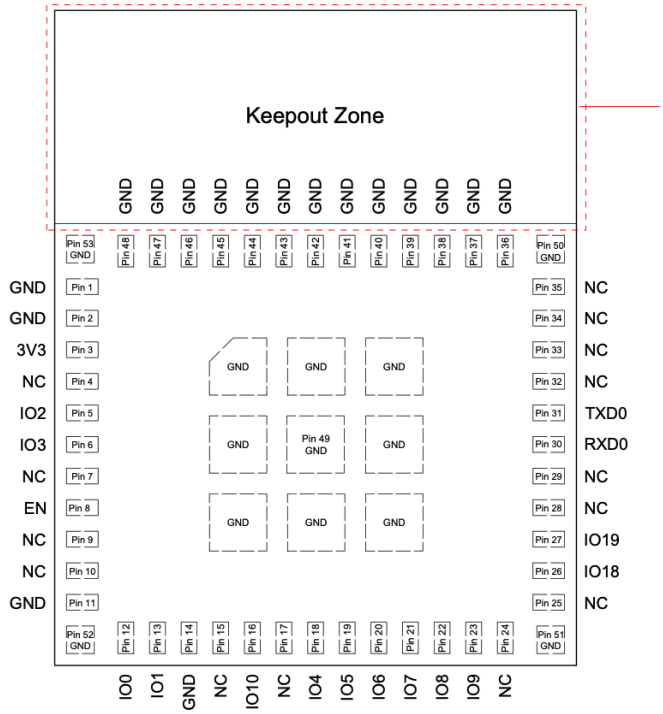
ANEXO A – *LAYOUTS* DE PINAGEM DO CHIP BQ76952 E DO MÓDULO ESP32-C3-MINI-1.

(a) Pinagem do chip BQ76952.



Fonte: (TEXAS INSTRUMENTS, 2020).

(b) Pinagem do módulo ESP32-C3-MINI-1.



Fonte: (ESPRESSIF SYSTEMS, 2022).