INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DE SANTA CATARINA - CÂMPUS FLORIANÓPOLIS
DEPARTAMENTO ACADÊMICO DE METAL MECÂNICA
CURSO DE GRADUAÇÂO EM ENGENHARIA MECATRÔNICA

JOSÉ GUALBERTO SANTOS LIMA

Sistema de Monitoramento por Geolocalização de Dispositivos IoT

FLORIANÓPOLIS, 2025.

INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DE SANTA CATARINA - CÂMPUS FLORIANÓPOLIS
DEPARTAMENTO ACADÊMICO DE METAL MECÂNICA
CURSO DE GRADUAÇÂO EM ENGENHARIA MECATRÔNICA

JOSÉ GUALBERTO SANTOS LIMA

Sistema de Monitoramento por Geolocalização de Dispositivos IoT

Trabalho de Conclusão de Curso submetido ao Instituto Federal de Educação, Ciência e Tecnologia de Santa Catarina como parte dos requisitos para obtenção do título de Engenheiro Mecatrônico.

Orientador:
Prof. VALDIR NOLL, Dr.Eng

FLORIANÓPOLIS, 2025.

Ficha de identificação da obra elaborada pelo autor,
através do Programa de Geração Automática da Biblioteca do IFSC.

Lima, José
Sistema de Monitoramento por Geolocalização de Dispositivos IoT / José Lima ; orientador, Valdir Noll, 2025.
67 p.

Trabalho de Conclusão de Curso (graduação) - Instituto Federal de Santa Catarina, Campus Florianópolis, Graduação em Engenharia mecatronica, Florianópolis, 2025.

Inclui referências.

1. Engenharia mecatronica. 2. Engennharia Mecatrônica. 3. Eletrônica. 4. Software. I. Noll, Valdir. II. Instituto Federal de Santa Catarina. Graduação em Engenharia mecatronica. III. Título.

Sistema de Monitoramento por Geolocalização de Dispositivos IoT

JOSÉ GUALBERTO SANTOS LIMA

Este trabalho foi julgado adequado para obtenção do título de Engenheiro em Mecatrônico e aprovado na sua forma final pela banca examinadora do Curso Engenharia Mecatrônica do Instituto Federal de Educação, Ciência e Tecnologia de Santa Catarina.

Florianópolis, 21 de agosto, 2025.

Banca Examinadora:

Documento assinado digitalmente
VALDIR NOLL
Data: 21/08/2025 21:05:58-0300
CPF: ***.839.000-**
Verifique as assinaturas em https://v.ufsc.br

Prof. Valdir Noll, Dr. Eng
Orientador
Instituto Federal de Santa Catarina

Documento assinado digitalmente
MAURICIO EDGAR STIVANELLO
Data: 21/08/2025 21:10:38-0300
CPF: ***.390.629-**
Verifique as assinaturas em https://v.ufsc.br

Prof. Maurício Edgar Stivanello, Dr. Eng.
Avaliador
Instituto Federal de Santa Catarina

Documento assinado digitalmente
GREGORY CHAGAS DA COSTA GOMES
Data: 25/08/2025 11:19:35-0300
CPF: ***.008.969-**
Verifique as assinaturas em https://v.ufsc.br

Gregory Chagas Da Costa Gomes, M. Eng.
Avaliador
Instituto Federal de Santa Catarina

Dedico este trabalho à minha mãe, que sempre foi meu alicerce, meu exemplo de força e de amor. Pela sua coragem, dedicação e por nunca deixar de acreditar em mim, mesmo nos momentos em que eu mesmo duvidei. Tudo o que conquistei até aqui carrega um pedaço do que aprendi com você.

AGRADECIMENTOS

Ao longo desta caminhada, tive o apoio de diversas pessoas que, de diferentes formas, contribuíram para que este trabalho se tornasse realidade. A todas elas, sou profundamente grato.

Mas, em especial, agradeço à minha namorada Hyasmin, por todo o carinho e paciência, e especialmente por me ajudar a manter o foco mesmo nos momentos de cansaço e desânimo. Sua presença foi essencial para que eu não desistisse e seguisse firme até o final.

Também agradeço ao meu amigo Samuel Cardoso, que foi meu parceiro de jornada ao longo de toda a graduação. Sempre disposto a colaborar, esteve presente em inúmeros trabalhos, apresentações e projetos, oferecendo apoio nas dificuldades e contribuindo diretamente para a concretização desta graduação.

Agradeço ainda ao meu professor orientador Valdir Noll, por sua orientação, disponibilidade e apoio ao longo deste trabalho.Sua experiência e conselhos foram fundamentais para a estruturação e conclusão deste projeto.

A vocês, minha sincera gratidão.

“O seu esforço não fará o menor sentido, se você não acredita em si mesmo.”

Might Guy

RESUMO

A crescente demanda por soluções de rastreamento em tempo real tem impulsionado o desenvolvimento de tecnologias de monitoramento aplicadas à logística, especialmente no varejo. Este trabalho apresenta o desenvolvimento de um sistema de monitoramento para distribuição de produtos, composto por um microcontrolador, um modem LTE, GPS e sensores Bluetooth de baixo consumo. O objetivo é oferecer uma solução de baixo custo e fácil implementação para empresas que realizam entregas de produtos destinados a testes e posterior devolução às filiais. O microcontrolador atua como unidade central, gerenciando a comunicação com o modem, GPS e sensores Bluetooth. O GPS fornece as coordenadas da trajetória em tempo real, transmitidas via rede celular, enquanto os sensores Bluetooth verificam a permanência dos produtos dentro de um raio de segurança de até 3 metros. Nos testes realizados, o sistema obteve comunicação estável via LTE-Cat M1, precisão de localização de até 3 metros e detecção confiável de etiquetas BLE.

Palavras-chave: Geolocalização. Rastreamento. Sensores. Automação. Logística.

ABSTRACT

The growing demand for real-time tracking solutions has driven the development of monitoring technologies applied to logistics, particularly in the retail sector. This work presents the development of a monitoring system for product distribution, composed of a microcontroller, an LTE modem, GPS, and low-power Bluetooth sensors. The goal is to provide a low-cost, easy-to-implement solution for companies that deliver products intended for testing and subsequent return to their branches. The microcontroller acts as the central unit, managing communication with the modem, GPS, and Bluetooth sensors. The GPS provides real-time trajectory coordinates, transmitted via the cellular network, while the Bluetooth sensors verify whether the products remain within a security radius of up to 3 meters. In the tests carried out, the system achieved stable communication via LTE-Cat M1, location accuracy of up to 3 meters, and reliable detection of BLE tags.

Keywords: Geolocation. Tracking. Sensors. Automation. Logistics.

LISTA DE FIGURAS

LISTA DE TABELAS

SUMÁRIO

1 INTRODUÇÃO

A transformação digital tem impulsionado mudanças significativas na forma como diversos setores operam, especialmente no contexto da logística e da distribuição de produtos. A adoção de soluções automatizadas tem permitido a substituição de processos manuais por sistemas mais ágeis, confiáveis e acessíveis, o que tem sido essencial para atender às novas demandas de eficiência e rastreabilidade no transporte de mercadorias (AL-SAQQA; COLEGAS, 2014).

Entre os principais vetores dessa transformação está a Internet das Coisas (Internet of Things – IoT), conceito proposto por Kevin Ashton em 1999, que envolve a interconexão de objetos físicos a sistemas computacionais por meio de sensores, comunicação em rede e tomada de decisão autônoma (ASHTON, 2009). A IoT permite que dispositivos atuem de forma inteligente, coletando, processando e transmitindo informações em tempo real, o que amplia as possibilidades de aplicação em setores como a logística e o varejo (CIUFFOLETTI, 2018).

Esse tipo de integração tecnológica tem permitido o desenvolvimento de sistemas mais compactos, modulares e com menor consumo energético, viabilizando aplicações que anteriormente dependiam de infraestrutura complexa e altos investimentos. A possibilidade de monitorar objetos ou bens de valor agregado, que precisam ser deslocados fisicamente, contribui diretamente para a melhoria da segurança, do planejamento logístico e da experiência do cliente.

No setor varejista, especialmente em segmentos como vestuário e calçados, a rastreabilidade de produtos durante o transporte ainda é limitada, devido ao custo elevado das soluções tradicionais e à falta de flexibilidade dos sistemas existentes. Essa limitação impacta principalmente pequenas e médias empresas, que operam com margens reduzidas e não dispõem de recursos para investir em tecnologias voltadas originalmente para cargas de alto valor agregado ou para veículos de grande porte. Como consequência, a gestão logística enfrenta dificuldades em garantir a visibilidade, o controle e a integridade dos produtos ao longo de todo o trajeto, resultando em perdas, atrasos e extravios não identificados.

1.1 Justificativa

A crescente demanda por entregas rápidas, rastreáveis e seguras tem impulsionado a busca por soluções logísticas mais eficientes e acessíveis. No entanto, boa parte das tecnologias disponíveis no mercado foi projetada para rastrear cargas de alto valor agregado ou veículos de grande porte, apresentando custos elevados e infraestrutura complexa, muitas vezes inviáveis para empresas de pequeno e médio porte.

Esse cenário é especialmente crítico no varejo, onde o transporte de itens como roupas, calçados e acessórios exige controle e visibilidade durante todo o trajeto, mas não justifica o investimento em sistemas robustos e caros. A ausência de soluções adaptadas a esse contexto resulta em perdas logísticas, extravios, atrasos não identificados e dificuldade de comprovar a integridade e o trajeto dos produtos transportados.

Além das limitações financeiras, muitas empresas enfrentam obstáculos técnicos, como a necessidade de operar em áreas com conectividade limitada, a dificuldade de integrar diferentes tipos de dados logísticos (como localização e proximidade entre volumes), e a falta de interfaces simples para visualização e análise das informações coletadas. Isso compromete não apenas a eficiência operacional, mas também a capacidade de resposta diante de problemas, impactando diretamente a qualidade do serviço prestado e a satisfação dos clientes.

Neste contexto, torna-se relevante o desenvolvimento de uma solução específica para o rastreamento de ativos no varejo, que seja de baixo custo, de fácil implementação e com baixo consumo de energia. Um sistema com essas características pode permitir que empresas menores adotem práticas modernas de gestão logística, anteriormente acessíveis apenas a grandes operadores.

1.2 Definição do Problema

Empresas do setor varejista, especialmente de pequeno e médio porte, enfrentam dificuldades para monitorar de forma eficiente e acessível o deslocamento de seus produtos durante o processo de entrega. Os sistemas disponíveis no mercado são, em grande parte, desenvolvidos para aplicações robustas, com foco em cargas de alto valor e frotas de grande escala, o que os torna economicamente inviáveis e tecnicamente desproporcionais quando aplicados ao transporte de itens como roupas, calçados e acessórios.

Além do custo elevado, essas soluções frequentemente exigem infraestrutura complexa e dependem de conectividade constante em áreas urbanas e rurais, o que limita sua aplicação em contextos onde há instabilidade de sinal ou baixa cobertura de rede. A ausência de integração entre informações sobre localização e outros dados relevantes – como a proximidade física entre os produtos e a central de controle – dificulta a gestão ativa das entregas e a identificação de desvios, atrasos ou extravios.

Como consequência, muitas empresas operam com baixa visibilidade sobre a posição e o estado dos produtos em trânsito, o que compromete o controle logístico, gera perdas operacionais e impacta negativamente a experiência do cliente.

Essa lacuna torna necessário o desenvolvimento de uma alternativa viável para o monitoramento de produtos de menor valor unitário, adaptada às limitações de

custo e infraestrutura enfrentadas por pequenos negócios. Ao focar na simplicidade, na acessibilidade e na adequação ao contexto varejista, uma solução com essas características pode viabilizar a rastreabilidade em ambientes com recursos limitados e promover maior controle e segurança nas entregas.

1.3 Objetivo Geral

Desenvolver um sistema autônomo, conectado a internet e de baixo custo, capaz de monitorar, em tempo real, o deslocamento de produtos no varejo durante o processo de entrega e devolução, detectando automaticamente eventuais desvios de rota ou afastamentos do ponto de controle, com foco na rastreabilidade, segurança e acessibilidade para pequenas e médias empresas.

1.4 Objetivos Específicos

Para atingir o objetivo geral proposto, este trabalho define os seguintes objetivos específicos:

a) Realizar uma revisão bibliográfica sobre soluções de monitoramento aplicadas à logística de produtos no varejo;

b) Levantar e analisar sistemas existentes no mercado, identificando suas características, limitações e custos;

c) Projetar uma solução integrada para rastreamento de produtos, considerando autonomia, conectividade e detecção de afastamentos;

d) Desenvolver um protótipo funcional de baixo custo que simule o processo de entrega e retorno dos produtos monitorados;

e) Implementar um mecanismo de alerta baseado na perda de proximidade entre o produto e o ponto de controle;

f) Propor melhorias na solução desenvolvida com base nos resultados dos testes realizados.

1.5 Estrutura do Trabalho

O Capítulo 1 apresenta a introdução ao tema, incluindo a justificativa, a definição do problema, os objetivos e a organização do documento. O Capítulo 2 expõe os conceitos teóricos necessários para o desenvolvimento do sistema, abordando temas como monitoramento, rastreamento de ativos e comunicação sem fio. O Capítulo 3 descreve o projeto e a construção do hardware utilizado, detalhando os componentes

empregados e sua integração. O Capítulo 4 trata do desenvolvimento do firmware responsável pela operação do sistema, incluindo a lógica de funcionamento e os processos de comunicação. No Capítulo 5, é descrito o sistema web de visualização, em que os dados de monitoramento são apresentados de forma acessível ao usuário. O Capítulo 6 reúne os testes realizados, os resultados obtidos e a análise do desempenho da solução proposta, finalizando com as considerações sobre o trabalho e sugestões para futuras melhorias.

2 FUNDAMENTAÇÃO TEÓRICA

2.1 Sistema de Posicionamento Global (GPS)

O sistema de posicionamento global (GPS) representa uma das tecnologias mais significativas para aplicações de monitoramento e rastreamento na atualidade, tendo se originado como um projeto militar denominado *NAVSTAR GPS*, desenvolvido pelo Departamento de Defesa dos Estados Unidos a partir de 1973, alcançando plena operacionalidade em 1995 (ASSIS, 2010). Este sistema revolucionário baseia-se em uma constelação de 24 satélites ativos distribuídos estrategicamente em seis planos orbitais distintos, cada um contendo quatro satélites com inclinação de 55 graus em relação ao equador terrestre, garantindo que pelo menos quatro satélites estejam visíveis em qualquer ponto do planeta a qualquer momento (GONçALVES; TAVARAYAMA, 2011). A arquitetura do sistema compreende três componentes essenciais: o segmento espacial, formado pelos satélites; o segmento de controle, composto por estações terrestres responsáveis pelo monitoramento e ajuste orbital; e o segmento do usuário, que inclui os diversos tipos de receptores GPS disponíveis no mercado (MONICO, 2000).

O princípio de operação do GPS reside no método de trilateração, onde os receptores calculam sua posição através da medição precisa do tempo que os sinais de rádio emitidos pelos satélites levam para alcançá-los (PAZ; CUGNASCA, 2010). Cada satélite transmite continuamente informações críticas que incluem suas efemérides (dados orbitais precisos), marcações temporais sincronizadas por relógios atômicos de extraordinária precisão, e códigos pseudo-aleatórios que permitem a identificação única de cada satélite. O receptor GPS realiza cálculos complexos comparando o tempo de emissão marcado no sinal com o momento de recepção, determinando assim a distância através da relação fundamental entre tempo e velocidade da luz (MACHADO, 2010). Receptores modernos apresentam arquitetura multicanal, permitindo o rastreamento simultâneo de múltiplos satélites – tipicamente até doze – o que melhora significativamente tanto a precisão quanto a confiabilidade do posicionamento obtido.

A precisão do sistema GPS, no entanto, está sujeita a diversas fontes de erro que devem ser consideradas em aplicações críticas como o monitoramento de frotas para distribuição de produtos. Entre os fatores mais relevantes destacam-se os efeitos ionosféricos, onde a variação na densidade eletrônica da ionosfera (quantificada pelo *Total Electron Content* ou TEC) pode causar atrasos significativos na propagação dos sinais, introduzindo erros que podem variar consideravelmente dependendo das condições atmosféricas (SALOMONI, 2008). Outros fatores limitantes incluem a oclusão de sinais por obstáculos físicos como edifícios ou vegetação densa, que podem bloquear completamente os sinais ou causar efeitos de multicaminho, além

da geometria dos satélites visíveis, onde configurações com baixo *Dilution of Precision* (DOP) tendem a fornecer posicionamento mais confiável.

No contexto de aplicações logísticas, a tecnologia GPS oferece capacidades fundamentais para o rastreamento em tempo real de objetos móveis, com desempenho sujeito a variações ambientais, número de satélites disponíveis e qualidade dos receptores utilizados. Sua integração com tecnologias de comunicação sem fio viabiliza a transmissão contínua das coordenadas geográficas, permitindo o acompanhamento remoto e a análise de rotas em tempo real.

2.2 Tecnologias de Comunicação Celular

A evolução da mobilidade e das aplicações conectadas tem estimulado o uso de sistemas embarcados em diferentes setores, especialmente na área logística, onde há interesse crescente em soluções que permitam acompanhar a movimentação de cargas em tempo real. Esse contexto favorece a adoção de tecnologias de comunicação sem fio, especialmente aquelas licenciadas por operadoras móveis, como as redes 2G, 3G, 4G e 5G, que proporcionam conectividade confiável em diferentes ambientes (MENDES, 2013).

A tecnologia de segunda geração (2G), baseada no padrão GSM, foi inicialmente concebida para chamadas de voz e envio de mensagens de texto. Posteriormente, com o suporte a tecnologias como GPRS e EDGE, passou a oferecer recursos limitados de conexão com a internet, sendo uma alternativa viável para aplicações que exigem baixa largura de banda, como sistemas de rastreamento e telemetria (MENDES, 2013). Mesmo com a chegada de tecnologias mais recentes, a 2G ainda é utilizada em função de sua abrangência e da existência de uma grande base instalada de dispositivos compatíveis.

Com o advento da terceira geração (3G), tornou-se possível utilizar a rede móvel para aplicações mais complexas e que demandam maior capacidade de transmissão de dados. Essa evolução ocorreu por meio da padronização do *Universal Mobile Telecommunications System* (UMTS), que sucedeu o GSM, e da posterior adoção do *High-Speed Packet Access* (HSPA), que aumentou significativamente as velocidades de transferência. Esses avanços possibilitaram o acesso a serviços multimídia e a uma comunicação mais robusta, refletindo uma mudança nas expectativas dos usuários, que passaram a exigir acesso constante a conteúdos em tempo real e maior interatividade .

Já a tecnologia de quarta geração (4G), conhecida como *Long Term Evolution* (LTE), foi desenvolvida para entregar taxas de transmissão mais altas e conexões mais estáveis. Esse padrão utiliza uma infraestrutura baseada em IP, o que facilita a integração de serviços como voz, dados e vídeo em uma única rede (CORDEIRO, 2012). Além disso, o 4G oferece ganhos importantes em eficiência espectral e suporte

à mobilidade, sendo ideal para aplicações em dispositivos móveis e soluções IoT que requerem comunicação contínua e em tempo real. Segundo (TAKEDA, 2013), o padrão estabelece metas de desempenho que incluem taxas de até 100 Mbps em mobilidade e até 1 Gbps em ambientes fixos, promovendo avanços significativos na experiência do usuário.

2.2.1 Sistema GSM para comunicações móveis

O sistema GSM (*Global System for Mobile Communications*) representa uma das bases da telefonia móvel digital moderna, tendo sua origem nos esforços conjuntos de países europeus para criar um padrão unificado de comunicação celular no início dos anos 1980. O projeto foi inicialmente conduzido pela Comissão Europeia de Correios e Telecomunicações (CEPT) e, posteriormente, passou para o Instituto Europeu de Normas de Telecomunicações (ETSI), que definiu as especificações técnicas responsáveis pela ampla adoção do padrão em escala global.

A arquitetura do GSM é baseada em uma estrutura modular composta por Estações Transceptoras Base (BTS), Controladores de Estações Base (BSC) e Centrais de Comutação Móvel (MSC). Essa estrutura permite o gerenciamento eficiente das comunicações, garantindo funções como autenticação, roteamento e localização dos dispositivos conectados à rede. O sistema emprega canais distintos para envio (*uplink*) e recebimento (*downlink*) de dados, utilizando técnicas de multiplexação como TDMA (*Time Division Multiple Access*) e FDMA (*Frequency Division Multiple Access*), que dividem os recursos do espectro de rádio de forma organizada e eficiente.

Apesar da evolução das tecnologias móveis, o GSM ainda permanece relevante em aplicações onde a cobertura ampla e a confiabilidade da comunicação são essenciais, como em projetos de monitoramento distribuído em regiões com infraestrutura limitada.

2.2.2 Categoria M1 em Redes 4G (LTE)

A Categoria M1 (Cat-M1) é uma das alternativas de comunicação dentro das redes móveis de quarta geração (4G), baseadas na tecnologia *Long Term Evolution* (LTE), voltadas especificamente para aplicações de *Internet das Coisas* (IoT). O LTE é a tecnologia que dá suporte ao padrão 4G, oferecendo maior largura de banda, menor latência e melhor cobertura em comparação às gerações anteriores.

A Cat-M1 foi projetada para atender à demanda crescente por soluções de conectividade que fossem simultaneamente econômicas, eficientes e compatíveis com mobilidade. Suas principais características incluem o baixo consumo de energia, capacidade de manter conexões contínuas mesmo com baixa taxa de dados e boa cobertura de sinal — inclusive em ambientes internos ou de difícil acesso.

Essa tecnologia se destaca em aplicações que exigem comunicação em tempo real com baixa latência, como sistemas de rastreamento logístico, monitoramento remoto de ativos móveis e telemetria distribuída. Outro benefício importante é a possibilidade de conexão simultânea de múltiplos dispositivos em uma mesma célula da rede, o que a torna ideal para aplicações em larga escala.

No contexto brasileiro, a adoção da Cat-M1 por operadoras que oferecem serviços *Machine-to-Machine* (M2M) fortalece seu papel como uma opção estratégica para projetos que exigem conectividade confiável, mobilidade e gerenciamento remoto de dispositivos embarcados.

2.2.3 NB IoT

Em sistemas de monitoramento voltados à distribuição de produtos, a escolha da tecnologia de conectividade deve considerar fatores como cobertura, consumo energético e compatibilidade com infraestrutura existente. Nesse contexto, o NB-IoT (Narrowband Internet of Things) surge como uma alternativa eficiente para aplicações que não exigem o envio constante de grandes volumes de dados nem latência reduzida
.

O NB-IoT é uma tecnologia de comunicação de baixa potência e baixo consumo de energia, desenvolvida especificamente para atender demandas da Internet das Coisas (IoT). Ele opera utilizando faixas estreitas de frequência e oferece taxas de transmissão reduzidas, o que contribui para a extensão da vida útil de baterias em dispositivos remotos e com limitação de energia (TALAVERA *et al.*, 2017). Isso é particularmente relevante em sistemas de monitoramento distribuído, onde os dispositivos muitas vezes permanecem em locais de difícil acesso e operam por longos períodos sem intervenção humana.

Uma das vantagens do NB-IoT é que ele se baseia na infraestrutura já existente das redes LTE, além de estar em conformidade com os padrões das futuras redes 5G. Contudo, sua maior limitação está na dependência dessas redes celulares, o que pode gerar instabilidades em áreas com pouca cobertura, como regiões rurais, zonas remotas ou locais com barreiras físicas, como estruturas metálicas.

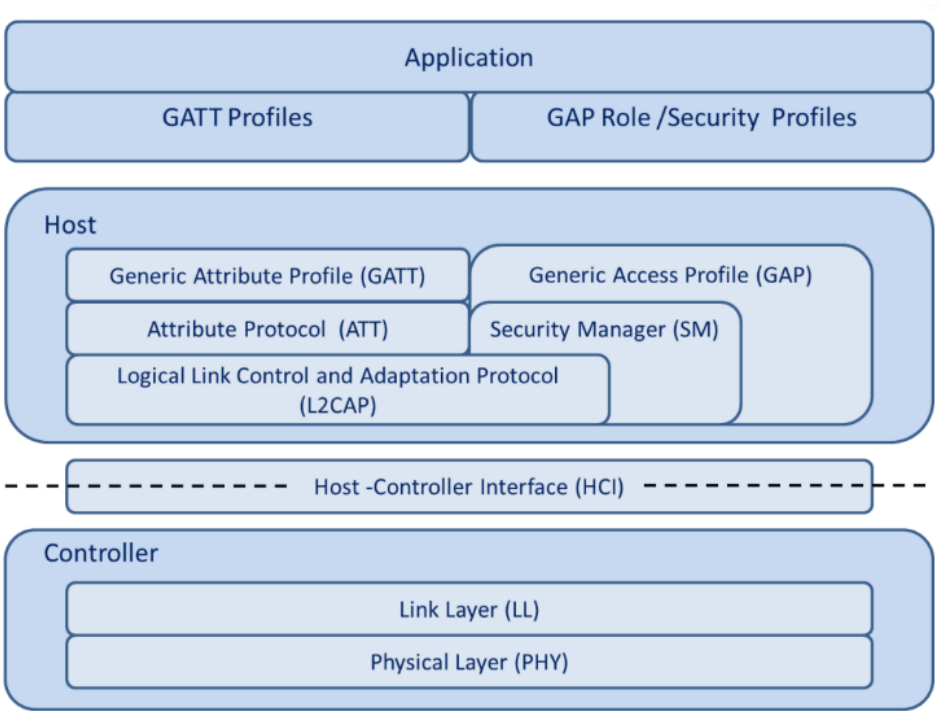
O uso de NB-IoT em aplicações de monitoramento tem se mostrado relevante em diversas áreas. De acordo com (TALAVERA *et al.*, 2017), entre 2006 e 2016, 62% dos artigos em inglês sobre IoT foram publicados como documentos acadêmicos, com destaque para aplicações de monitoramento remoto no setor agroindustrial. Entre essas aplicações, é comum o uso da tecnologia para o acompanhamento de parâmetros ambientais, como solo, ar, água, animais e plantas — áreas que compartilham requisitos semelhantes aos dos sistemas de rastreamento e monitoramento logístico, como durabilidade, economia de energia e confiabilidade de comunicação.

No cenário brasileiro, apesar do NB-IoT já estar sendo explorado comercialmente, sua adoção ainda é limitada. A operadora TIM é a única que atualmente disponibiliza essa tecnologia, mas a cobertura ainda não está plenamente ativa em todas as regiões do país. Isso se deve, em parte, ao fato de que o NB-IoT foi projetado inicialmente para dispositivos fixos, o que pode representar um desafio adicional quando aplicado a sistemas móveis.

2.3 Bluetooth Low Energy (BLE)

A tecnologia Bluetooth Low Energy (BLE), introduzida na versão 4.0 do Bluetooth, foi projetada com foco em aplicações de baixa potência, tornando-se uma alternativa eficiente para dispositivos IoT que demandam baixo consumo de energia e comunicação de curta distância (Bluetooth SIG, 2010). Diferentemente do Bluetooth Clássico, o BLE permite a operação prolongada com baterias de pequena capacidade, o que o torna especialmente vantajoso em cenários de rastreamento e monitoramento contínuo, como na logística de distribuição de produtos (GOMEZ; OLLER; PARADELLS, 2012). Para ampliar sua aplicação, dispositivos BLE podem operar em modo broadcasting, transmitindo dados de forma unidirecional sem a necessidade de vínculo com o receptor, ou em modo ponto a ponto, com troca bidirecional de informações (RENESAS, 2024). A Figura 1 apresenta a arquitetura BLE.

Figura 1 – Arquitetura do BLE



Fonte: Renesas, 2013

A arquitetura do BLE é estruturada por camadas e protocolos definidos

na especificação Bluetooth Core, sendo composta por controlador, host, camada de aplicação e interface HCI, o que garante interoperabilidade entre fabricantes (Bluetooth SIG, 2024). Dentro desse ecossistema, os dispositivos BLE podem ser organizados em três categorias: Clássico, LE (Low Energy) e Dual Mode, sendo os dispositivos Dual Mode os mais comuns em smartphones e laptops por suportarem ambas as versões (AFANEH, 2018). Esse tipo de compatibilidade é útil em sistemas distribuídos onde múltiplos tipos de dispositivos estão presentes, como sensores BLE utilizados no rastreamento de pacotes que se comunicam com uma unidade central, como a Raspberry Pi Pico W.

A utilização de Beacons BLE em projetos de monitoramento é uma solução prática para rastrear a movimentação de produtos em tempo real, sem depender de conexões constantes. Beacons funcionam como transmissores em broadcast, emitindo pacotes de dados que podem conter identificadores únicos e dados de localização estimada com base na força do sinal (ZAFARI; PAPAPANAGIOTOU, 2015). Esses pacotes são estruturados com preâmbulo, cabeçalho PDU, carga útil e CRC, podendo variar entre 8 e 47 bytes (Bluetooth SIG, 2010). Dentro da carga útil, 31 bytes estão disponíveis para dados, como endereço MAC e identificadores específicos como UUID, Major e Minor no protocolo iBeacon, utilizado pela Apple (STATLER, 2016). Esse tipo de identificação é valioso em ambientes logísticos para distinguir diferentes lotes, seções de armazenamento ou rotas de transporte.

2.4 Servidor Web e Dashboard em Tempo Real

Em sistemas de monitoramento baseados em Internet das Coisas (IoT), o servidor web desempenha um papel central na arquitetura da solução, sendo responsável pelo recebimento, processamento e disponibilização dos dados coletados pelos dispositivos remotos. Ele gerencia as comunicações com os nós sensores, armazena informações em bancos de dados estruturados e aplica regras de negócio para tratamento e validação dos dados. Sua estrutura geralmente inclui interfaces de programação de aplicações (APIs) para integração com outros sistemas e mecanismos de autenticação, criptografia e controle de acesso, garantindo a segurança e a integridade das informações.

Complementando o servidor, o dashboard em tempo real atua como a principal interface de visualização do sistema. Essa interface apresenta os dados monitorados de forma clara e acessível, permitindo ao usuário final interpretar os resultados rapidamente. Mapas interativos são utilizados para rastreamento geográfico de ativos, enquanto gráficos e tabelas auxiliam na análise temporal de eventos e no reconhecimento de padrões de operação. Além disso, é comum a presença de ferramentas para configuração de alertas e notificações, ampliando a capacidade de

resposta diante de condições anômalas ou falhas.

Para viabilizar a comunicação entre os dispositivos e o servidor web de forma leve e eficiente, é amplamente adotado o protocolo MQTT (Message Queuing Telemetry Transport). Baseado no modelo publish/subscribe, o MQTT é ideal para ambientes com largura de banda limitada e dispositivos com restrições de energia. Nesse modelo, os dispositivos sensores publicam mensagens em tópicos específicos, enquanto o servidor — ou outros sistemas interessados — se inscreve nesses tópicos para receber os dados. O broker MQTT atua como intermediário dessa comunicação, promovendo o desacoplamento entre os emissores e os receptores de mensagens e contribuindo para a escalabilidade e modularidade do sistema.

Além de seu papel como roteador de mensagens, o broker MQTT oferece funcionalidades adicionais que fortalecem sua robustez em sistemas de monitoramento em tempo real. Entre essas funcionalidades estão o controle de qualidade de serviço (QoS), com três níveis que determinam a confiabilidade da entrega das mensagens; o armazenamento de mensagens retidas, que permite que novos assinantes recebam imediatamente o último valor publicado; e a autenticação via credenciais ou certificados digitais, reforçando a segurança da comunicação. Com esses recursos, o broker MQTT se consolida como uma tecnologia-chave em aplicações IoT que exigem comunicação contínua, eficiente e segura entre múltiplos dispositivos distribuídos.

3 APLICAÇÃO

Durante a pandemia da COVID-19, as restrições de contato físico e a necessidade de distanciamento social impactaram significativamente a indústria de roupas e o varejo. Com as lojas físicas fechadas ou operando com limitações, muitas empresas registraram perdas expressivas nas vendas. Como alternativa, alguns varejistas adotaram um modelo de envio de roupas diretamente para clientes com maior poder aquisitivo: o consumidor recebia em casa uma seleção de peças para avaliação, escolhia o que desejava comprar e devolvia o restante.

Neste contexto, o sistema proposto se aplica como uma solução de rastreamento e controle de estoque durante esse processo de ida e retorno dos produtos. Na filial, os itens selecionados são acondicionados em uma caixa equipada com uma unidade central composta por uma Raspberry Pi Pico W e um módulo SIM7070G. Cada peça recebe uma etiqueta BLE com endereço MAC único, previamente cadastrada no sistema no momento da preparação do envio.

Quando a caixa sai da filial rumo ao cliente, o módulo SIM7070G fornece a localização GPS do transporte, permitindo acompanhar a rota e identificar desvios. Simultaneamente, o sistema realiza varreduras periódicas (scan) para verificar a presença de todas as tags BLE associadas ao envio, assegurando que nenhuma peça seja extraviada durante o trajeto.

Ao chegar à residência do cliente, a caixa permanece equipada com o módulo central e continua transmitindo dados de localização — que agora apontam a posição fixa no endereço do destinatário — e presença das tags BLE dentro de um raio de até 3 metros. Assim, caso qualquer peça saia desse alcance sem registro prévio no sistema, um alerta é gerado para a central de monitoramento.

Quando o cliente decide quais peças deseja adquirir, informa à loja, que identifica os MACs das tags correspondentes. As peças não adquiridas são devolvidas à caixa, e o sistema confirma novamente sua presença. Na etapa de retorno, o procedimento de rastreamento e monitoramento se repete, garantindo que o processo de devolução ocorra de forma segura e controlada.

Essa abordagem permite visibilidade total do ciclo de entrega e devolução, reduzindo perdas e extravios, e assegurando maior confiabilidade no processo logístico, especialmente em operações envolvendo produtos de valor variável e envio direto ao cliente.

3.1 Fluxo do Sistema de Monitoramento

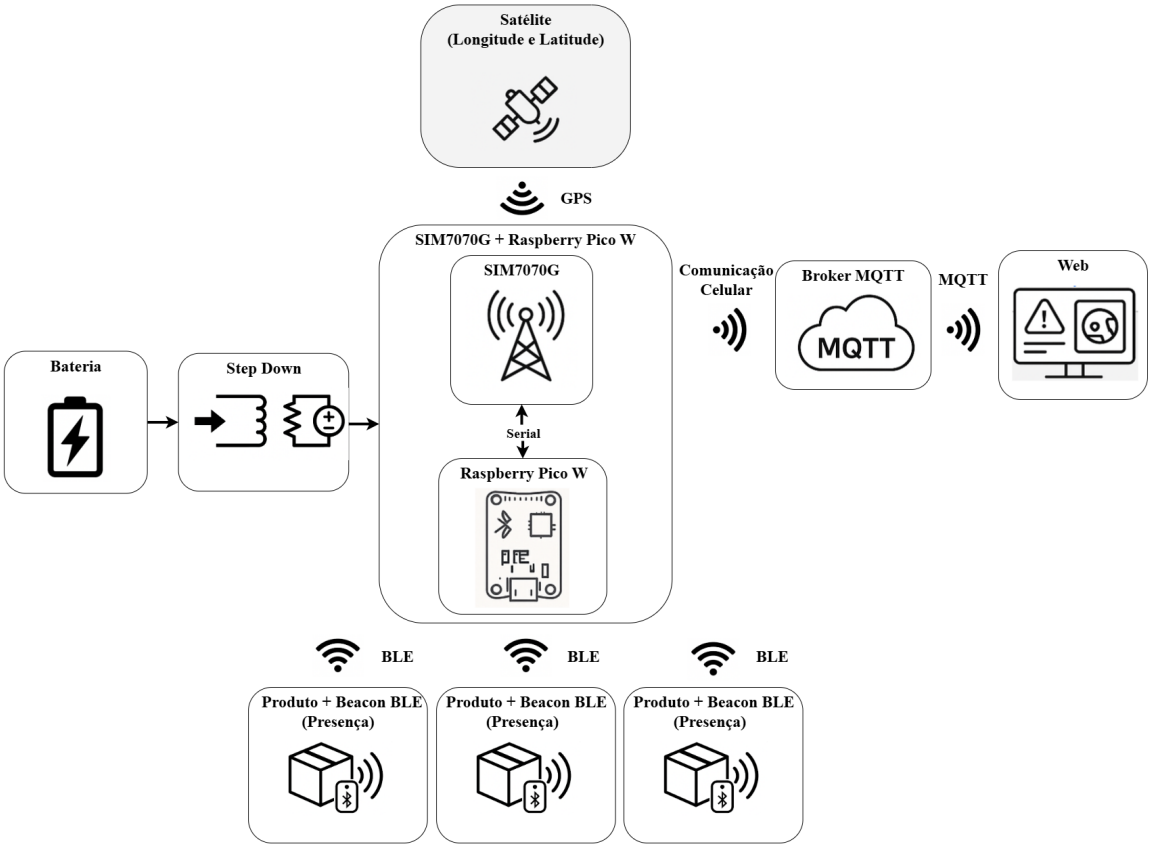
O sistema proposto estabelece um fluxo integrado de monitoramento de equipamentos que combina tecnologias de geolocalização, identificação por proximi-

dade e transmissão de dados. O processo inicia com a aquisição de coordenadas geográficas através do módulo GPS, que capta sinais de satélites para determinação posicional com precisão. Paralelamente, ocorre a identificação por proximidade via escaneamento periódico BLE das tags vinculadas aos equipamentos monitorados, garantindo a associação inequívoca entre a localização e o equipamento específico.

A transmissão dos dados coletados é realizada através de redes celulares IoT (LTE-M/NB-IoT), selecionadas por suas características de baixo consumo energético e ampla cobertura geográfica, fatores essenciais para aplicações móveis. O protocolo MQTT é empregado para o envio eficiente desses dados ao servidor central, assegurando a entrega confiável mesmo em condições de conectividade limitada.

A arquitetura final do sistema contempla um painel de controle web que permite a visualização em tempo real das informações coletadas, além do armazenamento de dados históricos para análise posterior. Essa estrutura integrada possibilita não apenas o monitoramento contínuo dos equipamentos, mas também a geração de relatórios e alertas automáticos, formando uma solução completa para gestão de ativos móveis. A Figura 2 apresenta de forma esquemática esse fluxo operacional, demonstrando a interligação entre os diversos componentes tecnológicos que compõem o sistema.

Figura 2 – Fluxo do sistema de monitoramento de equipamentos



Fonte: Do Autor, 2025

A partir do diagrama apresentado, observa-se que cada etapa do fluxo operacional está interligada, permitindo que as informações de localização e presença dos equipamentos sejam registradas e transmitidas de forma contínua. Essa integração garante que o sistema seja capaz de detectar desvios de rota, extravios ou afastamentos não autorizados em tempo hábil, possibilitando ações corretivas imediatas. Além disso, o armazenamento dos dados históricos viabiliza a análise de desempenho e a otimização dos processos logísticos, contribuindo para maior eficiência e segurança nas operações.

4 DESENVOLVIMENTO DO HARDWARE

Este capítulo apresenta o desenvolvimento do hardware utilizado no sistema de monitoramento proposto, abordando os componentes físicos, suas funções, interligações e critérios de seleção. Por meio de diagramas e descrições técnicas, busca-se fornecer uma representação clara da composição do sistema, evidenciando a lógica de funcionamento, a distribuição de energia e a comunicação entre os módulos. Além de documentar as especificações adotadas, esta seção serve como referência para a replicação do protótipo e para a análise da conformidade com os objetivos definidos.

4.1 Análise dos componentes

Este tópico apresenta uma análise comparativa entre os componentes utilizados no sistema proposto e outras alternativas viáveis disponíveis no mercado. A seleção dos dispositivos adotados considerou critérios como custo, disponibilidade nacional, consumo energético, facilidade de integração com microcontroladores e compatibilidade com aplicações de monitoramento logístico. As tabelas a seguir organizam essas comparações de forma categorizada, com destaque para controladoras, módulos de comunicação e fontes de alimentação.

Tabela 1 – Comparativo de controladoras principais

Componente	**Características**	**Observações (Motivo da Escolha)**
Raspberry Pi Pico W	Baixo consumo, suporte a MicroPython, Wi-Fi integrado	Escolhida por oferecer boa relação custo-benefício, baixo consumo e facilidade de programação em MicroPython, além de atender às necessidades do sistema.
ESP32	Wi-Fi e BLE integrados, mais processamento, maior consumo	Avaliada como alternativa, mas descartada por apresentar consumo energético mais elevado e recursos acima do necessário para a aplicação.
Arduino MKR GSM 1400	Conectividade celular nativa, maior custo	Considerada inviável devido ao custo elevado e recursos que ultrapassam a demanda do projeto.

Tabela 2 – Comparativo de módulos de comunicação e rastreamento

Componente	**Características**	**Observações (Motivo da Escolha)**
SIM7070G	GPS, LTE-M, NB-IoT e BLE integrados, baixo consumo	Escolhido por integrar múltiplas tecnologias em um único módulo, reduzindo complexidade e consumo energético, além de atender plenamente aos requisitos do sistema.
SIM7600E-H	4G LTE com GPS, maior desempenho e consumo	Considerado mais potente, porém descartado devido ao consumo mais elevado e recursos além do necessário para a aplicação.
Quectel BG95	NB-IoT e Cat M1, menor consumo, menor disponibilidade	Boa opção em consumo, mas a baixa disponibilidade no mercado nacional inviabilizou sua adoção.
GPS + ESP32 (Wi-Fi/BLE)	Solução econômica, depende de Wi-Fi local	Alternativa mais barata, porém não atende ao requisito de comunicação contínua em locais sem Wi-Fi, inviabilizando seu uso.

Tabela 3 – Comparativo de fontes de alimentação

Componente	**Características**	**Observações (Motivo da Escolha)**
2x 18650 + MT3608	Alta capacidade, boost para 5,2 V com MT3608	Escolhido por fornecer boa autonomia energética e permitir ajuste de tensão para atender ao módulo SIM7070G, com fácil reposição das baterias.
LiPo + PowerBoost 1000C	Carregamento USB, regulação embutida, baixa corrente	Boa solução integrada, mas descartada devido à menor capacidade e corrente insuficiente para picos de consumo do sistema.
Baterias AAA ou alcalinas	Alta disponibilidade, baixa densidade energética	Opção simples e barata, porém inviável devido à baixa autonomia e necessidade de trocas frequentes.
Baterias com PCM/PMIC	Proteção integrada, exige maior conhecimento técnico	Oferece segurança adicional, mas maior complexidade de implementação e custo não justificaram o uso.

Com base na análise apresentada, optou-se pela utilização da Raspberry Pi Pico W como controladora principal devido ao baixo consumo energético, à compatibilidade com MicroPython e à disponibilidade nacional, características que equilibram desempenho e custo. Para comunicação e rastreamento, o SIM7070G foi selecionado por integrar GPS, LTE-M, NB-IoT e BLE em um único módulo, reduzindo a complexidade de integração e o consumo total do sistema. A alimentação é fornecida por duas células 18650 associadas a um conversor MT3608 configurado para 5,2 V, garantindo maior autonomia e capacidade de suprir os picos de corrente exigidos pelo modem durante a transmissão de dados. Essas escolhas atendem aos requisitos de baixo custo, eficiência energética e viabilidade de implementação para aplicações de monitoramento logístico de pequeno e médio porte.

4.2 Raspberry Pi Pico W (Controladora Mestre)

A Raspberry Pi Pico W é responsável por centralizar o controle do sistema. Atuando como unidade mestre, ela realiza a comunicação serial com o módulo SIM7070G através da interface UART, utilizando comandos AT para requisitar dados de localização e estabelecer conexões MQTT.

Além disso, a Pico W processa os dados GPS recebidos, organiza as mensagens a serem transmitidas e monitora o estado dos periféricos conectados. Quando o sistema não está em uso, ela pode ser configurada para operar em modos de economia de energia, aumentando a autonomia do equipamento.

Figura 3 – Raspberry Pi Pico W

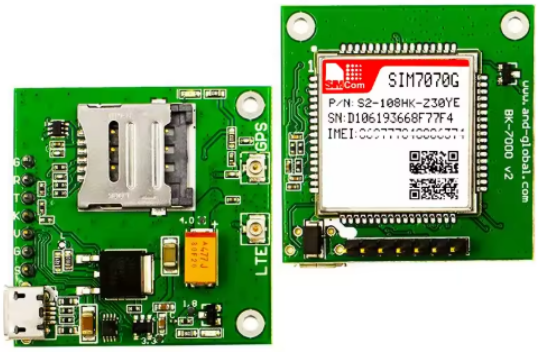
Fonte: Foundation (2023)

4.3 Modulo de comunicação de dados GPS e 4G

O SIM7070G desempenha a função de coletar coordenadas de GPS e enviar os dados ao servidor MQTT via conexão LTE-M. Esse módulo oferece suporte a múltiplos protocolos (MQTT, TCP/IP, HTTP), o que permite a integração direta com sistemas de IoT baseados em nuvem (Texto 2).

O módulo possui duas antenas externas conectadas via IPEX — uma destinada à recepção do sinal LTE e outra ao GPS —, o que melhora significativamente a qualidade da comunicação em áreas com sinal fraco (Luthfi; Karna; Mayasari, 2019).

Figura 4 – SIM7070G



Fonte: AND Wireless 2025.

4.4 Baterias 18650 e Gerenciamento de Energia

O sistema de alimentação foi projetado para garantir estabilidade e autonomia em ambientes móveis. Para isso, são utilizadas duas baterias 18650 de íons de lítio ligadas em série, formando um pack 2S com tensão nominal de 7,4 V e capacidade combinada de 3300 mAh. A ligação em série aumenta a tensão disponível, permitindo maior flexibilidade no uso de conversores de tensão dedicados aos diferentes componentes do sistema.

O circuito conta com um sistema de proteção integrado por meio de um módulo BMS (Battery Management System), que atua automaticamente para evitar sobrecarga (acima de 8,4 V), descarga profunda (abaixo de 6,0 V), sobrecorrente e curto-circuito. Essa proteção assegura a integridade das células ao longo do tempo, mesmo em situações de variação de carga ou falhas temporárias.

A energia fornecida pelo pack é regulada por dois conversores buck: um conversor LM2596 ajustado para fornecer 5 V ao módulo SIM7070G, e um conversor MP1584 configurado para entregar 3,3 V à Raspberry Pi Pico W. Ambos os conversores foram calibrados com o auxílio de um multímetro para garantir estabilidade mesmo durante picos de consumo, especialmente em transmissões de dados via LTE, que exigem maior corrente do módulo de comunicação.

A distribuição da energia é feita de forma dedicada: a Raspberry Pi Pico W recebe os 3,3 V diretamente do MP1584 via pino VSYS, enquanto o SIM7070G é alimentado pelos 5 V provenientes do LM2596. Essa separação evita interferências entre os componentes e contribui para a confiabilidade do sistema.

Os testes realizados indicaram que, com ciclos de envio configurados para intervalos de 15 minutos, a autonomia do sistema pode ultrapassar sete dias, dependendo da eficiência da rede celular e da qualidade das baterias utilizadas. Além disso, o uso de cabos com bitola apropriada (preferencialmente 22 AWG ou superior) se mostrou essencial para minimizar perdas por queda de tensão e garantir um desempenho estável do conjunto.

Figura 5 – Bateria

Fonte:Eletroparts, 2025

Figura 6 – MP1584

Fonte:Eletroparts, 2025

Figura 7 – LM2596

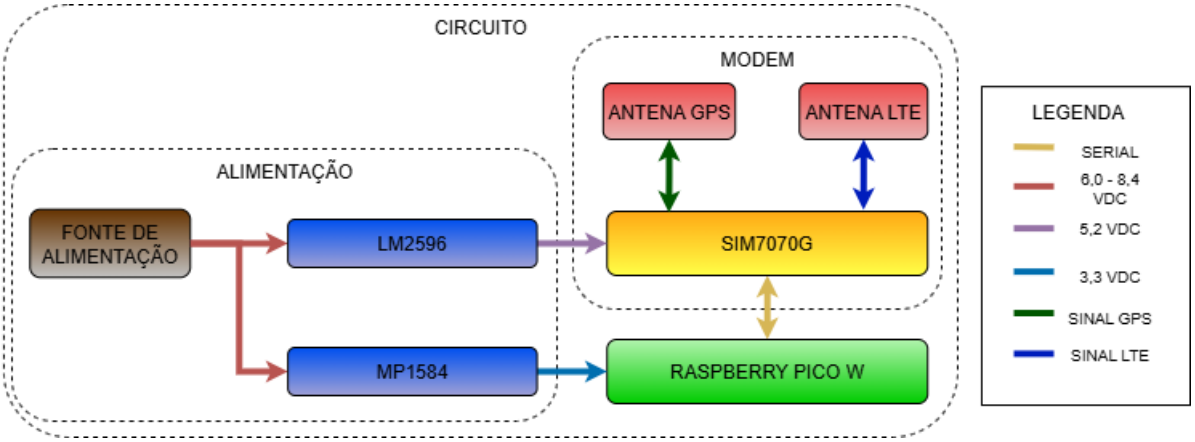
Fonte:Eletroparts, 2025

4.5 Diagrama de Sinais

A Figura 8 apresenta o diagrama de sinais que descreve a estrutura do sistema desenvolvido, destacando a interação entre os principais componentes eletroeletrônicos. A representação permite identificar o fluxo de sinais e os níveis de tensão utilizados, além das conexões responsáveis pela alimentação e comunicação entre os módulos.

A fonte de alimentação fornece energia diretamente à Raspberry Pi Pico W e ao conversor buck-boost. Este, por sua vez, eleva a tensão para 5,2 V, valor necessário para o funcionamento do módulo SIM7070G. A comunicação entre a Pico W e o SIM7070G ocorre por meio de interface serial, utilizada para controle e envio de dados. O SIM7070G está conectado a duas antenas externas, responsáveis pela recepção dos sinais GPS e LTE.

Figura 8 – Diagrama de sinais



Fonte: Do Autor, 2025

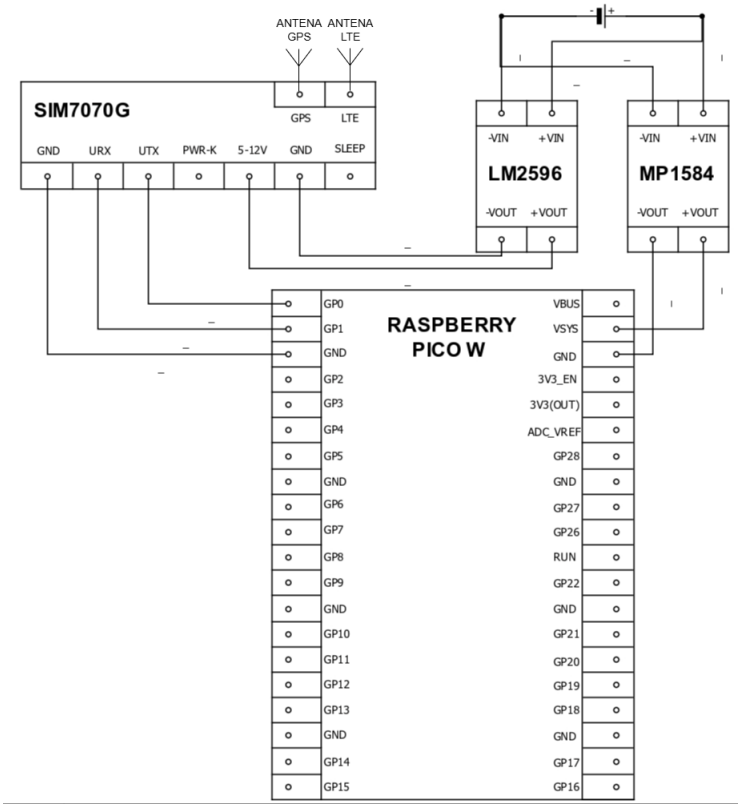
4.6 Circuito elétrico

A Figura 9 apresenta o diagrama elétrico do sistema embarcado, representando as conexões entre os principais componentes do circuito. A Raspberry Pi Pico W atua como unidade de controle, realizando a comunicação serial com o módulo SIM7070G por meio dos pinos GPIO0 (TX) e GPIO1 (RX), responsáveis pelo envio e recepção de dados.

A alimentação do SIM7070G é fornecida por um conversor MT3608, que eleva a tensão das baterias de 3,0 V a 4,2 V para uma saída estabilizada de 5,2 V, compatível com os requisitos operacionais do módulo. Já a Raspberry Pi Pico W é alimentada diretamente pela fonte, utilizando o pino VSYS, que suporta a faixa de tensão das baterias.

O esquema também evidencia as conexões de aterramento compartilhado entre os módulos, garantindo referência comum para os sinais elétricos. A disposição das ligações reflete a topologia adotada na montagem do protótipo físico, servindo como base para a implementação prática do sistema.

Figura 9 – Circuito elétrico

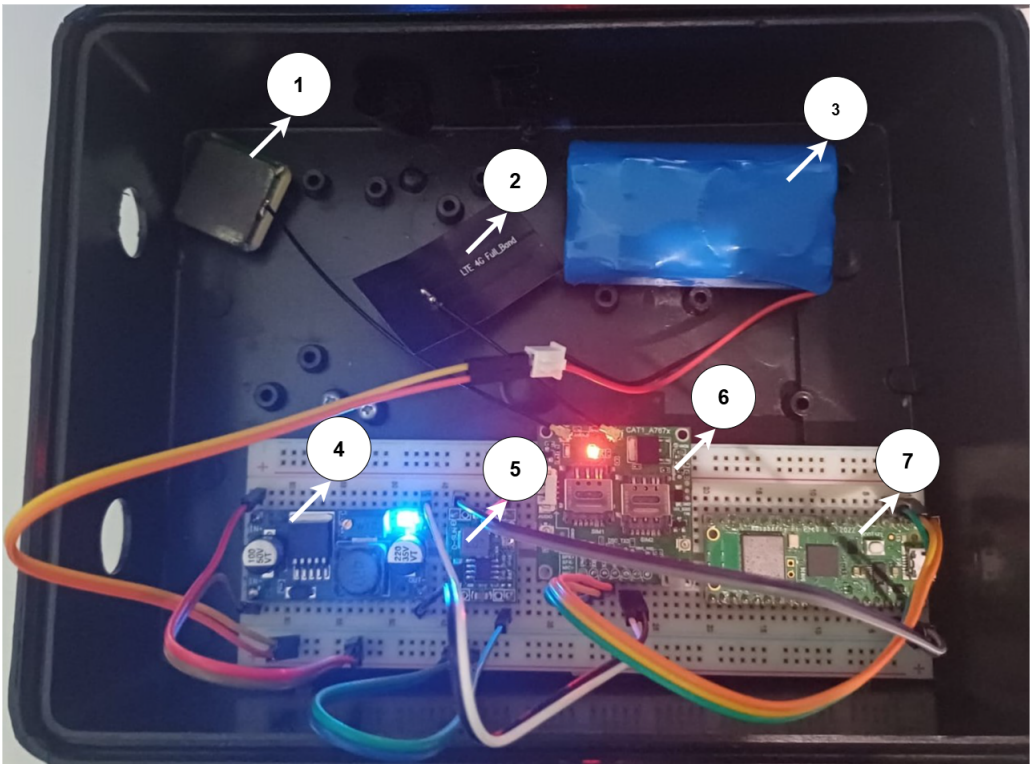


Fonte: Do Autor, 2025

4.7 Protótipo

A montagem do protótipo foi realizada com foco na funcionalidade e na integração dos componentes utilizados no sistema de monitoramento. Todos os módulos foram organizados em uma caixa plástica, com interligações feitas por meio de uma protoboard para facilitar ajustes e testes. O sistema é alimentado por uma bateria Li-ion e conta com dois conversores de tensão para fornecer os níveis adequados a cada módulo. A Figura 10 apresenta o protótipo final montado.

Figura 10 – Protótipo físico do sistema montado



Fonte: Do Autor, 2025

Legenda da Figura 10:

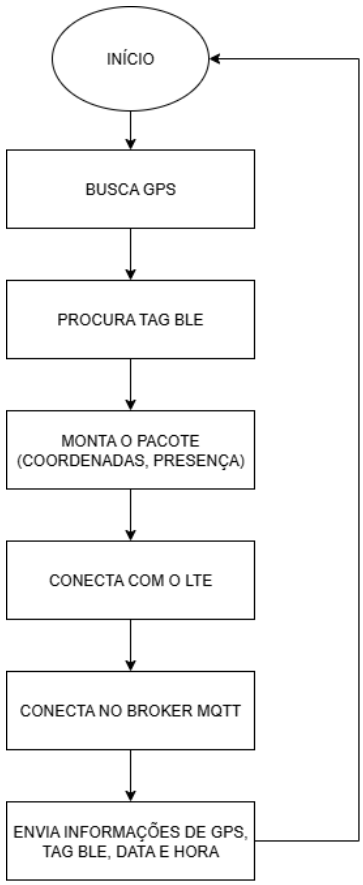
1. Antena GPS;
2. Antena LTE;
3. Bateria Li-ion (2S);
4. Conversor buck LM2596 (5 V);
5. Conversor buck MP1584 (3,3 V);
6. Módulo SIM7070G (LTE CAT-M e GNSS);
7. Raspberry Pi Pico W.

5 DESENVOLVIMENTO DO FIRMWARE

Nesta etapa do projeto, foi desenvolvido o firmware embarcado responsável pela coleta e transmissão de dados de localização geográfica e pela detecção de sinal Bluetooth. A implementação foi realizada na Raspberry Pi Pico W, escolhida por seu baixo consumo energético, conectividade Wi-Fi integrada e compatibilidade com bibliotecas em MicroPython, o que facilita a prototipação e a integração com os módulos utilizados.

O sistema opera em ciclos sucessivos, executando sequencialmente a leitura das coordenadas GPS, a verificação da presença de um dispositivo BLE previamente definido e, por fim, a transmissão dessas informações por meio do protocolo MQTT. A lógica geral de funcionamento do firmware está representada na Figura 11, que ilustra as etapas principais de execução e os pontos de verificação realizados em cada ciclo.

Figura 11 – Fluxograma Geral



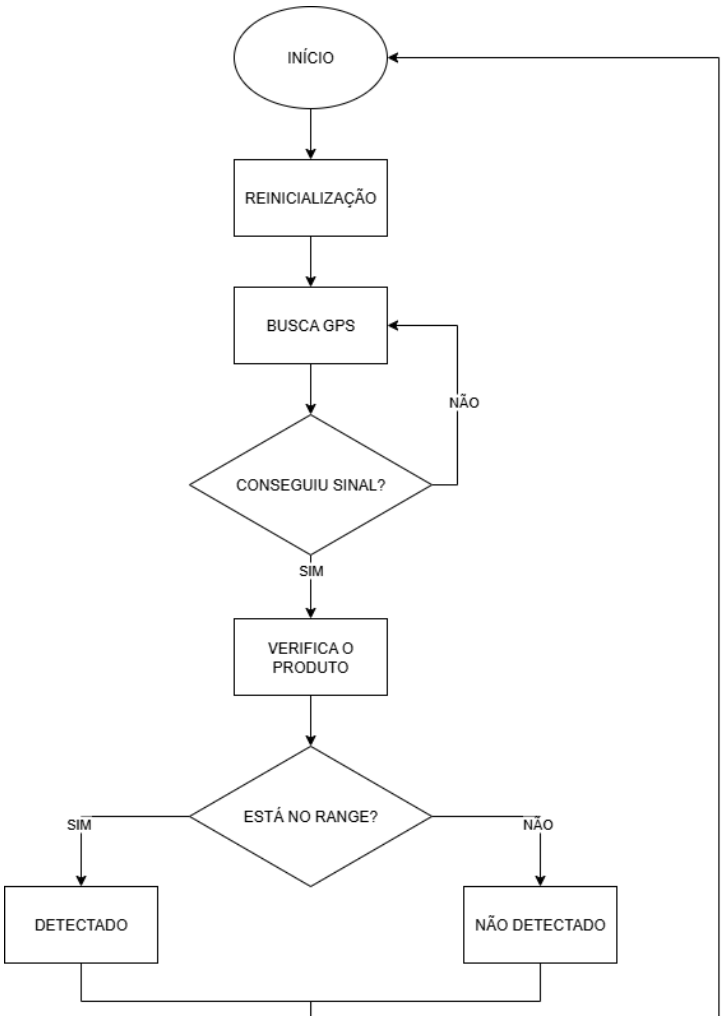
Fonte: Do autor, 2025

5.1 Arquitetura Geral do Firmware

A arquitetura do firmware foi projetada para operar em ciclos periódicos com duração aproximada de 60 segundos. Cada ciclo é responsável por executar uma série

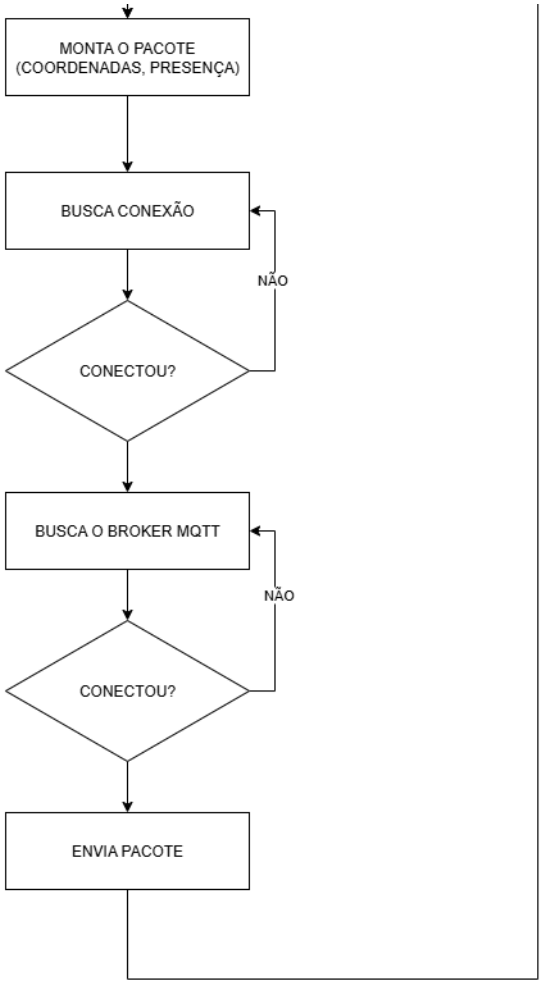
de etapas organizadas em uma ordem lógica, conforme ilustrado nas Figuras 12 e 13. Esse fluxograma apresenta a sequência de operações realizadas pelo sistema embarcado, desde a inicialização dos periféricos até o envio dos dados ao broker

Figura 12 – Fluxograma da Rotina de Coleta e Envio de Dados – Parte 1



Fonte: Do autor, 2025

Figura 13 – Fluxograma da Rotina de Coleta e Envio de Dados – Parte 2

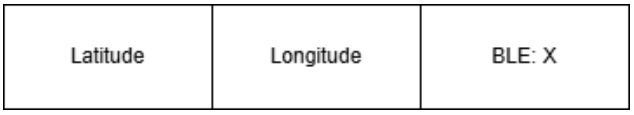


Fonte: Do autor, 2025

Com a conexão ativa, o GPS é ativado e o sistema aguarda a fixação da posição. Quando os dados de latitude e longitude válidos são obtidos, o firmware os extrai do formato NMEA e os converte para coordenadas decimais. Em paralelo, a controladora executa um escaneamento BLE com duração de aproximadamente 5 segundos, buscando detectar um dispositivo com endereço MAC previamente conhecido. O resultado da detecção é armazenado em uma variável booleana.

Com as informações coletadas, é montada uma string no formato <latitude>,<longitude>,BLE:<x>, onde `x` representa `1` caso o dispositivo BLE tenha sido detectado ou `0` caso contrário. Essa estrutura está representada na Figura 14. A mensagem é então enviada ao broker MQTT remoto, encerrando o ciclo atual antes do início de um novo.

Figura 14 – Estrutura da mensagem enviada via protocolo MQTT



Fonte: Do Autor, 2025

5.2 Comandos AT Utilizados no Firmware

A comunicação entre a Raspberry Pi Pico W e o módulo SIM7070G ocorre por meio da interface UART, utilizando comandos AT (Attention Commands). Esses comandos representam um conjunto padronizado de instruções utilizadas para operar funcionalidades de modems celulares e de posicionamento, como configuração de rede, ativação de GPS e envio de dados.

No decorrer da execução do firmware, diferentes comandos AT são empregados em momentos distintos. Inicialmente, são utilizados comandos como `AT+COPS`, `AT+CGDCONT` e `AT+CGATT` para verificar e configurar a conexão com a rede celular. Na etapa de obtenção das coordenadas geográficas, o sistema envia instruções como `AT+CGNSPWR` e `AT+CGNSINF`, responsáveis por ativar e consultar o módulo GPS. Por fim, os dados obtidos são encaminhados a um broker MQTT utilizando comandos como `AT+SMCONF`, `AT+SMCONN`, `AT+SMSTATE` e `AT+SMPUB`, que estruturam a conexão e o envio da mensagem no protocolo desejado.

A tabela 4 apresenta um resumo dos comandos utilizados, com suas respectivas funções e parâmetros aplicados ao longo do ciclo de funcionamento do firmware.

Tabela 4 – Comandos AT utilizados no firmware embarcado

Comando AT	**Descrição**
AT+COPS=0,2,"72405",7	Seleciona manualmente a operadora (Claro - MCC/MNC 72405).
AT+CGDCONT=1,"IP","claro.com.br"	Define o APN da operadora para conexão com a internet.
AT+COPS?	Consulta a operadora atualmente registrada.
AT+CGDCONT?	Consulta os perfis de contexto PDP configurados.
AT+CGATT?	Verifica se o módulo está anexado à rede.
AT+SNPDPID=0	Consulta o ID do contexto PDP (opcional).
AT+CNACT=0,0	Desativa a conexão PDP.
AT+CNACT=0,1	Ativa a conexão PDP.
AT+CFUN=0	Desativa temporariamente o módulo (modo avião).
AT+CFUN=1,1	Reativa o módulo com reinicialização total.
AT+SMCONF="CLIENTID","ID"	Define o identificador do cliente MQTT.
AT+SMCONF="KEEPTIME",60	Define o tempo de keep-alive para conexão MQTT.
AT+SMCONF="URL", "test.mosquitto.org","1883"	Define o endereço e porta do broker MQTT separando o host e a porta com vírgula.
AT+SMCONF="CLEANSS",1	Configura o uso de sessão limpa para a conexão MQTT.
AT+SMCONF="QOS",1	Define o nível de qualidade de serviço (QoS) da publicação.
AT+SMCONF="TOPIC","sim7070g"	Define o tópico MQTT a ser utilizado.
AT+SMCONF="MESSAGE","oi"	Define a mensagem a ser publicada no tópico.
AT+SMCONF="RETAIN",1	Define se a mensagem será retida no broker.
AT+SMCONN	Estabelece conexão com o broker MQTT.
AT+SMSTATE?	Verifica o estado da conexão MQTT.
AT+SMSUB="sim7070g",1	Realiza a inscrição (subscribe) no tópico especificado.
AT+SMPUB="sim7070g",2,1,1	Publica mensagem no tópico MQTT (seguido da carga útil).

5.3 Comunicação com o Módulo GPS

A comunicação com o módulo SIM7070G foi realizada por meio de comandos AT, seguindo os padrões estabelecidos para controle de modems LTE. A interface UART utiliza os pinos GPIO 0 (TX) e GPIO 1 (RX) da Raspberry Pi Pico W, garante uma comunicação direta e confiável. A inicialização do módulo começa com um reset via GPIO dedicado, seguido pela verificação do SIM card com o comando AT+CPIN?.

Uma vez validado o chip, o firmware ativa a conexão com a rede de pacotes utilizando AT+CGATT=1 e define o ponto de acesso da operadora com o comando AT+CGDCONT=1,"IP",«APN>". Após estar conectado, são iniciadas as etapas de configuração MQTT, começando com AT+CMQTTSTART para ativar o cliente MQTT embutido no módulo. O firmware então utiliza os comandos AT+CMQTTACCQ, AT+CMQTTCONNECT e, finalmente, AT+CMQTTPUB para enviar as informações coletadas.

Para obter as coordenadas GPS, o comando AT+CGNSINF (ou AT+CGPSINF=0) retorna dados em formato NMEA. A conversão desses dados para o formato decimal foi realizada diretamente no firmware, utilizando fórmulas matemáticas apropriadas. O sistema também conta com verificações periódicas do status da conexão; em caso de falha, o módulo é reinicializado com o comando AT+CFUN=1,1, garantindo a retomada da funcionalidade.

5.4 Escaneamento BLE com a Raspberry Pi Pico W

A detecção da tag BLE é realizada por meio da funcionalidade Bluetooth embutida na Raspberry Pi Pico W, acessada por bibliotecas compatíveis com MicroPython, como bluetooth e aioble. A biblioteca aioble é uma implementação de Bluetooth Low Energy para MicroPython baseada em tarefas assíncronas, que permite executar operações como escaneamento e identificação de dispositivos BLE de forma eficiente e sem bloqueios.

O firmware executa varreduras com duração de cinco segundos, durante os quais coleta todos os dispositivos BLE próximos e armazena seus endereços MAC temporariamente na memória. A tag BLE utilizada possui um endereço MAC fixo, previamente cadastrado no firmware. Durante o processo de escaneamento, o sistema verifica se esse endereço foi identificado entre os dispositivos detectados. Esse procedimento ocorre em paralelo à leitura das coordenadas GPS e é executado por uma tarefa assíncrona, evitando interferências nas demais funções do sistema.

Essa detecção permite verificar se o objeto monitorado permanece nas proximidades do dispositivo rastreador, mesmo que a posição geográfica via GPS não sofra alterações frequentes.

5.5 Publicação via MQTT

Após a obtenção da localização e da verificação da presença BLE, os dados são organizados em uma única string e é enviada via protocolo MQTT para um broker remoto, sendo esse o meio principal de comunicação entre o firmware e o servidor responsável por armazenar e tratar os dados.

O processo inicia com o comando AT+CMQTTSTART, que ativa o cliente MQTT interno. Em seguida, o comando AT+CMQTTACCQ=0,"client_id"configura o identificador do cliente, enquanto AT+CMQTTCONNECT=0,"tcp://<broker>:1883",60,1 estabelece a conexão com o broker no endereço IP definido, utilizando a porta padrão 1883. Para definir o tópico de publicação e o conteúdo da mensagem, são utilizados os comandos AT+CMQTTTOPIC e AT+CMQTTPAYLOAD, seguidos por AT+CMQTTPUB, que efetua a publicação com qualidade de serviço (QoS) 1, garantindo a entrega da mensagem.

O tópico MQTT utilizado no projeto foi nomeado como sim7070g, tanto no firmware quanto no servidor Flask responsável por receber e tratar os dados publicados — cuja estrutura será abordada em detalhes no Capítulo 5. Em caso de falha na publicação ou perda de conexão, o sistema tenta automaticamente restabelecer a conexão e repetir o processo, garantindo a robustez da comunicação.

5.6 Gerenciamento de Energia

Embora a Raspberry Pi Pico W não possua modos de economia de energia tão avançados quanto microcontroladores como o ESP32, foram implementadas estratégias para reduzir o consumo de energia ao máximo, prolongando a autonomia do dispositivo em campo. A principal medida adotada foi a execução de ciclos de leitura e envio com intervalo de 10 segundos, evitando funcionamento contínuo e desnecessário dos componentes.

Além disso, foi implementado o desligamento temporário do módulo SIM7070G por meio do comando AT+CFUN=0 durante os períodos de inatividade entre os ciclos, contribuindo para a redução do consumo. O uso de uasyncio.sleep() entre os ciclos também possibilita a execução assíncrona sem bloquear os demais processos, garantindo que o dispositivo opere de forma eficiente mesmo com tarefas concorrentes.

Tais medidas, somadas à simplicidade do hardware e ao controle rigoroso da comunicação, tornam o sistema adequado para aplicações móveis e autônomas.

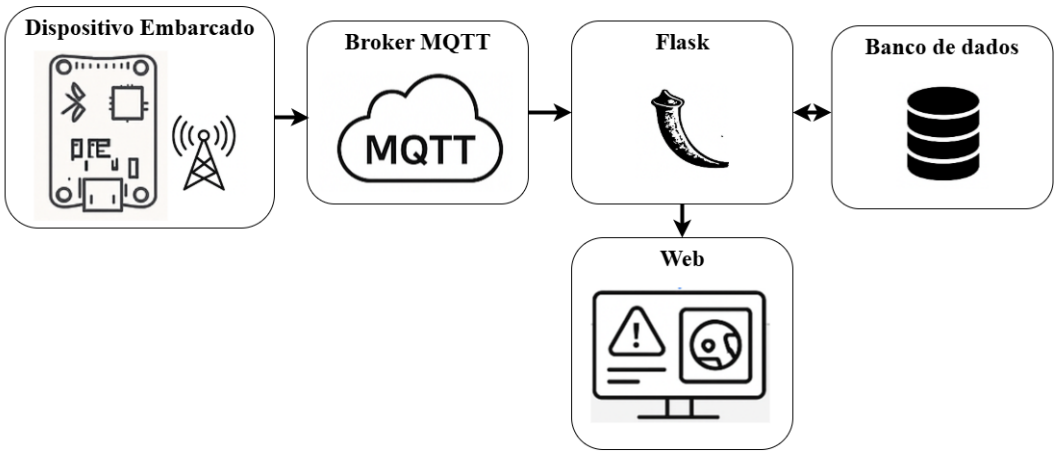
6 DESENVOLVIMENTO DO SISTEMA WEB

O sistema web desenvolvido neste projeto foi projetado para permitir o monitoramento remoto da localização e da presença do objeto rastreado. Para isso, adotou-se uma arquitetura baseada em um backend leve utilizando o framework Flask, escrito em Python, e um frontend construído com HTML, JavaScript e a API do Google Maps. Essa combinação possibilita que os dados enviados pelo dispositivo embarcado sejam processados, armazenados e exibidos em tempo real por meio de um mapa interativo acessível via navegador.

A aplicação é hospedada em uma instância de nuvem (AWS), onde o servidor Flask recebe as mensagens publicadas via protocolo MQTT, processa essas informações e armazena os dados em um banco estruturado. O frontend, por sua vez, consome essas informações por meio de requisições HTTP e as apresenta ao usuário, utilizando marcadores no mapa para indicar a posição do objeto rastreado e exibindo também o estado da detecção BLE.

A Figura 15 ilustra a arquitetura do sistema web, destacando a relação entre os principais componentes, incluindo o dispositivo embarcado, o broker MQTT, o servidor Flask hospedado na AWS e a interface de visualização no navegador por meio da API do Google Maps.

Figura 15 – Diagrama da arquitetura Web



Fonte: Do Autor, 2025

A Figura 16 apresenta a interface inicial do sistema web, com o mapa interativo e os elementos de controle da visualização.

Figura 16 – Interface inicial do sistema web

Fonte: Do Autor, 2025

6.1 Servidor Remoto

O backend foi responsável por receber, processar e armazenar os dados enviados via protocolo MQTT. Uma instância do cliente MQTT foi implementada em Python, utilizando a biblioteca paho-mqtt, que se conecta ao mesmo broker utilizado pelo dispositivo embarcado. A cada nova publicação recebida no tópico `sim7070g`, o backend extrai os dados da mensagem, que incluem latitude, longitude e o status de detecção BLE, além de anexar um carimbo de data e hora no momento da recepção.

A estrutura do banco de dados utilizada para esse armazenamento está ilustrada na Figura 17, onde é possível visualizar os campos relacionados à posição geográfica, presença do BLE e timestamp.

Figura 17 – Visualização da estrutura do banco de dados SQLite

[2025-05-21 00:42:11] Tópico: sim7070g, Mensagem: -27.623764,-48.525745,BLE:1

[2025-05-21 00:41:25] Tópico: sim7070g, Mensagem:

[2025-05-21 00:38:52] Tópico: sim7070g, Mensagem: -27.623973,-48.525649,BLE:1

[2025-05-21 00:38:44] Tópico: sim7070g, Mensagem:

[2025-05-21 00:36:38] Tópico: sim7070g, Mensagem: -27.623792,-48.525659,BLE:1

[2025-05-21 00:36:30] Tópico: sim7070g, Mensagem:

[2025-05-21 00:34:21] Tópico: sim7070g, Mensagem: -27.624278,-48.525631,BLE:1

Fonte: Do Autor, 2025

Essas informações são armazenadas localmente em um banco de dados

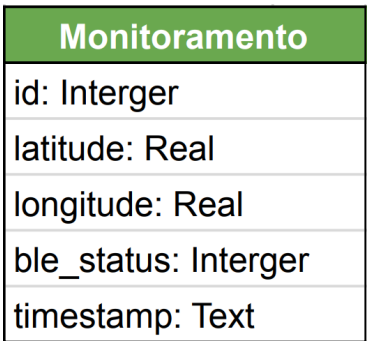
SQLite, escolhido por sua simplicidade de uso e compatibilidade nativa com o Flask. O banco possui uma tabela que registra as coordenadas geográficas, a presença ou ausência da tag BLE e a marca temporal. Essa estrutura permite o rastreamento cronológico completo do trajeto percorrido pelo ativo monitorado, bem como a identificação de momentos em que a tag esteve ou não próxima ao dispositivo.

6.1.1 Armazenamento dos Dados

Para armazenar os dados recebidos, foi utilizada uma estrutura de banco de dados relacional, implementada com SQLite, pela sua leveza e simplicidade de integração com aplicações Flask. A tabela principal contém os seguintes campos:

- `id` (INTEGER): identificador único de cada registro, com incremento automático;
- `latitude` (REAL): coordenada geográfica de latitude enviada pelo dispositivo;
- `longitude` (REAL): coordenada geográfica de longitude correspondente;
- `ble_status` (INTEGER): valor binário representando a presença (1) ou ausência (0) da tag BLE;
- `timestamp` (TEXT): data e hora da recepção da mensagem no servidor, armazenada em formato ISO 8601 (`YYYY-MM-DD HH:MM:SS`).

Figura 18 – Estrutura relacional para armazenamento dos dados



Fonte: Do Autor (2025).

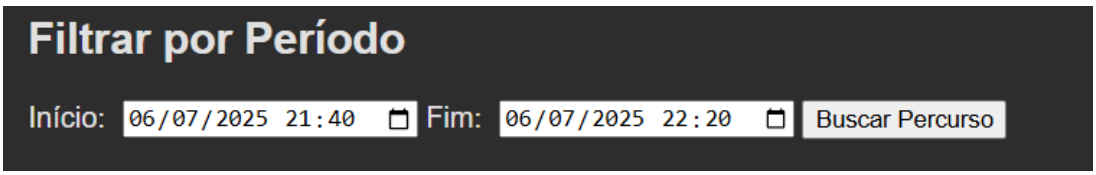
A Figura 18 apresenta a estrutura definida para o armazenamento dos dados. Essa estrutura foi escolhida por sua clareza e facilidade de consulta, permitindo filtragens simples por data, localização ou status BLE. O uso de tipos como REAL e INTEGER garante que os dados numéricos sejam armazenados com a precisão necessária para aplicações de monitoramento geográfico.

6.2 Interface de usuário

No frontend, foi desenvolvida uma página web que se comunica com o backend por meio de requisições HTTP (RESTful). Ao carregar a página, uma requisição GET é feita para o servidor Flask, que responde com os dados mais recentes ou com um histórico filtrado conforme os parâmetros definidos. A interface gráfica exibe os pontos geográficos em um mapa interativo fornecido pela Google Maps API, com marcadores customizados para cada ponto de coleta.

Uma das funcionalidades incorporadas foi o filtro por data e hora, que permite ao usuário selecionar um intervalo específico e visualizar apenas os dados geográficos correspondentes a esse período. O sistema consulta o banco de dados e traça o percurso feito pelo objeto rastreado dentro da faixa de tempo escolhida, representando-o como uma linha sobre o mapa. Essa funcionalidade está representada na Figura 19 – Interface com filtro de período e linha do percurso.

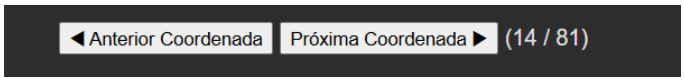
Figura 19 – Filtro de data e hora



Fonte: Do Autor, 2025

Outra adição foi a possibilidade de navegação manual entre coordenadas geográficas válidas, por meio de botões de "Próxima"e "Anterior". Essa funcionalidade permite ao usuário avançar ponto a ponto ao longo do trajeto monitorado, visualizando cada posição com seus respectivos marcadores e status BLE. A Figura 20 ilustra o uso desses controles de navegação de coordenadas.

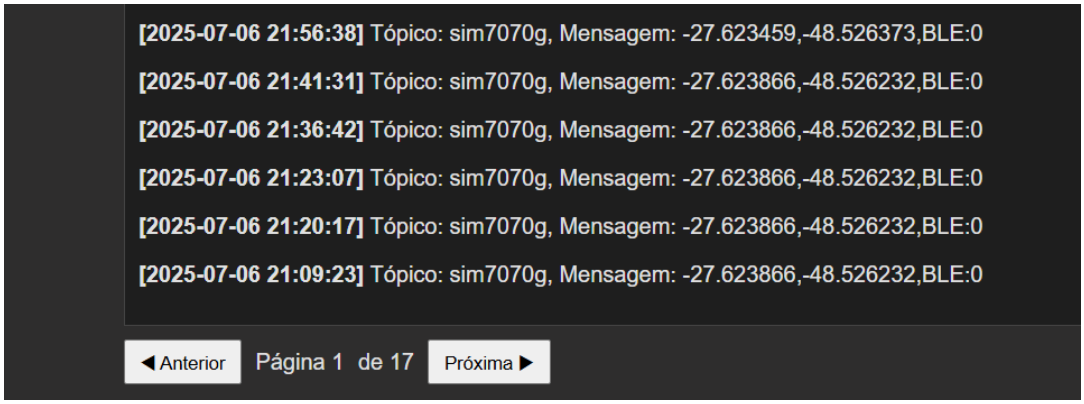
Figura 20 – Navegação de coordenadas



Fonte: Do Autor, 2025

Também foi implementado um sistema de paginação de mensagens, em que os dados recebidos do dispositivo são exibidos em blocos de até 20 registros por página. O usuário pode navegar entre as páginas utilizando botões específicos, permitindo uma visualização mais controlada e organizada das informações. Esse recurso é demonstrado na Figura 21 – Sistema de paginação das mensagens recebidas.

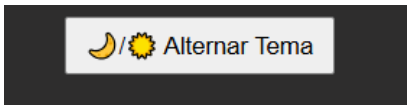
Figura 21 – Páginas do histórico



Fonte: Do Autor, 2025

Adicionalmente, a interface passou a contar com um botão para alternância entre os modos claro e escuro, permitindo maior conforto visual em diferentes ambientes. O tema escolhido pelo usuário é salvo localmente e mantido entre sessões. A Figura 22 mostra o seletor dos modos claro e escuro.

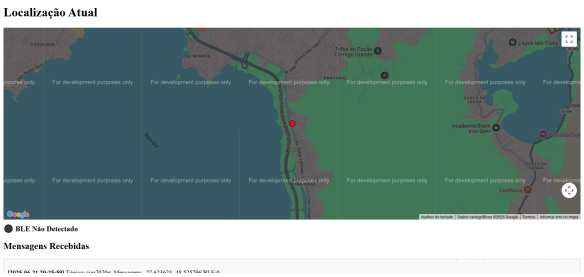
Figura 22 – Seletor de tema



Fonte: Do Autor, 2025

A Figura 23 mostra a exibição de um ponto em que a tag BLE não foi detectada, enquanto a Figura 24 representa o caso em que o dispositivo estava próximo do ativo monitorado no momento da coleta.

Figura 23 – Tag não detectada (BLE:0)



Fonte: Do Autor, 2025

Figura 24 – Tag detectada (BLE:1)

Fonte: Do Autor, 2025

Cada marcador representa uma posição geográfica em que o objeto foi monitorado, com diferenciação visual baseada no status da tag BLE. Por exemplo, marcadores em verde indicam que a tag estava presente (BLE:1), enquanto marcadores em vermelho representam momentos em que a tag não foi detectada (BLE:0). Além disso, um painel lateral apresenta os dados numéricos da última posição, incluindo data, hora, coordenadas e o status da conexão BLE.

O sistema web também foi projetado para ser leve e responsivo, permitindo acesso tanto em navegadores de desktop quanto em dispositivos móveis. O HTML foi estruturado com base no padrão `index.html`, enquanto o estilo visual foi definido com CSS simples. Toda a lógica dinâmica da página, como o carregamento do mapa, foi implementada em JavaScript, sem uso de frameworks adicionais, com o objetivo de reduzir dependências e manter o código mais portátil.

Durante os testes do sistema, foi possível validar a integração completa entre o firmware do dispositivo embarcado e o servidor web. Cada vez que o dispositivo enviava uma nova posição via MQTT, o backend atualizava o banco de dados local e, ao recarregar a página, o frontend refletia as novas posições automaticamente. Essa sincronização foi comprovada por meio das Figuras 23 e 24, que demonstram a atualização visual conforme a presença da tag BLE.

7 TESTES E RESULTADOS

Este capítulo apresenta os testes realizados para validar o funcionamento do sistema embarcado desenvolvido, com foco nos três principais blocos: aquisição de coordenadas geográficas via GPS, detecção de sinal BLE e transmissão de dados por MQTT utilizando a rede celular LTE-Cat M1. Os testes com NB-IoT e GSM ainda não foram realizados, mas estão previstos como etapas futuras. A seguir, são descritos os experimentos conduzidos, os resultados obtidos e as respectivas análises.

7.1 Teste de Aquisição de Coordenadas GPS

Para validar a funcionalidade do módulo GPS embarcado no SIM7070G, foram realizados testes com foco na sequência de comandos AT para ativação do receptor e obtenção de dados geográficos.

A execução se inicia com o comando `AT+CGNSPWR=1`, que liga o módulo GPS. Em seguida, é utilizado o comando `AT+CGNSINF` de forma repetida, aguardando o retorno de coordenadas válidas. A Tabela 5 apresenta trechos dos logs capturados durante esse processo.

Tabela 5 – Trecho do log de ativação e leitura de GPS

Comando/Resposta	Observação
AT+CGNSPWR=1	GPS ativado
AT+CGNSINF	Aguardando fixação do sinal
AT+CGNSINF	Dados válidos: lat = -27.6245, lon = -48.5254

Os testes empiricamente demonstraram que o tempo de fixação variou entre 15 e 60 segundos, dependendo das condições do ambiente (visada ao céu, interferências e clima).

7.2 Teste de Conectividade e Envio via MQTT

Para verificar a estabilidade da comunicação por rede celular, foi utilizado o modo LTE-Cat M1 do SIM7070G. O comando `AT+CNACT=0,1` foi usado para iniciar a conexão com a rede de dados, com a resposta `OK` indicando sucesso. Em seguida, a publicação da mensagem foi realizada com o comando `AT+SMPUB`, conforme estrutura:

```
AT+SMPUB="sim7070g",<tam>,1,1
<mensagem>
```

A mensagem enviada seguiu o formato `latitude,longitude,BLE:x`, sendo o valor `x` igual a 1 se a tag BLE foi detectada, e 0 caso contrário.

7.2.1 LTE-Cat M1

Durante os testes, o tempo médio entre o envio do comando de ativação da conexão e a resposta positiva foi inferior a 5 segundos. Após a conexão, a publicação da mensagem MQTT ocorreu com latência também inferior a 2 segundos. A Tabela 6 ilustra parte do log capturado.

Tabela 6 – Trecho do log de conexão e envio em LTE-Cat M1

Comando/Resposta	Observação
AT+CNACT=0,1	Conexão com a rede celular
OK	Conexão estabelecida
AT+SMPUB=...	Comando de envio MQTT
Mensagem publicada com sucesso!	Confirmação do envio

7.3 Teste de Detecção BLE

Para validar a capacidade de detecção de proximidade via BLE, foram realizados testes com uma tag Bluetooth posicionada a diferentes distâncias da Raspberry Pi Pico W. O endereço MAC da tag foi previamente cadastrado no firmware, que realiza leituras periódicas de RSSI e calcula uma estimativa de distância. A detecção foi bem-sucedida em todas as tentativas realizadas a até 3 metros de distância.

A Tabela 7 mostra leituras de RSSI e estimativas de distância para uma distância real de 3 metros.

Tabela 7 – RSSI e distância estimada para 3 metros reais

Leitura	RSSI (dBm)	Distância Estimada (m)
1	-91	5.41
2	-89	4.64
3	-90	5.01
4	-85	3.41
5	-92	5.84
6	-93	6.31

Apesar das flutuações no valor estimado de distância, a presença da tag foi detectada em todos os testes. Para a finalidade do sistema proposto, a detecção da presença no ambiente é suficiente, não sendo necessário calcular uma distância precisa.

8 CONSIDERAÇÕES FINAIS

Este trabalho teve como objetivo o desenvolvimento e validação de um sistema embarcado para monitoramento de localização e presença utilizando GPS, comunicação MQTT via rede celular e detecção de proximidade por meio de BLE. As etapas de projeto, implementação e testes demonstraram a viabilidade da solução proposta em ambiente de testes controlado, utilizando a tecnologia LTE-Cat M1 para envio de dados, o módulo SIM7070G como interface de comunicação e a Raspberry Pi Pico W como unidade de controle.

Durante os testes de aquisição GPS, foi possível observar que o tempo médio de fixação variou conforme as condições ambientais, mas apresentou resultados consistentes. A detecção da tag BLE foi realizada com sucesso a distâncias de até 3 metros, mesmo com variações nos valores de RSSI e estimativas de distância. A comunicação via MQTT utilizando LTE-Cat M1 apresentou baixa latência aceitável e estabilidade na conexão com o broker remoto.

De modo geral, o sistema demonstrou ser funcional e adequado para aplicações de rastreamento em tempo real que exigem apenas a confirmação da presença de um ativo monitorado em determinado local, sem necessidade de geolocalização precisa ou variação contínua de dados.

8.1 Sugestões para Trabalhos Futuros

Com base nos resultados obtidos, algumas possibilidades de aprimoramento e expansão do projeto são sugeridas:

- Realização de testes com as tecnologias NB-IoT e GSM, a fim de comparar latência, consumo energético e estabilidade com a rede LTE-Cat M1.
- Integração com sistemas de autenticação no dashboard, permitindo acesso restrito e gestão de múltiplos dispositivos.
- Inclusão de sensores adicionais no sistema embarcado, como acelerômetros ou sensores de abertura, para ampliar a capacidade de monitoramento do objeto rastreado.
- Implementação de armazenamento em nuvem, visando maior disponibilidade e escalabilidade dos dados.
- Avaliação da autonomia energética em uso contínuo com diferentes fontes de alimentação, como baterias de lítio ou painéis solares.

- Desenvolvimento de um invólucro físico mais robusto para o dispositivo, adequado a ambientes externos ou sujeitos a impactos.
- Na interface do usuário implementar autenticação de usuários, visualização de múltiplos ativos e exportação de relatórios.

REFERÊNCIAS

AFANEH, M. *Intro to Bluetooth Low Energy*. [S.l.]: Novel Bits, 2018. 22

AL-SAQQA, S.; COLEGAS. Computação e automação de processos. *Journal of Emerging Technologies*, v. 9, n. 2, p. 55–63, 2014. 13

ASHTON, K. *That 'Internet of Things' Thing*. 2009. RFID Journal. 13

ASSIS, J. C. *Sistemas de Navegação por Satélite*. [S.l.]: Editora Técnica, 2010. 17

Bluetooth SIG. *The Bluetooth Core Specification, version 4.0*. San Jose, CA, USA: [s.n.], 2010. 21, 22

Bluetooth SIG. *Informações institucionais sobre Bluetooth (consultado em 2024)*. 2024. Acesso via site oficial. Verificar URL, se aplicável. 22

CIUFFOLETTI, A. Internet of things: tendências e desafios. *Revista Internacional de Tecnologia*, v. 12, n. 3, p. 11–19, 2018. 13

CORDEIRO, G. R. *4G: Quarta geração de telefonia móvel*. Curitiba: Universidade Federal do Paraná, 2012. 18

FOUNDATION, R. P. *Raspberry Pi Pico W Datasheet*. 2023. Disponível em: https://www.raspberrypi.com/products/raspberry-pi-pico/. 30

GOMEZ, C.; OLLER, J.; PARADELLS, J. Overview and evaluation of bluetooth low energy: An emerging low-power wireless technology. *Sensors*, MDPI, v. 12, n. 9, p. 11734–11753, 2012. 21

GONçALVES, P.; TAVARAYAMA, L. Arquitetura do sistema gps. *Revista de Engenharia e Tecnologia*, v. 8, n. 2, p. 45–52, 2011. 17

MACHADO, C. A evolução da navegação e a precisão do gps. *Ciência e Navegação*, v. 5, n. 1, p. 23–29, 2010. 17

MENDES, J. R. R. *5G: A quinta geração*. Portugal: Faculdade de Engenharia - Universidade do Porto, 2013. 18

MONICO, J. F. G. *Posicionamento pelo GNSS: descrição, fundamentos e aplicações*. [S.l.]: Editora UNESP, 2000. 17

PAZ, R.; CUGNASCA, C. *História da Navegação e Geolocalização*. [S.l.]: Editora Acadêmica, 2010. 17

RENESAS. *Introdução — Manual de Referência da Plataforma SW DA14585/DA14531*. [S.l.], 2024. Disponível em: https://lpccs-docs.renesas.com/UM-B-119_DA14585-DA14531_SW_Platform_Reference/Introduction/Introduction.html#protocol-stack. Acesso em: 11 abr. 2024. Disponível em: https://lpccs-docs.renesas.com/UM-B-119_DA14585-DA14531_SW_Platform_Reference/Introduction/Introduction.html#protocol-stack. 21

SALOMONI, C. S. Gps e ionosfera: Estudo do comportamento do tec e de sua influência no posicionamento com gps na região brasileira em períodos de alta e baixa atividade solar. *Dissertação de Mestrado, UFRGS*, 2008. 17

STATLER, S. Geofencing: Everything you need to know. In: *Beacon Technologies*. [S.l.]: Springer, 2016. p. 307–316. 22

TAKEDA, L. N. *Evolução da tecnologia móvel até 2013*. [S.l.]: Faculdade de Engenharia, Campus de Guaratinguetá, 2013. 19

TALAVERA, J. M. *et al.* Review of iot applications in agro-industrial and environmental fields. *Computers and Electronics in Agriculture*, Elsevier, v. 142, 2017. 20

ZAFARI, F.; PAPAPANAGIOTOU, I. Enhancing ibeacon based micro-location with particle filtering. In: *Globecom*. [S.l.: s.n.], 2015. 22

APÊNDICES

APÊNDICE A – CÓDIGO-FONTE DO FIRMWARE

A seguir é apresentado o código-fonte do firmware desenvolvido para a Raspberry Pi Pico W, responsável pela leitura das coordenadas GPS, detecção da tag BLE e envio das informações por MQTT utilizando o módulo SIM7070G.

Código A.1 – Código do firmware embarcado

```
from machine import UART, Pin
import time
import bluetooth
import ubinascii

# --- CONFIGURAÇÕES ---
TARGET_MAC = ffff118382ce
uart = UART(0, baudrate=115200, tx=Pin(0), rx=Pin(1))
ble = bluetooth.BLE()
ble.active(True)

# --- FUNÇÃO BLE ---
beacon_detectado = False

def bt_scan_callback(event, data)
    global beacon_detectado
    if event == 5  # Evento de descoberta de dispositivo
        addr_type, addr, adv_type, rssi, adv_data = data
        mac_addr = .join(ubinascii.hexlify(addr, ).decode().split())
        if mac_addr.lower() == TARGET_MAC
            print(fBeacon encontrado! MAC {mac_addr}, RSSI {rssi} dBm)
            beacon_detectado = True

def verificar_beacon()
    global beacon_detectado
    beacon_detectado = False
    ble.gap_scan(3000, 30000, 30000)
    ble.irq(bt_scan_callback)
    time.sleep(4)  # Aguarda o scan acontecer
    ble.gap_scan(None)  # Para o scan
    return beacon_detectado

# --- FUNÇÕES UART ---
def enviar_comando_at(comando, timeout=2000)
    tentativas = 0
    max_tentativas = 3
    resposta_invalida = '+CGNSINF 0,,,,,,,,,,,,,,,,,,'

    while tentativas  max_tentativas
        comando_enviado = comando + 'rn'
        uart.write(comando_enviado.encode())
        print(fComando enviado {comando.strip()})
        time.sleep_ms(100)

        start = time.ticks_ms()
        resposta = b''

        while time.ticks_diff(time.ticks_ms(), start)  timeout
            if uart.any()
```

```
                resposta += uart.read()

        try
            resposta_decodificada = resposta.decode('utf-8').strip()
            if resposta_decodificada !=  and resposta_invalida not in
    resposta_decodificada
                print(fResposta recebida {resposta_decodificada})
                return resposta_decodificada
            else
                print(fResposta inválida ou vazia recebida {resposta_decodificada})
        except UnicodeDecodeError
            print(fErro ao decodificar a resposta {resposta})

        tentativas += 1
        print(fTentativa {tentativas} falhou.)

    print(Máximo de tentativas atingido. Reiniciando o modem...)
    time.sleep(120)
    loop()
    return

def reboot()
    print(Reiniciando o modem...)
    enviar_comando_at('AT+CFUN=0')
    enviar_comando_at('AT+CFUN=1,1')
    time.sleep(10)

def configurar_modem_gps()
    print(Configurando o modem GPS...)
    enviar_comando_at('AT+CGNSPWR')
    enviar_comando_at('AT+CGNSPWR=1')

    coordenadas_gps = None
    while True
        resposta = enviar_comando_at('AT+CGNSINF')
        if +CGNSINF 1,1 in resposta
            coordenadas = resposta.split(,)[35]
            print(fCoordenadas GPS obtidas {coordenadas})
            coordenadas_gps = coordenadas
            break
        time.sleep(2)

    enviar_comando_at('AT+CGNSPWR=0')
    print(GPS desativado.)
    return coordenadas_gps

def configurar_modem()
    print(Configurando o modem...)
    enviar_comando_at('AT+COPS=0,2,72405,7')
    enviar_comando_at('AT+CGDCONT=1,IP,claro.com.br')
    enviar_comando_at('AT+COPS')
    enviar_comando_at('AT+CGDCONT')
    enviar_comando_at('AT+CGATT')
    enviar_comando_at('AT+SNPDPID=0')
    enviar_comando_at('AT+CNACT=0,0')
    enviar_comando_at('AT+CNACT=0,1')

def enviar_post_celular(coordenadas_gps, beacon)
```

```
    if not coordenadas_gps
        print(Coordenadas GPS não disponíveis!)
        return

    latitude, longitude = coordenadas_gps
    ble_status = BLE1 if beacon else BLE0
    mensagem = f{latitude},{longitude},{ble_status}

    print(Configurando MQTT...)
    enviar_comando_at('AT+SMCONF=CLIENTID,ID')
    enviar_comando_at('AT+SMCONF=KEEPTIME,60')
    enviar_comando_at('AT+SMCONF=URL,test.mosquitto.org,1883')
    enviar_comando_at('AT+SMCONF=CLEANSS,1')
    enviar_comando_at('AT+SMCONF=QOS,1')
    enviar_comando_at('AT+SMCONF=TOPIC,sim7070g')

    print(Conectando ao Broker MQTT...)
    resposta = enviar_comando_at('AT+SMCONN', timeout=5000)
    if +SMSTATE 1 in enviar_comando_at('AT+SMSTATE')
        print(Conexão MQTT estabelecida com sucesso!)
    else
        while True
            enviar_comando_at('AT+SMCONN', timeout=5000)
            if +SMSTATE 1 in enviar_comando_at('AT+SMSTATE')
                print(Conexão MQTT estabelecida com sucesso!)
                break

    print(Publicando mensagem com coordenadas...)
    enviar_comando_at(f'AT+SMPUB=sim7070g,{len(mensagem)},1,1')
    enviar_comando_at(mensagem)
    print(Mensagem publicada com sucesso!)

# --- LOOP PRINCIPAL ---
def loop()
    reboot()
    coordenadas = configurar_modem_gps()
    time.sleep(2)
    beacon = verificar_beacon()
    configurar_modem()
    enviar_post_celular(coordenadas, beacon)
    time.sleep(60)

while True
    loop()
```

APÊNDICE B – CÓDIGO-FONTE DO BACKEND

A seguir é apresentado o código-fonte do backend desenvolvido em Python com o framework Flask. Este componente é responsável por receber mensagens via MQTT, armazenar os dados em banco SQLite e fornecer as informações ao frontend via requisições HTTP.

Código B.1 – Código do backend em Flask

```
import threading
import sqlite3
from flask import Flask, jsonify
from flask_cors import CORS
import paho.mqtt.client as mqtt

# Configuração do MQTT
BROKER = "test.mosquitto.org"
PORT = 1883
TOPIC = "sim7070g"

# Estado global da localização e BLE
latest_status = {"lat": 0, "lng": 0, "ble": 0}

# Nome do banco de dados
DB_NAME = 'mqtt_data.db'

# Inicializar o banco de dados
def init_db():
    conn = sqlite3.connect(DB_NAME)
    cursor = conn.cursor()
    cursor.execute('''
        CREATE TABLE IF NOT EXISTS messages (
            id INTEGER PRIMARY KEY AUTOINCREMENT,
            topic TEXT NOT NULL,
            message TEXT NOT NULL,
            timestamp DATETIME DEFAULT CURRENT_TIMESTAMP
        )
    ''')
    conn.commit()
    conn.close()

# Salvar mensagem no banco
def save_message(topic, message):
    conn = sqlite3.connect(DB_NAME)
    cursor = conn.cursor()
    cursor.execute('INSERT INTO messages (topic, message) VALUES (?, ?)', (topic,
    message))
    conn.commit()
    conn.close()

# Buscar mensagens do banco
def get_messages():
    conn = sqlite3.connect(DB_NAME)
    cursor = conn.cursor()
    cursor.execute('SELECT * FROM messages ORDER BY timestamp DESC')
    messages = cursor.fetchall()
```

```
198     conn.close()
199     return messages
200 
201 # Callback de conexão do MQTT
202 def on_connect(client, userdata, flags, rc):
203     if rc == 0:
204         print("Conectado ao broker MQTT")
205         client.subscribe(TOPIC)
206     else:
207         print(f"Falha na conexão MQTT, código: {rc}")
208 
209 # Callback de mensagem recebida do MQTT
210 def on_message(client, userdata, msg):
211     global latest_status
212     mensagem = msg.payload.decode()
213     print(f"Mensagem recebida no tópico {msg.topic}: {mensagem}")
214 
215     save_message(msg.topic, mensagem)
216 
217     try:
218         partes = mensagem.split(",")
219         if len(partes) == 3 and "BLE:" in partes[2]:
220             lat = float(partes[0])
221             lng = float(partes[1])
222             ble = int(partes[2].split(":")[1])
223             latest_status = {"lat": lat, "lng": lng, "ble": ble}
224         else:
225             print("Formato inesperado. Ignorando dados.")
226     except Exception as e:
227         print("Erro ao processar mensagem:", e)
228 
229 # Iniciar cliente MQTT
230 mqtt_client = mqtt.Client()
231 mqtt_client.on_connect = on_connect
232 mqtt_client.on_message = on_message
233 mqtt_client.connect(BROKER, PORT, 60)
234 
235 # Configuração Flask
236 app = Flask(__name__)
237 CORS(app)
238 
239 # Endpoint para localização atual
240 @app.route('/location', methods=['GET'])
241 def get_location():
242     return jsonify(latest_status)
243 
244 # Endpoint para mensagens salvas
245 @app.route('/messages', methods=['GET'])
246 def fetch_messages():
247     messages = get_messages()
248     return jsonify([
249         {"id": msg[0], "topic": msg[1], "message": msg[2], "timestamp": msg[3]}
250         for msg in messages
251     ])
252 
253 # Threads para Flask e MQTT
254 def start_flask():
255     app.run(host='0.0.0.0', port=3000)
```

```
256
257 def start_mqtt():
258     mqtt_client.loop_forever()
259
260 if __name__ == "__main__":
261     init_db()
262     flask_thread = threading.Thread(target=start_flask)
263     mqtt_thread = threading.Thread(target=start_mqtt)
264     flask_thread.start()
265     mqtt_thread.start()
```

APÊNDICE C – CÓDIGO DO FRONTEND WEB

A seguir é apresentado o código-fonte utilizado para a visualização dos dados de localização e status BLE no navegador, desenvolvido em HTML e JavaScript com integração à API do Google Maps.

Código C.1 – Código do frontend web

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Google Maps e Mensagens MQTT</title>
    <script src="https://maps.googleapis.com/maps/api/js?key=
    AIzaSyD4nAwAE6imqm_efT2S-TPFUfKM9dDkIz4"></script>
    <style>
        #map {
            width: 100%;
            height: 500px;
        }
        #messages {
            margin-top: 20px;
            padding: 10px;
            border: 1px solid #ccc;
            background: #f9f9f9;
        }
        #status {
            margin-top: 10px;
            font-size: 1.2em;
            font-weight: bold;
        }
    </style>
</head>
<body>
    <h1>Localização Atual</h1>
    <div id="map"></div>
    <div id="status">Carregando status BLE...</div>

    <h2>Mensagens Recebidas</h2>
    <div id="messages"></div>

    <script>
        let map, marker;

        function initMap(lat = 0, lng = 0) {
            const location = { lat, lng };
            map = new google.maps.Map(document.getElementById('map'), {
                zoom: 15,
                center: location,
            });
            marker = new google.maps.Marker({
                position: location,
                map: map,
                icon: getMarkerIcon(0) // BLE padrão: não detectado
            });
        }
```

```
314
315         function getMarkerIcon(ble) {
316             // Retorna um ícone colorido dependendo do BLE
317             const color = ble === 1 ? 'green' : 'red';
318             return {
319                 path: google.maps.SymbolPath.CIRCLE,
320                 scale: 8,
321                 fillColor: color,
322                 fillOpacity: 0.8,
323                 strokeColor: 'black',
324                 strokeWeight: 1,
325             };
326         }
327
328         function updateMap(lat, lng, ble) {
329             const location = { lat, lng };
330             map.setCenter(location);
331             marker.setPosition(location);
332             marker.setIcon(getMarkerIcon(ble));
333
334             const statusText = ble === 1
335                 ? '   BLE Detectado (próximo)'
336                 : '  BLE Não Detectado';
337             document.getElementById('status').textContent = statusText;
338         }
339
340         async function fetchLocation() {
341             try {
342                 const response = await fetch('http://localhost:3000/location');
343                 const data = await response.json();
344                 updateMap(data.lat, data.lng, data.ble);
345             } catch (error) {
346                 console.error('Erro ao buscar localização:', error);
347             }
348         }
349
350         async function fetchMessages() {
351             try {
352                 const response = await fetch('http://localhost:3000/messages');
353                 const messages = await response.json();
354                 const messagesDiv = document.getElementById('messages');
355                 messagesDiv.innerHTML = messages
356                     .map(msg => `<p><strong>[${msg.timestamp}]</strong> Tópico: ${
    msg.topic}, Mensagem: ${msg.message}</p>`)
357                     .join('');
358             } catch (error) {
359                 console.error('Erro ao buscar mensagens:', error);
360             }
361         }
362
363         initMap();
364         setInterval(fetchLocation, 5000);
365         setInterval(fetchMessages, 5000);
366     </script>
367 </body>
368 </html>
```

ANEXOS

ANEXO A – TÍTULO

ANEXO B – TÍTULO