INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DE SANTA CATARINA – CÂMPUS FLORIANÓPOLIS
DEPARTAMENTO ACADÊMICO DE ELETRÔNICA
CURSO DE GRADUAÇÃO EM ENGENHARIA ELETRÔNICA

SARAH GIOVANNA BARARUA SANTOS

ESTUDO E APLICAÇÃO DO CC2652R7 EM REDES IoT BASEADAS NO PROTOCOLO ZIGBEE

FLORIANÓPOLIS, 2026.

INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DE SANTA CATARINA – CÂMPUS FLORIANÓPOLIS
DEPARTAMENTO ACADÊMICO DE ELETRÔNICA
CURSO DE GRADUAÇÃO EM ENGENHARIA ELETRÔNICA

SARAH GIOVANNA BARARUA SANTOS

ESTUDO E APLICAÇÃO DO CC2652R7 EM REDES IoT BASEADAS NO PROTOCOLO ZIGBEE

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 Bacharel em Engenharia Eletrônica.

Orientador:
Prof. Renan Augusto Starke, Dr.Eng.

FLORIANÓPOLIS, 2026.

Ficha de identificação da obra elaborada pelo autor.

Santos, Sarah Giovanna Bararua
Estudo e Aplicação do CC2652R7 em Redes IoT Baseadas
no Protocolo Zigbee / Sarah Giovanna Bararua Santos; orientação
de Renan Augusto Starke. - Florianópolis, SC,
2026.
61 p.

Trabalho de Conclusão de Curso (TCC) - Instituto Federal
de Santa Catarina, Câmpus Florianópolis. Bacharelado
em Engenharia Eletrônica. Departamento Acadêmico
de Eletrônica.
Inclui Referências.

1. Internet das Coisas. **2. Zigbee.** 3. CC2652R7.
4. HTU21D. I. Starke, Renan Augusto. II. Instituto Federal
de Santa Catarina. III. Estudo e Aplicação do CC2652R7
em Redes IoT Baseadas no Protocolo Zigbee.

ESTUDO E APLICAÇÃO DO CC2652R7 EM REDES IoT BASEADAS NO PROTOCOLO ZIGBEE

SARAH GIOVANNA BARARUA SANTOS

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

Florianópolis, 12 de Fevereiro de 2026.

Banca Examinadora:

gov.br Documento assinado digitalmente
RENAN AUGUSTO STARKE
Data: 19/02/2026 16:03:44-0300
Verifique em https://validar.iti.gov.br

Prof. Renan Augusto Starke, Dr. Eng.

gov.br Documento assinado digitalmente
MATHEUS LEITZKE PINTO
Data: 20/02/2026 14:36:25-0300
Verifique em https://validar.iti.gov.br

Prof. Matheus Leitzke Pinto, Msc. Eng.

gov.br Documento assinado digitalmente
DANIEL LOHMANN
Data: 19/02/2026 20:06:53-0300
Verifique em https://validar.iti.gov.br

Prof. Daniel Lohmann, Msc. Eng.

Dedico este trabalho aos meus pais,
pelo incentivo constante e por todo o esforço
dedicado à minha formação.
Sem o apoio de vocês, nada disso seria possível.

AGRADECIMENTOS

Primeiramente, agradeço a Deus, por ter me dado a vida, a fé e a força necessária para superar os obstáculos e persistir diante das dificuldades ao longo desta caminhada. Sua presença constante foi o alicerce que me permitiu chegar até aqui.

À minha família, que sempre foi meu porto seguro. Agradeço pelo amor incondicional, pelo incentivo diário e por acreditarem no meu potencial mesmo nos momentos mais desafiadores. Vocês são a razão de todo o meu esforço e dedicação.

Aos meus amigos da faculdade, Diana, Marina e João, que tornaram essa jornada acadêmica mais leve e significativa. Obrigada pelas horas de estudo compartilhadas, pelo companheirismo, pelas risadas nos momentos de descontração e pelo apoio mútuo que tornou os dias difíceis mais suportáveis. A amizade de vocês ocupa um lugar especial nesta conquista.

Ao meu orientador, Prof. Renan Augusto Starke, pela paciência, pela sabedoria transmitida e pela orientação precisa em cada etapa deste trabalho. Agradeço não apenas pelo conhecimento técnico, mas também pela confiança depositada e pelo incentivo.

Ao Instituto Federal de Santa Catarina (IFSC), e a todos os professores do Departamento Acadêmico de Eletrônica, pela oportunidade de ensino de qualidade, pela infraestrutura disponibilizada e por contribuírem diretamente para a minha formação profissional e humana.

“Para que todos vejam e saibam,
considerem e juntamente entendam
que a mão do senhor fez isso.”
Isaías 41:20

RESUMO

Este trabalho apresenta o estudo e a aplicação do microcontrolador CC2652R7, da Texas Instruments, em uma rede de Internet das Coisas (*Internet of Things* - IoT) baseada no protocolo Zigbee, com foco no desenvolvimento de um sistema de sensoriamento ambiental. Inicialmente, são abordados os conceitos fundamentais de IoT, o funcionamento da arquitetura Zigbee, as operações de baixo consumo (*Low Power*) e sua integração com *gateways* baseados em MQTT, destacando-se o uso do *software* Zigbee2MQTT como interface entre a rede de sensores e a camada de aplicação. A metodologia adotada envolveu a configuração do ambiente de desenvolvimento utilizando o SimpleLink SDK e o Code Composer Studio, bem como a configuração da pilha de protocolos Z-Stack, adaptando um código base para operar como um dispositivo final (*Zigbee End Device*) com perfil de sensoriamento. A validação de rede foi realizada inicialmente através do sensor interno do microcontrolador, seguida pela implementação prática da leitura de temperatura e umidade por meio do sensor HTU21D, utilizando o barramento I2C. Os resultados demonstraram a correta transmissão bidirecional dos dados pela rede Zigbee e sua visualização *online*, validando a interoperabilidade do sistema com um coordenador de terceiros. A análise de consumo energético com a ferramenta EnergyTrace evidenciou a importância da adequada gestão de periféricos, alcançando um consumo médio de 2,09 mA no estágio final. Os resultados obtidos comprovam a viabilidade técnica da solução proposta, alcançando uma autonomia estimada de 41 dias em uma bateria de 2000 mAh. Apontam, contudo, que o contraste com o padrão comercial em microamperes (uA) exige uma futura aplicação de estratégias de gerenciamento de energia mais otimizadas para maximizar a autonomia em aplicações reais.

Palavras-chave: Internet das Coisas. Zigbee. CC2652R7. HTU21D.

ABSTRACT

This work presents an investigation and practical application of the Texas Instruments CC265R7 microcontroller in an Internet of Things (IoT) network based on the Zigbee protocol, focusing on the development of an environmental sensing system. Initially, fundamental IoT concepts are discussed, along with the operation of the Zigbee architecture, low-power operation, and its integration with MQTT-based gateways, highlighting the use of Zigbee2MQTT as an interface between the sensor network and the application layer. The methodology included configuring the development environment using the SimpleLink SDK and Code Composer Studio, as well as setting up the Z-Stack protocol stack and adapting a base code to operate as a Zigbee End Device with a sensing profile. Network validation was first performed using the microcontroller's internal sensor, followed by the practical implementation of temperature and humidity measurements were performed using the HTU21D sensor via the I2C bus. The results validated the correct bidirectional transmission of data across the Zigbee network and its online visualization, confirming the system's interoperability with a third-party coordinator. Energy consumption analysis using the EnergyTrace tool highlighted the importance of appropriate peripheral management, achieving an average current consumption of 2.09 mA in the final stage.The obtained results demonstrated the technical feasibility of the proposed solution, reaching an estimated autonomy of 41 days with a 2000 mAh battery. However, comparison with commercial microampere-level standards indicates the need for more optimized energy-management strategies to maximize autonomy in real-word applications.

Keywords: Internet of Things. Zigbee. CC265R7. HTU21D.

LISTA DE FIGURAS

LISTA DE QUADROS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

AWS	Amazon Web Services
BLE	*Bluetooth Low Energy*
CCS	Code Composer Studio
I2C	Circuito Inter-Integrado
IDE	Ambiente de Desenvolvimento Integrado
IFSC	Instituto Federal de Santa Catarina
IoT	*Internet of Things* (Internet das Coisas)
LWT	*Last Will and Testament*
MQTT	*Message Queuing Telemetry Transport*
RH	Umidade Relativa
SCL	*Serial Clock*
SDA	*Serial Data*
SDK	*Software Development Kit*
SoC	*System-on-Chip*
TI	Texas Instruments
UART	Receptor Transmissor Assíncrono Universal
ZCL	*Zigbee Cluster Library*
ZC	*Zigbee Coordinator*
ZED	*Zigbee End Device*

SUMÁRIO

1. INTRODUÇÃO

A Internet das Coisas (*Internet of Things* - IoT) vem se tornado um tema de grande interesse entre gigantes da tecnologia e no meio empresarial (INTERNET OF THINGS WIKI, 2025), pois representa uma rede de dispositivos interconectados capazes de coletar, processar e compartilhar dados. Esses dispositivos, que podem incluir sensores, *software*, módulos de conectividade e outros componentes eletrônicos, comunicam-se entre si permitindo automação e monitoramento *online*. Nesse contexto, os protocolos desempenham papel fundamental. Assim, o Zigbee destaca-se como uma solução amplamente utilizada devido ao baixo consumo de energia, confiabilidade e suporte a redes em malha (*mesh*), o que torna adequado para aplicações que exigem escalabilidade e operação contínua.

Para atender às exigentes demandas de *hardware* dessas redes IoT, o mercado dispõe de soluções como o CC2652R7 da Texas Instruments (TI) (TEXAS INSTRUMENTS, 2025), um microcontrolador de alto desempenho com rádio transceptor integrado e suporte multiprotocolo. Projetado para atender demandas de aplicações de IoT que exigem baixo consumo de energia, comunicação sem fio e capacidade de sensoriamento. Ele integra suporte nativo para importantes protocolos de comunicação sem fio, como Thread, Zigbee e Bluetooth 5.2 Low Energy (BLE) (TEXAS INSTRUMENTS, 2025), sendo amplamente direcionado para aplicações de automação predial, automação industrial, infraestrutura de rede, eletrônicos pessoais e aplicações médicas. Contudo, a estruturação de um ecossistema inteligente completo exige não apenas o hardware de borda, mas também a sua integração eficiente com plataformas de aplicações superior, o que frequentemente é realizado por meio de *gateways* conversores baseados em MQTT, como o *software* Zigbee2MQTT.

Embora o CC2652R7 apresente recursos avançados e seja reconhecido pelo seu desempenho em redes Zigbee (TEXAS INSTRUMENTS, 2025), a documentação oferecida pela Texas Instruments, assim como alguns exemplos do kit de desenvolvimento de *software* oficial, SimpleLink SDK (*Software Development Kit)* (TEXAS INSTRUMENTS, 2025), fornece apenas bases introdutórias, deixando uma carência de materiais que tratem de implementações aplicadas e autônomas,

especialmente no que tange à integração com gateways terceiros e à otimização do consumo energético (*Low Power*) para dispositivos finais alimentados a bateria.

Diante desse cenário, este trabalho busca explorar o potencial da CC2652R7 em uma aplicação funcional completa de sensoriamento ambiental de temperatura e umidade, Ao abranger desde a adaptação do *firmware* e a integração com sensores externos até a validação da comunicação via Zigbee2MQTT e a análise do perfil de consumo de energia com a ferramenta EnergyTrace, o trabalho visa contribuir com novos resultados e referências práticas que possam servir de apoio para desenvolvedores e pesquisadores em futuros projetos.

1.1 Justificativa

O crescimento das aplicações de internet das coisas (IoT) tem impulsionado a busca por soluções de comunicação sem fio que sejam eficientes, confiáveis e econômicas energeticamente. Entre essas tecnologias, o Zigbee se destaca por sua robustez, suporte à rede em malhas e capacidade de operar com baixo consumo de energia, sendo essas características essenciais para sistema de monitoramento de automação em larga escala. No entanto, apesar do seu potencial, observa-se uma escassez de estudos aplicados que abordam o uso prático do CC2652R7 como nó de sensoriamento em ambientes reais.

A plataforma da Texas Instruments oferece recursos avançados para redes Zigbee, mas a documentação disponível e os exemplos fornecidos pelo SimpleLink SDK não aprofundam implementações completas envolvendo integração com infraestruturas externas geralmente utilizadas para o gerenciamento e distribuição dos dados coletados, como soluções baseadas em MQTT ou plataformas de monitoramento como Zigbee2MQTT (ZIGBEE2MQTT, 2025). Assim, torna-se necessário desenvolver materiais que demonstrem, na prática, a viabilidade em desempenho e limitação desse *hardware* em aplicação IoT. Desta forma, este trabalho justifica-se pela necessidade de ampliar o conhecimento técnico sobre a utilização do CC2652R7 em redes de sensoriamento, fornecendo subsídios para estudantes, desenvolvedores e pesquisadores que buscam referências.

1.2 Definição do Problema

Como o microcontrolador CC2652R7 pode ser utilizado para medir umidade e temperatura e transmitir esses dados via Zigbee?

1.3 Objetivo Geral

Implementar um sistema funcional de sensoriamento de temperatura e umidade utilizando o CC2652R7 para a aquisição de dados com comunicação Zigbee.

1.4 Objetivos Específicos

a) Apresentar as principais características do CC2652R7 e seu potencial para aplicações em internet das coisas (IoT).

b) Descrever o funcionamento do protocolo Zigbee e sua aplicação para redes de sensoriamento sem fio.

c) Configurar o ambiente de desenvolvimento, utilizando o SimpleLink SDK, e implementar o código necessário para operação do CC2652R7 em uma rede Zigbee.

d) Validar a aplicação por meio da leitura e análise do sensor de temperatura interno do microcontrolador.

e) Identificar, integrar e testar o sensor externo de temperatura e umidade HTU21D via barramento I2C, avaliando seu funcionamento junto com a comunicação Zigbee e o *gateway* Zigbee2MQTT.

f) Avaliar o desempenho e o consumo energético do CC2652R7 durante a execução das aplicações desenvolvidas, utilizando a ferramenta EnergyTrace.

1.5 Estrutura do Trabalho

Esse trabalho está organizado em cinco capítulos. No primeiro capítulo, apresenta-se a contextualização do tema, a definição do problema de pesquisa, a justificativa para o desenvolvimento do projeto e os objetivos (geral e específicos). No segundo capítulo aborda-se os conceitos fundamentais para a compreensão do trabalho, incluindo IoT, características do protocolo Zigbee, arquitetura de *gateways* e os principais aspectos técnicos do microcontrolador CC2652R7 e do sensor utilizado. No terceiro capítulo descreve-se os materiais e ferramentas utilizados, o ambiente de desenvolvimento, a configuração dos dispositivos e as etapas de desenvolvimento do projeto. O quarto capítulo apresenta os resultados obtidos a partir dos testes práticos realizados, incluindo a validação da comunicação, leitura dos sensores e análise do consumo de energia nos diferentes cenários. Por fim, o quinto capítulo apresenta a conclusão, considerações finais e possíveis ideias para futuros projetos encontrados após análises feitas neste trabalho.

2 FUNDAMENTAÇÃO TEÓRICA

Este capítulo apresenta os conceitos fundamentais para compreensão e o desenvolvimento do trabalho, com foco em redes de sensores sem fio aplicadas à IoT. Inicialmente, discute-se o conceito de IoT, abordando sua origem, evolução e relevância no cenário atual. Em seguida, uma revisão dos protocolos de comunicação envolvidos no sistema: Zigbee, responsável pela comunicação entre dispositivos, e MQTT, utilizado no *gateway* Zigbee2MQTT para publicação dos dados em plataforma externa. Além disso, apresentam-se as características técnicas do microcontrolador CC2652R7, utilizado para coleta e transmissão de dados, e o funcionamento do sensor externo responsável pela medição de umidade e temperatura, o HTU21D, garantindo conhecimento e compreensão para etapas seguintes.

2.1 Internet das Coisas (IoT)

A IoT representa um estágio evolutivo da tecnologia da informação, caracterizado pela extensão da conectividade de rede para além dos computadores e *smartphones* tradicionais. O conceito fundamenta-se na interconexão de objetos físicos que, equipados com sensores e *software*, tornam-se capazes de coletar dados e interagir com o ambiente e outros sistemas pela internet (INTERNETOFTHINGSWIKI, 2025). Essa mudança de paradigma permitiu a transição de sistemas isolados para ecossistemas inteligentes, impulsionando avanços significativos em áreas como automação residencial, saúde e indústria.

Para compreender o funcionamento desses ecossistemas, é comum adotar uma arquitetura de referência dividida em camadas fundamentais: a camada de percepção (ou de borda), responsável pela coleta de dados físicos; a camada de rede, encarregada da transmissão dessas informações; e a camada de aplicação, onde ocorre o processamento e a entrega de valor ao usuário. Nesse contexto, a comunicação sem fio desempenha um papel vital na viabilidade da infraestrutura, pois elimina a necessidade de cabeamento complexo e permite a escalabilidade de sensores em locais de difícil acesso (TEXAS INSTRUMENTS, 2020).

A eficiência dessa arquitetura está diretamente relacionada à seleção apropriada das tecnologias de conectividade. Em contextos que requerem monitoramento contínuo aliado a baixo consumo energético, a comunicação entre os dispositivos da camada de percepção exige o uso de protocolos otimizados para redes locais e a aplicação de técnicas em *hardware* e *software* para colocar os processadores em modo de baixa potência (*sleep mode*). Paralelamente, a integração dos dados coletados com a camada de aplicação ou com a nuvem demanda padrões de transporte de mensagens leves e confiáveis, capazes de funcionar adequadamente mesmo em ambientes com largura de banda restrita.

Dessa forma, a implementação de uma solução IoT robusta geralmente combina diferentes padrões de comunicação para atender aos requisitos de cada camada. Neste trabalho, destacam-se o protocolo Zigbee, atuando na rede de sensores sem fio, e o protocolo MQTT, responsável pelo transporte das informações através do *gateway*. As características técnicas e as especificidades de cada uma dessas tecnologias serão detalhadas nas seções subsequentes.

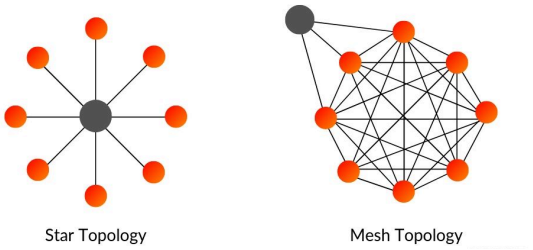
2.2 Protocolo Zigbee

Diferenciando-se de outras tecnologias sem fio por seu baixo consumo de energia e sua arquitetura otimizada, o protocolo Zigbee consolidou-se como uma solução fundamental para a IoT. Trata-se de um protocolo sem fio baseado na especificação IEEE 802.15.4 e operando, na sua maioria, na banda de 2.4 GHz. Segundo a *Connectivity Standards Alliance* (2025, tradução nossa), "Zigbee é a única solução completa de IoT — desde a rede *mesh* até a linguagem universal que permite que objetos inteligentes funcionem em conjunto". Sua principal atribuição é proporcionar o monitoramento e controle de redes onde a autonomia das baterias é crítica, permitindo que dispositivos operem por um longo período de tempo sem manutenção.

A lógica e topologia de rede garantem a eficiência do sistema. Diferentemente de redes conhecidas, como Wi-Fi, que utilizam topologia estrela, onde todos os dispositivos finais se comunicam com um ponto central. A rede Zigbee utiliza a topologia em malha (*mesh*). Nessa configuração, os dispositivos não apenas enviam o seu próprio dados, mas também podem atuar como roteadores, transmitindo as

mensagens de outros nós. Isso cria múltiplos caminhos possíveis para a informação trafegar, aumentando o alcance e a tolerância a falhas na rede. Na Figura 1, observam-se essas topologias.

Figura 1 - Topologias malha (*mesh)* e estrela (*star*).



Fonte: boosthigh (2025).

Para compreender a dinâmica ilustrada na Figura 1, deve-se observar qeu cada círculo representa um nó da rede e as linhas indicam os alcance de comunicação sem fio. O protocolo classifica esses nós em três tipos lógicos de dispositivos, conforme detalhado nos guias da Texas Instruments(2025):

1. Coordenador (Zigbee *Coordinator*): Dispositivo raiz da rede, responsável por iniciá-la, selecionar o canal de frequência e gerir chaves de segurança.
2. Roteador (Zigbee *Router*): Atua como nó intermediário, estendendo o alcance da rede.
3. Dispositivo Final (*End Device*): São sensores ou atuadores alimentados por bateria. Entram em modo "sono" (*sleep mode*) para economizar energia, comunicando com seu roteador associado ou coordenador quando necessário para transmitir dados coletados, reportar mudanças de estado ou verificar o recebimento de mensagens pendentes.

Na topologia *mesh*, a comunicação de um nó com outro é realizada por meio de algoritmos de roteamento dinâmico, em que os roteadores repassam as mensagens de forma colaborativa até o destino. Devido a esse mecanismo, se um nó falhar, a camada da rede encontra automaticamente uma rota alternativa para entregar os dados.

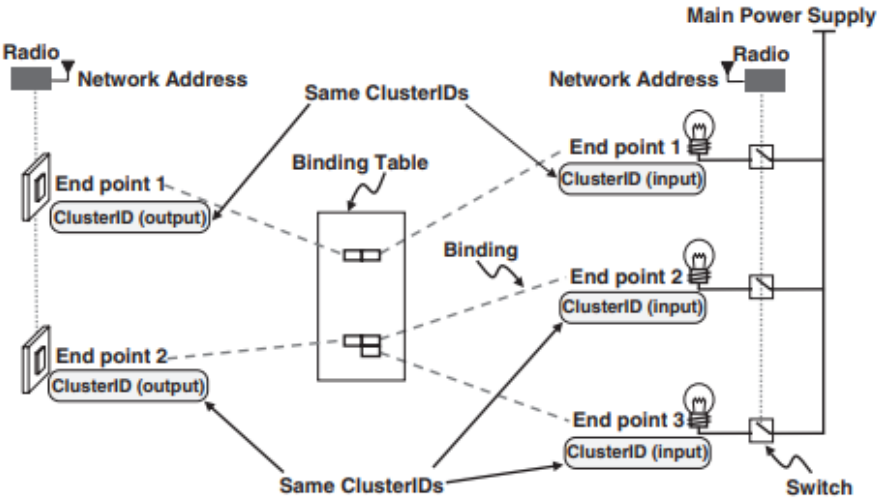
Para que diferentes fabricantes pudessem trabalhar juntos, como, por exemplo, a integração entre uma lâmpada de fabricante X e um interruptor de

fabricante Y, o protocolo utiliza o Zigbee Cluster Library (ZCL) (ZIGBEE ALLIANCE, 2021). A especificação ZCL define um "*cluster*" como um conjunto de atributos e comandos padronizados relacionados a uma função específica. Por exemplo, o *Cluster On*/*Off* define os atributos para ler o estado de um dispositivo (ligado/desligado) e os comandos para alterar esse estado. Esta padronização permite a interoperabilidade no protocolo Zigbee.

Nessa estrutura, a comunicação é organizada por meio de um modelo cliente/servidor, no qual cada *cluster* possui um lado servidor, que armazena os dados ou o estado real do dispositivo, e um lado cliente, responsável pelo envio de comandos para consulta ou alteração dessas informações. No contexto deste trabalho, essa arquitetura é aplicada através dos *clusters* de Medição de Temperatura (*Temperature Measurement*) e de Umidade Relativa (*Relative Humidity Measurement*), o sensor implementa o lado servidor, reportando dados lidos pelo *hardware*, enquanto o coordenador ou *gateway* atua como cliente, recebendo e processando essa informação.

Para exemplificar, a Figura 2 ilustra um processo de *binding* em um cenário clássico de controle de iluminação, demonstrando um mecanismo que segue a mesma lógica utilizada pelos sensores deste trabalho. Para compreender os elementos da imagem, observa-se que cada dispositivo físico possui o seu próprio identificador na rede (*Network Address*). A esse endereço de rádio estão atrelados os pontos de extremidade (*End points*), que representam as aplicações lógicas individuais dentro de um mesmo dispositivo físico (como os botões independentes de um interruptor múltiplo). Cada *Endpoint*, por sua vez, carrega um identificador de função (*ClusterID*). O vínculo ocorre de forma automática porque os interruptores (*clusters* de saída) são mapeados para as lâmpadas (*clusters* de entrada) que compartilham o mesmo código de função (*Same ClusterIDs*). Todo esse mapeamento de endereços e rotas fica registrado em uma tabela de vinculação (*Binding Table*), que é tipicamente armazenada de forma centralizada na memória do Coordenador da rede.

Figura 2 - Relações de *Binding* em um exemplo de controle de iluminação.



Fonte: Farahani (2008, p.116).

Para viabilizar essa comunicação de forma eficiente, o protocolo utiliza o mecanismo de *binding*, que estabelece uma associação lógica direta entre *endpoints* de dispositivos diferentes, conectando *clusters* de saída a *clusters* de entrada e eliminando a necessidade de especificar explicitamente o endereço de destino em cada transmissão. Ao contrário do *broadcast*, o *binding* recupera automaticamente o endereço do nó associado na camada de rede, garantindo transmissões direcionadas (*unicast*). Essas associações são armazenadas em uma tabela de *binding*, garantindo o roteamento automático das mensagens, reduzindo o uso de transmissão em *broadcast*, minimizando a latência e o consumo de energia e aumentando a autonomia e a robustez da rede. Essa robustez ocorre porque, após a configuração inicial das rotas, o tráfego de dados passa a ser realizado diretamente entre os nós ou repassado pelos roteadores intermediários, permitindo que dispositivos vinculados mantenham comunicações básicas mesmo na ausência temporária do coordenador (ZIGBEE ALLIANCE, 2021).

Os comandos para estabelecer esse vínculo são padronizados pelo protocolo Zigbee e podem ser enviados por qualquer dispositivo, inclusive pelo coordenador da rede. Uma das principais vantagens de se estabelecer esse bind é a possibilidade de automatizar o fluxo de informações por meio do recurso de reporting (relatório automático). Com o vínculo estabelecido, as atualizações de dados entre cliente e servidor podem ser automatizadas sem a necessidade de o

coordenador interrogar o dispositivo constantemente. Esse envio de relatórios pode ser configurado de duas formas principais: por mudança de estado, em que o dispositivo transmite proativamente uma atualização sempre que o valor do atributo sofre uma variação significativa; ou de forma temporizada, enviando os dados em intervalos de tempo máximos e mínimos previamente definidos. Esse mecanismo atua em conjunto com as estratégias de baixo consumo, pois garante que o rádio seja ativado apenas quando estritamente necessário, otimizando o tempo do sensor em modo de suspensão.

A evolução dessa tecnologia tem facilitado no quesito de aplicações reais, desde automação residencial (*Smart Home*) e iluminação inteligente (*Smart Lighting*) até sistemas industriais complexos, como medição avançada de energia (*Smart Metering*) (ZIGBEE ALLIANCE, 2021).

2.3 Gateway

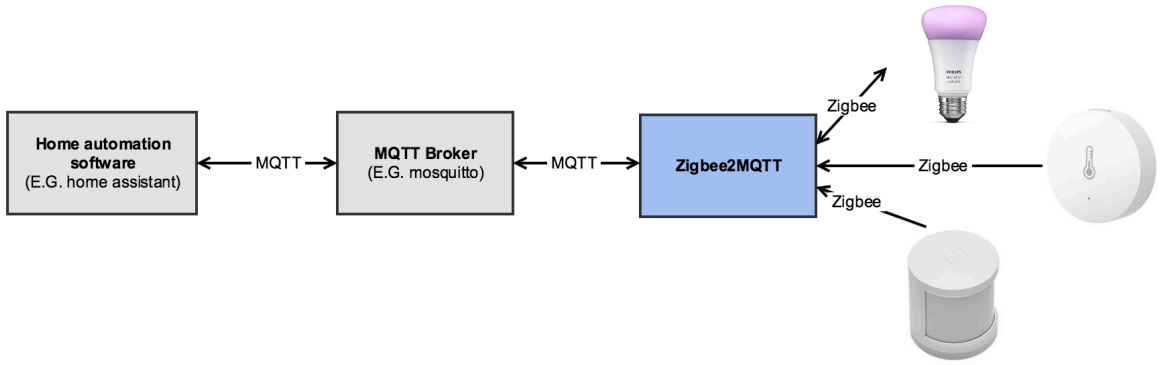
No contexto da IoT, a comunicação entre dispositivos finais (como sensores e atuadores) e o sistema de gestão na nuvem nem sempre é viável ou eficiente, dependendo crucialmente do *gateway*. Esse dispositivo atua como um intermediário traduzindo protocolos locais para protocolos de rede robustos baseados em IP. Para realizar essa função de transporte de dados, o *gateway* utiliza o protocolo MQTT *(Message Queuing Telemetry Transport*), muito utilizado em mensagens IoT devido a sua arquitetura leve, ideal para conectar dispositivos remotos com recursos limitados de código de largura de banda de rede (MQTT.ORG, 2025). Segundo a Amazon Web Services (AWS), o MQTT foi inicialmente projetado para monitoramento de oleodutos através de satélites, necessitando alta confiabilidade e resiliência em ambientes de conectividade crítica (AWS, 2025). Para garantir a integridade do sistema de monitoramento, o protocolo conta com especificações técnicas como o *Last Will and Testament* (LWT) que permitem ao servidor enviar notificações automáticas imediatas caso o sistema detecte interrupções inesperadas de conexões com os sensores (IBM, 2025).

No cenário de automação residencial (*Smart Home*), a implementação de *gateways* baseadas em *software* livre mais popular é o Zigbee2MQTT. Esta solução em *software* permite o uso de adaptador USB Zigbee genérico, um dispositivo de

hardware conhecido como *dongle*, que atua como o rádio coordenador da rede, para controlar dispositivos de diferentes fabricantes, sem a necessidade de *gateways* das próprias marcas, traduzindo os pacotes da rede Zigbee diretamente para mensagens MQTT padronizadas (ZIGBEE2MQTT, 2025). Além disso, o Zigbee2MQTT disponibiliza uma interface web opcional para o gerenciamento e monitoramento de rede. Essa abordagem facilita a integração com plataformas como *Home Assistant*, onde o sistema centraliza as informações em um painel local e privado, executado em um servidor físico residencial como um Raspberry Pi ou computador dedicado. Esta configuração garante a autonomia do sistema, eliminando a dependência de servidores externos de terceiros para o seu funcionamento (KOENKK, 2025).

A Figura 3 apresenta a arquitetura típica de comunicação entre dispositivos Zigbee, o *gateway* responsável pela conversão dos dados e os serviços baseados em MQTT, evidenciando o fluxo completo entre sensores e as aplicações de automação. Nessa estrutura, o Zigbee2MQTT realiza a tradução dos pacotes da rede Zigbee para mensagens MQTT, que são encaminhadas ao *broker*, um componente *software* responsável por gerenciar e distribuir essas informações aos sistemas conectados. O *Home Assistant*, por sua vez, desempenha o papel de camada de aplicação, recebendo os dados, exibindo-os ao usuário e executando rotinas de automação, além de enviar comandos aos dispositivos por meio do mesmo fluxo de comunicação.

Figura 3 - Arquitetura de comunicação entre dispositivos Zigbee, *gateway* Zigbee2MQTT, *broker* MQTT e *software* de automação.



Fonte: Koenkk (2025).

Enquanto Zigbee2MQTT atende mercado residencial e entusiastas, o cenário industrial exige *hardware* dedicado e robusto, como soluções desenvolvidas pela Khomp. Dispositivos dessa categoria, como por exemplo a linha ITG 200, são projetados para operar em ambientes de aplicações críticas, se encarregando da telemetria de sensores de larga escala e convertendo os dados recolhidos para transmissão via *Ethernet* ou rede móveis (KHOMP, 2025). A aplicação dessa tecnologia na manufatura é fundamental para eliminar barreiras de dados, permitindo que informações sejam transmitidas *online* para sistema de gestão. Isso viabiliza estratégias avançadas de manufaturas preditivas e eficiência operacional (BEVYWISE, 2025).

2.4 Microcontrolador CC2652R7

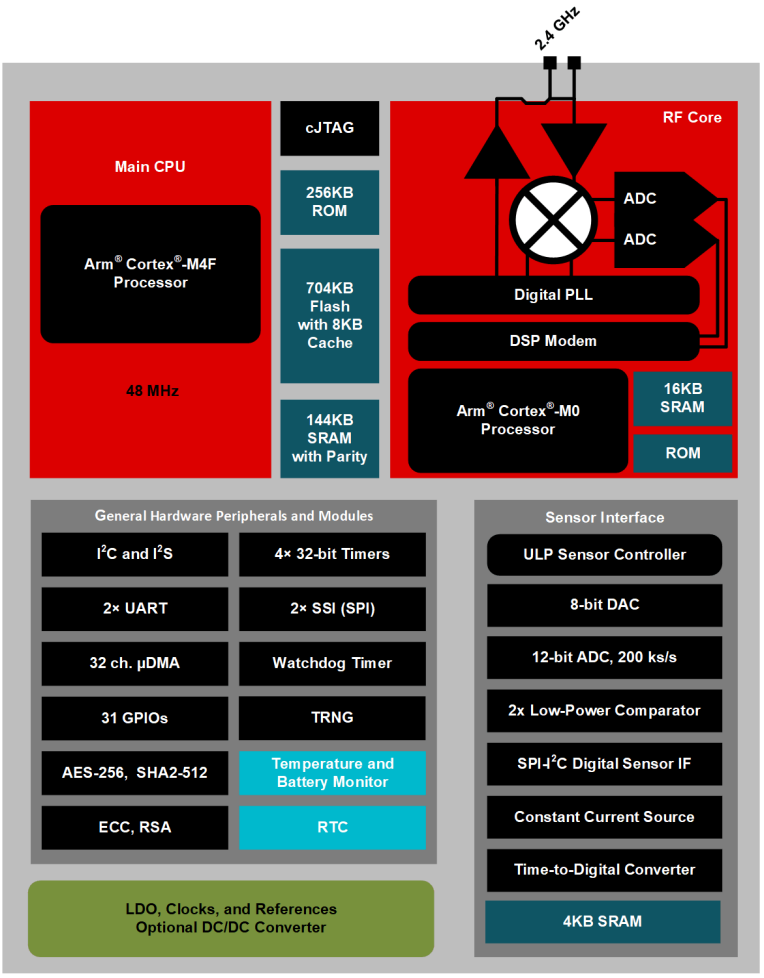
O CC2652R7 é um microcontrolador desenvolvido pela Texas Instruments e integrante da plataforma *SimpleLink*, que de acordo com a própria fabricante, é um "Microcontrolador sem fio multiprotocolo de 2.4 GHz" (TEXAS INSTRUMENTS, 2025). Posiciona-se estrategicamente em um vasto leque de mercados, sendo utilizado em automação predial, automação industrial, infraestrutura de rede, eletrônicos pessoais e aplicações médicas, onde a estabilidade e o baixo consumo são cruciais (TEXAS INSTRUMENTS, 2025).

A fabricante define o dispositivo como de característica *System-on-Chip* (SoC) otimizado para operação de baixa potência (TEXAS INSTRUMENTS, 2025). Composto por um processador Arm® Cortex®-M4F de 48 MHz, além de um núcleo Arm® Cortex®-M0 dedicado a rádio e um controlador de sensor autônomo de 16 bits, capaz de operar enquanto o sistema operacional está em repouso (TEXAS INSTRUMENTS, 2025).

Para suportar operações complexas e atualizações remotas, o dispositivo possui uma arquitetura de memória composta por 704 KB de flash, 256 KB de ROM para bibliotecas e *drivers*, e 144 KB de SRAM protegida por paridade (TEXAS INSTRUMENTS, 2025). O dispositivo também é equipado com diversos periféricos, incluindo interfaces seriais como UART e I2C, módulos de criptografía acelerada por *hardware* (AES, ECC), temporizadores, comparadores e sensores internos de

monitoramento de sistema, conforme detalhado no diagrama de blocos da Figura 4 (TEXAS INSTRUMENTS, 2025).

Figura 4 - Diagrama de blocos do CC2652R7 SoC.



Fonte: TEXAS INSTRUMENTS (2025).

Como apresentado na Figura 4, o microcontrolador integra módulos essenciais para a metodologia deste trabalho. Um elemento relevante explorado neste trabalho é o bloco "*Temperature and Battery Monitor*" (Monitor de temperatura e bateria), destinado ao monitoramento do próprio sistema e compensação de desempenho (TEXAS INSTRUMENTS, 2025). Esse módulo permite obter as leituras de temperatura interna diretamente do próprio SoC, sem a necessidade de sensores externos, e será empregado em uma das aplicações desenvolvidas neste trabalho.

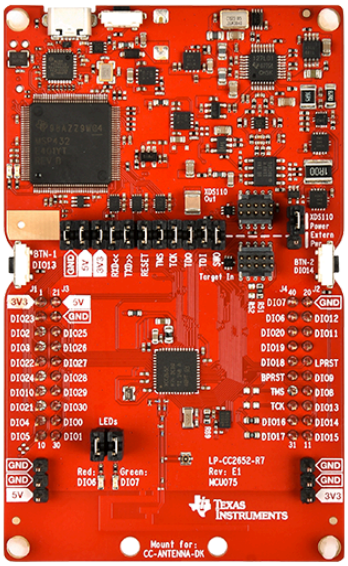
Além disso, a Figura 4 evidencia a presença da interface I2C, utilizada para comunicação com componentes externos, como sensores ambientais e módulos de

expansão (TEXAS INSTRUMENTS, 2025). Esse recurso será utilizado quando houver a necessidade de ampliar a capacidade de monitoramento do sistema.

Outro aspecto relevante é a arquitetura de baixa potência, representada pelo bloco "*ULP Sensor Controller*" (Controlador de Sensor de Ultra Baixa Potência) e por um conjunto de modos de gerenciamento de energia. Esses mecanismos possibilitam a operação eficiente em aplicações que exigem autonomia prolongada, e serão explorados no trabalho para avaliar o comportamento energético do microcontrolador, com apoio das ferramentas de desenvolvimento da Texas Instruments, incluindo o Code Composer Studio™ e recursos do SDK (TEXAS INSTRUMENTS, 2025).

Além do microcontrolador, a Texas Instruments disponibiliza o kit de desenvolvimento CC2652R7 *LaunchPad*, que integra o CC2652R7, circuito de depuração *on-board* XDS110, interface USB e antena para 2,4 GHz. Esse kit é totalmente compatível com o *Code Composer Studio* e com o *SimpleLink Low Power SDK*, permitindo programação, depuração e análise de consumo energético de forma simplificada. Dessa maneira, o *LaunchPad* facilita a prototipagem e validação de aplicações Zigbee, reduzindo o tempo de desenvolvimento e possibilitando testes práticos dos recursos do dispositivo (TEXAS INSTRUMENTS, 2025). A Figura 5 ilustra o kit CC2652R7 *LaunchPad* utilizado neste trabalho.

Figura 5 - Kit de Desenvolvimento CC2652R7 *LaunchPad*.



Fonte: TEXAS INSTRUMENTS (2025)

Para dar suporte ao desenvolvimento de *firmware*, a Texas Instruments fornece o ambiente *SimpleLink* SDK, que inclui a implementação certificada da pilha Zigbee (Z-Stack) acompanhada de diversos códigos de exemplo (como o *zed_light* e *zed_temperaturesensor*). Embora esses projetos básicos sejam ferramentas excelentes para validar o *hardware* e compreender o comissionamento inicial na rede, eles apresentam limitações na arquitetura quando utilizados para produtos finais de IoT.

Os códigos de exemplo dos kits *LaunchPad* são voltados para demonstrar funcionalidades isoladas e geralmente dependem de interação manual, como o pressionamento de botões. Além disso, a configuração padrão de *reporting* e *polling* não é otimizada para operação contínua e autônoma. Quando utilizados sem adaptação, esses exemplos podem apresentar falhas no tratamento de sensores externos, configuração estática de atributos e erros de comunicação com coordenadores genéricos, exigindo a reestruturação da lógica e das máquinas de estado para garantir funcionamento autônomo e robusto.

2.5 Estratégias de Baixo Consumo em Redes Sem Fio

Em aplicações de IoT alimentadas por bateria, a eficiência energética é fundamental. O conceito de *Low Power* em microcontroladores com rádio integrado foca na redução do tempo em que o processador e o módulo de RF (radiofrequência) ficam ativos. Para isso, adota-se a estratégia de *Duty Cycling*, que alterna o dispositivo entre curtos períodos de atividade (para transmissão, recepção e processamento de dados) e longos intervalos de suspensão profunda (*sleep* ou *standby*) (ANASTASI et al., 2009; IEEE, 2020).

No contexto do protocolo Zigbee, os dispositivos finais com limitação de energia (ZED, *Zigbee End Devices*) operam majoritariamente como *Sleepy End Devices*. Nessa configuração, o rádio permanece desligado durante a maior parte do tempo, desconectando-se logicamente da rede para economizar energia. O dispositivo desperta apenas em intervalos previamente configurados para realizar o polling, consultando o roteador ou coordenador responsável sobre a existência de mensagens pendentes, ou de maneira assíncrona para transmitir o relatório de uma leitura de sensor (CONNECTIVITY STANDARDS ALLIANCE, 2023).

Para ilustrar o nível de exigência de aplicações reais, produtos Zigbee comerciais desenvolvidos para automação residencial (como sensores de temperatura e umidade disponíveis no mercado) são projetados para operar durante anos utilizando apenas baterias de célula tipo moeda, como a CR2032, que tem uma capacidade típica de cerca de 220 mAh. Para atingir essa autonomia, o consumo médio de corrente desses dispositivos comerciais é mantido rigorosamente na faixa de poucos microamperes (µA), geralmente variando entre 2 µA e 10 µA no modo ativo, enquanto suas correntes de repouso (*deep sleep*) frequentemente ficam entre 1 µA e 3 µA.

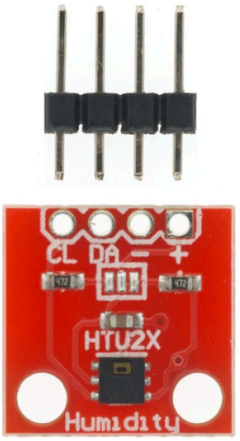
Esse referencial comercial estabelece o rigor técnico necessário no desenvolvimento do *firmware*. Como o consumo durante a transmissão ou recepção ativa do rádio pode alcançar a faixa de dezenas de miliamperes (mA) (IEEE, 2020), a viabilidade do produto depende de uma máquina de estados extremamente otimizada. Ela deve ser capaz de maximizar o tempo em modo de suspensão e garantir que as janelas de comunicação sejam tão curtas e esporádicas quanto a aplicação permitir (ANASTASI et al., 2009).

2.6 Sensor HTU21D

O HTU21D é um sensor digital de umidade e temperatura fabricado pela TE Connectivity, que se destaca no mercado por ser um componente "*plug and play*" dedicado a aplicações que exigem confiabilidade e precisão. De acordo com a folha de dados do fabricante, este componente é um sensor que disponibiliza dados digitais calibrados e compensados, utilizando a interface de comunicação I2C (TE CONNECTIVITY, 2025). A comunicação ocorre por meio de comandos específicos, tais como *Trigger Temperature Measurement, Trigger Humidity Measurement,* leitura do registrador de usuário e comando de reinicialização (*Soft Reset*), os quais permitem ao microcontrolador solicitar medições, configurar parâmetros de operação e monitorar o estado do sensor. A transmissão digital de 16 bits assegura a integridade dos dados ao eliminar a suscetibilidade a ruídos e variações de tensão comuns em sinais analógicos. Complementarmente, o protocolo I2C utiliza somas de verificação (*checksum* CRC) para que o microcontrolador valide a precisão dos bits recebidos e descarte leituras corrompidas.

Estas características são fundamentais para a arquitetura do sistema proposto, pois permite a conexão direta com o microcontrolador sem a necessidade de circuitos complexos de condicionamento de sinal. As características físicas podem ser observadas na Figura 6.

Figura 6 - Sensor HTU21D em módulo comercial.



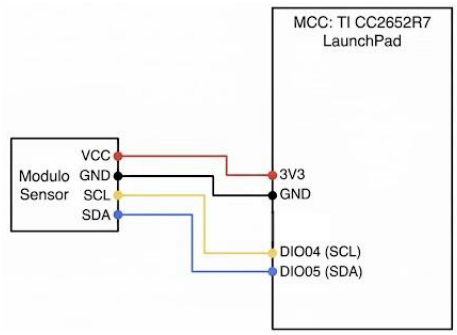
Fonte: makerhero (2025).

Projetado para operar com baixo consumo de energia e apresentando tempo de resposta, de aproximadamente 5s tanto para umidade quanto para temperatura, fatores essenciais para dispositivos IoT alimentados por bateria, o HTU21D oferece resolução e a precisão que permitem monitorar variações no ambiente, tendo uma faixa de operação completa de 0 a 100% de umidade relativa (RH) com precisão de ±2% (na faixa de 20% a 80%) e uma faixa de temperatura estendida de −40°C a +125°C com precisão de ±0,3°C (a 25°C) (TE CONNECTIVITY, 2025) tornando-o apto para aplicações que vão desde automação residencial até monitoramento industrial.

Neste trabalho, a finalidade do HTU21D é atuar como o sensor ambiental externo conectado ao periférico I2C do microcontrolador CC2652R7. Embora o CC2652R7 possua um sensor de temperatura interno, este é destinado prioritariamente ao monitoramento térmico do próprio silício. Assim, o uso de um componente externo tem o objetivo de expandir a capacidade de monitoramento do sistema, proporcionando a leitura da umidade relativa do ar e uma leitura de temperatura ambiente mais fiel, desacoplado do aquecimento do processador. O

módulo utilizado dispõe de resistores *pull-up* de 4,7 kΩ integrados às linhas SDA e SCL. Essa característica assegura o correto funcionamento da comunicação I2C sem a necessidade de componentes externos adicionais. A Figura 7 apresenta o diagrama de conexões entre sensor HTU21D e o microcontrolador CC2652R7.

Figura 7 - Diagrama de conexão do sensor HTU21D ao microcontrolador CC2652R7.



Fonte: Elaboração própria (2025).

2.7 Resumo

A partir da fundamentação teórica exposta, foram apresentados os conceitos fundamentais que sustentam o desenvolvimento do trabalho. A revisão bibliográfica sobre IoT e protocolos de comunicação evidenciou a importância de tecnologias padronizadas e eficientes, justificando a escolha do Zigbee para rede de sensores sem fio e do MQTT para camada de transporte e integração com o *gateway*.

A análise técnica do microcontrolador CC2652R7 demonstrou sua adequação aos requisitos do projeto, destacando-se pelo processador dedicado a rádio e pelos recursos de gerenciamento de energia. Nesse contexto, a revisão sobre estratégias de *Low Power* estabeleceu as diretrizes para a operação eficiente na faixa de microamperes, evidenciando também a necessidade de adaptar os códigos de exemplo da Texas Instruments, que originalmente apresentam limitações para uma operação autônoma e contínua.

Simultaneamente, o estudo das especificações do sensor HTU21D forneceu os subsídios necessários para a implementação da etapa de aquisição de dados ambientais, agregando parâmetros fundamentais de precisão e rápido tempo de resposta ao sistema.

Com a base teórica consolidada, o próximo capítulo detalhará a metodologia aplicada para configurar o ambiente de desenvolvimento, implementar o *firmware* e integrar esses componentes em uma solução funcional.

3 METODOLOGIA

Este capítulo descreve os materiais, ferramentas e os procedimentos experimentais adotados para o desenvolvimento do sistema de monitoramento de temperatura e umidade utilizando o microcontrolador CC2652R7 e o protocolo Zigbee. A metodologia foi dividida em etapas que abrangem a configuração do ambiente, a seleção e adaptação dos códigos de exemplo (*firmware*), a integração com o *gateway* Zigbee2MQTT e a análise de consumo energético.

3.1 Materiais e Configuração do Ambiente

Para o desenvolvimento do firmware, utilizou-se o ambiente de desenvolvimento integrado (IDE) Code Composer Studio (CCS), em conjunto com o *SimpleLink Low Power* SDK, fornecido pela Texas Instruments. Esse ambiente possui exemplos prontos, ferramentas de configuração gráfica de periféricos e pinos do microcontrolador (SysConfig) e recursos de depuração adequados ao desenvolvimento de aplicações Zigbee.

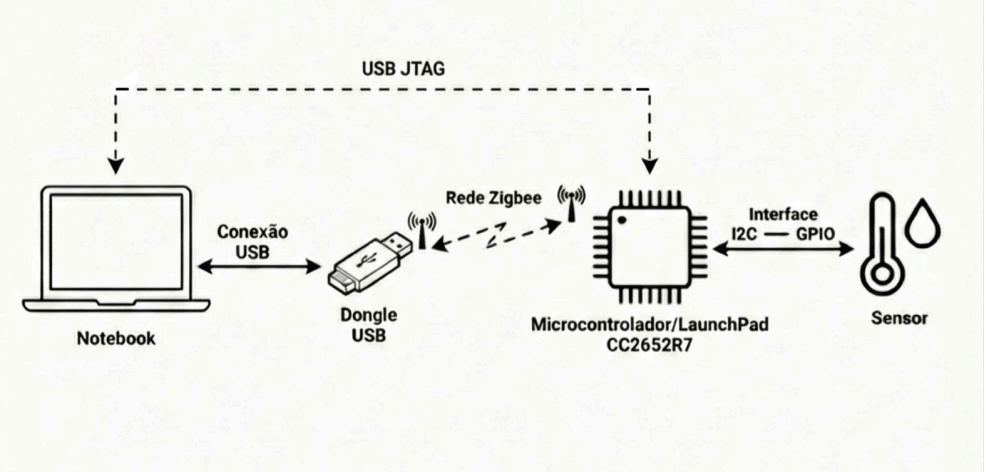
3.1.1 Dispositivos: ZC, ZR e ZED

No decorrer do desenvolvimento, analisaram-se as diferenças entre os exemplos fornecidos pelo SDK, quais permitem configurar o microcontrolador para operar como *Zigbee Coordinator* (ZC), *Zigbee Router* (ZR) e *Zigbee End Device* (ZED). Para este projeto, o *gateway* (Zigbee2MQTT) atua como o coordenador, sendo responsável por iniciar e gerenciar a rede, enquanto o CC2652R7 foi configurado utilizando exemplos do tipo ZED, visando à operação como um nó sensor alimentado por bateria.

A Figura 8 apresenta o diagrama de conexões do sistema experimental utilizado neste trabalho. Nela, observa-se o computador host executando o serviço Zigbee2MQTT, conectado a um dongle USB Zigbee responsável pela função de coordenador de rede. Conforme detalhado na fundamentação teórica, este dispositivo exerce a função de coordenador de rede, atuando como a interface física responsável por estabelecer a malha Zigbee e gerenciar a comunicação com os dispositivos finais. O microcontrolador CC2652R7 atua como dispositivo final (ZED),

comunicando-se sem fio com o coordenador, enquanto o sensor HTU21D é conectado ao microcontrolador por meio do barramento I2C para aquisição de dados.

Figura 8 - Diagrama de conexões do sistema: Notebook, Dongle USB, LaunchPad e Sensor.



Fonte: Elaboração própria (2025)

3.1.2 Configuração do *Gateway* Zigbee2MQTT

A inicialização da infraestrutura de recepção dos dados baseou-se no projeto *open-source* Zigbee2MQTT, utilizando um adaptador USB coordenador conectado a um computador *host*. Todo o processo de inicialização e gerenciamento direto do serviço é realizado por meio de linhas de comando no terminal do sistema operacional do computador *host*. Para este trabalho, o serviço foi executado através do comando "*npm start",* conforme evidenciado no *log* apresentado na Figura 9 nas primeiras linhas do terminal.

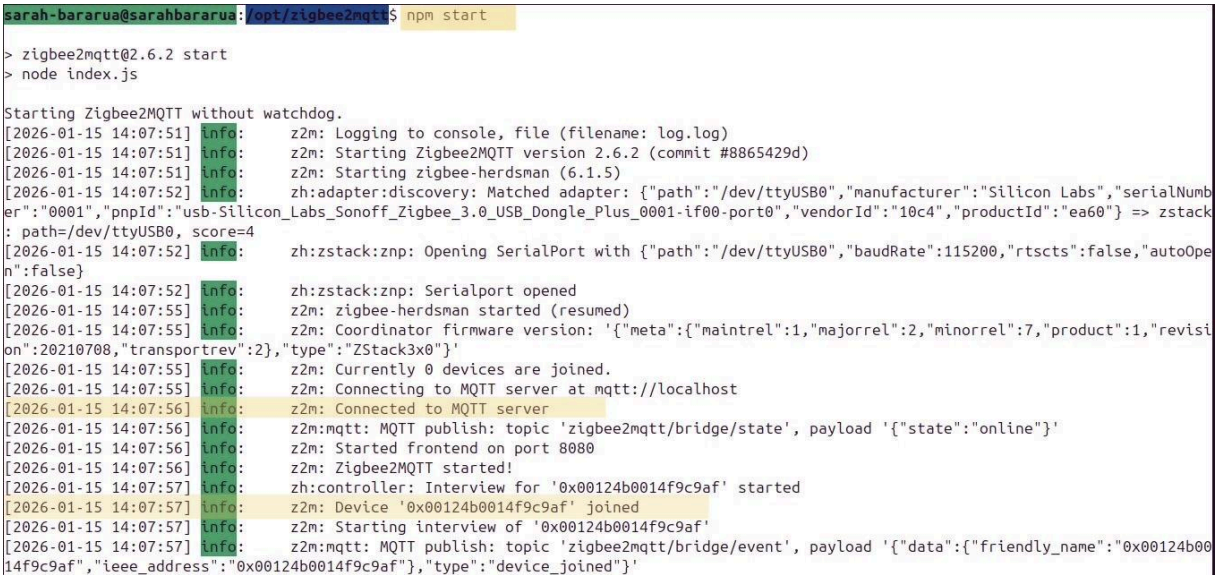
Após a inicialização dos *drivers* do adaptador, habilitou-se a permissão de entrada de novos dispositivos na rede (*permit join*). O microcontrolador CC2652R7, ao ser energizado, iniciou o processo de associação (*beacon request*), sendo reconhecido pelo coordenador. Este evento pode ser observado na Figura 9, conforme destacado na linha 14:07:57, pela mensagem "*Device '0x00124b0014f9c9af' joined*".

Adicionalmente, observa-se no *log* a conexão bem sucedida com o *broker* MQTT, o serviço de software responsável pelo gerenciamento das mensagens, evidenciada na linha 14:07:56 pela mensagem "*Connected to MQTT server*",

indicando que a infraestrutura de comunicação estava plenamente operacional no momento da associação com dispositivo final.

A Figura 9 apresenta o log de execução no terminal do sistema operacional, detalhando a sequência cronológica de eventos da infraestrutura. Nas linhas iniciais, observa-se a inicialização dos serviços lógicos e a conexão com o *broker* MQTT. Na sequência, o registro confirma a entrada do dispositivo final na rede Zigbee após o processo de associação. Essa visualização comprova a integração bem-sucedida entre o hardware coordenador, o serviço de tradução de protocolos e o nó sensor remoto.

Figura 9 - Inicialização do serviço Zigbee2MQTT e associação do dispositivo final.



Fonte: Elaboração própria (2025)

3.2 Desenvolvimento do Firmware

O desenvolvimento do código para o CC2652R7 seguiu uma abordagem iterativa e experimental, partindo de exemplos fornecidos pelo SDK da Texas Instruments e evoluindo para a aplicação final com sensores externos.

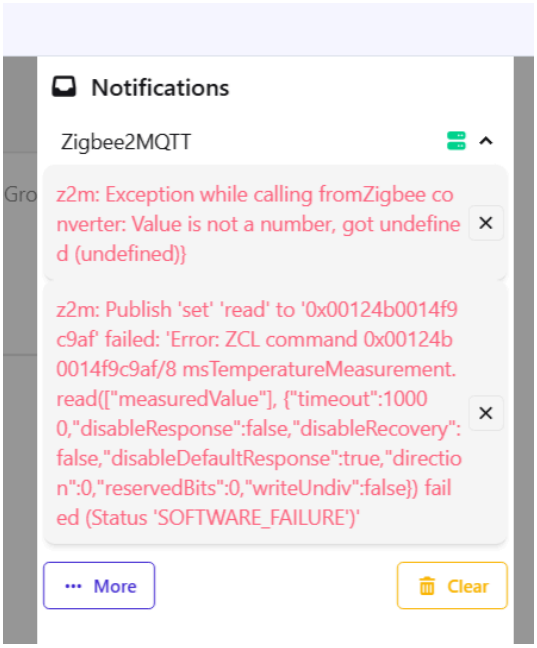
3.2.1 Testes Iniciais e Validação de Comunicação

Como ponto de partida, optou-se por utilizar o exemplo nativo de "Sensor de Temperatura" (*zed_temperaturesensor*) fornecido pelo SimpleLink SDK, destinado à leitura do sensor interno de temperatura do microcontrolador. A escolha desde

exemplo teve como objetivo validar, simultaneamente, o funcionamento da pilha Zigbee e a capacidade de envio de dados de sensoriamento sem a necessidade de *hardware* externo.

Durante os testes, observou-se que, embora o dispositivo fosse corretamente reconhecido pelo Zigbee2MQTT, havia falhas críticas na atualização dos dados. É importante mencionar que, mesmo com o serviço sendo executado no terminal, o Zigbee2MQTT disponibiliza uma interface gráfica web. Embora o monitoramento possa ser realizado diretamente no terminal, a interface gráfica proporciona uma visualização mais intuitiva e estruturada do status de rede, das conexões ativas, dos valores das medições em tempo real e das notificações do sistema. No cenário testado o dispositivo comunicava-se com o Zigbee2MQTT, porém o valor de temperatura permanecia estático em uma medição pré-definida, não sofrendo alterações mesmo quando a leitura era solicitada manualmente através do console de desenvolvedor (*Dev Console*) presente nessa interface web, resultando na notificação de erros conforme apontado na Figura 10. Na imagem, observa-se o erro '*Exception while calling fromZigbee converter*', indicando que o serviço recebeu um valor indefinido (*undefined*) em vez de um dado numérico válido. Adicionalmente, a notificação aponta uma falha no comando '*msTemperatureMeasurement*', evidenciando que a pilha Zigbee do microcontrolador não foi capaz de processar ou responder à requisição de leitura dentro do tempo limite (*timeout*)."

Figura 10 - Notificação de erro de leitura manual no Dev Console do Zigbee2MQTT.



Fonte: Elaboração própria (2025)

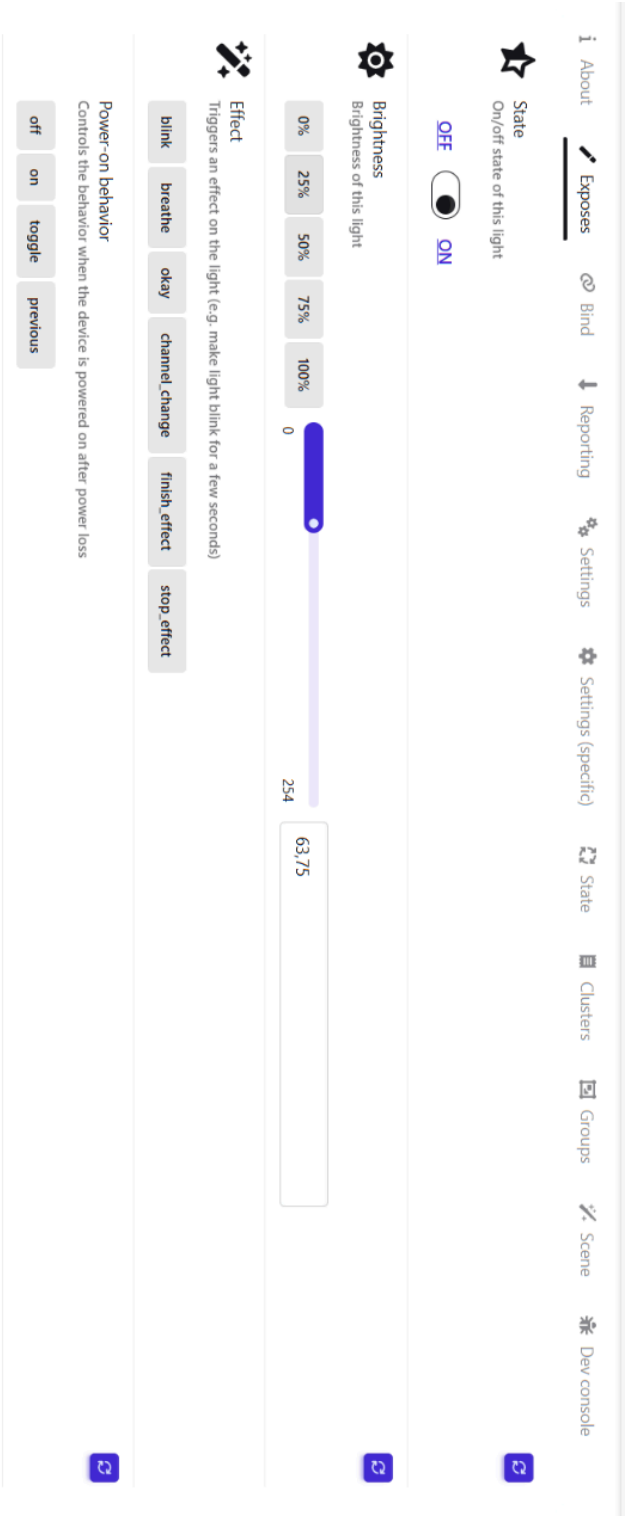
Após análise, diagnosticou-se que o código de exemplo possuía limitações na rotina de atualização periódica e resposta a comandos de leitura (*polling*), o que inviabilizava sua utilização direta como base para o trabalho. Optou-se, então, por buscar um exemplo mais funcional.

3.2.2 Adaptação do Exemplo

Após a identificação das limitações do primeiro exemplo, foi selecionado um código exemplo de "Controle de Luz/Interruptor" (*zed_light*), também disponibilizado no *SimpleLink* SDK. Esse exemplo foi escolhido por apresentar uma implementação mais funcional de comunicação Zigbee, incluindo atualizações dinâmicas de estados e resposta adequada a comandos enviados pelo coordenador. A funcionalidade é verificada fisicamente pela manipulação dos pinos de GPIO do microcontrolador CC2652R7. O *firmware* processa comandos de rádio para alterar o nível lógico das saídas, controlando periféricos e monitorando interrupções de *hardware* para reportar mudanças ao coordenador.

Os testes realizados com esse código demonstram comunicação estável com o Zigbee2MQTT, conforme demonstrado na Figura 11. A apresentação completa dos parâmetros da lâmpada na aba *Exposes* da interface web atua como um indicador técnico de confiabilidade da pilha Zigbee. Isso ocorre porque a exibição correta dos controles na interface exige conclusão bem-sucedida da entrevista (*interview*) e a troca de *clusters* durante o pareamento. Falhas nessa comunicação resultam em uma interface vazia. Portanto, o fato de a interface exibir os parâmetros padrões de uma lâmpada confirmou que a comunicação bidirecional estava operante. A partir dessa validação, o código de luz passou a ser utilizado como base para modificações subsequentes do trabalho.

Figura 11 - Interface do Zigbee2MQTT exibindo o estado do dispositivo *zed_light*.



Fonte: Elaboração própria (2025)

Com a comunicação Zigbee devidamente validada, iniciou-se o processo de modificação do código base para incluir a leitura do sensor interno de temperatura

do microcontrolador. As rotinas de leitura foram incorporadas ao projeto, mantendo a estrutura funcional do exemplo.

Quadro 1 - Função implementada para leitura do sensor de temperatura interno do CC2652R7.

```
static int16_t zclSampleLight_readTemperature(void)
{
    AONBatMonEnable();
    float temp_em_float = AONBatMonTemperatureGetDegC();
    int16_t temp_para_zigbee = (int16_t)(temp_em_float * 100.0f);
    return temp_para_zigbee;
}
```

Fonte: Elaboração própria (2025)

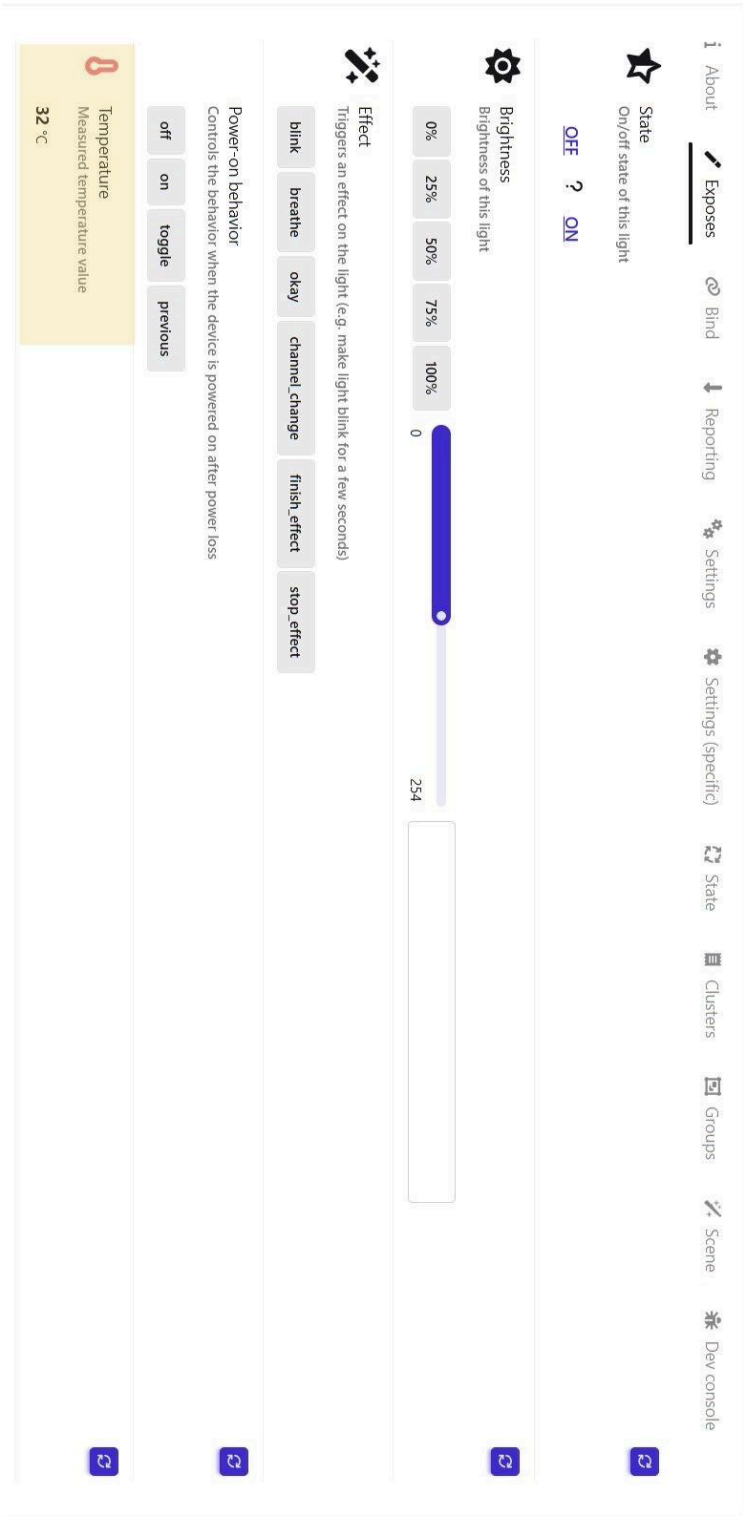
O Quadro 1 apresenta a função responsável pela leitura da temperatura interna do microcontrolador. Para isso, o código utiliza a função AONBatMonEnable(), nativa da biblioteca da Texas Instruments, que é responsável por habilitar e energizar o periférico interno de monitoramento de temperatura. Em seguida, a leitura é realizada e o valor retornado é convertido para o formato exigido pelo *cluster* Zigbee. Visto que a especificação do protocolo determina que a temperatura deve ser transmitida como um número inteiro com precisão fixa de duas casas decimais, realiza-se a multiplicação do valor em ponto flutuante (*float*) por 100 antes da conversão, possibilitando sua transmissão pelo coordenador.

Os testes realizados mostraram que a comunicação Zigbee manteve-se compatível e os dados foram recebidos corretamente pelo *gateway*. Observou-se que a leitura do sensor interno indica variação de até aproximadamente 10°C superior em relação à temperatura ambiente, comportamento esperado para sensores internos de microcontroladores, cuja função é monitoramento do próprio chip e não medição de temperatura ambiente.

O resultado visual dessa implementação pode ser observado no destaque da Figura 12, onde o Zigbee2MQTT apresenta o dispositivo modificado. Embora o foco seja a telemetria, os seletores de estado (*State*), brilho (*Brightness*) e efeito (*Effect*) permanecem visíveis por serem recursos nativos do modelo de iluminação (*zed_light*) utilizado como base. Essa estratégia permitiu validar a comunicação bidirecional do nó, confirmando que o dispositivo permanece apto a receber comandos do coordenador enquanto transmite os dados de temperatura interna revelados na interface. Esta integração confirma que a temperatura interna foi

recebida com sucesso, o que valida a estratégia de reaproveitar a estrutura de endpoints do modelo de iluminação para a transmissão de dados de sensores.

Figura 12 - Interface do Zigbee2MQTT exibindo a funcionalidade híbrida: controle de iluminação (código base) e leitura de temperatura interna (adaptação).



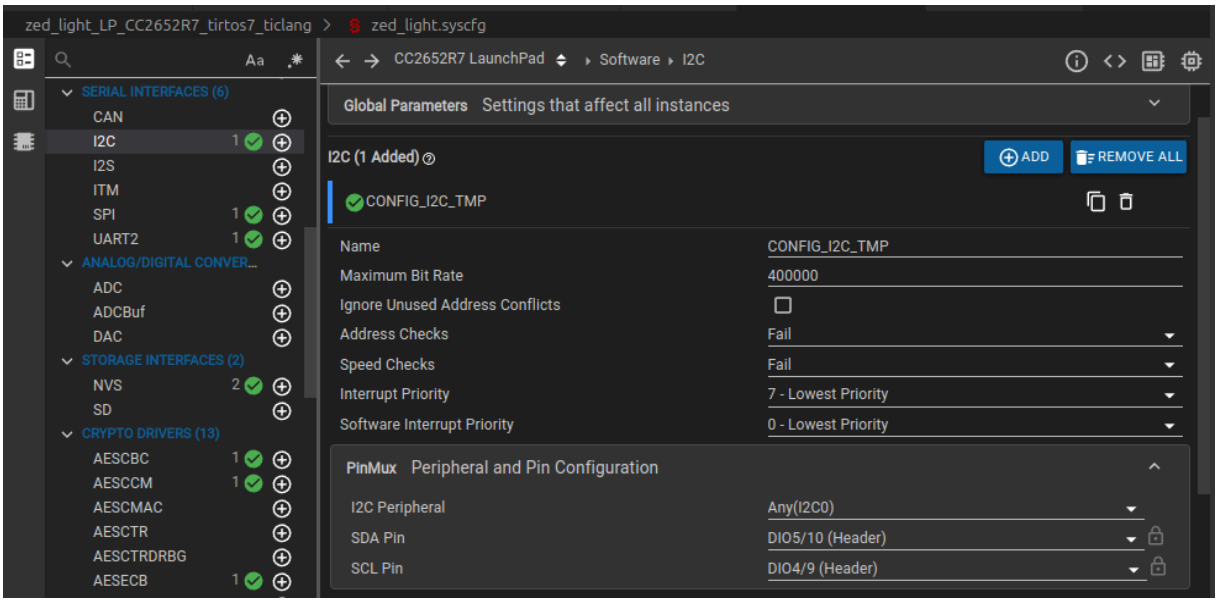
Fonte: Elaboração própria (2025)

Essa etapa permitiu validar simultaneamente a leitura do sensor interno e o envio dinâmico dos dados pela rede Zigbee.

3.2.3 Integração do Sensor Externo (HTU21D)

Validada a transmissão via Zigbee, a etapa seguinte constituiu na integração do sensor externo de temperatura e umidade por meio da interface I2C. Inicialmente, a interface foi habilitada no SysConfig, uma ferramenta de configuração visual da Texas Instruments que simplifica a alocação de periféricos e a geração de código de inicialização de hardware. Na Figura 13, observa-se a instância do periférico CONFIG_I2C_TMP a 400kbps, com o mapeamento físico via PinMux dos sinais SCL e SDA nos pinos GPIO4 e GPIO5, respectivamente

Figura 13 - Configuração dos pinos da interface I2C na ferramenta SysConfig.



Fonte: Elaboração própria (2025)

No código foram implementadas as rotinas de inicialização do I2C, bem como as funções responsáveis pela leitura de temperatura e umidade do sensor HTU21D. A função apresentada no Quadro 2 realiza a comunicação I2C com o sensor, executa a conversão dos valores brutos conforme as equações definidas em *datasheet* e armazena os resultados no formato exigido pela *Zigbee Cluster Library (ZCL).*

Quadro 2 - Implementação da leitura de temperatura e umidade do sensor HTU21D via barramento I2C.

```
static void zclSampleLight_readAllSensors(void)
{
    // --- PREPARAR I2C (Externo) ---
    I2C_Handle      i2c;
    I2C_Params      i2cParams;
    I2C_Transaction i2cTransaction;
    uint8_t         txBuffer[1];
    uint8_t         rxBuffer[3];
    bool            status;
    uint16_t        currentHumidity = 0; // Variável local para umidade

    I2C_Params_init(&i2cParams);
    i2cParams.bitRate = I2C_400kHz;
    i2c = I2C_open(CONFIG_I2C_TMP, &i2cParams);

    if (i2c != NULL) {

        // >>> LEITURA A: TEMPERATURA <<<
        txBuffer[0] = TRIGGER_TEMP_MEASURE_HOLD;
        i2cTransaction.targetAddress = HTU21D_ADDR;
        i2cTransaction.writeBuf = txBuffer;
        i2cTransaction.writeCount = 1;
        i2cTransaction.readBuf = rxBuffer;
        i2cTransaction.readCount = 3;

        status = I2C_transfer(i2c, &i2cTransaction);
        if (status) {
            uint16_t rawTemp = (rxBuffer[0] << 8) | (rxBuffer[1] & 0xFC);
            float tempExtFloat = -46.85 + 175.72 * ((float)rawTemp /
65536.0);
            currentTempExternal = (int16_t)(tempExtFloat * 100);
        }
        Task_sleep(1000 / Clock_tickPeriod);

        // >>> LEITURA B: UMIDADE <<<
        txBuffer[0] = TRIGGER_HUM_MEASURE_HOLD; // Comando 0xE5
        i2cTransaction.writeBuf = txBuffer; // Reusa o buffer

        status = I2C_transfer(i2c, &i2cTransaction);
        if (status) {

            // Fórmula do Datasheet HTU21D
            uint16_t rawHum = (rxBuffer[0] << 8) | (rxBuffer[1] & 0xFC);
            float humFloat = -6.0 + 125.0 * ((float)rawHum / 65536.0);
            // Ajustes de limites (0 a 100%)
            if (humFloat < 0) humFloat = 0;
            if (humFloat > 100) humFloat = 100;
            // Converte para Zigbee (50.5% -> 5050)
            currentHumidity = (uint16_t)(humFloat * 100);
        }
        I2C_close(i2c);
    }
    // --- ATUALIZAR ZIGBEE ---

    // Temperatura
    if (currentTempExternal != 0) {
```

```
        zclSampleLight_updateTemperatureAttribute(currentTempExternal);
    } else {
        zclSampleLight_updateTemperatureAttribute(currentTempInternal);
    }

    // Umidade
    if (currentHumidity != 0) {
        zclSampleLight_updateHumidityAttribute(currentHumidity);
    }
}
```

Fonte: Elaboração própria (2025)

O Quadro 2 apresenta a função *zclSampleLight_readAllSensors* que é responsável por coordenar o processo de aquisição dos dados do sensor HTU21D. Inicialmente, realiza a configuração e abertura do periférico I2C do microcontrolador na frequência de 400 kHz, utilizando a estrutura *I2C_Transaction* para gerenciar a comunicação no barramento. Em seguida, são enviados os comandos de leitura de temperatura e umidade, e os valores brutos recebidos são convertidos conforme as equações definidas no *datasheet* do fabricante.

O código também utiliza a função *Task_sleep* para respeitar o tempo necessário de conversão do sensor. Por fim, os dados são formatados de acordo com os padrões da ZCL, sendo multiplicados por 100 e convertidos para valores inteiros, de modo a preservar casas decimais de precisão sem a utilização de pontos flutuantes.

Após a aquisição dos dados, os valores de temperatura e umidade são associados aos respectivos *clusters* Zigbee de medição, permitindo sua correta interpretação e publicação pelo *gateway* Zigbee2MQTT. Diferente das configurações de interface realizadas anteriormente no computador *host*, o Quadro 3 apresenta as modificações efetuadas diretamente no código-fonte (*firmware*) do microcontrolador CC2652R7. Nesta etapa, declaram-se os atributos *msTemperatureMeasurement* e *msRelativeHumidity*, vinculando-os a variáveis internas de memória e definindo sua visibilidade para a rede.

Quadro 3 - Configuração dos *Clusters* Zigbee para os sensores de temperatura e umidade.

```
// Dentro de zclSampleLight_Attrs[]:

    // --- TEMPERATURA ---
    {ZCL_CLUSTER_ID_MS_TEMPERATURE_MEASUREMENT,
     {ATTRID_MS_TEMPERATURE_MEASURED_VALUE, ZCL_DATATYPE_INT16,
```

```
ACCESS_CONTROL_READ | ACCESS_REPORTABLE,
     (void *)&zclSampleLight_TemperatureMeasuredValue}},

    // --- UMIDADE ---
    {ZCL_CLUSTER_ID_MS_RELATIVE_HUMIDITY,
     {ATTRID_MS_RELATIVE_HUMIDITY_MEASURED_VALUE, ZCL_DATATYPE_UINT16,
ACCESS_CONTROL_READ | ACCESS_REPORTABLE,
     (void *)&zclSampleLight_HumidityMeasuredValue}},

// Lista de CLUSTERS DE ENTRADA (Serviços do Dispositivo)
const cId_t zclSampleLight_InClusterList[] =
{
  ZCL_CLUSTER_ID_GENERAL_BASIC,
  ZCL_CLUSTER_ID_GENERAL_IDENTIFY,
  ZCL_CLUSTER_ID_GENERAL_GROUPS,
  ZCL_CLUSTER_ID_GENERAL_SCENES,
  ZCL_CLUSTER_ID_GENERAL_ON_OFF,

  // Adicionado: Clusters de Medição
  ZCL_CLUSTER_ID_MS_TEMPERATURE_MEASUREMENT,
  ZCL_CLUSTER_ID_MS_RELATIVE_HUMIDITY
};
```

Fonte: Elaboração própria (2025)

Conforme apresentado no Quadro 3, na definição dos atributos destaca-se o uso da *flag ACCESS_REPORTABLE*. Essa configuração é crítica para dispositivos alimentados por bateria, pois habilita o mecanismo de relatório automático (*reporting*), permitindo que o sensor envie dados proativamente apenas quando houver mudanças significativas ou intervalos de tempo definidos, economizando energia. Além disso, observa-se a tipagem rigorosa dos dados, sendo utilizado o tipo INT16 para temperatura, possibilitando valores negativos, e UINT16 para umidade relativa. A inclusão final desses identificadores na lista *zclSampleLight_InClusterList* constitui a etapa que torna efetivamente esses serviços visíveis e acessíveis ao coordenador durante o processo de descoberta e configuração de rede.

Essa definição no *firmware* é o que permite ao dispositivo informar ao coordenador, durante o processo de entrevista (*interview*), quais tipos de dados ele é capaz de prover. O processo de validação ocorreu de forma gradual: Inicialmente, configurou-se apenas o *cluster* de temperatura (*msTemperatureMeasurement*). Após a confirmação da leitura correta, adicionou-se o *cluster* de umidade (*msRelativeHumidity*). Por fim, todas as funcionalidades remanescentes do exemplo original de controle de luz, código referente a botões e acionamento de LEDs, foram removidas, resultando em uma aplicação dedicada ao sensoriamento.

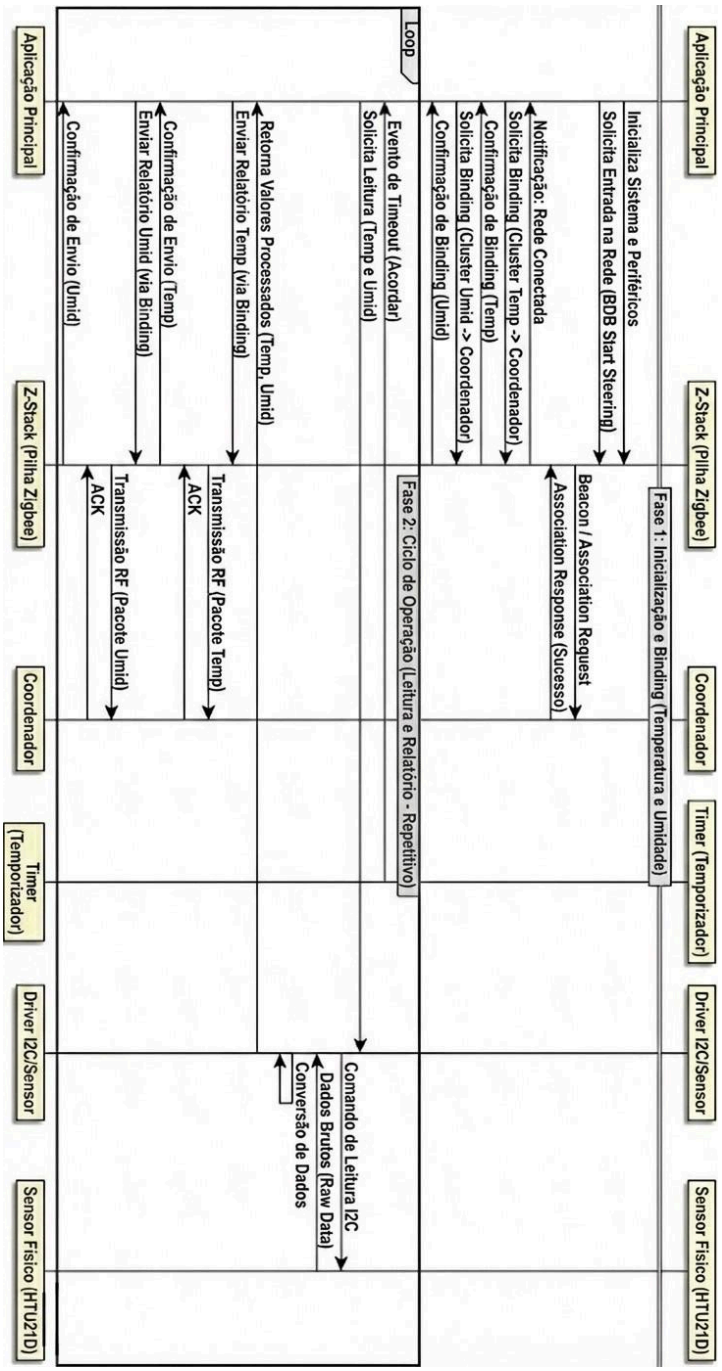
3.3 Lógica de Controle e Fluxo de Execução

A lógica de controle da aplicação foi organizada em um modelo orientado a eventos, visando otimizar o gerenciamento de energia e a resposta do sistema. Após a energização, o microcontrolador realiza a inicialização dos módulos essenciais, incluindo a pilha Zigbee, e verifica na memória não volátil se existem credenciais de rede. Caso o dispositivo seja novo, é iniciado o processo de ingresso na rede; caso contrário, a comunicação é restabelecida automaticamente.

Uma vez integrado à rede, o sistema entra em um ciclo de operação baseado em eventos temporizados. A cada expiração do temporizador, ocorre a aquisição dos dados dos sensores via I2C, seguida do processamento das informações e da atualização dos atributos Zigbee. Quando aplicável, os dados são transmitidos ao coordenador da rede. Finalizado o ciclo, o sistema retorna ao estado de espera (*sleep*).

A Figura 14 apresenta o diagrama de sequência que sintetiza essas etapas, servindo como apoio visual para a compreensão do fluxo geral de execução do projeto.

Figura 14 - Diagrama de Sequência do *Firmware.*



Fonte: Elaboração própria (2025)

3.4 Análise de Consumo Energético

Para avaliar o desempenho da solução, foi utilizada a ferramenta *EnergyTrace*, uma tecnologia desenvolvida pela Texas Instruments e integrada ao Code Composer Studio e ao depurador XDS110 da placa *LaunchPad*. Essa ferramenta realiza a medição do consumo de corrente do microcontrolador e retorna

os resultados em dois formatos complementares: gráficos de perfil em tempo real, que permitem a identificação rápida de picos de corrente durante os estados de transmissão, recepção ou processamento, e tabelas estatísticas detalhadas, que fornecem dados quantitativos exatos de energia total, corrente média e corrente máxima no período analisado.

Esta ferramenta permite a captura de perfis de consumo em tempo real, possibilitando identificar a corrente drenada durante os estados de rádio (transmissão/recepção) e nos períodos de baixo consumo (*standby* ou *sleep*).

As medições foram realizadas de forma comparativa em três cenários distintos, a fim de analisar o impacto das modificações no *firmware*:

1. Código Exemplo: Medição do código do exemplo original da "*zed_light*" fornecido pela TI. Este cenário serve como base para o consumo do Zigbee sem periféricos adicionais.
2. Código com Sensor Interno: Medição após a adição da leitura de temperatura interna, avaliando o custo energético do processamento adicional.
3. Código Final (Sensor Externo): Medição do *firmware* final, com comunicação I2C e leitura do sensor externo ativa.

O perfil de consumo foi analisado nos três estágios distintos de desenvolvimento para quantificar o impacto da adição de periféricos e rotinas de leitura. Em cada estágio foram executadas duas medições distintas com duração de 233 segundos. A escolha desse valor não foi arbitrária, mas sim determinada pelo limite máximo de tempo de captura contínua suportado pelo *buffer* da ferramenta EnergyTrace durante as sessões de depuração nessa configuração de hardware. Para uma análise de baixo consumo (*Low Power*), operar no tempo limite máximo da ferramenta é a abordagem metodológica ideal, pois garante a captura do maior número possível de ciclos alternados entre transmissão ativa e modos de repouso (*sleep*). Quanto maior a janela de tempo amostrada, mais diluídos ficam os ruídos de inicialização e mais fiel se torna o valor da corrente média, refletindo com maior precisão o comportamento real do dispositivo em longo prazo.

O primeiro cenário é sem forçar a comunicação, o dispositivo operou em seu comportamento padrão, enviando relatórios periódicos conforme configurado, mas

sem intervenção externa. O segundo cenário foi forçando a comunicação, durante a captura o dispositivo foi acionado repetidas vezes para forçar o envio imediato de pacotes de dados, simulando um cenário de alto tráfego ou alarme. Os dados quantitativos obtidos nestes testes são detalhados no Capítulo 4 (Apresentação dos Resultados).

4 APRESENTAÇÃO DOS RESULTADOS

Este capítulo apresenta os resultados obtidos a partir da implementação prática do *firmware* no microcontrolador CC2652R7, evidenciando a comunicação bem sucedida com o sensor e a rede Zigbee. Os testes realizados tiveram como foco a validação funcional da aplicação Zigbee adaptada, contemplando desde o reconhecimento dos novos dados até a recepção efetiva de telemetria, os quais foram validados por meio da interface HTML do Zigbee2MQTT.

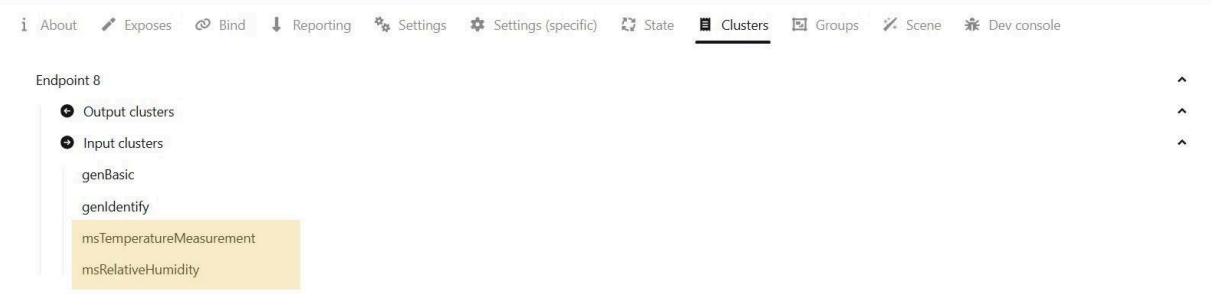
Adicionalmente, é apresentada a análise do consumo de energia do sistema, obtido por meio da ferramenta do *Energy Trace*. Os resultados de consumo são usados para comparação entre as três versões de *firmware*: o código base fornecido nos exemplos do SDK, a versão adaptada para leitura do sensor interno do microcontrolador e a versão final, com leitura do sensor externo HTU21D.

4.1 Análise e discussão dos resultados

A primeira etapa de validação consistiu em verificar o processo de entrevista (*interview*) do dispositivo logo após o pareamento com a rede. O objetivo era confirmar se as alterações realizadas no arquivo zcl_samplelight_data.c (definição dos *endpoints* e *clusters*) seriam corretamente interpretadas pelo coordenador via Zigbee2MQTT.

Observou-se que o dispositivo foi aceito na rede e, diferentemente do exemplo original *zed_light,* que expunha apenas funcionalidades de iluminação, o sistema identificou as novas capacidades de medição. A Figura 15 apresenta a lista de *clusters* ativos reconhecidos pelo *software* de gerenciamento, onde foram destacados os *clusters* de Medição de Temperatura (*msTemperatureMeasurement*) e Umidade Relativa (*msRelativeHumidity*) para evidenciar que ambos foram instanciados corretamente no *Endpoint* configurado.

Figura 15 - Interface exibindo *cluster* de temperatura e umidade.

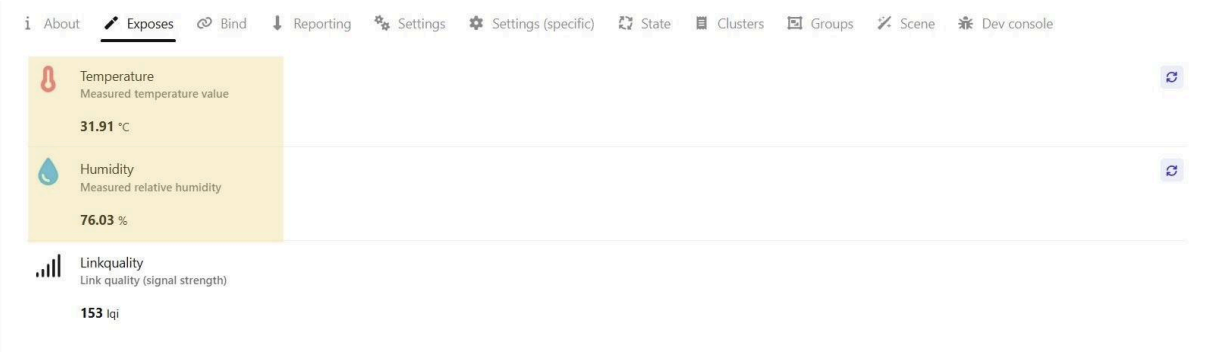


Fonte: Elaboração própria (2025)

Após a validação da estrutura de dados, verificou-se o funcionamento da comunicação I2C e a conversão dos dados do sensor HTU21D. O *firmware* foi configurado para enviar relatórios automáticos (*Attribute Reporting*) baseados na variação dos valores lidos.

A Figura 16 demonstra a aba de exposição de dados ("*Exposes*") do Zigbee2MQTT em operação. É possível observar que o sistema recebe os pacotes Zigbee, decodifica o *payload* hexadecimal e apresenta os valores de temperatura e umidade em formato decimal, conforme destacado na Figura 16.

Figura 16 - Interface de telemetria exibindo os valores de temperatura e umidade recebidos em tempo real.



Fonte: Elaboração própria (2025)

Os dados apresentados na interface mostram-se coerentes com as condições ambientais no momento da execução, indicando que as rotinas de leitura do barramento I2C e as fórmulas de conversão (conforme *datasheet* do sensor) foram implementadas com sucesso. Além disso, a atualização autônoma desses valores

na interface comprova a eficácia do mecanismo de relatórios automáticos configurado na pilha Z-Stack. Diferente de um modelo de consulta contínua (*polling*), no qual o *gateway* precisa solicitar ativamente cada leitura, esse mecanismo permite que o próprio dispositivo final inicie a transmissão dos dados de forma inteligente. A configuração garante que novos pacotes sejam enviados ao coordenador apenas quando ocorrem variações significativas nas grandezas medidas ou quando um intervalo de tempo limite é atingido. Dessa forma, assegura-se o monitoramento contínuo e em tempo real do ambiente de maneira totalmente autônoma, dispensando requisições manuais e otimizando o uso do rádio, característica essencial para nós sensores de baixo consumo

4.2 Análise de Consumo Energético

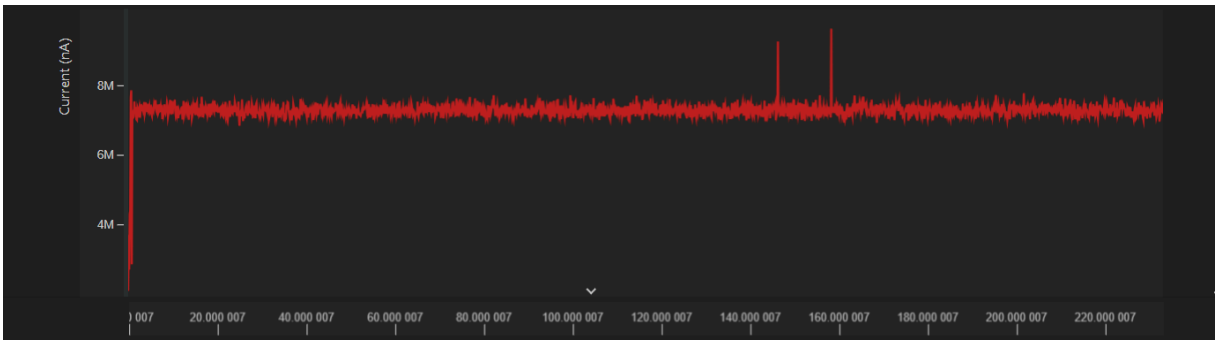
Com o objetivo de avaliar o impacto das modificações realizadas ao longo do desenvolvimento no consumo energético do sistema, foram conduzidas medições utilizando a ferramenta *EnergyTrace*. A análise considera diferentes versões de *firmware*, permitindo comparar o comportamento energético desde o código base fornecido pelo SDK até a versão final otimizada para sensoriamento ambiental e operação com baixo consumo. Para fins de organização e clareza, os resultados são apresentados em três estágios distintos, descritos nas subseções a seguir.

4.2.1 Estágio 1 - Código exemplo

Neste estágio, foram realizadas medições de consumo energético utilizando o código base fornecido pelo SDK, com o objetivo de caracterizar o comportamento do sistema sem modificações funcionais. Inicialmente, o *firmware* foi executado sem forçar a transmissão de dados, permitindo observar o consumo associado apenas à manutenção da rede Zigbee

A Figura 17 o apresenta um consumo energético nesse estágio, evidenciando o consumo constante devido aos periféricos ligados, com pequenos picos de manutenção de rede.

Figura 17 - Gráfico do *EnergyTrace* - Cenário base sem forçar comunicação.

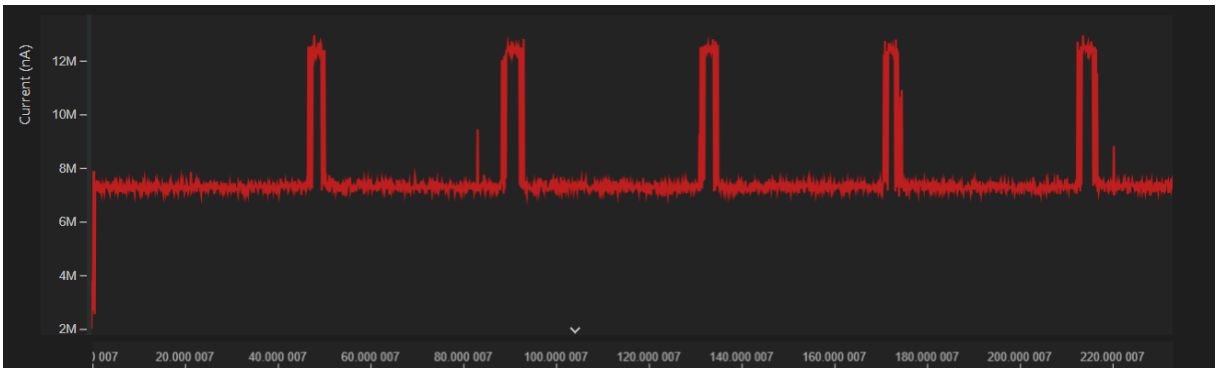


Fonte: Elaboração própria (2025)

Na sequência, foi avaliado o comportamento do sistema quando a comunicação Zigbee foi forçada manualmente, provocando transmissão adicionais de dados. A Figura 18 ilustra o perfil de consumo desse cenário.

Observa-se que forçar a comunicação resulta na presença mais frequente de de picos de correntes, o que impacta diretamente o consumo energético do sistema.

Figura 18 - Gráfico do *EnergyTrace* - Cenário base forçando comunicação.



Fonte: Elaboração própria (2025)

Embora os gráficos ilustrem o perfil visual de consumo, os valores quantitativos exatos foram extraídos da tabela de estatísticas gerada automaticamente pela ferramenta *Energy Trace* ao final de cada captura. A análise desses dados tabulados indica que o ato de forçar a comunicação elevou o consumo médio da corrente em aproximadamente 5% (de 7,29 mA para 7,66 mA). Outro ponto observado foi o aumento da corrente máxima, que saltou de 18,21 mA para 21,01 mA, demonstrando o custo instantâneo associado à transmissão Zigbee.

Com intuito de sintetizar os resultados obtidos nesse estágio, a Tabela 1 apresenta os indicadores de consumo energético para ambos os cenários avaliados.

Tabela 1 – Tipos de energia analisados para o primeiro estágio.

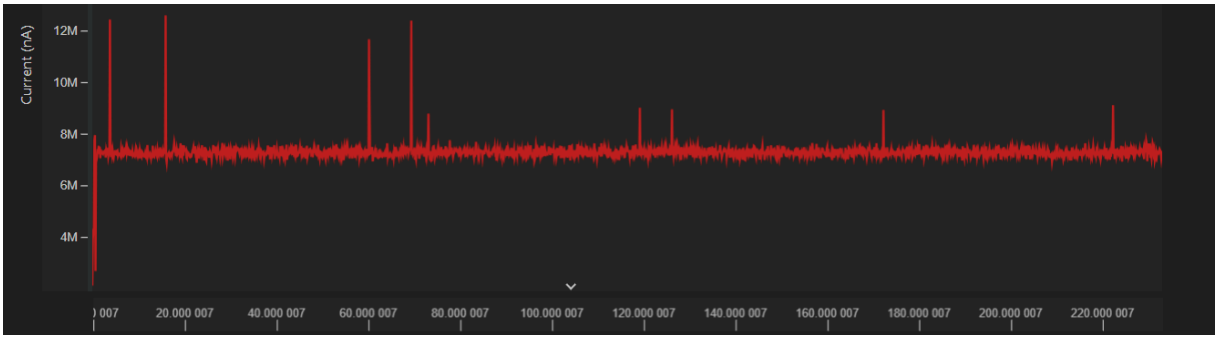
Métrica (233s)	Sem Forçar Comunicação	Forçando Comunicação	Variação
Energia Total	5604,63 mJ	5887,81 mJ	5,05 %
Corrente Média	7,29 mA	7,66 mA	5,05 %
Corrente Máxima	18,21 mA	21,01 mA	15,46 %

Fonte: Elaboração própria (2026).

4.2.2 Estágio 2 - Código com Sensor Interno

Neste estágio é adicionado a leitura do sensor interno do microcontrolador. A Figura 19 apresenta o perfil de consumo energético nesse cenário, evidenciando um comportamento semelhante ao observado no estágio anterior, o que indica que a leitura do sensor interno, por si só, possui baixo impacto no consumo de energia do sistema.

Figura 19 - Gráfico do *EnergyTrace* - Sensor interno sem forçar comunicação.

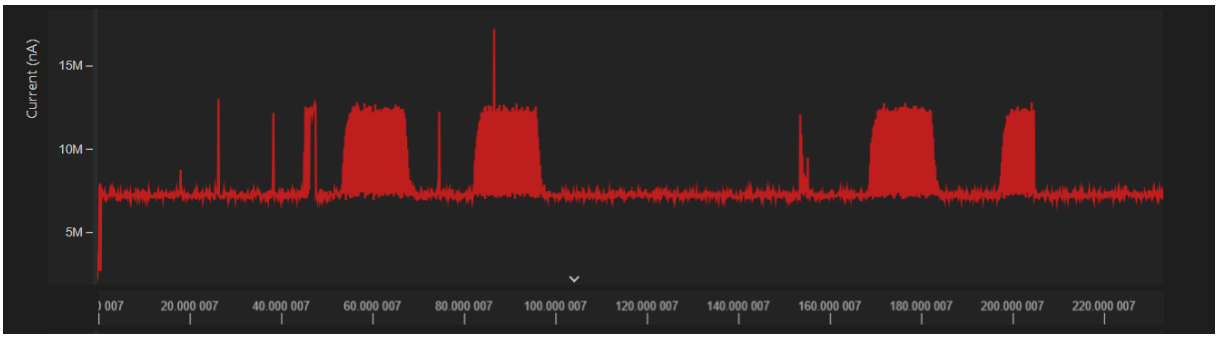


Fonte: Elaboração própria (2025)

Na sequência, foi analisado o comportamento do sistema com a comunicação Zigbee forçada manualmente, de modo a avaliar o impacto combinado do processamento dos dados do sensor e da transmissão. A Figura 20 ilustra o perfil de consumo correspondente a esse cenário.

Observa-se um aumento na densidade e na intensidade dos picos de corrente, associados às etapas de processamento da leitura do sensor, formatação do pacote de dados e ativação do rádio para transmissão, o que resulta em maior demanda energética em relação ao cenário sem forçamento de comunicação.

Figura 20 - Gráfico do *EnergyTrace* - Sensor interno forçando comunicação.



Fonte: Elaboração própria (2025)

A análise quantitativa dos dados indica que a adição da leitura do sensor interno tornou o processo de forçamento de comunicação mais custoso energeticamente. O aumento do consumo médio foi de aproximadamente 8,50% em relação ao estado de repouso, valor superior ao observado no estágio anterior (5,05%). Esse resultado sugere que a sobrecarga de processamento associada à leitura e formatação dos dados, somada à transmissão Zigbee, consome mais energia do que o envio de pacotes sem carga útil significativa.

Com o objetivo de sintetizar os resultados obtidos neste estágio, a Tabela 2 apresenta os principais indicadores de consumo energético para os dois cenários avaliados.

Tabela 2 – Tipos de energia analisados para o segundo estágio.

Métrica (233s)	**Sem Forçar Comunicação**	**Forçando Comunicação**	**Variação**
Energia Total	5605,74 mJ	6081,99 mJ	8,50 %
Corrente Média	7,29 mA	7,91 mA	8,50 %
Corrente Máxima	18,13 mA	20,98 mA	15,73 %

Fonte: Elaboração própria (2026).

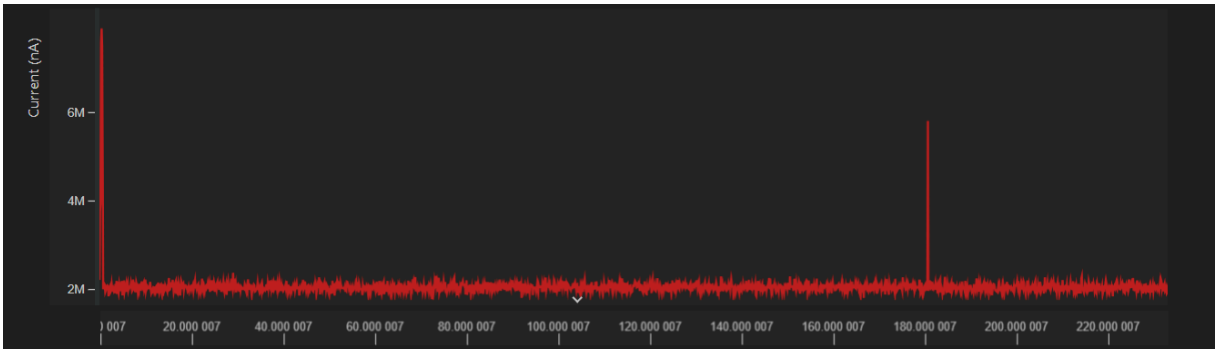
4.2.3 Estágio 3 - Código com Sensor Externo

Na etapa final do desenvolvimento, foram aplicadas otimizações voltadas à utilização do sistema como um produto real alimentado por bateria. Nessa fase, foi implementada a comunicação I2C com o sensor externo HTU21D e foram removidas funcionalidades e periféricos herdados do exemplo original de controle de

iluminação, resultando em um *firmware* dedicado exclusivamente ao sensoriamento ambiental.

A Figura 21 apresenta o perfil de consumo energético do sistema operando sem forçar a comunicação Zigbee. Observa-se uma redução significativa da corrente basal, que se mantém próxima de 2 mA, evidenciando o impacto positivo do desligamento de periféricos não utilizados e da utilização mais eficiente dos modos de baixo consumo do microcontrolador.

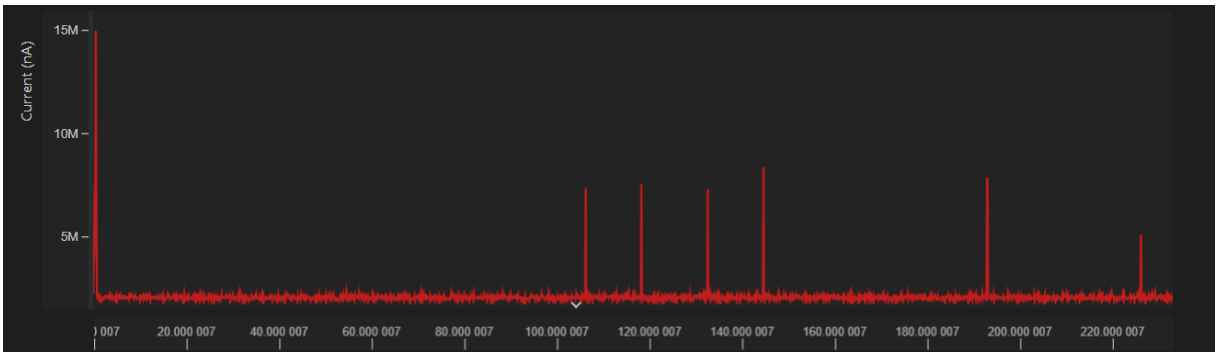
Figura 21 - Gráfico do *EnergyTrace* - Sensor externo sem forçar comunicação.



Fonte: Elaboração própria (2025)

Para efeito de comparação, a Figura 22 ilustra o perfil de consumo quando a comunicação Zigbee é forçada. Em relação aos estágios anteriores, nota-se que os picos de corrente associados à transmissão tornam-se mais estreitos, indicando que a maior parte do tempo de operação ocorre em modo de baixo consumo.

Figura 22 - Gráfico do EnergyTrace - Sensor externo forçando comunicação.



Fonte: Elaboração própria (2025)

Este estágio apresentou os resultados mais expressivos em termos de eficiência energética. A corrente média basal foi reduzida para 2,09 mA,

representando uma diminuição significativa em relação aos estágios anteriores. Além disso, a diferença entre os cenários com e sem forçamento da comunicação mostrou-se desprezível (inferior a 0,1%), indicando que o sistema tornou-se suficientemente eficiente para absorver o custo energético das transmissões eventuais.

Observou-se apenas um leve aumento na corrente máxima, da ordem de 1,44%, o que confirma a ativação do rádio durante a transmissão. No entanto, o impacto na energia total consumida foi diluído pela eficiência geral do sistema, especialmente pelo uso adequado dos modos de *sleep* e pela leitura pontual do sensor externo.

A Tabela 3 apresenta os principais indicadores de consumo energético obtidos neste estágio.

Tabela 3 – Tipos de energia analisados para o terceiro estágio.

Métrica (233s)	**Sem Forçar Comunicação**	**Forçando Comunicação**	**Variação**
Energia Total	1604,82 mJ	1604,13 mJ	-0,04 %
Corrente Média	2,09 mA	2,09 mA	-0,05 %
Corrente Máxima	17,51 mA	17,77 mA	1,44 %

Fonte: Elaboração própria (2026).

A comparação entre os três estágios evidencia a importância da gestão adequada de periféricos em dispositivos IoT. A transição do estágio 1 (código base) para o estágio 3 (código otimizado com sensor externo) resultou em uma redução de aproximadamente 71,3% no consumo médio de corrente, passando de 7,29 mA para 2,09 mA. Enquanto nos estágios 1 e 2 a transmissão de dados impunha um custo adicional visível, da ordem de 5% a 8%, no cenário otimizado do estágio 3 o sistema demonstrou-se robusto o suficiente para que transmissões eventuais não causem impacto significativo na autonomia total.

Com base na corrente média de 2,09 mA visto no estágio 3, é possível projetar a autonomia do dispositivo para um cenário de aplicação contínua. Considerando o consumo na ordem de miliamperes, foi dimensionado a fonte de alimentação com uma bateria de 2000 mAh. A vida útil estimada (T_{Vida}) foi calculada

pela razão entre a capacidade total da bateria (C_{bat}) e a corrente média de consumo (I_{Med}), conforme a equação 1:

$$T_{Vida} = \frac{C_{bat}}{I_{Med}} \quad (1)$$

Substituindo pelos valores experimentais do estágio 3:

$$T_{Vida} = \frac{2000mAh}{2mA} = 1000\ h \quad (2)$$

Convertendo para dias de operação:

$$T_{vida} = \frac{1000h}{24} = 41\ dias \quad (3)$$

O resultado de 41 dias indica que, embora a otimização tenha reduzido o consumo comparado aos estágios anteriores, a autonomia final ainda é limitada para produtos Zigbee comerciais. No mercado de automação residencial e industrial, um dispositivo final sem fio (*End Device*) é considerado viável quando apresenta uma autonomia mínima de um ou dois anos operando com baterias com capacidade inferior, como as células tipo moeda.

Para que essa meta comercial seja atingida, o consumo médio aceitável de um nó sensor deve operar estritamente na faixa de pouco microamperes, variando tipicamente entre 2 µA e 10 µA. Em contrapartida, o protótipo desenvolvido, mesmo em sua versão mais eficiente (estágio 3), apresentou um consumo médio de 2,09 mA, um valor que é ordens de grandeza superior ao referencial de mercado.

Esse contraste evidencia que o sistema atual é plenamente funcional do ponto de vista de comunicação telemétrica e sensoriamento, mas, em sua forma atual, demandaria baterias de alta capacidade para manter a operação, tornando-o pouco aplicável em cenários com restrição severa de espaço e energia. Portanto, esse resultado reforça a necessidade de trabalhos futuros focados na implementação de rotinas de suspensão profunda (deep sleep) mais otimizadas no firmware, capazes de desligar completamente os periféricos ociosos e aproximar o consumo do microcontrolador CC2652R7 da faixa de microamperes exigida pelo setor.

5 CONSIDERAÇÕES FINAIS

O presente trabalho teve como objetivo principal o estudo e a aplicação do microcontrolador CC2652R7, da Texas Instruments, em redes de Internet das Coisas (IoT), utilizando o protocolo Zigbee.

A fundamentação teórica permitiu compreender a complexidade do protocolo Zigbee baseado em *clusters* e atributos. Evidenciou-se que a arquitetura hierárquica do protocolo, embora robusta e amplamente adotada, impõe uma curva de aprendizado acentuada ao desenvolvedor, exigindo o domínio de mecanismos como *binding*, *reporting* e gerenciamento de *endpoints* para a correta implementação de dispositivos interoperáveis.

No desenvolvimento prático, a metodologia adotada, baseada na análise estrutural e adaptação do código de exemplo *zed_light*, mostrou-se eficaz. Foi possível modificar com sucesso a estrutura de dados da Z-*Stack*, alterando os descritores do dispositivo para que este deixasse de ser reconhecido como um atuador de iluminação e passasse a operar como um dispositivo sensor multisensorial. A integração do *driver* I2C para o sensor externo HTU21D, bem como a correta exposição dos *clusters* de medição de temperatura (*msTemperatureMeasurement*) e umidade (*msRelativeHumidity*), comprovaram a flexibilidade do *hardware* e do *firmware* desenvolvidos.

Os resultados obtidos validaram a interoperabilidade da solução proposta. O dispositivo foi capaz de realizar o comissionamento em uma rede Zigbee gerenciada por um coordenador de terceiros, baseado no Zigbee2MQTT, e transmitir dados telemétricos de forma estável e coerente. No que se refere ao consumo energético, a análise realizada por meio da ferramenta *EnergyTrace* demonstrou que, embora o modo de suspensão (*standby*) do CC2652R7 atue de forma eficaz na redução da corrente basal, o consumo médio global ainda se mantém elevado para os requisitos mais restritivos de dispositivos finais alimentados por bateria em redes Zigbee.

Conclui-se, portanto, que o microcontrolador CC2652R7, aliado às ferramentas de desenvolvimento *SimpleLink* SDK e SysConfig, constitui uma solução robusta e versátil para aplicações de IoT. O trabalho atingiu seus objetivos

principais ao entregar um protótipo funcional e documentar de forma sistemática o processo de desenvolvimento sobre a pilha Zigbee da Texas Instruments. Entretanto, a análise do desempenho energético indica a necessidade de refinamentos futuros no *firmware*, especialmente no gerenciamento de periféricos e estratégias de baixo consumo, a fim de maximizar a vida útil da bateria e atender plenamente aos requisitos de operação prolongada em aplicações de baixa potência

5.1 Sugestões para trabalhos futuros

Como continuidade deste trabalho, uma possibilidade de evolução consiste na implementação de mecanismos dinâmicos de *reporting* Zigbee, com ajuste adaptativo dos intervalos de envio de dados em função da variação das grandezas medidas. Essa abordagem tem potencial para reduzir transmissões desnecessárias e, consequentemente, o consumo energético do sistema, preservando a relevância das informações disponibilizadas ao coordenador da rede.

Adicionalmente, recomenda-se a ampliação do sistema para a incorporação de múltiplos sensores externos, possibilitando a avaliação do comportamento do *firmware* em cenários de maior complexidade e maior carga de dados. A integração de recursos de segurança do protocolo Zigbee, como mecanismos de autenticação, também se apresenta como um tema pertinente para trabalhos futuros, especialmente em aplicações de monitoramento ambiental em ambientes críticos.

Por fim, a validação do sistema em cenários reais de longa duração, com medições contínuas de consumo energético e análise da vida útil da bateria ao longo do tempo, permitiria consolidar os resultados obtidos em ambiente controlado e aproximar o protótipo desenvolvido de uma solução com maior grau de maturidade para aplicações industriais ou comerciais.

REFERÊNCIAS

AMAZON WEB SERVICES. **O que é MQTT?** Disponível em: https://aws.amazon.com/pt/what-is/mqtt/. Acesso em: 04 dez. 2025.

ANASTASI, Giuseppe et al. *Energy conservation in wireless sensor networks: A survey*. *Ad **Hoc Networks***, v. 7, n. 3, p. 537–568, 2009. Disponível em: https://www.sciencedirect.com/science/article/abs/pii/S1570870508000954?via%3Dihub . Acesso em: 16 fev. 2026

BEVYWISE. **MQTT use cases: manufacturing solutions.** Disponível em: https://www.bevywise.com/mqtt-usecases/manufacturing-solutions.html. Acesso em: 04 dez. 2025.

CONNECTIVITY STANDARDS ALLIANCE (CSA). **Zigbee solution.** Disponível em: https://csa-iot.org/all-solutions/zigbee/. Acesso em: 04 dez. 2025.

CONNECTIVITY STANDARDS ALLIANCE. ***Zigbee Specification***. Davis, 2023. Disponível em: https://csa-iot.org. Acesso em: 16 fev. 2026.

EMQX. **MQTT topics and wildcards: a beginner's guide.** 2023. Disponível em: https://www.emqx.com/en/blog/advanced-features-of-mqtt-topics. Acesso em: 04 dez. 2025.

FARAHANI, Shahin. **ZigBee wireless networks and transceivers.** Burlington: Newnes/Elsevier, 2008. Disponível em: https://e2echina.ti.com/cfs-file/__key/communityserver-discussions-components-files/104/ZigBee-Wireless-Networks-and-Transceivers.pdf. Acesso em: 04 dez. 2025.

GISLASON, Drew. **Zigbee wireless networking.** Oxford: Newnes, 2008.

IBM. **MQTT V3.1 protocol specification.** Disponível em: https://public.dhe.ibm.com/software/dw/webservices/ws-mqtt/mqtt-v3r1.html. Acesso em: 04 dez. 2025.

IEEE. **IEEE Std 802.15.4-2020: IEEE Standard for Low-Rate Wireless Networks**. New York, 2020. Disponível em: https://standards.ieee.org/standard/802_15_4-2020.html . Acesso em: 16 fev. 2026.

INTERNETOFTHINGSWIKI. **Internet of Things – definition.** Disponível em: https://internetofthingswiki.com/internet-of-things-definition/. Acesso em: 04 dez. 2025.

KHOMP. **IoT gateways e sensores de telemetria.** Disponível em: https://www.khomp.com/. Acesso em: 04 dez. 2025.

KOENKK. **Zigbee2MQTT repository.** GitHub. Disponível em: https://github.com/Koenkk/zigbee2mqtt. Acesso em: 04 dez. 2025.

MQTT.ORG. **MQTT: the standard for IoT messaging.** Disponível em: https://mqtt.org/. Acesso em: 04 dez. 2025.

PROJETO DE REDES. **As redes com ZigBee.** Disponível em: https://site.projetoderedes.com.br/artigos/artigo_zigbee.php. Acesso em: 04 dez. 2025.

TE CONNECTIVITY. **HTU2X humidity sensors.** Disponível em: https://www.ttieurope.com/content/dam/tti-europe/manufacturers/te-connectivity/resources/c1de7b32-1394-42ed-9516-4647502d9aed.pdf. Acesso em: 05 dez. 2025.

TEXAS INSTRUMENTS. **CC2652R7 SimpleLink™ multiprotocol 2.4 GHz wireless MCU datasheet (Rev. B).** 2025. Disponível em: https://www.ti.com/lit/ds/symlink/cc2652r7.pdf. Acesso em: 16 set. 2025.

TEXAS INSTRUMENTS. **CC2652R7: product information and support.** 2025. Disponível em: https://www.ti.com/product/CC2652R7. Acesso em: 16 set. 2025.

TEXAS INSTRUMENTS. **Getting started with Code Composer Studio™ v20.** Vídeo. Disponível em: https://www.ti.com/video/6367048605112. Acesso em: 16 set. 2025.

TEXAS INSTRUMENTS. **SimpleLink Low Power F2 SDK.** GitHub repository. 2025. Disponível em: https://github.com/TexasInstruments/simplelink-lowpower-f2-sdk. Acesso em: 16 set. 2025.

TEXAS INSTRUMENTS. **SimpleLink™ low power SDK: software development kit (SDK).** Disponível em: https://www.ti.com/tool/SIMPLELINK-LOWPOWER-SDK. Acesso em: 16 set. 2025.

TEXAS INSTRUMENTS. **Welcome to CC2652R7 LaunchPad.** TI Developer Zone. Disponível em: https://dev.ti.com/lp-cc2652r7. Acesso em: 16 set. 2025.

TEXAS INSTRUMENTS. **Wireless connectivity technology selection guide.** 2020. Disponível em: https://www.ti.com/lit/wp/sway012/sway012.pdf. Acesso em: 04 dez. 2025.

WANG, Chonggang; JIANG, Tao; ZHANG, Qian. **Zigbee network protocols and applications.** Boca Raton: Auerbach Publications, 2016.

ZIGBEE ALLIANCE. **Zigbee Cluster Library specification.** 2021. Disponível em: https://zigbeealliance.org/wp-content/uploads/2019/12/07-5123-06-zigbee-cluster-library-specification.pdf. Acesso em: 04 dez. 2025.

ZIGBEE2MQTT. **Zigbee2MQTT documentation.** Disponível em: https://www.zigbee2mqtt.io/. Acesso em: 04 dez. 2025.