UNIVERSIDADE FEDERAL DE SÃO CARLOS
CENTRO DE CIÊNCIAS EXATAS E DE TECNOLOGIA - CCET
DEPARTAMENTO DE ENGENHARIA ELÉTRICA - DEE

Carlos Henrique Basali Rocha

Otimização de consumo energético em casas inteligentes com ESP32 e TagoIO

SÃO CARLOS-SP
2025

Carlos Henrique Basali Rocha

Otimização de consumo energético em casas inteligentes com ESP32 e TagoIO

Trabalho de conclusão de curso apresentado ao Departamento de Engenharia Elétrica da Universidade Federal de São Carlos, para obtenção do título de bacharel em Engenharia Elétrica.

Orientador: Prof.Dr. Celso Ap.de França

São Carlos-SP
2025

AGRADECIMENTOS

Agradeço, primeiramente, a minha mãe, dona Eliane. Muito obrigado por todo o incentivo, apoio e companheirismo em todos os momentos, inclusive nos mais difíceis. Costumo pensar que, provavelmente, eu não estaria escrevendo esta monografia se não fosse por você.

Ao meu pai, João, que me ensinou a ser quem eu sou. Sinto sua falta.

À minha namorada, amigos e família, que estiveram comigo ao longo de toda esta jornada. Que passaram por momentos felizes, tristes e, claro, de aprendizagem ao meu lado nos últimos 8 anos.

Obrigado, também, ao meu orientador, o professor dr. Celso Aparecido de França, pelos conselhos, sugestões e paciência desde o começo do trabalho.

Por fim, agradeço a todos os meus colegas de curso e de projetos de extensão que me ajudaram e fazem parte dessa trajetória.

RESUMO

Este trabalho aborda o gerenciamento de energia em residências inteligentes, destacando o problema do desperdício de energia elétrica e a necessidade de soluções eficientes para monitoramento e otimização do consumo. O objetivo principal é desenvolver e simular um sistema baseado no ESP32 e na plataforma TagoIO, utilizando o protocolo MQTT para transmissão de dados de sensores e atuadores em tempo reall. A justificativa do estudo está na crescente demanda por eficiência energética e na acessibilidade das tecnologias IoT e computação em nuvem, que permitem um controle mais eficaz dos equipamentos residenciais. A metodologia envolve a implementação de uma simulação e um protótipo, onde sensores coletam dados de temperatura, umidade e movimento e enviam as informações para um dashboard na TagoIO, facilitando a visualização e análise dos dados. Os testes realizados demonstraram que o sistema é capaz de fornecer informações sobre o ambiente e os equipamentos controlados, permitindo uma melhor compreensão dos padrões de uso e contribuindo para um consumo mais eficiente e sustentável.

Palavras-chave: Internet das Coisas; gerenciamento de energia; ESP32; TagoIO; MQTT; monitoramento energético; computação em nuvem.

ABSTRACT

This work addresses energy management in smart homes, highlighting the problem of electricity waste and the need for efficient solutions for monitoring and optimizing consumption. The main objective is to develop and simulate a system based on the ESP32 and the TagoIO platform, using the MQTT protocol for real-time data transmission from sensors and actuators. The study is justified by the growing demand for energy efficiency and the accessibility of IoT and cloud computing technologies, which enable more effective control of household appliances. The methodology involves the implementation of a simulation and a prototype, where sensors collect temperature, humidity, and motion data and send the information to a dashboard in TagoIO, facilitating data visualization and analysis. The tests carried out demonstrated that the system can provide relevant information about the environment and controlled devices, allowing for a better understanding of usage patterns and contributing to more efficient and sustainable energy consumption.

Keywords: Internet of Things; energy management; ESP32; TagoIO; MQTT; energy monitoring; cloud computing.

LISTA DE FIGURAS

LISTA DE SIGLAS

API – *Application Programming Interface* (Interface de Programação de Aplicações)

ESP – *Expressif Systems Platform* (Plataforma da Expressif Systems, fabricante do ESP32)

GPIO – *General-Purpose Input/Output* (Entrada/Saída de Uso Geral)

HTTP – *Hypertext Transfer Protocol* (Protocolo de Transferência de Hipertexto)

IoT – *Internet of Things* (Internet das Coisas)

LED – *Light-Emitting Diode* (Diodo Emissor de Luz)

MQTT – *Message Queuing Telemetry Transport* (Protocolo de Telemetria para IoT)

SHEMS – *Smart Home Energy Management System* (Sistema de Gerenciamento de Energia Residencial Inteligente)

SSR – *Solid State Relay* (Relé de Estado Sólido)

TCP/IP – *Transmission Control Protocol/Internet Protocol* (Protocolo de Controle de Transmissão/Protocolo de Internet)

WOKWi – Simulador Online para Microcontroladores

SUMÁRIO

1 INTRODUÇÃO

O aumento do consumo de energia elétrica nas últimas décadas tem impulsionado a busca por soluções mais eficientes para monitoramento e otimização do uso de eletricidade. Segundo ZHOU e et al. (2016), em residências, o desperdício energético frequentemente ocorre devido à falta de informações detalhadas sobre o consumo e à ausência de mecanismos automatizados para controle. Nesse contexto, as tecnologias de Internet das Coisas (IoT) e computação em nuvem surgem como ferramentas promissoras para aprimorar a gestão do consumo elétrico, permitindo a coleta, transmissão e análise de dados em tempo real.

Este trabalho propõe o desenvolvimento e a simulação de um sistema de gerenciamento de energia residencial inteligente, utilizando o ESP32 integrado à plataforma TagoIO. A comunicação entre o microcontrolador e a nuvem ocorre por meio do protocolo MQTT (Message Queuing Telemetry Transport), garantindo a transmissão eficiente dos dados captados pelos sensores. A partir dessas informações, os usuários podem monitorar os dados por meio de um dashboard interativo, possibilitando a identificação de padrões de uso e auxiliando na tomada de decisões para um consumo mais eficiente.

A justificativa para o desenvolvimento deste projeto está na necessidade de reduzir desperdícios energéticos e melhorar a eficiência do consumo residencial por meio de tecnologias acessíveis e de fácil implementação. O uso de IoT associado à computação em nuvem permite a criação de soluções flexíveis e escaláveis, tornando-se uma alternativa viável para a automação e controle energético em residências.

Para alcançar esse objetivo, foi desenvolvido uma simulação, onde sensores de temperatura, umidade e movimento são responsáveis pela captação dos dados, e um protótipo, onde um sensor de movimento verifica presenças no local, enviam os dados para a nuvem através do ESP32. Os dados são processados e exibidos na TagoIO, permitindo a análise gráfica das informações e possibilitando automações estratégicas para economia de energia.

Por fim, este estudo busca demonstrar a viabilidade do uso de sistemas inteligentes para monitoramento e otimização do consumo elétrico, contribuindo para um uso mais sustentável e eficiente da energia elétrica em residências.

1.1 CONTEXTUALIZAÇÃO

A crescente demanda global por energia, aliada à preocupação com o meio ambiente, impulsiona a busca por soluções mais eficientes. A integração de dispositivos eletrônicos em nossas vidas, tanto em ambientes urbanos quanto residenciais, torna a eficiência energética uma necessidade cada vez mais urgente. Conceitos extremamente importantes para o futuro das cidades surgiram a partir desta demanda, como o *Smart Grid*, que busca uma maior eficiência energética, integração com fontes renováveis de energia, melhoria da qualidade de fornecimento e redução de custos.

Se o *Smart Grid* representa a visão macro de uma rede elétrica mais inteligente e eficiente, as casas inteligentes representam a microvisão deste conceito, trazendo a eficiência energética para dentro dos lares. Em um geral, as casas inteligentes buscam a otimização do consumo energético residencial, reduzindo os custos e aumentando a qualidade de vida e conforto dos moradores. Neste contexto, surge, também, o conceito de sistemas de gerenciamento energético em casas inteligentes, que se torna um sistema essencial para o sucesso da gestão de demanda em *Smart Grids*, ZHOU e et al. (2016).

1.2 JUSTIFICATIVA

Segundo ZHOU e et al. (2016), o desenvolvimento de sistemas de gerenciamento de energia residenciais inteligentes vem se tornando cada vez mais uma prioridade global a fim de atender a demanda de consumo mais limpo e eficiente de energia elétrica e este conceito se torna um sistema essencial para o sucesso da gestão de demanda em *Smart Grids*. Buscar um consumo mais eficiente de energia elétrica é um caminho para reduzir os custos residenciais e o impacto ambiental causado pela geração de energia elétrica, enquanto que os recursos se esgotam.

Em paralelo, desenvolver uma solução de baixo custo e de fácil implementação torna este conceito mais acessível para todas as casas e nos deixa mais perto do futuro das *Smart Cities*.

Por fim, automação, eficiência energética e *IoT* são assuntos que interessam

o autor desde o ensino médio, período onde se formou em técnico de Mecatrônica e ajudou a desenvolver um protótipo de estufa automatizada como TCC.

1.3 OBJETIVOS

Este projeto tem como objetivo desenvolver e simular um sistema de automação para otimização do consumo energético em residências, utilizando o simulador Wokwi, o microcontrolador ESP32 e a plataforma TagoIO, que permite monitorar, controlar e reduzir o uso de energia elétrica de forma eficiente e prática. A implementação de uma simulação de um ambiente residencial será apresentada a fim de validar a solução proposta e a eficácia do sistema na automação do consumo energético e no gerenciamento de dispositivos elétricos. Em um segundo momento, será apresentado um protótipo simples de comunicação via MQTT envolvendo um sensor e o ESP32. Para isso, os objetivos específicos são reduzir desperdícios no consumo de energia em tempo real, automatizar o controle de dispositivos elétricos, desenvolver e configurar dashboards interativos na plataforma TagoIO, construir e testar um protótipo simplificado e identificar áreas de melhoria e otimização para aprimorar a eficiência do sistema e sua aplicabilidade em ambientes residenciais reais.

1.4 ESTRUTURA DO TRABALHO

O trabalho está dividido em cinco capítulos. Este capítulo inicial contextualiza e justifica o trabalho. No segundo capítulo é apresentada a fundamentação teórica sobre o assunto obtida através da revisão bibliográfica. O terceiro capítulo mostra o desenvolvimento e a implementação do trabalho. No capítulo quatro são relatados e comentados os resultados obtidos. Por fim, as conclusões finais são apresentadas no capítulo cinco.

2 FUNDAMENTAÇÃO TEÓRICA E REVISÃO DE LITERATURA

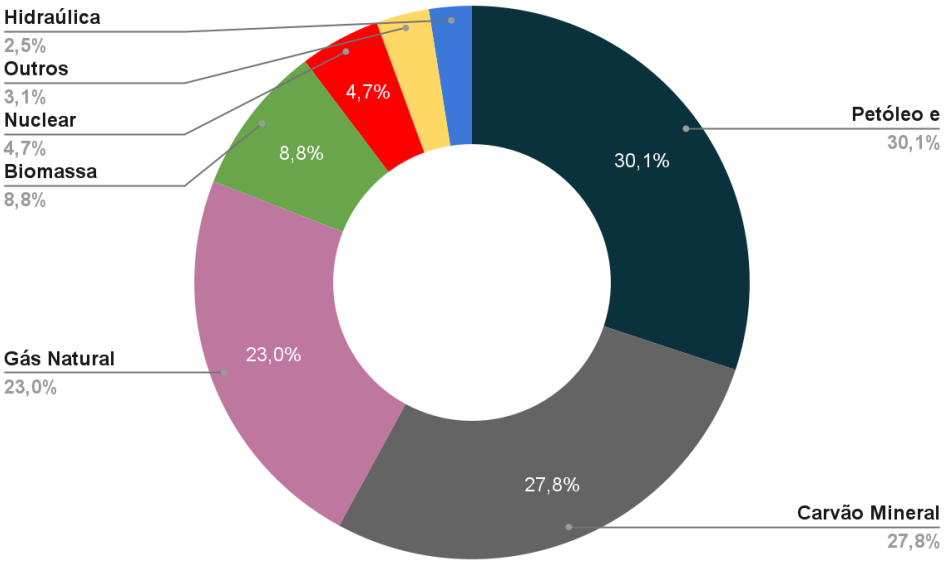
Este capítulo está focado na fundamentação teórica e revisão de literatura sobre temas importantes para o trabalho. Primeiro é abordado como é a atual matriz energética no Mundo e no Brasil e uma visão recente do consumo de energia elétrica no país. Em seguida, são apresentados os conceitos de *Smart Grids,* Internet das Coisas e Computação em Nuvem. No terceiro momento, foram realizadas pesquisas sobre o software e os hardwares que fazem parte da solução proposta e como podemos comunicar o físico e o digital utilizando protocolos de comunicação como o HTTP e o MQTT. Para finalizar, artigos acadêmicos que também trabalharam sobre automação residencial, medição de grandezas elétricas e redução do consumo e desperdício de energia elétrica foram estudados.

2.1 MATRIZ ENERGÉTICA NO BRASIL E NO MUNDO

A energia elétrica pode ser gerada de diversas maneiras atualmente. Ela pode ser dividida entre duas categorias: energia gerada através de fontes renováveis, como a biomassa e hidráulica, e energia gerada a partir de fontes não renováveis, como o carvão e o petróleo. Cada cidade, país e continente lidam com a geração de energia de forma diferente, onde cada fonte contribui com mais ou menos impacto no total da geração, conforme as características do local. O conjunto de fontes de geração de energia elétrica em uma determinada região é o que se chama de matriz energética.

A matriz energética mundial, como mostrado no Figura 1, é composta por várias fontes de energia, renováveis ou não, porém, segundo a Empresa de Pesquisa Energética (EPE), em 2022, 80,9% de toda energia gerada no mundo foi proveniente das fontes não renováveis de petróleo e derivados, carvão mineral e gás natural.

Figura 1: Matriz energética mundial em 2022

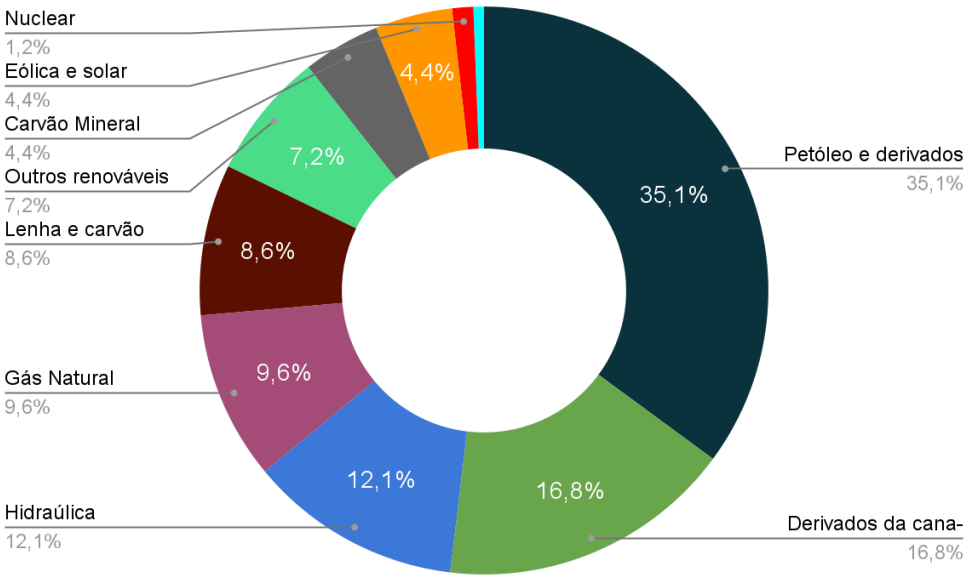


Fonte: adaptado de EPE (2024)

É possível ver na Figura 2 que o Brasil, no entanto, é um país que se aproveita mais das fontes renováveis. Devido às características hidrográficas e geológicas brasileiras, as fontes lenha e carvão vegetal, hidráulica, derivados de cana e eólica são mais abundantes e, segundo EPE (2024), são utilizadas em 41,9% da geração de energia no país. Incluindo outras fontes, quase metade da matriz energética brasileira é formada por fontes renováveis. Apesar disso, a principal fonte de energia ainda é o petróleo, com 35,1% de toda a geração.

Estas matrizes energéticas nos mostram que o mundo, mesmo com todos os esforços e avanços tecnológicos para geração de energia renovável, continua dependente de fontes não renováveis. E embora o Brasil esteja mais avançado na utilização de fontes renováveis em relação à média global, ainda é preciso desenvolver e aplicar soluções que visam substituir o uso de fontes não renováveis, o gerenciamento de consumo de energia elétrica e a otimização dos recursos energéticos.

Figura 2: Matriz energética brasileira em 2023



Fonte: adaptado de EPE (2024).

2.2 AUMENTO DO CONSUMO DE ENERGIA ELÉTRICA NO BRASIL

O Brasil tem uma das maiores populações do mundo e toda variação no consumo de energia elétrica do país é relevante. Segundo FÉLIX (2024), em dezembro de 2024, a projeção do aumento de consumo de energia elétrica do mesmo ano foi de 5,3% em relação ao ano de 2023, alcançando 79.899 MW médios. Para os próximos anos, a projeção é “para 2025, estima-se uma alta de 3,5% em comparação com 2024, atingindo 82.690 MW médios. Em 2029, a carga deverá atingir 94.157 MW médios.” (FÉLIX, 2024).

Segundo a Câmara de Comercialização de Energia Elétrica, CCEE (2024), o aumento do consumo de energia elétrica no primeiro trimestre de 2024 foi de 5% em comparação com o mesmo período do ano anterior. Ainda, este aumento foi causado principalmente pela alta atividade de setores como o de serviços, comércio e indústrias alimentícias e pelo aumento do calor em todas as regiões do país, impulsionando o uso de equipamentos de refrigeração, como ventilador e ar condicionado.

Embora 2024 tenha sido um ano com muito calor, estudiosos nos dizem que o futuro próximo será ainda mais quente e extremo. BERNARDES (2025) expõe que

2024 foi o ano mais quente no mundo desde o começo dos registros, em 1850. Bernardes também cita em seu texto a física e professora da USP Luciana Rizzo: “Importante pontuar que o clima possui uma inércia. A temperatura está aumentando, e vai continuar aumentando por um tempo, mesmo que as emissões de carbono sejam zeradas amanhã. É como tentar parar um caminhão em alta velocidade” (Luciana Rizzo).

Este aumento de temperatura, que deve continuar por alguns anos, pode aumentar ainda mais a utilização de equipamentos de refrigeração que, por sua vez, pode gerar um grande aumento no consumo de energia elétrica. Com isso, reduzir o desperdício de energia, como em casos onde o equipamento está ligado sem uma pessoa no local, e aumentar a eficiência destes equipamentos pode ajudar a minimizar o impacto tanto na rede elétrica e na geração de energia do país, quanto no bolso da população.

2.3 *SMART GRID*

A *Smart Grid* é uma abordagem moderna para a rede elétrica, que faz uso de tecnologias de automação, computação e comunicação para monitorar e controlar o sistema de forma mais eficiente. Contudo, não deve ser considerada uma tecnologia ou equipamento específico, trata-se de um conceito que promove a integração entre a geração, transmissão e distribuição de energia com redes digitais de comunicação e processamento de dados, criando um ecossistema conectado por dispositivos eletrônicos inteligentes.

Segundo FALCÃO (2010), essa modernização traz diversas funcionalidades, como:

- Auto-recuperação: Habilidade de identificar, analisar e corrigir falhas na rede de forma autônoma.
- Participação ativa dos consumidores: Integração dos hábitos e equipamentos dos consumidores nos processos de planejamento e operação da rede elétrica.
- Resistência a ataques: Capacidade de lidar com ameaças físicas ou cibernéticas de forma eficaz.
- Qualidade de energia aprimorada: Entrega de energia elétrica com padrões

que atendam às exigências tecnológicas modernas.

- Suporte a fontes diversificadas de energia: Compatibilidade com diferentes tipos e tamanhos de fontes de energia, incluindo renováveis, com fácil integração (plug-and-play).
- Sustentabilidade: Redução das perdas energéticas e uso de fontes mais amigáveis ao meio ambiente.
- Resposta à demanda: Ajuste remoto do consumo dos dispositivos para equilibrar a oferta e a demanda.
- Incentivo a mercados competitivos: Facilitação da microgeração e promoção de mercados de energia descentralizados.

A *Smart Grid* é um conceito que não pode ser implementado, ao todo, em um curto período de tempo. Sua transição saudável acontece de maneira gradual e, geralmente, seguindo etapas específicas. Primeiro, instala-se dispositivos inteligentes com possibilidade de integração IoT. Depois, cria-se uma infraestrutura de comunicação eficiente e realiza-se a integração e interoperabilidade dos sistemas e dispositivos. Por fim, são utilizadas ferramentas analíticas avançadas e se faz a otimização da operação da rede elétrica.

Contudo, a introdução do conceito de *Smart Grid* representa uma mudança de paradigma na forma de lidar com a geração e distribuição de energia elétrica. Quando bem implementada, representa uma evolução significativa no setor de energia, oferecendo uma rede mais resiliente, eficiente e sustentável, capaz de atender às demandas energéticas e ambientais do futuro.

2.4 SISTEMAS DE GERENCIAMENTO DE ENERGIA EM CASAS INTELIGENTE

O gerenciamento de energia residencial envolve a utilização de estratégias e tecnologias que permitem um controle mais eficiente do consumo elétrico em uma casa. O objetivo principal é minimizar desperdícios, reduzir custos e garantir um uso mais racional dos recursos energéticos disponíveis. Isso pode ser alcançado por meio da automação de dispositivos, monitoramento em tempo real e implementação de regras inteligentes para controle de cargas.

Entre os exemplos de gerenciamento de energia, destacam-se sistemas de monitoramento de consumo, que utilizam sensores para medir a corrente elétrica e enviar dados para plataformas na nuvem. Outra abordagem comum é o uso de dispositivos inteligentes, como lâmpadas e tomadas controláveis remotamente, que permitem desligamento automático quando não há necessidade de uso. Além disso, sistemas de agendamento de carga podem priorizar o funcionamento de determinados aparelhos em horários de menor tarifa, otimizando o custo da energia elétrica utilizada.

De acordo com LIU e et al. (2016), como qualquer outro sistema de gerenciamento de energia, um sistema de gerenciamento de energia em casas inteligentes, *smart home energy management system* (SHEMS), tem como objetivos finais conservar energia, reduzir custos e melhorar o conforto. Basicamente, o SHEMS oferece cinco serviços principais: monitoramento, registro, controle, gerenciamento e alarmes.

2.5 INTERNET DAS COISAS E COMPUTAÇÃO EM NUVEM

Segundo FALCÃO (2010), Internet das Coisas (IoT) e de Computação em Nuvem são conceitos que vêm sendo cada vez mais discutidos e utilizados tanto na indústria, quanto nas residências. Com o avanço tecnológico de equipamentos elétricos e eletrônicos e da computação, está mais fácil e simples conectar dispositivos via WI-FI e visualizar os dados e controlá-los via internet, sem restrição de tempo e espaço.

A IoT possibilita a conexão de dispositivos físicos à internet, permitindo a troca de dados entre sensores, atuadores e plataformas de gerenciamento. No caso de locais com muitos dispositivos IoT, é comum o uso de um microcontrolador ou uma placa de desenvolvimento, como o Arduíno e o ESP32, como central do sistema, coletando dados dos outros dispositivos e enviando-os para a nuvem.

A computação em nuvem entra nesse contexto como o meio pelo qual os dados são processados, armazenados e visualizados de forma remota. A TagoIO é um exemplo de plataforma de IoT baseada em nuvem que permite a conexão com a central do sistema local, além de possibilitar a criação de automação e controle dos dispositivos.

2.6 *SOFTWARES*

Uma parte essencial de toda aplicação IoT são os *softwares*. Eles estão presentes desde a simulação e configuração do microcontrolador até a visualização e análise dos dados na nuvem. Os principais dois *softwares* utilizados neste trabalho foram a plataforma de nuvem TagoIO e o simulador de eletrônica Wokwi.

2.6.1 Plataforma TAGOIO

No conceito IoT, depois de enviar dados dos dispositivos para a nuvem, é preciso um software para receber, armazenar e processar estes dados. Hoje, existem várias plataformas web disponíveis na internet que são específicas para esta aplicação ou podem ser utilizadas neste contexto. Dentre elas, pode-se encontrar plataformas gratuitas ou plataformas que exijam assinatura paga. Empresas multinacionais como a Google, Amazon e a Microsoft possuem sua própria plataforma e oferecem o serviço principalmente para grandes aplicações. Contudo, algumas plataformas possuem acesso gratuito ao serviço com versões limitadas e de aprendizagem, como a TagoIO e a ThinkSpeak.

Para pequenas aplicações, a TagoIO se apresenta como uma alternativa interessante. A plataforma possui um plano gratuito com até 5 dispositivos conectados, 5 *dashboards* e 30 dias de armazenamento de dados. Além desta versão, a plataforma dispõe dos planos Starter e Scale, por assinatura. Em um *dashboard*, é possível visualizar os dados em formatos de gráficos, tabelas e grandes números e customizar a disposição e design, como no exemplo da Figura 3.

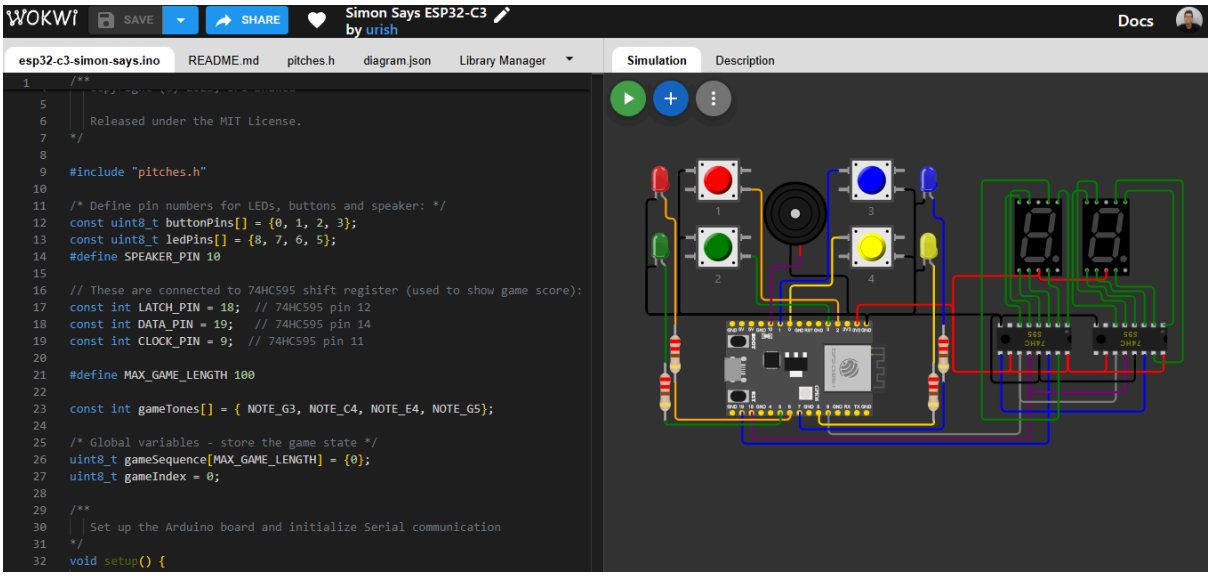
Figura 3: Exemplo de *dashboard* na plataforma TagoIO.

Fonte: Adaptado de TAGOIO (2025)

2.6.2 Simulador Wokwi

O Wokwi é uma plataforma *web* de simulação eletrônica. Nele, é possível simular circuitos com placas de desenvolvimento como o Arduino e microcontroladores como o ESP32, além de componentes como resistores, sensores, botões, *leds* etc. Para uso pessoal e de aprendizagem, o simulador é gratuito. A Figura 4 mostra um projeto disponível para edição e uso dentro da plataforma, criado pelo usuário “urish".

Figura 4: Simon Says ESP32-C3 by urish



Fonte: WOKWI

Segundo WOKWI (2024), o simulador é o mais avançado simulador de ESP32 do mundo e possui algumas características interessantes para a simulação de aplicações *IoT,* como:

- Simulação de Wi-Fi - Conecte seu projeto simulado à internet. Você pode usar MQTT, HTTP, NTP e muitos outros protocolos de rede.
- Analisador Lógico Virtual - Capture sinais digitais em sua simulação (por exemplo, UART, I2C, SPI) e analise-os em seu computador.
- Depuração avançada com GDB - Poderoso depurador Arduino e Raspberry Pi Pico para usuários avançados.
- Simulação de cartão SD - Armazene e recupere arquivos e diretórios do seu código. Os membros do Club também podem fazer upload de arquivos binários (como imagens)
- API dos chips - Crie seus próprios chips e peças personalizados e compartilhe-os com a comunidade.

- Integração no Visual Studio Code - Simule seus projetos incorporados diretamente no VS Code.

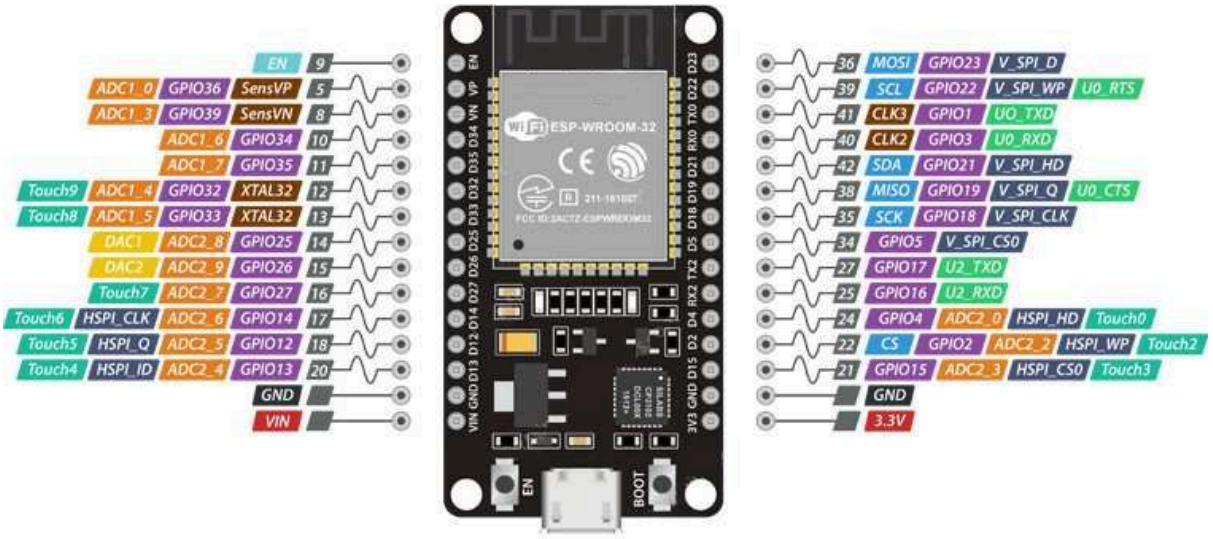
2.7 HARDWARE

Juntamente com o *software* de computação em nuvem, dispositivos físicos compõem um sistema IoT. Microcontroladores, sensores, atuadores e outros equipamentos eletrônicos podem ser usados de infinitas formas dentro de uma indústria ou residência para a aplicação IoT.

2.7.1 Microcontrolador ESP32

Em uma automação residencial, o microcontrolador atua como o centro de controle do sistema, coletando dados dos sensores de corrente, compilando as automações e comunicando-se com a plataforma na nuvem via Wi-Fi. O ESP32 tem capacidade de processamento e conectividade sem fio, sendo ideal para aplicações IoT de baixo consumo energético e alto desempenho. A Figura 5 apresenta os vários pinos que o microcontrolador possui.

Figura 5: Pinagem do ESP32



Fonte: Adaptado de ROBOCORE (2025)

Este microcontrolador possui as seguintes especificações:

- Processador: Xtensa® Dual-Core 32-bit LX6

- Memória Flash Programável: 4 MB
- Memória RAM: 520 KBytes
- Memória ROM: 448 KBytes
- Clock Máximo: 240 MHz
- Pinos Digitais GPIO: 25 (todos com PWM)
- Resolução do PWM: até 16 bits (ajustável via código)
- Wireless 802.11 b/g/n: 2.4 GHz (antena integrada)
- Modos de Operação: Access Point / Estação / Access Point+Estação
- Bluetooth Low Energy: padrão 4.2 integrado
- Tensão de Alimentação Externa: 4,5 V a 9 V (o módulo possui regulador integrado para 3,3 V)
- Dimensões: 54 x 29 x 13 mm (c x l x a - com barra de pinos)

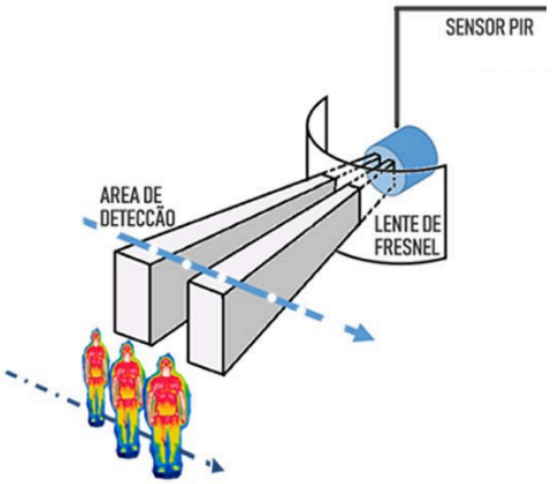
2.7.2 Sensor de movimento (PIR) HC-SR501

O sensor de movimento é usado para medir a movimentação de algum objeto no ambiente e é comumente utilizado para medir se um local está vazio ou não. Como explica SIMÕES (2018), o HC-SR501, visto na Figura 6, possui um sensor piroelétrico de alta sensibilidade que consegue detectar a radiação infravermelha emitida pelos corpos. Quanto mais quente está um corpo, maior será a radiação infravermelha emitida. Internamente, esse sensor piroelétrico é dividido em duas metades, formando dois elementos de medição, de modo que uma variação da radiação infravermelha será detectada por esses elementos piroelétricos, como ilustrado na Figura 7.

Figura 6: Sensor HC-SR501

Fonte: Autoria própria

Figura 7: Detecção do sensor HC-SR501.



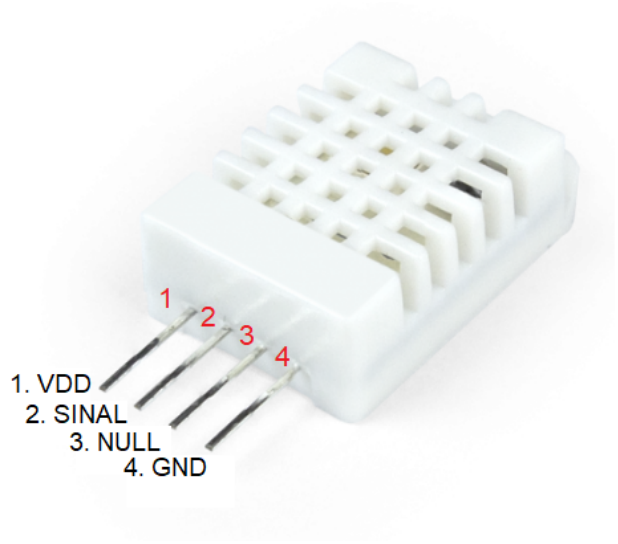
Fonte: Adaptado de (SIMÕES, 2018)

O sensor é alimentado com uma tensão de 5 a 12 volts em corrente contínua, consome 65 mA, opera entre as temperaturas -15º e 70º e tem um alcance de detecção de até 6 metros.

2.7.3 Sensor De Temperatura e Umidade

Um sensor de temperatura e umidade é um dispositivo que combina dois tipos de sensores: um para a temperatura, que pode ser um termistor ou termopar, e outro para a umidade, usando tecnologias como sensores capacitivos ou resistivos. O sensor de temperatura funciona alterando sua resistência elétrica de acordo com a variação de temperatura: quando a temperatura muda, a resistência do material também muda. O sensor de umidade, por sua vez, detecta as alterações nas propriedades elétricas de um material que absorve ou libera água, como a capacitância ou resistência, conforme a umidade do ambiente varia. Esses sensores transformam as medições em sinais elétricos que podem ser lidos e processados por sistemas eletrônicos, permitindo monitorar as condições ambientais. Eles são comumente usados em áreas como automação, controle de clima, agricultura e dispositivos conectados à Internet das Coisas (IoT).

Figura 8: Sensor de temperatura e umidade DHTT22



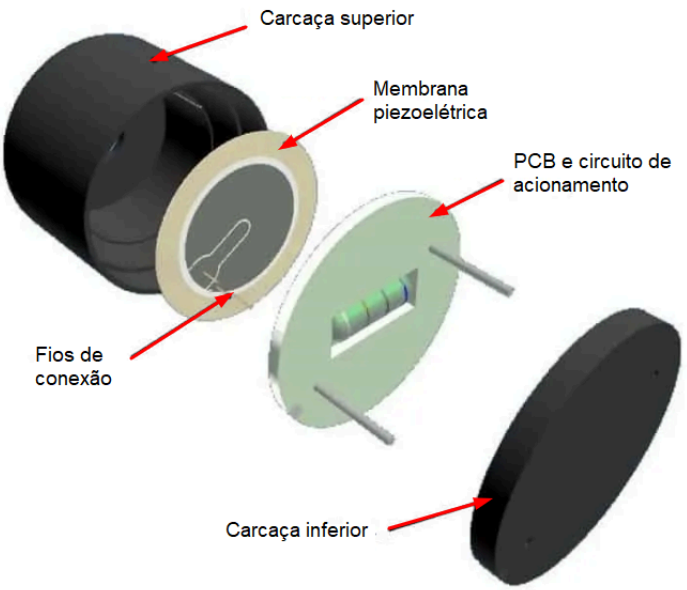
Fonte: Adaptado de ROBOCORE

O DHT22, apresentado na Figura 8, mede as duas grandezas e possui comunicação digital de um fio. O sensor é calibrado e não requer componentes extras para funcionar, sendo possível obter os dados sem complicações. Fora isso, tem especificações que se adequam a aplicações *IoT* residenciais, como *range* de temperatura de -40ºC a 80ºC, precisão de umidade: +-2% e precisão de temperatura: +-0.5ºC (ROBOCORE).

2.7.4 Buzzer piezoelétrico

O Buzzer piezoelétrico, Figura 9, é um dispositivo emissor de som que pode servir como um alerta sonoro em diversas aplicações. É um componente bastante simples e requer poucos e comuns materiais para ser produzido, o que o torna barato e pequeno. No buzzer piezoelétrico, um cristal sofre deformação mecânica quando submetido a uma tensão elétrica. Quando um sinal elétrico alternado é aplicado ao buzzer, o cristal vibra rapidamente, transmitindo essa vibração para uma membrana metálica acoplada. Essa membrana movimenta o ar ao seu redor, gerando ondas sonoras audíveis. A frequência do sinal elétrico determina o tom do som produzido, permitindo que o buzzer emita diferentes frequências sonoras dependendo da aplicação.

Figura 9: Estrutura de um buzzer piezoelétrico



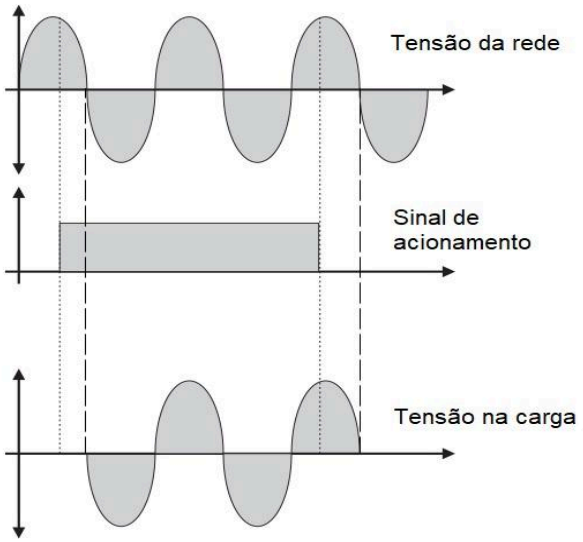
Fonte: Adaptado de AUTOCORE (2021)

2.7.5 Relés de Estado Sólido (SSR)

Esses relés são usados para controlar o estado de dispositivos elétricos, permitindo que sejam ligados ou desligados automaticamente com base nas regras de automação configuradas. O relé de estado sólido (SSR) é acionado por um emissor de luz, em geral um LED infravermelho, enquanto os contatos são trocados por fototransistores, fotodiodos ou outros dispositivos sensíveis à luz. Ao ser energizado, o LED emite um sinal luminoso que é recebido pelo receptor, que atua para comutar um dispositivo de maior potência, como TRIACs ou MOSFET de potência (RH MATERIAIS ELÉTRICOS, 2023)

SSR opera ativando ou desativando a alimentação de uma carga ao receber um sinal de controle em sua entrada. No entanto, a condução só se inicia quando a tensão da rede atinge o valor zero, reduzindo ruídos elétricos e protegendo o circuito contra picos de corrente. Da mesma forma, o desligamento ocorre na próxima passagem por zero da tensão. Esse processo garante um chaveamento mais suave e eficiente, com um atraso máximo de 8,3 milissegundos. Apesar dessas vantagens, o SSR é compatível apenas com tensões alternadas, não sendo possível utilizá-lo para chaveamento de corrente contínua. Na Figura 10 é possível entender melhor a operação do SSR.

Figura 10: Acionamento do SSR



Fonte: Adaptado de (RH MATERIAIS ELÉTRICOS, 2023)

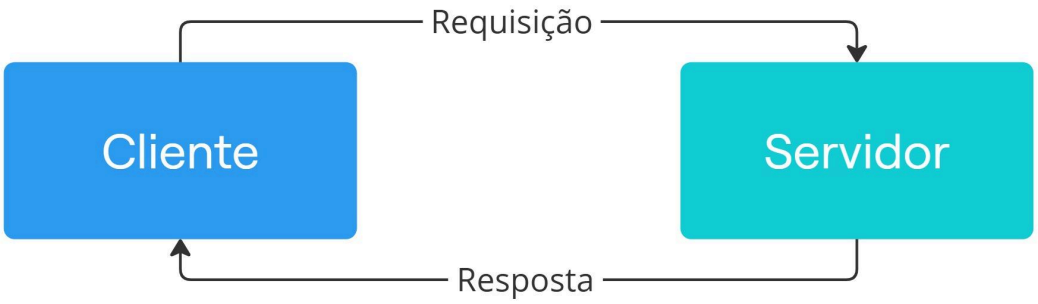
2.8 INTEGRAÇÃO ESP32 e TAGOIO

Para a integração do ESP32 com a plataforma TagoIO, são utilizados conceitos de Internet das Coisas (IoT), computação em nuvem e protocolos de comunicação, permitindo o monitoramento e controle remoto de dispositivos. Para que tenha dados a serem visualizados e processados em uma plataforma em nuvem, é preciso integrar a parte física e a digital do sistema. Para isso, pode-se utilizar os protocolos de comunicação que sejam compatíveis com os dispositivos. No caso do ESP32 e da TagoIO, dois protocolos são comumente usados: HTTP e MQTT.

2.8.1 Protocolo de comunicação HTTP

Hypertext Transfer Protocol (HTTP) é um protocolo de comunicação muito utilizado nos dias de hoje em diversas aplicações mas, principalmente, nos computadores e na Internet. Funciona da forma requisição-resposta, onde um cliente, como navegadores e dispositivos IoT, envia uma requisição a um servidor na internet e recebe uma resposta direta sobre o que foi pedido, como ilustrado na Figura 11.

Figura 11 : Representação da comunicação entre cliente e servidor



Fonte: Autoria própria

Segundo BORDIGNON (2023), o protocolo HTTP apresenta características que garantem sua eficiência e ampla utilização. Ele é simples, pois suas mensagens são legíveis e fáceis de interpretar sem a necessidade de equipamentos especiais. Sua extensibilidade permite a adição de novos cabeçalhos de forma intuitiva, facilitando sua evolução. Além disso, é um protocolo *stateless*, ou seja, não mantém estado entre solicitações, tornando cada requisição independente. Por fim, sua confiabilidade é assegurada pela integração com o protocolo TCP/IP, garantindo a integridade dos dados transmitidos e minimizando perdas de informação.

Os principais métodos utilizados em uma comunicação via HTTP são:

- *GET*: Solicita dados do servidor. Exemplo: acessar uma página web.
- *POST*: Envia dados para o servidor. Exemplo: preencher um formulário.
- *PUT*: Atualiza recursos no servidor. Exemplo: modificar dados em um banco de dados.
- *DELETE*: Remove um recurso do servidor. Exemplo: excluir uma conta de usuário.

2.8.2 Protocolo de comunicação MQTT

O MQTT (*Message Queuing Telemetry Transport*) é um protocolo de comunicação leve e eficiente, projetado para a troca de mensagens entre dispositivos conectados à Internet das Coisas *(IoT*). Ele segue o modelo publicador-assinante *(publish-subscribe*) e utiliza um servidor central chamado "*broker*" para gerenciar as mensagens. Um dispositivo publicador (*publisher*) envia uma mensagem para um tópico específico no broker, que atua como um intermediário central. Dispositivos assinantes (*subscribers*) podem se inscrever

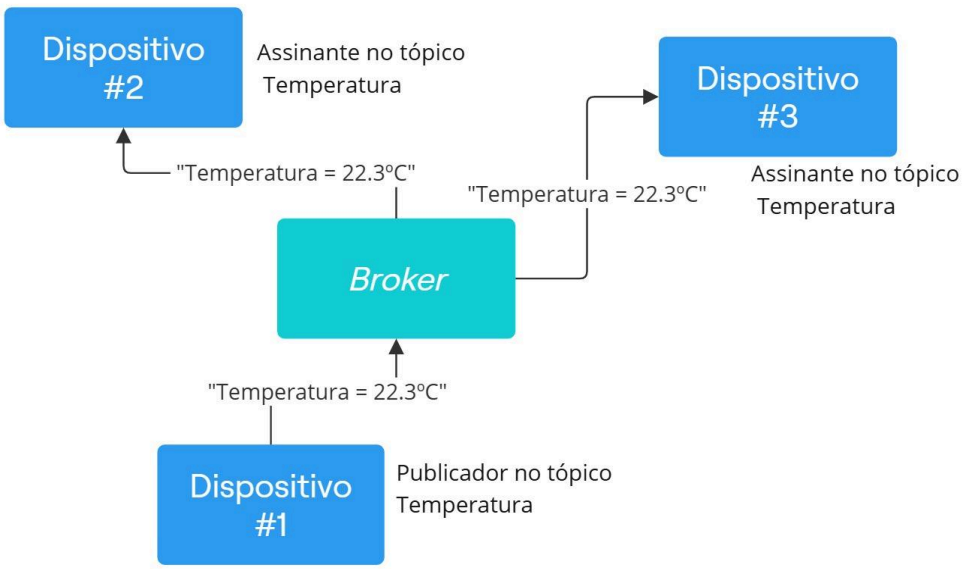
nesses tópicos para receber as mensagens correspondentes. Quando o *broker* recebe uma mensagem de um publicador, ele automaticamente distribui essa informação para todos os assinantes do tópico, garantindo uma comunicação eficiente e descentralizada entre dispositivos *IoT*. A Figura 12 mostra como é o fluxo desta comunicação.

O MQTT oferece três níveis de serviço de QoS *(Quality of Service*). No nível 0, não há garantia de entrega; no nível 1, as mensagens são entregues pelo menos uma vez; e no nível 2, as mensagens são entregues exatamente uma vez, garantindo a entrega sem duplicatas, embora com maior sobrecarga. Esses níveis permitem adaptar a confiabilidade da entrega de mensagens conforme necessário (BORDIGNON, 2023).

Principais características do protocolo MQTT:

- Leveza: Consome pouca largura de banda, ideal para dispositivos com recursos limitados.
- Baixo consumo de energia: Perfeito para sensores e dispositivos IoT que operam com baterias.
- Qualidade de Serviço (QoS): Permite diferentes níveis de entrega das mensagens, garantindo confiabilidade.
- Comunicação assíncrona: Mensagens são enviadas e recebidas independentemente do tempo de resposta do outro dispositivo.

Figura 12: Representação da comunicação pelo MQTT



Fonte: Adaptado de BORDIGNON (2023)

2.9 ARTIGOS DE SISTEMAS DE GERENCIAMENTO DE ENERGIA

Para embasar o projeto, foram analisados artigos relacionados a sistemas de gerenciamento de energia em residências inteligentes, encontrados nas bases Elsevier e Google Scholar.

O artigo *"A Real-Time Energy Consumption and Monitoring for Domestic Houses by using Internet of Things Approach",* AZMAN e RAHMAN (2022), apresenta uma solução prática e acessível para monitoramento em tempo real do consumo energético, utilizando sensores de corrente conectados a um microcontrolador Arduino Wemos D1 R1 e à plataforma ThingSpeak. O principal objetivo é permitir que os usuários controlem o uso de energia e evitam desperdícios. Apesar de suas vantagens, como simplicidade e baixo custo, a solução depende de uma conexão estável à internet e apresenta limitações relacionadas à precisão das medições devido a fatores externos.

O segundo artigo, *"Development of IoT-Based Smart Home Application with Energy Management"*, PRATHYUSHA e BISWAJIT (2023), propõe um sistema de automação residencial inteligente, que integra múltiplos dispositivos *IoT* para controle remoto de luzes, ventiladores e exaustores, com simulação no Tinkercad. A solução visa aumentar o conforto e reduzir o consumo de energia, permitindo testes antes da implementação real. Contudo, a complexidade da integração de vários dispositivos pode aumentar custos e dificultar a manutenção, além de o próprio sistema consumir energia para operar, o que pode impactar a eficiência esperada.

Já o terceiro artigo, *"Smart Home Energy Management Systems: Concept, Configurations, and Applications***"**, ZHOU e et al. (2016), fornece uma visão abrangente sobre os conceitos básicos, configurações e aplicações de sistemas de gerenciamento de energia, destacando a integração com tecnologias renováveis e redes inteligentes *smart grids*. Ressalta-se a importância de estratégias de agendamento de consumo e o desenvolvimento de tecnologias mais acessíveis e eficientes. No entanto, o artigo apresenta limitações práticas por não incluir estudos de caso detalhados.

2.10 ESTUDO COMPARATIVO

Os artigos do estudo bibliográfico foram comparados e ajudaram na definição da solução do projeto.

AZMAN e RAHMAN (2022) e PRATHYUSHA e BISWAJIT (2023) trazem em seus respectivos artigos algumas soluções práticas que podem ser aplicadas em uma residência inteligente e integrada à nuvem. Os primeiros apresentam um protótipo de dispositivo para medição em tempo real de corrente utilizando o sensor não-invasivo SCT013, o Arduino Wemos D1 R1 com um módulo Wi-Fi e integrando com o software de armazenamento de dados em nuvem ThingSpeak. Os segundos desenvolveram um modelo de casa inteligente que contempla múltiplos dispositivos e cômodos, com o foco em reduzir desperdícios de energia elétrica. Os dois estudos se complementam, já que um mostra uma solução para medição de corrente e consumo de energia elétrica e o outro é um modelo de integração de dispositivos e estratégias de otimização no consumo de energia e conforto dos moradores, e darão a base inicial para o desenvolvimento da solução deste projeto.

Por outro lado, ZHOU e et al. (2016) introduziram alguns conceitos e configurações de sistemas de gerenciamento de energia em casas inteligentes e de estratégias de agendamento de consumo de energia elétrica. O artigo levanta alguns pontos interessantes que serão considerados ao longo deste projeto, como o conceito e arquitetura de um sistema de gerenciamento de energia residencial, aplicação de estratégias de agendamento para estes sistemas e estratégias de respostas a demandas baseadas em preço e em incentivo.

Cada um destes projetos e estudos, embora sobre o mesmo tema, trouxeram conceitos e estratégias diferentes de como medir, controlar e otimizar o consumo de energia elétrica em uma residência. O conteúdo destes artigos deram uma base inicial de conhecimento para a estruturação da solução proposta e foram referência ao longo do desenvolvimento dela.

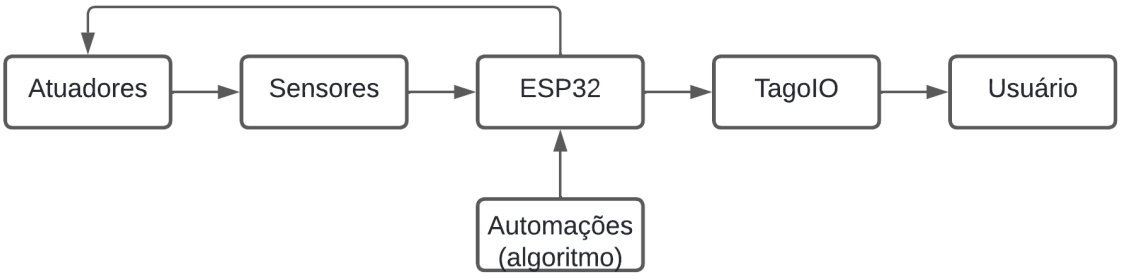
3 DESENVOLVIMENTO DO TRABALHO

Neste capítulo será apresentado como o trabalho foi desenvolvido. Os principais pontos são a instalação e configuração do MQTT Broker, configuração da plataforma TagoIO, criação do dashboard e o desenvolvimento da simulação e da implementação física.

3.1 DESCRIÇÃO DA SOLUÇÃO

A solução proposta visa otimizar o consumo de energia elétrica em uma residência por meio de medições de consumo, automações, controle remoto e estratégias de economia de consumo. Utilizando o microcontrolador ESP32 simulado como central do sistema e a plataforma de dados em nuvem TagoIO como ferramenta de análise e controle, o projeto simula o monitoramento em tempo real dos sensores de umidade, temperatura e presença. Automações para desligamento de uma lâmpada, ar-condicionado e umidificador em cômodos vazios serão implementados, a fim de reduzir o desperdício de energia. A interface com o usuário foi composta por *dashboards* interativos no TagoIO, permitindo controle e visualização dos dados transmitidos. A Figura 13 apresenta o diagrama geral da solução.

Figura 13: Diagrama geral da solução



.Fonte: Autoria própria

O projeto também incluirá o desenvolvimento de um protótipo simplificado de transmissão de dados via MQTT.

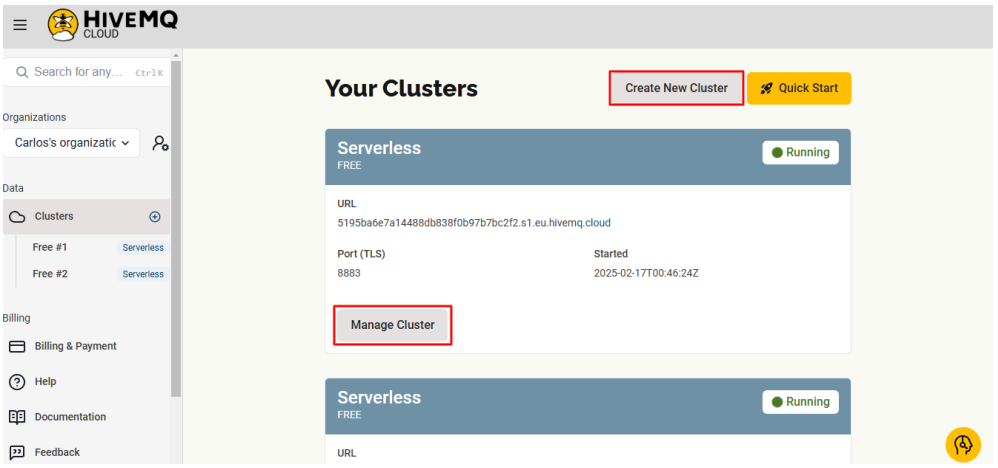
3.2 CONEXÃO COM A PLATAFORMA TAGOIO

O envio de dados para a plataforma TagoIO foi realizado através do protocolo de comunicação MQTT. Foi escolhido este protocolo por suas características que favorecem aplicações IoT: baixo consumo de energia, leveza, qualidade de serviço e comunicação assíncrona. Assim, foi preciso configurar o publicador, o broker e o assinante.

3.2.1 MQTT Broker HiveMQ

A TagoIO possui um MQTT Broker próprio que pode ser usado pelos usuários para a comunicação dos dispositivos, porém, desde julho de 2024, o broker se tornou exclusivo para usuários pagantes. Com isso, foi necessário o uso de um broker externo. Foi escolhido o *broker* web da plataforma *HiveMQ Cloud*, precisando apenas de criar uma conta gratuita e, posteriormente, um novo *Cluster.* Pôde-se gerenciar o *cluster* clicando no botão “Manage Cluster”, como visto na Figura 14.

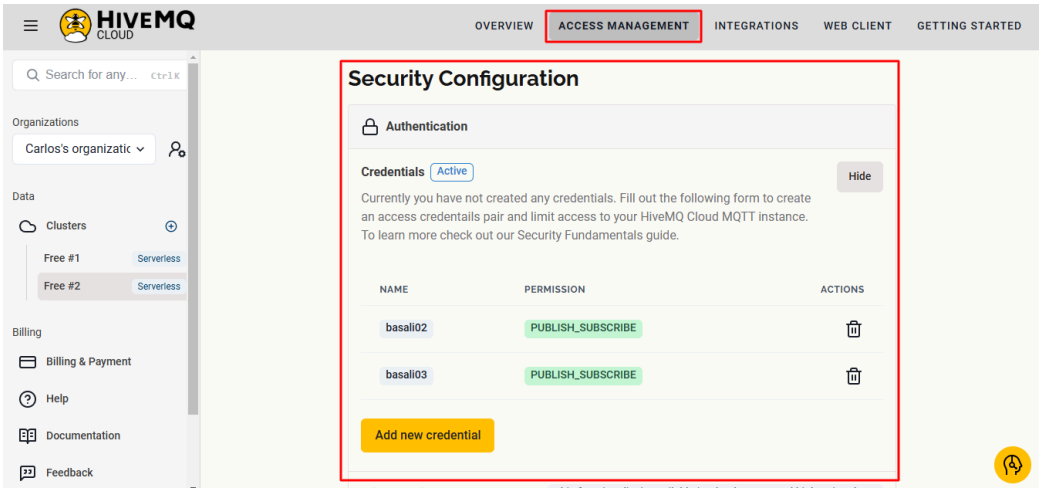
Figura 14: *Cluster* na plataforma HiveMQ



Fonte: Autoria própria.

No gerenciamento, pode-se encontrar as informações de *URL*, endereço do broker, e *Port,* necessárias para a comunicação. Além disso, foi criada na aba *Access Management* as credenciais de acesso para o broker, também necessárias para a comunicação e segurança dos dados transmitidos. A Figura 15 mostra onde encontrar as informações.

Figura 15: Configuração de segurança do MQTT *Broker*

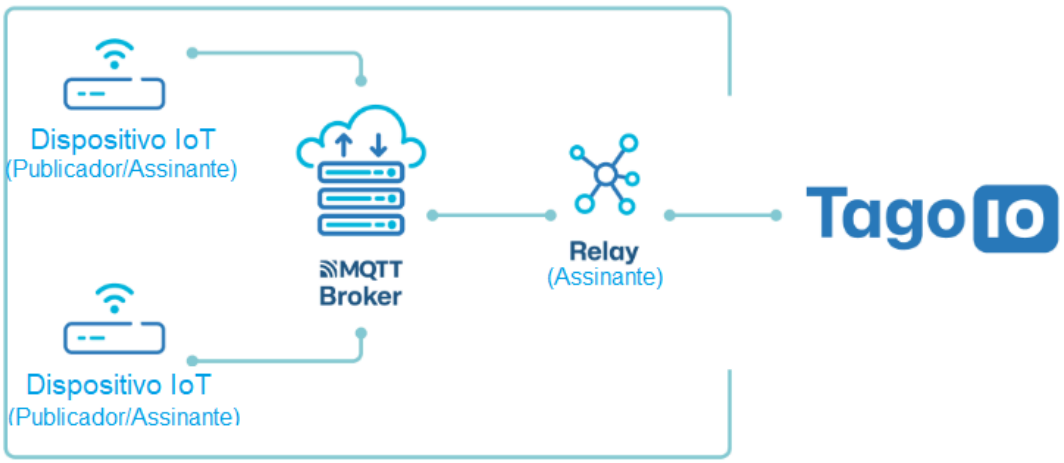


Fonte: Autoria própria.

3.2.2 Configuração TagoIO

No caso da utilização de um *broker* externo na comunicação via MQTT, a configuração da plataforma TagoIO exige que os dados passem pelo software MQTT *Relay* antes de chegar na nuvem. Este software age entre o MQTT *Broker* e a plataforma para facilitar a integração e o fluxo de dados. A Figura 16 apresenta o fluxo de dados neste caso. A TagoIO publicou um tutorial de como estabelecer a comunicação entre o MQTT *Broker* e a plataforma, passando pelo *Relay*. O tutorial pode ser visto em TAGOIO (2024).

Figura 16: Fluxo de transmissão de mensagens via MQTT entre dispositivos e TagoIO.

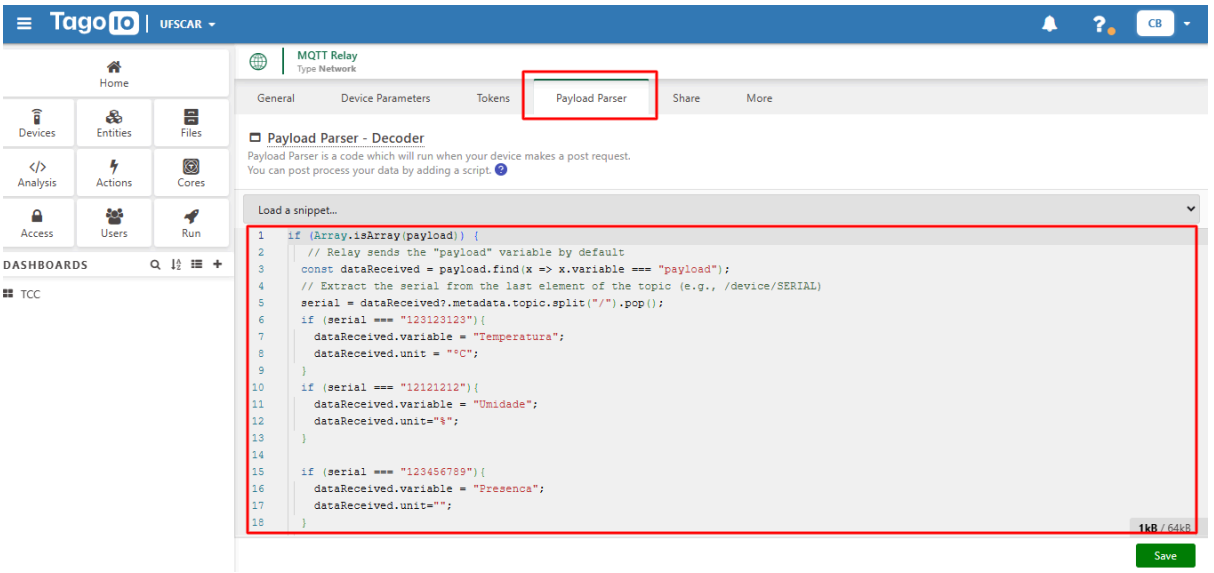


Fonte: Adaptado de TAGOIO (2024)

Depois de baixar o MQTT Relay pelo link disponibilizado no tutorial e instalá-lo no computador, foi feita a configuração na plataforma. O próximo passo é a criação do MQTT *Relay Network,* que recebe a informação enviada pelo MQTT *Relay* :

- Depois de *logar* na plataforma da TagoIO, clica-se no ícone do usuário no canto superior direito e depois em "*Integrations*";
- Passe para a aba *Network* e clique em "*Add Network*". Adicione um nome e crie o seu *Network*;
- Habilite o *Serial Number* no canto superior direito;
- Na aba "Tokens", clique-se para gerar um token;
- Abra o Network criado e, na aba "*Payload Parser*", adicione o código que fará a captura dos dados enviados e salve. A Figura 17 mostra onde deve ser adicionado o código.

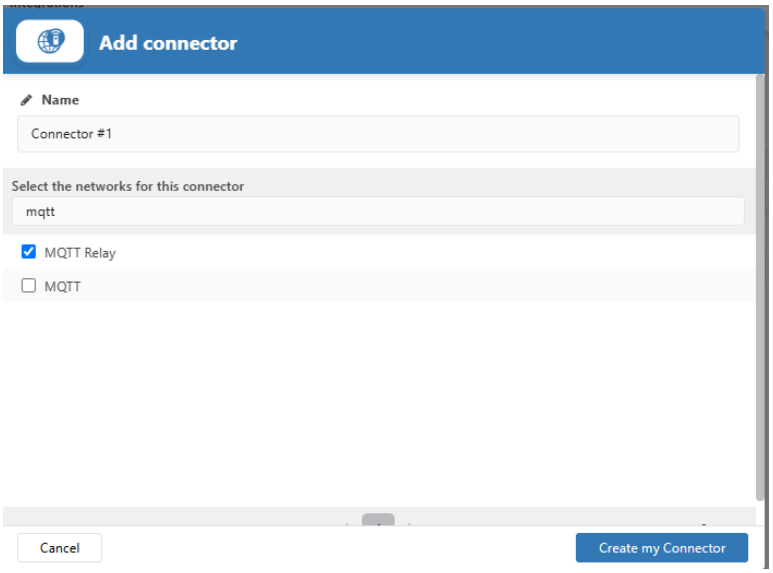
Figura 17: *Payload Parser*



Fonte: Autoria própria.

Novamente na página *"Integrations",* é preciso criar um conector que será usado com o *Relay.* Na aba "Connector", clique no botão "Add Connector" no canto superior direito, digite um nome, selecione o *network* criado anteriormente, crie um novo conector, e salve. Na Figura 18 é mostrado a criação do conector de nome "*Connector* #1", criado do desenvolvimento.

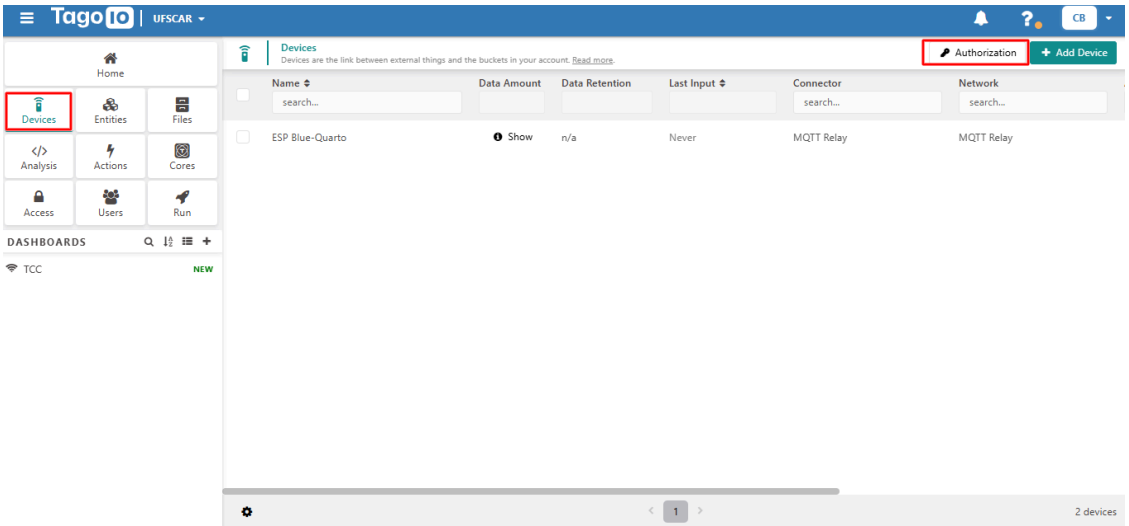
Figura 18: Acionando um novo conector



Fonte: Autoria própria.

Em seguida, cria-se a autorização para conexão com os dispositivos da TagoIO. Na tela inicial, clica-se em “Devices”, à esquerda da tela, e depois em “*Authorization*”, no canto superior direito, seguindo a Figura 19. Então, clica-se em “*Generate*” para gerar a autorização e definir um nome para ela.

Figura 19: Gerando uma autorização.

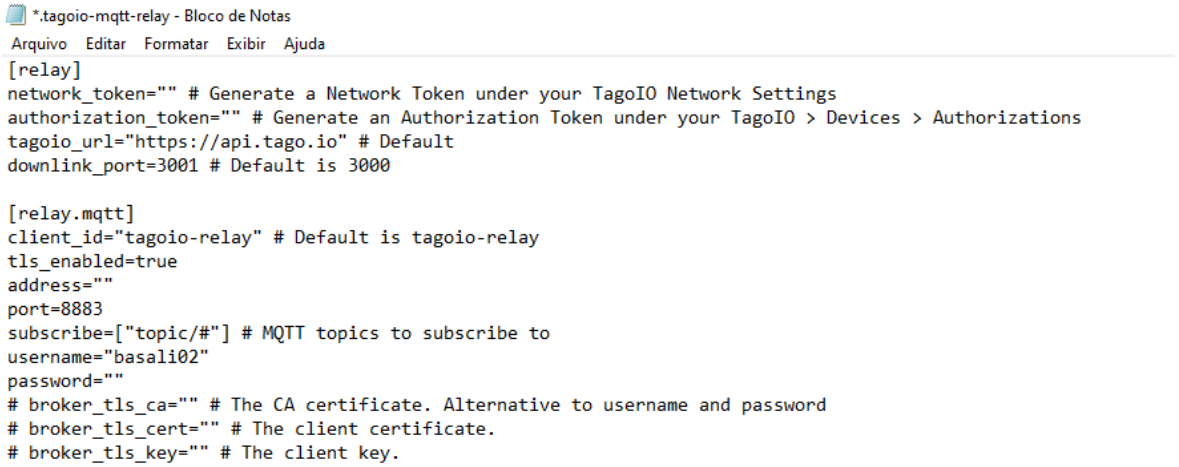


Fonte: Autoria própria.

Finalizando as configurações na plataforma, é preciso adicionar as

informações de segurança e endereço no software do MQTT Relay. Procura-se no computador o arquivo de texto “.tagoio-mqtt-relay” que foi baixado ao longo da instalação do software e altera-se o texto com os corretos tokens do network e autorização que acabaram de serem criados, e com o endereço *URL,* o *Port e* com as credenciais do MQTT *Broker*. Segue a Figura 20 como exemplo do arquivo.

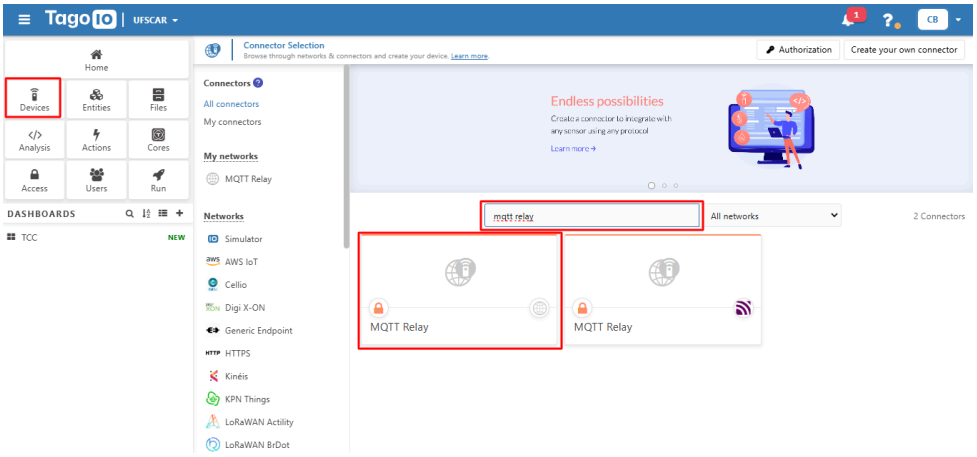
Figura 20: tagoio-mqtt-relay



Fonte: Autoria própria.

Por último, cria-se um dispositivo específico para receber os dados do *Relay* e ser possível apresentá-los no *dashboard*. No menu, clica-se em “*Devices*” e depois em “Add Device" no canto superior direito. Pesquisa-se pelo dispositivo “MQTT *Relay*”, assim como na Figura 21, e o seleciona.

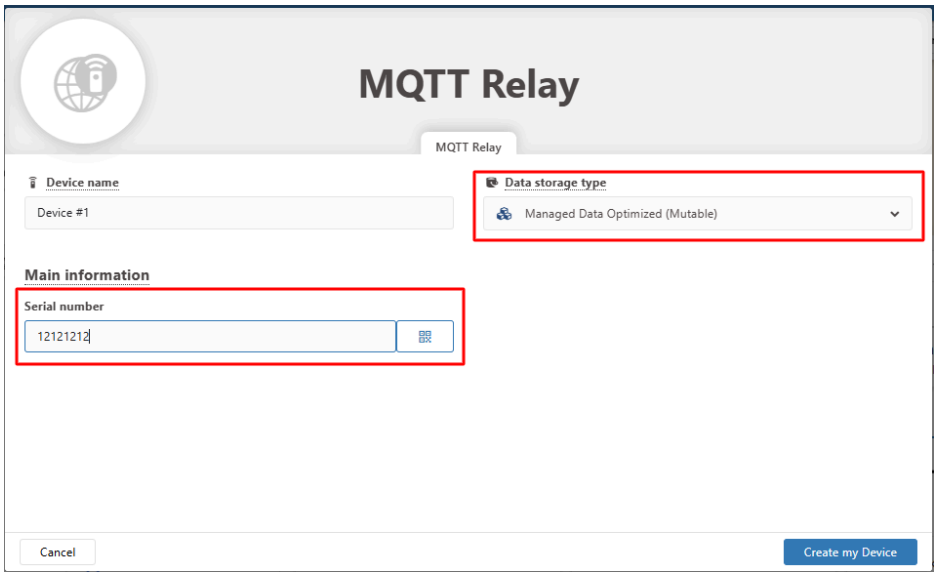
Figura 21: Dispositivo MQTT *Relay*



Fonte: Autoria própria.

Então, nomeia-se o dispositivo com o nome desejado e, no campo '*Data storage type'*, seleciona-se a opção *"Managed Data Optimized (Mutable)*" para ser possível definir um número serial para o dispositivo. Por fim, é preciso definir o Serial number do dispositivo, que será usado posteriormente, e clicar em *"Create my Device*" para criar o dispositivo. A Figura 22 apresenta um exemplo de número serial que pode ser usado. O nome utilizado para o dispositivo foi "ESP *Blue*-Quarto".

Figura 22: Criando o dispositivo MQTT *Relay*



Fonte: Autoria própria.

Depois de realizar todos estes procedimentos, finaliza-se as configurações da plataforma para a comunicação MQTT utilizando um Broker externo e é possível receber e ver os dados enviados através do dispositivo criado. Sendo assim, a próxima etapa foi a configuração e desenvolvimento do dispositivo publicador: a ESP32.

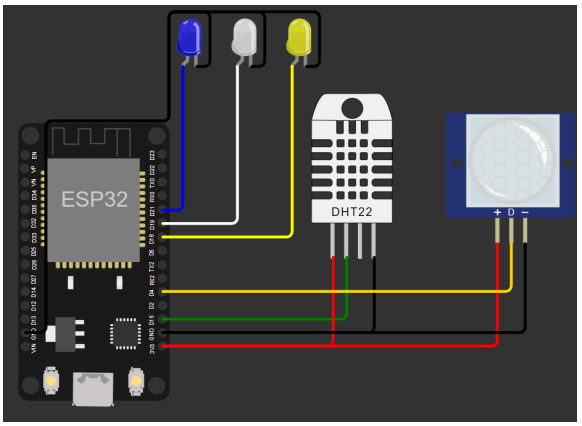
3.3 DESENVOLVIMENTO DA ESP32

Com a finalidade de operar como um intermédio dos sensores e o broker MQTT, utilizou-se uma ESP32 WROOM-32, para isso utilizando o simulador WOKWI, foi implementado um circuito esquemático juntamente com o *firmware* que foi enviado ao controlador.

3.3.1 Conexões

O projeto tem como objetivo a automação de um ambiente residencial, utilizando uma placa ESP32. O esquema do circuito é representado abaixo na Figura 23.

Figura 23: Esquema do circuito simulado.



Fonte: Autoria própria.

A ESP32 é responsável pelo monitoramento de temperatura, umidade e presença, além do acionamento automatizado dos dispositivos elétricos, lâmpada, ar-condicionado e umidificador, conforme as condições detectadas. Para isso, utiliza o sensor DHT22, conectado ao pino 15, para a medição da temperatura e umidade, e o sensor de presença PIR HC-SR501, ligado ao pino 4, para detectar movimentos no ambiente. Os leds amarelo, branco e azul foram usados para representar os dispositivos elétricos, respectivamente.

3.3.2 Código Fonte

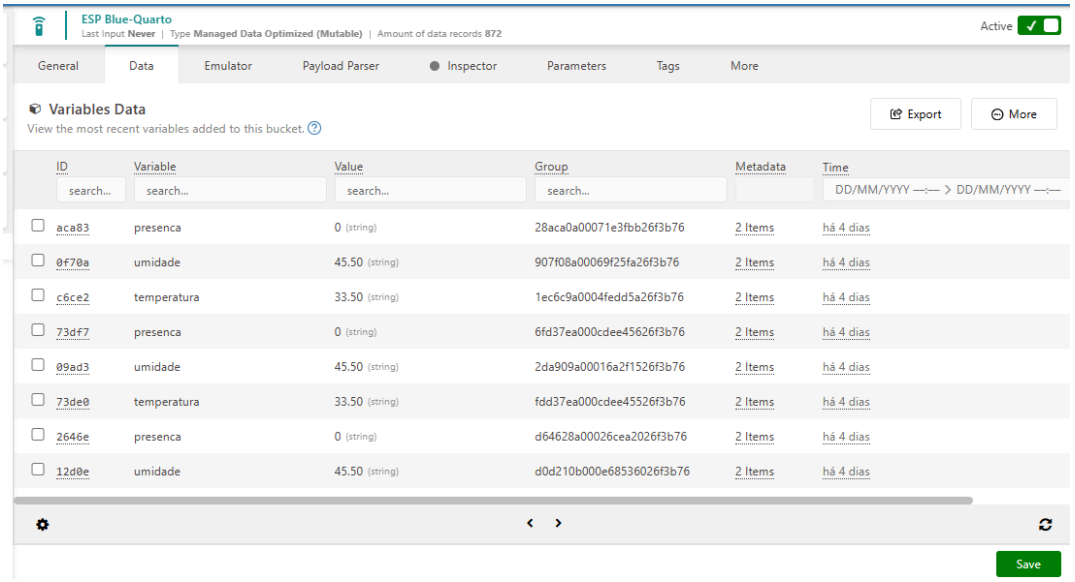
Como dito anteriormente, a comunicação de dados ocorre via protocolo MQTT, permitindo a interação com a plataforma TagoIO para armazenamento e visualização remota.

Inicialmente, a ESP32 se conecta à rede Wi-Fi configurada, garantindo a comunicação com o broker MQTT. Para isso, a biblioteca “WiFi.h” é utilizada, sendo necessário fornecer o SSID e a senha da rede. O protocolo seguro TLS é aplicado, garantindo uma comunicação criptografada com o MQTT *Broker*.

Para a leitura dos sensores, o código utiliza as bibliotecas “*DHTesp.h*” para o sensor de temperatura e umidade DHT22 e as funções nativas do ESP32 para a leitura do sensor de presença PIR HC-SR501. O DHT22 fornece leituras precisas de temperatura e umidade, fundamentais para a automação do sistema de climatização. O sensor PIR detecta movimento no ambiente, permitindo o controle inteligente da iluminação e demais dispositivos.

A comunicação com o *Broker* ocorre por meio da biblioteca “*PubSubClient.h*”, que gerencia a publicação e a assinatura de tópicos. A ESP32 envia periodicamente os valores de temperatura e umidade em formato JSON, bem como o estado do sensor de presença. A função “*snprintf*” é utilizada para formatar os valores antes da publicação, garantindo que os dados sejam transmitidos corretamente. Além dos dados dos sensores, também é enviado o estado de cada equipamento através de tópicos próprios. A Figura 24 mostra que é possível visualizar os dados recebidos pelo dispositivo na TagoIO entrando nele e clicando na aba “*Data*”.

Figura 24: Dados obtidos pelo dispositivo criado.



Fonte: Autoria própria.

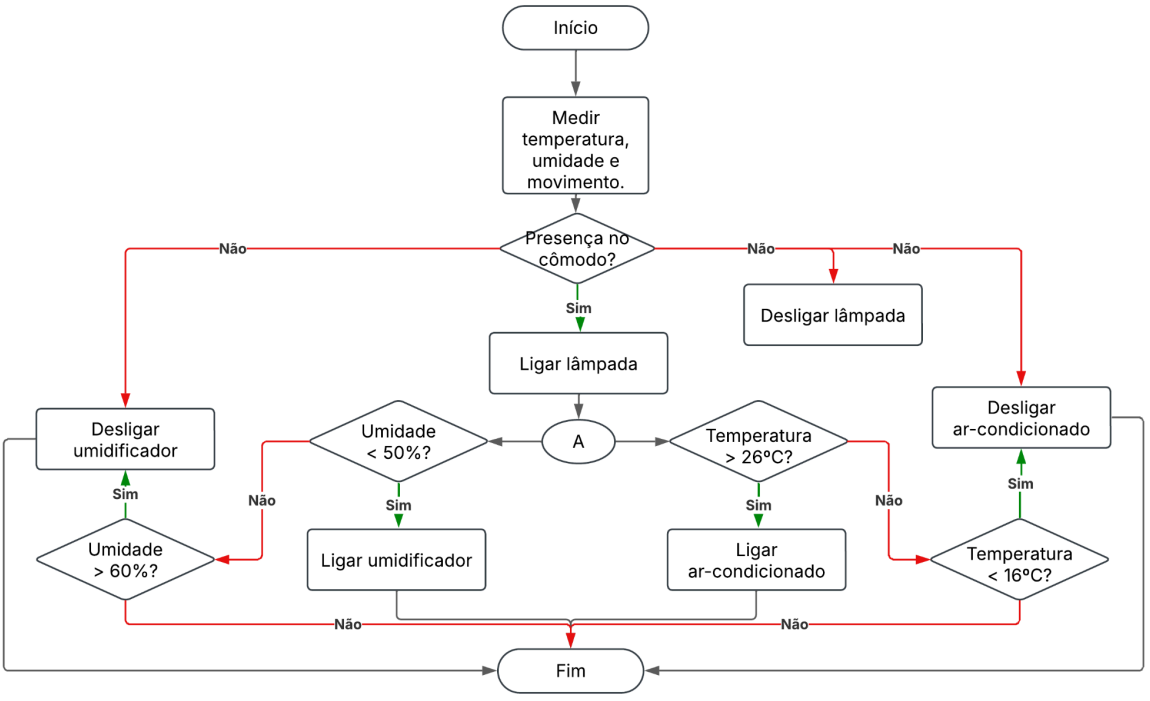
Com base nas leituras dos sensores, o código implementa a lógica de acionamento dos dispositivos elétricos. O ar-condicionado é ligado caso a temperatura ultrapasse 26°C e seja detectada presença no ambiente, desligando-se quando a temperatura estiver abaixo de 16°C ou após um período sem detecção de

movimento. O umidificador segue lógica semelhante, sendo ativado quando a umidade está abaixo de 50% e desligado acima de 60%. A iluminação do ambiente também é controlada automaticamente conforme a presença de pessoas. O fluxograma da Figura 25 mostra a lógica da automação.

Todas as funções foram implementadas na IDE do Arduino, garantindo compatibilidade com a ESP32 e facilitando o desenvolvimento do projeto. O código pode ser expandido para incluir novos sensores ou funcionalidades, tornando a automação do ambiente ainda mais eficiente.

O código desenvolvido para este projeto está disponível nos Apêndices deste documento, permitindo a análise detalhada e possíveis modificações conforme a necessidade.

Figura 25: Fluxograma da lógica da automação.



Fonte: Autoria própria.

3.4 *DASHBOARD*

Para a criação do *dashboard* na TagoIO, inicialmente, foi acessado o menu "*Dashboards"*, onde foi selecionada a opção "*Create Dashboard*" para iniciar a

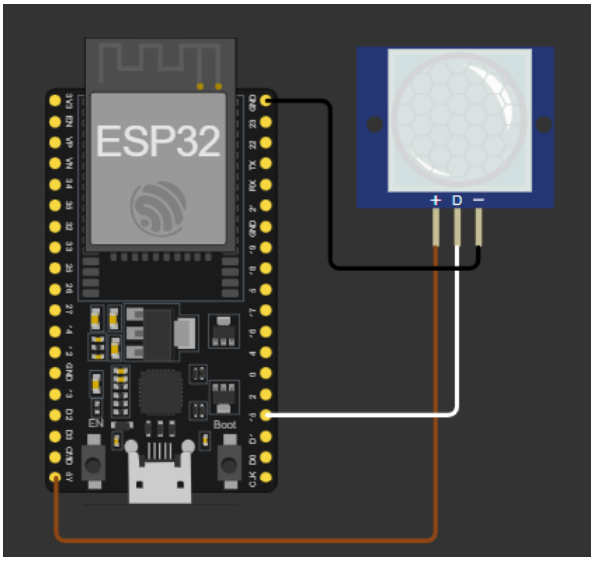
configuração. Em seguida, foi atribuído o nome “ TCC” ao painel, bem como um ícone de *Wi-Fi* para facilitar sua identificação dentro da plataforma.

Após a criação do dashboard, foram adicionados *widgets*, responsáveis pela exibição dos dados recebidos. Para isso, utilizou-se a opção “*Add Widget*”, permitindo a escolha entre diferentes tipos de visualização, como gráficos, indicadores numéricos, tabelas ou mapas. Para este dashboard, foi escolhido os *widgets* “*line*”, “*display*” e “*push button*”. Cada widget foi configurado individualmente, associando-o ao dispositivo cadastrado previamente e à variável de interesse: temperatura, umidade, presença, umidificador, ar-condicionado e lâmpada. Além disso, ajustes adicionais foram realizados para personalizar a aparência e disposição visual.

3.5 IMPLEMENTAÇÃO FÍSICA

Além do desenvolvimento de um sistema simulado de aplicação de automação residencial, foi desenvolvido um circuito simples de comunicação via MQTT. Utilizando um ESP32 e um sensor PIR HC-SR501 físicos, foi implementada a comunicação para envio do estado do sensor, podendo estar em 1 (detecção de movimento) ou 0 (sem detecção de movimento). Na Figura 26 é possível visualizar as conexões do circuito.

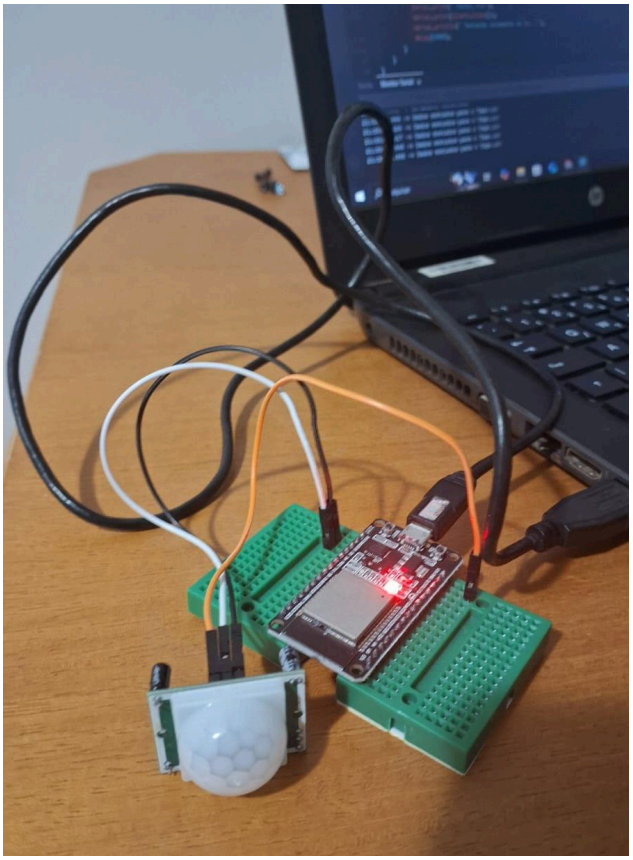
Figura 26: Conexões do circuito físico.



Fonte: Autoria própria.

O código fonte adicionado a ESP32 física foi bem parecido com o código utilizado na simulação porém mais simples e utilizando a IDE do Arduino como compilador. Assim como na simulação, inicia-se com a conexão com a internet e com o MQTT Broker. Para a leitura do sensor, foi utilizado as bibliotecas nativas do ESP32 e para comunicação com o broker também ocorre via biblioteca "*PubSubClient.h*". A transmissão dos dados ocorre da mesma forma que na simulação. A Figura 27 apresenta o circuito implementado e o código desenvolvido também está disponível nos Apêndices.

Figura 27: Circuito físico implementado.



Fonte: Autoria própria.

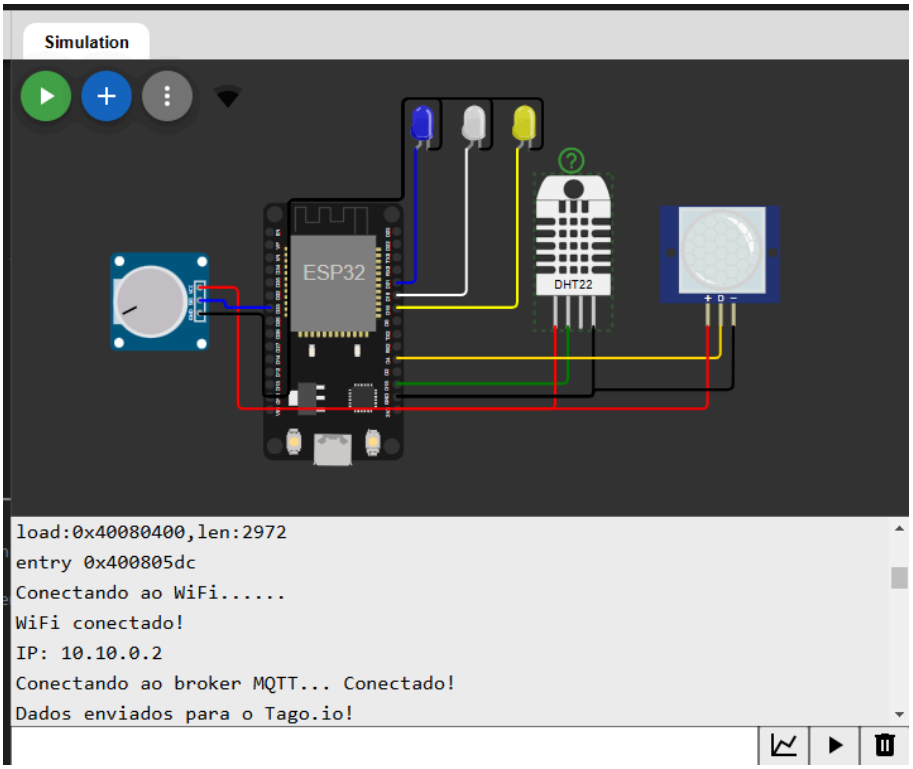
4 RESULTADOS

Neste capítulo, são apresentados os resultados obtidos com o envio de dados pela simulação e pela implementação física.

4.1 TESTES COM O SIMULADOR

O teste do envio de dados para a plataforma TagoIO foi realizado pelo simulador no Wokwi. Após inicializar o MQTT *Relay* no computador, foi iniciada a simulação e o envio de dados. No monitor serial, foi verificada a conexão com a internet e ao Broker e o envio dos primeiros dados. A Figura 28 mostra o monitor serial na inicialização.

Figura 28: Inicialização da simulação.

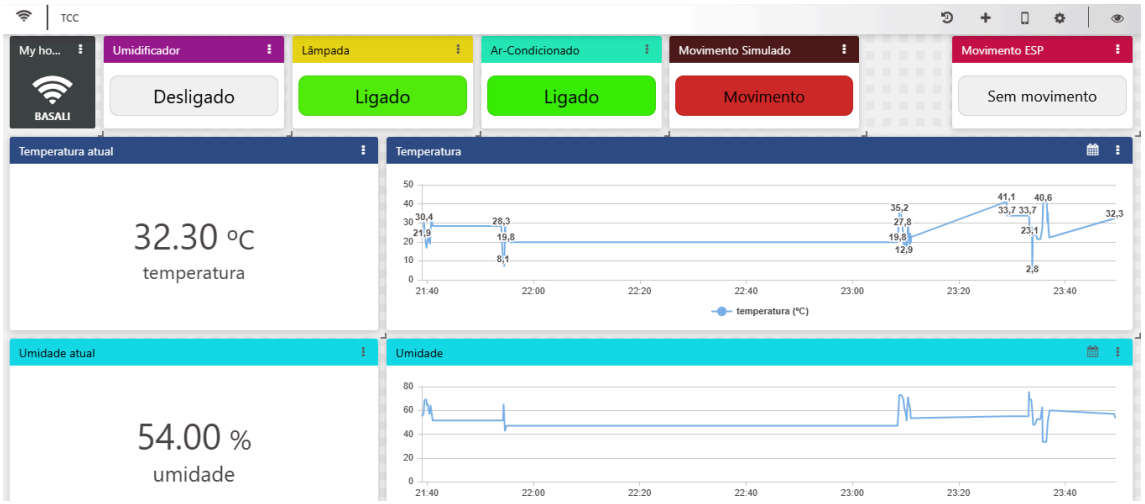


Fonte: Autoria própria.

Os dados foram obtidos e lidos com sucesso pela plataforma e imediatamente foi possível visualizá-los no *dashboard* “TCC”. A fim de testes da lógica condicional para acionamento e desligamento dos equipamentos, foram

simulados 4 cenários de estado dos sensores. No cenário 1, ilustrado pela Figura 29, os sensores mandaram uma leitura de temperatura igual a 32.3 ºC, 54% de umidade e presença no cômodo. Neste cenário, a lâmpada e o ar-condicionado foram ligados e o umidificador se manteve desligado, como esperado.

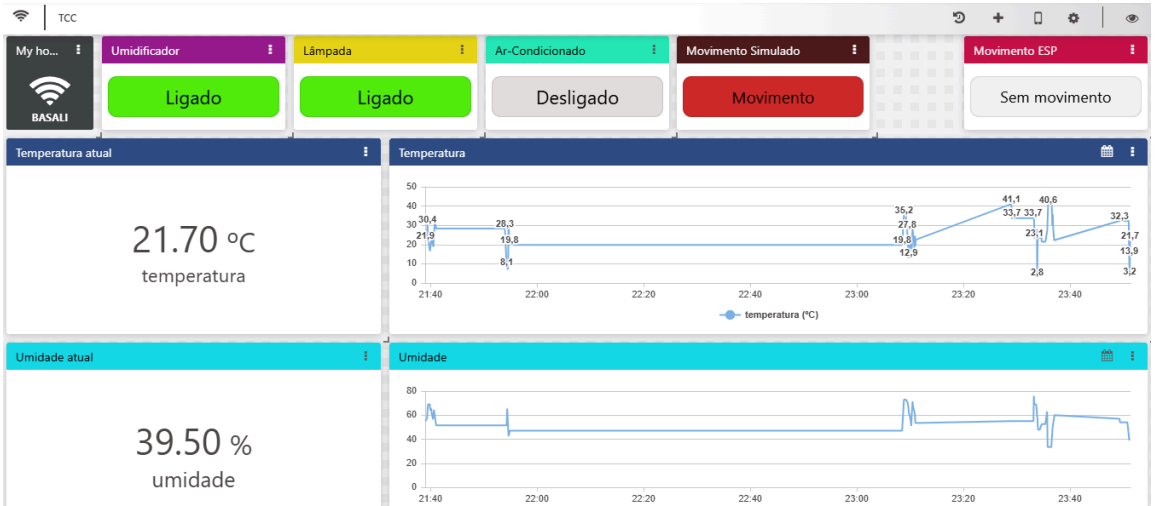
Figura 29: Cenário 1



Fonte: Autoria própria.

No cenário 2, ilustrado pela Figura 30, os sensores mandaram uma leitura de temperatura igual a 21.7 ºC, 39.5% de umidade e presença no cômodo. Neste cenário, a lâmpada e o umidificador foram ligados e o ar-condicionado foi desligado, como esperado.

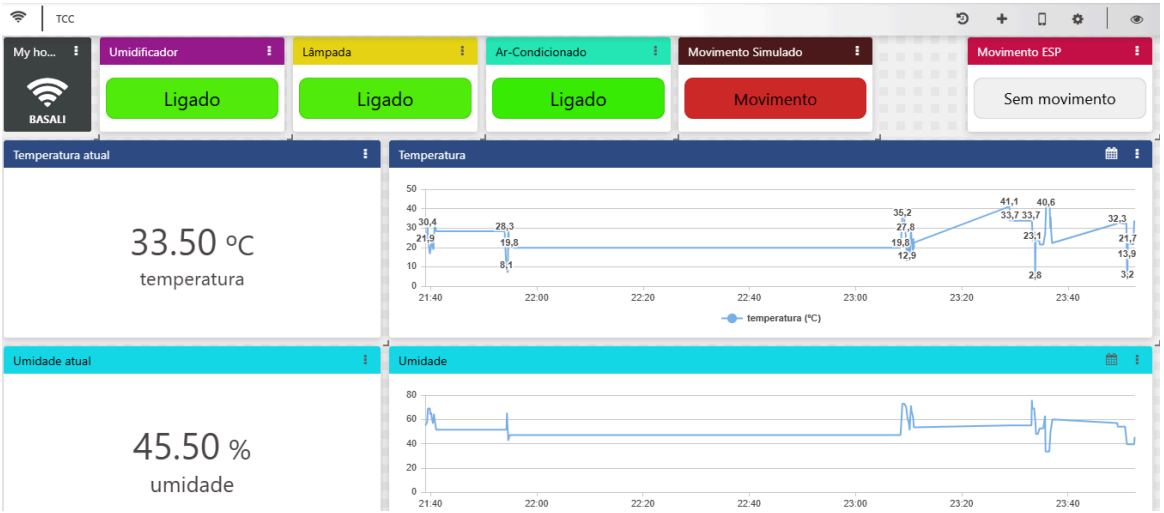
Figura 30: Cenário 2.



Fonte: Autoria própria.

No cenário 3, ilustrado pela Figura 31, os sensores mandaram uma leitura de temperatura igual a 33.5 ºC, 45.5% de umidade e presença no cômodo. Neste cenário, os três equipamentos foram ligados, como esperado.

Figura 31: Cenário 3.



Fonte: Autoria própria.

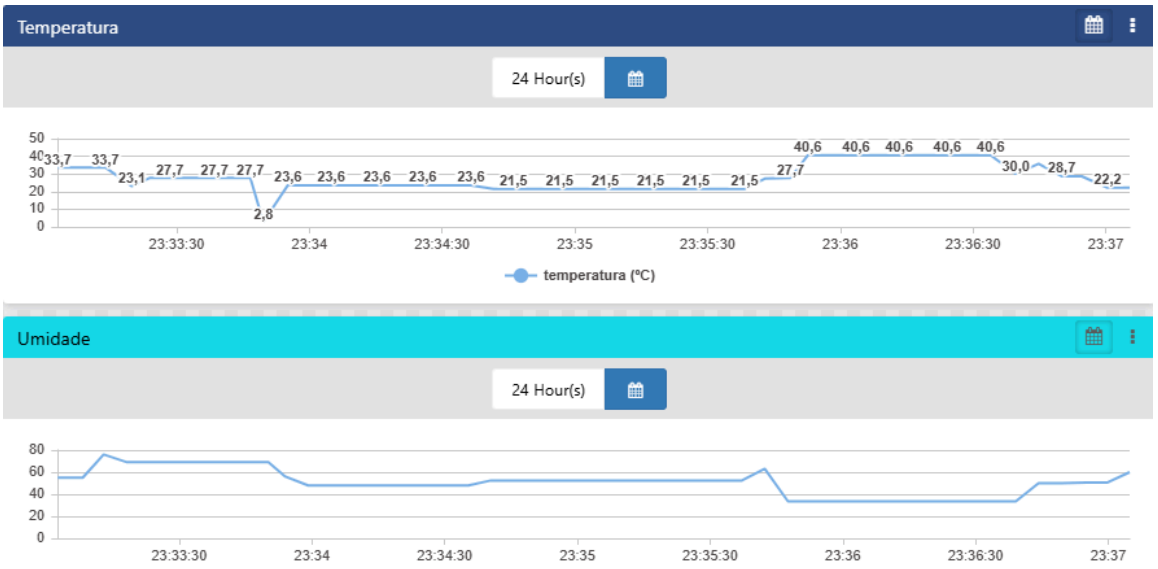
E no cenário 4, ilustrado pela Figura 32, os sensores mandaram as mesmas leituras de temperatura e de umidade do cenário 3, mas sem presença no cômodo. Neste cenário, os três equipamentos foram desligados, como esperado.

Figura 32: Cenário 4.

Fonte: Autoria própria.

Além do estado de cada equipamento e das medições de temperatura e umidade em tempo real, no dashboard é possível visualizar as medições de um determinado período em um gráfico linha. A figura 33 apresenta um exemplo da visualização ao longo do tempo.

Figura 33: Visualização ao longo do tempo de temperatura e umidade.

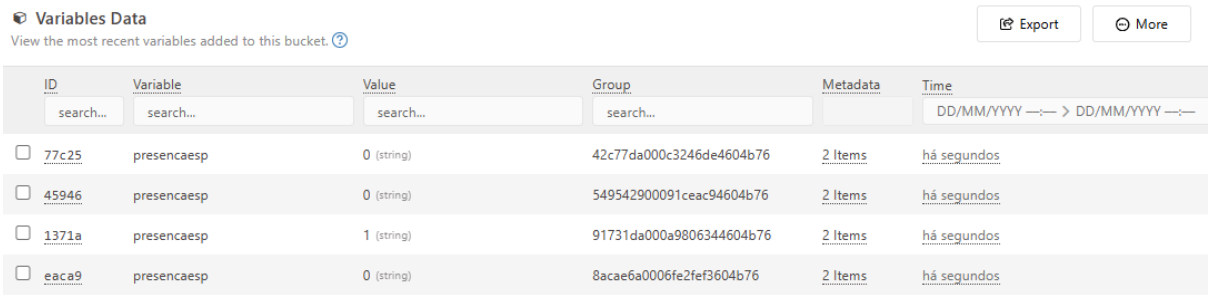


Fonte: Autoria própria.

4.2 ENVIO DE DADOS UTILIZANDO A ESP32 FÍSICA

O comportamento da ESP32 física é o mesmo que o simulado e pode-se enviar os dados da mesma forma e para o mesmo dispositivo. Na plataforma TagoIO, é possível visualizar o valor da variável “presencaesp” a cada envio de dados, dentro do dispositivo já criado. A Figura 34 mostra os dados recebidos para teste.

Figura 34: Dados teste recebidos.



Fonte: Autoria própria.

Com a variável recebendo os valores do sensor PIR físico, pode-se notar a mudança em seu respectivo botão dentro do *dashboard*, o botão “Movimento ESP”, como apresentado na Figura 35.

Figura 35: Botão “Movimento ESP”

Fonte: Autoria própria.

5 CONCLUSÃO

Este trabalho apresentou a simulação de uma aplicação IoT de baixo custo voltada para a automação residencial com o objetivo de reduzir o desperdício de energia elétrica. Além disso, foi demonstrado um exemplo prático de comunicação via protocolo MQTT, tecnologia amplamente utilizada em sistemas IoT residenciais e industriais.

O ESP32 mostrou-se uma excelente escolha para o processamento e a transmissão de dados, destacando-se por sua facilidade de programação, baixo custo e conectividade integrada. A simulação realizada no WOKWi foi suficiente para validar a implementação, apresentando-se como uma ferramenta eficaz para o desenvolvimento de projetos baseados no ESP32, além de ser gratuita e oferecer um comportamento muito próximo ao de um microcontrolador físico.

A TagoIO se mostrou uma plataforma robusta para soluções IoT, com ferramentas intuitivas e um ambiente de desenvolvimento amigável. Mesmo com as limitações do plano gratuito, foi possível explorar seus recursos de dashboards e armazenamento de dados. A plataforma ainda dispõe de funcionalidades avançadas, como “Actions” e “Analysis”, que podem ampliar as possibilidades de automação e análise de dados em projetos futuros.

A implementação da comunicação MQTT foi bem-sucedida tanto na simulação quanto no teste com o ESP32 físico. Os dados foram transmitidos corretamente do publicador para o broker, chegando ao assinante de forma eficiente, comprovando a confiabilidade desse protocolo na comunicação entre dispositivos IoT.

Como sugestão para trabalhos futuros, recomenda-se a utilização de um plano avançado na TagoIO ou de uma plataforma alternativa que permita o envio de dados via MQTT sem restrições, possibilitando a exploração completa dos recursos de automação da plataforma. Além disso, uma implementação física mais completa, incluindo sensores adicionais, como o DHT22 (para temperatura e umidade) e o SCT-013, pode tornar o sistema ainda mais eficiente. O SCT-013O é um sensor de corrente responsável por medir a corrente elétrica consumida por um equipamento e pode ser utilizado em aplicações IoT para identificação de consumos excedentes em uma residência .Também seria interessante o controle de mais atuadores, como ventiladores, exaustores e alertas sonoros, para aprimorar o gerenciamento do

consumo energético.

Dessa forma, este estudo demonstra a viabilidade e eficiência da IoT na automação residencial, abrindo caminho para futuras melhorias e implementações mais abrangentes.

REFERÊNCIAS

AUTOCORE. Conhecendo a Fundo o Buzzer. 2021. Disponível em: https://autocorerobotica.blog.br/conhecendo-a-fundo-o-buzzer/. Acesso em: 31 jan. 2025.

AZMAN, Muhammad Asyraaf; RAHMAN, Rahisham Abd. A Real-Time Energy Consumption and Monitoring for Domestic Houses by using Internet of Things Approach. *Evolution in Electrical and Electronic Engineering*, v. 3, n. 2, p. 710-719, 2022. Disponível em: https://doi.org/10.30880/eeee.2022.03.02.084. Acesso em: 02 set. 2024.

BERNARDES, Júlio. Temperatura global aumenta 1,6°C e segue subindo: "É como tentar parar um caminhão em alta velocidade". *Jornal da USP*, 2025. Disponível em: https://jornal.usp.br/ciencias/temperatura-global-aumenta-16c-e-segue-subindo-e-como-tentar-parar-um-caminhao-em-alta-velocidade/. Acesso em: 29 jan. 2025.

BORDIGNON, Lucas Muller. Análise de protocolos de comunicação para a eficiência energética em sistemas IoT. 2023. 56f. Monografia (Graduação em Engenharia de Computação) - Universidade Federal de Santa Maria, Santa Maria, 2023. Disponível em: http://repositorio.ufsm.br/handle/1/31391. Acesso em: 31 jan. 2025.

CCEE - CÂMARA DE COMERCIALIZAÇÃO DE ENERGIA ELÉTRICA. Consumo de energia no Brasil subiu 5% no 1º trimestre de 2024, aponta CCEE. *CCEE*, 2024. Disponível em: https://www.ccee.org.br/en/web/guest/-/consumo-de-energia-no-brasil-subiu-5-no-1-trimestre-de-2024-aponta-ccee?utm_source=chatgpt.com. Acesso em: 29 jan. 2025.

CUT. Queda brusca da umidade relativa do ar: Saiba o que fazer para proteger sua saúde. *Central Única dos Trabalhadores*, 2021. Disponível em: https://www.cut.org.br/noticias/queda-brusca-da-umidade-relativa-do-ar-saiba-o-que-fazer-para-proteger-sua-saude-d9d6. Acesso em: 30 jan. 2025.

EPE - EMPRESA DE PESQUISA ENERGÉTICA. Matriz Energética e Elétrica. 2024. Disponível em: https://www.epe.gov.br/pt/abcdenergia/matriz-energetica-e-eletrica. Acesso em: 29 jan. 2025.

FALCÃO, Djalma M. et al. Integração de tecnologias para viabilização da smart grid. *III Simpósio Brasileiro de Sistemas Elétricos*, p. 1-5, 2010. Disponível em: http://www.cricte2004.eletrica.ufpr.br/odilon/te339/artigo_SMART_GRID.PDF. Acesso em: 28 jan. 2025.

FÉLIX, Thiago. Brasil pode aumentar em mais de 3% o consumo de energia anual. *CNN Brasil*, 2024. Disponível em: https://www.cnnbrasil.com.br/economia/money/brasil-pode-aumentar-em-mais-de-3-o-consumo-de-energia-anual/?utm_source=chatgpt.com. Acesso em: 29 jan. 2025.

LIU, Yuanyuan; QIU, Bo; FAN, Xiaodong; ZHU, Haijing; HAN, Bochong. Review of Smart Home Energy Management Systems. *Energy Procedia*, v. 104, p. 504-508, 2016. ISSN 1876-6102. Disponível em: https://doi.org/10.1016/j.egypro.2016.12.085. Acesso em: 01 fev. 2025.

PRATHYUSHA, M. R.; BISWAJIT, Bhowmik. Development of IoT-Based Smart Home Application with Energy Management. *Proceedings of the International Conference on Sustainable Communication Networks and Application (ICSCNA 2023)*. 2023. p. 367-371. IEEE Xplore. DOI: 10.1109/ICSCNA58489.2023.10370276. Acesso em: 02 set. 2024.

RH MATERIAIS ELÉTRICOS. Como funciona um Relé de Estado Sólido?. 2023. Disponível em: https://www.rhmateriaiseletricos.com.br/como-funciona-um-rele-de-estado-solido. Acesso em: 31 jan. 2025.

ROBOCORE. ESP32 - WiFi + Bluetooth. Disponível em: https://www.robocore.net/wifi/esp32-wifi-bluetooth. Acesso em: 30 jan. 2025.

ROBOCORE. Sensor de Corrente Não Invasivo 100A SCT-013. Disponível em: https://www.robocore.net/sensor-corrente-tensao/sensor-de-corrente-nao-invasivo-100a-sct-013. Acesso em: 31 jan. 2025.

SIMÕES, Haroldo. Detector de Presença com Sensor PIR HC-SR501. Módulo Eletrônica. 2018. Disponível em: https://blog.moduloeletronica.com.br/detector-de-presenca-com-sensor-pir-hc-sr501/

. Acesso em: 31 jan. 2025.

TAGOIO. Connecting your MQTT Broker to TagoIO. Disponível em: https://help.tago.io/portal/en/kb/articles/tagoio-mqtt-relay. Acesso em: 12 dez. 2024.

TAGOIO. The robust platform to easily build scalable IoT solutions. Disponível em: https://tago.io/. Acesso em: 30 jan. 2025.

WOKWI. World's most advanced ESP32 simulator. Disponível em: https://wokwi.com/. Acesso em: 14 nov. 2024.

ZHOU, Bin; LI, Wentao; CHAN, Ka Wing; CAO, Yijia; KUANG, Yonghong; LIU, Xi; WANG, Xiong. Smart Home Energy Management Systems: Concept, configurations, and applications. *Renewable and Sustainable Energy Reviews*. DOI: 10.1016/j.rser.2016.03.047. Disponível em: https://www.sciencedirect.com. Acesso em: 02 set. 2024.

APÊNDICES

Apêndice A: Código Fonte da ESP32 simulada.

```
#include <WiFi.h>
#include <WiFiClientSecure.h>
#include <PubSubClient.h>
#include <DHT.h>
#include "DHTesp.h"

// Configuração da Rede WiFi
const char* ssid = "Wokwi-GUEST";
const char* password = "";

// Configuração do HiveMQ Cloud
const char* mqtt_server =
"03f771ead3b34380811b3037bead2888.s1.eu.hivemq.cloud";
const int mqtt_port = 8883;
const char* mqtt_user = "basali02";
const char* mqtt_password = "";

// Configuração do Sensor de Temperatura e Umidade - DHT22
#define DHT_PIN 15 // Pino de dados do DHT22
DHTesp dhtSensor;

// Configuração do Sensor de Presença PIR - HC-SR501
#define PIR_PIN 4  // Pino conectado ao OUT do PIR

// Configuração dos Equipamentos de Saida
#define AC_PIN 21 // Pino do Ar-Condicionado
#define HM_PIN 19 // Pino do Umidificador
#define LP_PIN 18 // Pino da Lâmpada

// VARIAVEIS GLOBAIS
int time_presence = 0;
```

```
// Cliente WiFi Seguro (TLS)
WiFiClientSecure espClient;
PubSubClient client(espClient);

void setupWiFi() {
    Serial.print("Conectando ao WiFi...");
    WiFi.begin(ssid, password);
    while (WiFi.status() != WL_CONNECTED) {
        delay(500);
        Serial.print(".");
    }
    Serial.println("\nWiFi conectado!");
    Serial.print("IP: ");
    Serial.println(WiFi.localIP());
}

void reconnectBroker() {
    while (!client.connected()) {
        Serial.print("Conectando ao broker MQTT...");

        if (client.connect("ESP32_HiveMQ", mqtt_user,
mqtt_password)) {
            Serial.println(" Conectado!");
            client.subscribe("test/topic");
        } else {
            Serial.print(" Falha, rc=");
            Serial.print(client.state());
            Serial.println(" Tentando novamente em 5s...");
            delay(5000);
        }
    }
}

void setup() {
    Serial.begin(115200);
    setupWiFi();

    espClient.setInsecure();  // Apenas para testes!

    client.setServer(mqtt_server, mqtt_port);
    reconnectBroker();
```

```
    dhtSensor.setup(DHT_PIN, DHTesp::DHT22);  // Inicializa o
DHT22

    pinMode(PIR_PIN, INPUT);  // Define o pino do PIR - HC-SR501
como entrada
    pinMode(AC_PIN, OUTPUT);  // Define o Pino do Ar-Condicionado
como saída
    pinMode(HM_PIN, OUTPUT);  // Define o Pino do Umidificador
como saída
    pinMode(LP_PIN, OUTPUT);  // Define o Pino da Lâmpada como
saída
}

void loop() {
    if (!client.connected()) {
        reconnectBroker();
    }
    client.loop();

    // Leitura de temperatura e umidade
    TempAndHumidity  data = dhtSensor.getTempAndHumidity(); //
Leitura da Temperatura e Umidade

    // Criando mensagem JSON para temperatura
    char payload[100];
    snprintf(payload, sizeof(payload), "%.2f", data.temperature);
    client.publish("topic/123123123", payload);

    // Criando mensagem JSON para umidade
    snprintf(payload, sizeof(payload), "%.2f", data.humidity);
    client.publish("topic/12121212", payload);


    int estado = digitalRead(PIR_PIN);  // Lê o estado do sensor
de Presença

    if(estado == HIGH)     //Variável acumuladora para
identificação de tempo sem detectar alguem
    time_presence = 0;
    else time_presence++;
```

```
    if (estado == HIGH) {
        Serial.println("Movimento detectado!");

        char payload[50];
        snprintf(payload, sizeof(payload), "1");
        client.publish("topic/123456789", payload);
    } else {
        char payload[50];
        snprintf(payload, sizeof(payload), "0");
        client.publish("topic/123456789", payload);
    }

    Serial.println("Dados enviados para o Tago.io!");

    //Lógica de Acionamento dos Equipamentos da casa

    if(data.temperature > 26.00 && estado == HIGH){ // Lógica de
Liga/Desliga do Ar-Condicionado
    digitalWrite(AC_PIN, HIGH);
    snprintf(payload, sizeof(payload), "1");
    client.publish("topic/23232323", payload);
    }
    else if(data.temperature < 16.00 || time_presence == 2){
    digitalWrite(AC_PIN, LOW);
    snprintf(payload, sizeof(payload), "0");
    client.publish("topic/23232323", payload);
    }
    if(data.humidity < 50.00 && estado == HIGH){ // Lógica de
Liga/Desliga do Umificador
    digitalWrite(HM_PIN, HIGH);
    snprintf(payload, sizeof(payload), "1");
    client.publish("topic/56565656", payload);
    }
    else if(data.humidity > 60.00 || time_presence == 2){
    digitalWrite(HM_PIN, LOW);
    snprintf(payload, sizeof(payload), "0");
    client.publish("topic/56565656", payload);
    }
    if(estado == HIGH){
    digitalWrite(LP_PIN, HIGH);
```

```
    snprintf(payload, sizeof(payload), "1");
    client.publish("topic/87878787", payload);
    }
    else if(time_presence == 2){
    digitalWrite(LP_PIN, LOW);
    snprintf(payload, sizeof(payload), "0");
    client.publish("topic/87878787", payload);
    }
    delay(5000);  // Publica a cada 5 segundos
}
```

Apêndice B: Código Fonte da ESP32 física.

```
#include <WiFi.h>
#include <WiFiClientSecure.h>
#include <PubSubClient.h>

// Configuração da Rede WiFi
const char* ssid = "BASALE";  // Rede
const char* password = "";          // Senha

//  Configuração do HiveMQ Cloud
const char* mqtt_server =
"03f771ead3b34380811b3037bead2888.s1.eu.hivemq.cloud";
const int mqtt_port = 8883;  // TLS ativo
const char* mqtt_user = "basali02";
const char* mqtt_password = "Basali02";

// Configuração do Sensor de Presença PIR - HC-SR501
#define PIR_PIN 15  // Pino conectado ao OUT do PIR

// Cliente WiFi Seguro (TLS)
WiFiClientSecure espClient;
PubSubClient client(espClient);

void setupWiFi() {
    Serial.print("Conectando ao WiFi...");
    WiFi.begin(ssid, password);
    while (WiFi.status() != WL_CONNECTED) {
        delay(500);
        Serial.print(".");
    }
    Serial.println("\nWiFi conectado!");
```

```
    Serial.print("IP: ");
    Serial.println(WiFi.localIP());
}

void reconnectBroker() {
    while (!client.connected()) {
        Serial.print("Conectando ao broker MQTT...");

        if (client.connect("ESP32_HiveMQ", mqtt_user,
mqtt_password)) {
            Serial.println(" Conectado!");
            client.subscribe("test/topic");
        } else {
            Serial.print(" Falha, rc=");
            Serial.print(client.state());
            Serial.println(" Tentando novamente em 5s...");
            delay(5000);
        }
    }
}

void setup() {
    Serial.begin(115200);
    setupWiFi();

    espClient.setInsecure();  // Apenas para testes!

    client.setServer(mqtt_server, mqtt_port);
    reconnectBroker();

    pinMode(PIR_PIN, INPUT);  // Define o pino do PIR - HC-SR501
como entrada
}

void loop() {
    if (!client.connected()) {
        reconnectBroker();
    }
    client.loop();

    int estado = digitalRead(PIR_PIN);  // Lê o estado do sensor
```

```
de Presença

    if (estado == HIGH) {
        Serial.println("Movimento detectado!");

        char payload[50];
        snprintf(payload, sizeof(payload), "1");
        client.publish("topic/45454545", payload);
    } else {
        char payload[50];
        snprintf(payload, sizeof(payload), "0");
        client.publish("topic/45454545", payload);
    }

    Serial.println("Dados enviados para o Tago.io!");

    delay(5000);  // Publica a cada 5 segundos
}
```

Apêndices C: Código do *Payload Parser* na plataforma TagoIO

```
if (Array.isArray(payload)) {
   // Relay sends the "payload" variable by default
  const dataReceived = payload.find(x => x.variable ===
"payload");
  // Extract the serial from the last element of the topic (e.g.,
/device/SERIAL)
  serial = dataReceived?.metadata.topic.split("/").pop();
  if (serial === "123123123"){
    dataReceived.variable = "Temperatura";
    dataReceived.unit = "°C";
  }
  if (serial === "12121212"){
    dataReceived.variable = "Umidade";
    dataReceived.unit="%";
  }
  if (serial === "123456789"){
    dataReceived.variable = "Presenca";
    dataReceived.unit="";
  }
```

```
   if (serial === "222222222"){
    dataReceived.variable = "Corrente";
    dataReceived.unit="";
  }
  if (serial === "23232323"){
    dataReceived.variable = "ArCond";
    dataReceived.unit="";
  }
  if (serial === "56565656"){
    dataReceived.variable = "Humidificador";
    dataReceived.unit="";
  }
  if (serial === "87878787"){
    dataReceived.variable = "Lampada";
    dataReceived.unit="";
  }
   if (serial === "45454545"){
    dataReceived.variable = "PresencaEsp";
    dataReceived.unit="";
  }
}
```