UNIVERSIDADE DA INTEGRAÇÃO INTERNACIONAL DA LUSOFONIA AFRO-BRASILEIRA
INSTITUTO DE ENGENHARIAS E DESENVOLVIMENTO SUSTENTÁVEL

GABRIEL OLIVEIRA DOS SANTOS
JANAINA SOUSA DE OLIVEIRA

MICROCONTROLADORES E DSPs: Desenvolvimento de protótipo IoT integrando ESP8266, MQTT e aplicativo Android para leitura de temperatura e controle de LEDs

REDENÇÃO
2025

1. INTRODUÇÃO E JUSTIFICATIVA

A Internet das Coisas (IoT) tem se consolidado cada vez mais como uma das áreas mais relevantes dentro do campo dos sistemas embarcados, permitindo que dispositivos físicos sejam monitorados e controlados remotamente por meio de redes de comunicação. Segundo BOCA et al. (2023), a IoT integra sensores, atuadores e serviços em nuvem para coletar e trocar dados em tempo real, permitindo o monitoramento e a otimização de processos. Nesse contexto, o estudo e a implementação de microcontroladores conectados assumem papel importante na formação de profissionais de engenharia e tecnologia.

A disciplina de Microcontroladores e DSPs exige que o estudante desenvolva competências relacionadas à programação em baixo nível, à integração de hardware e software e à utilização de protocolos de comunicação eficientes. O microcontroller ESP8266, utilizado neste projeto, destaca-se por oferecer conectividade Wi-Fi integrada, baixo custo e facilidade de programação, o que o torna adequado tanto para fins educacionais quanto para prototipagem rápida. A integração com o protocolo MQTT também se mostra pertinente, uma vez que esse protocolo é amplamente empregado em aplicações IoT devido à sua leveza e confiabilidade em redes de baixa largura de banda, conforme descrito por ROISENBERG (2022).

O desenvolvimento de um protótipo que realiza leitura de um sensor, transmite dados via MQTT e permite controle remoto através de um aplicativo Android contribui diretamente para a consolidação dos conteúdos trabalhados na disciplina. Esse tipo de atividade permite ao estudante aplicar conceitos fundamentais de interfaces analógicas, conversão A/D, timers, comunicação em rede, armazenamento local e segurança de dados. Além disso, reforça a compreensão de arquiteturas de microcontroladores e práticas de desenvolvimento de firmware robusto, que são importantes na área de sistemas embarcados.

A escolha do tema também se justifica pelo crescimento acelerado do ecossistema IoT em escala global. De acordo com reportagem da Forbes Brasil, publicada por PACETE (2022), estima-se que mais de 27 bilhões de dispositivos IoT estarão conectados até 2025, demonstrando a expansão contínua dessa tecnologia e a necessidade de profissionais qualificados para trabalhar com soluções embarcadas conectadas. Dessa forma, o projeto apresentado neste relatório não apenas atende às exigências acadêmicas da disciplina, mas também contribui para a formação de competências alinhadas às demandas atuais do mercado tecnológico.

2. OBJETIVOS

Esta seção apresenta o objetivo geral e os objetivos específicos do projeto, destacando o que se pretende alcançar com o desenvolvimento do protótipo IoT e sua integração entre hardware, nuvem e aplicativo móvel.

2.1. OBJETIVO GERAL

Desenvolver um protótipo IoT capaz de simular a leitura de temperatura e permitir o controle remoto de LEDs por meio de comunicação MQTT entre o ESP8266 e um aplicativo Android.

2.2. OBJETIVOS ESPECÍFICOS

- Montar o circuito com o ESP8266, potenciômetro, LEDs e demais componentes necessários.
- Programar o microcontrolador para ler a temperatura simulada a cada 2,5 segundos.
- Enviar os dados ao broker MQTT e receber comandos do aplicativo.
- Desenvolver o app Android para visualizar a leitura e controlar os LEDs.
- Validar a comunicação bidirecional entre app e ESP8266.

3. MATERIAIS E MÉTODOS

Esta seção apresenta os recursos utilizados no desenvolvimento do protótipo, bem como os procedimentos adotados para sua implementação. São descritos os componentes de hardware, os softwares empregados e as etapas seguidas para integrar a leitura de temperatura simulada, a comunicação via MQTT e o controle remoto através do aplicativo Android. O objetivo é detalhar o processo de construção do sistema, desde a montagem física do circuito até a programação e os testes funcionais.

3.1. MATERIAIS UTILIZADOS

Os componentes de hardware e softwares apresentados a seguir foram utilizados no desenvolvimento do projeto.

3.1.1. Hardware utilizado

Os elementos de hardware foram selecionados para compor um circuito simples, modular e adequado para testes de leitura e envio de dados.

- ESP8266 NodeMCU V3: microcontrolador com Wi-Fi integrado, responsável por realizar leituras analógicas, processar os dados simulados e estabelecer comunicação com o broker MQTT.
- Protoboard: placa de ensaio que possibilita montar o circuito sem solda, facilitando modificações e experimentações.
- 1 Potenciômetro linear de 100kΩ: utilizado como simulador de temperatura, gerando variações de tensão conforme a resistência é ajustada.
- 1 Resistor de 1kΩ: componente destinado a limitar a corrente elétrica no circuito, garantindo a proteção dos dispositivos conectados.
- LED comum: atuador controlado remotamente pelo aplicativo Android, permitindo verificar o funcionamento bidirecional da comunicação MQTT.
- Jumpers rígidos ou normais: cabos empregados na interligação dos

componentes, possibilitando a montagem e reorganização do protótipo.

3.1.2. Softwares utilizado

As ferramentas de software permitiram desenvolver o firmware, estabelecer a comunicação em nuvem e criar o aplicativo móvel de controle.

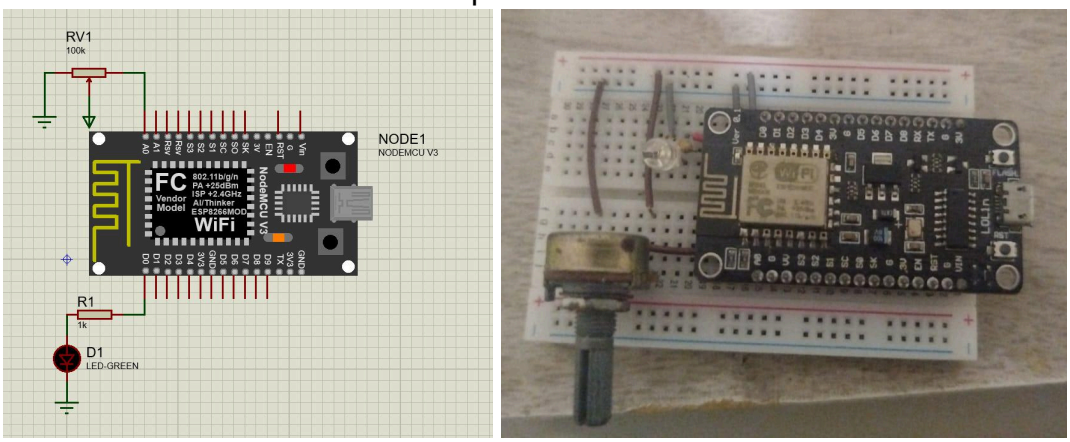
- Arduino IDE: utilizado para programar o ESP8266, compilar o código e realizar o upload do firmware.
- MQTT: protocolo leve de troca de mensagens, empregado para comunicação entre o microcontrolador, o broker e o aplicativo.
- Cloud HiveMQ: broker MQTT em nuvem que gerencia os tópicos utilizados no envio e recebimento de dados.
- Android Studio: ambiente de desenvolvimento utilizado para criar o aplicativo Android responsável pela visualização e controle do sistema.
- SQLite (nativo do Android) – utilizado para persistência de dados do usuário e armazenamento das credenciais do servidor MQTT.

3.2. MÉTODOS

O desenvolvimento do protótipo foi realizado em etapas sequenciais envolvendo a montagem do circuito, a programação do microcontrolador ESP8266, a configuração do serviço MQTT na nuvem e a implementação do aplicativo Android responsável pelo monitoramento e controle do sistema.

Inicialmente, o circuito foi montado em uma protoboard utilizando o ESP8266 NodeMCU V3 como unidade principal de processamento, conforme mostrado na Figura 1.

FIGURA 1 – Diagrama esquemático e montagem do circuito inicial com ESP8266 e potenciômetro



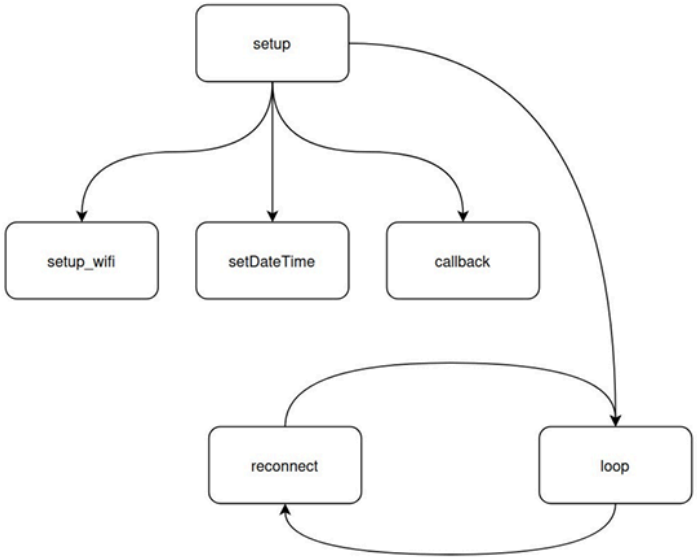
Fonte: acervo pessoal, 2025.

Um potenciômetro linear de 100 kΩ foi conectado à entrada analógica do microcontrolador, atuando como um sensor de temperatura simulado, uma vez que sua variação de resistência permite gerar diferentes níveis de tensão. Além disso, um resistor de 1 kΩ foi inserido para limitar a corrente no acionamento do LED, garantindo

proteção ao componente e ao próprio microcontrolador. Toda a interligação do circuito foi realizada por meio de jumpers rígidos ou tradicionais, facilitando adaptações durante os testes.

Após a montagem do circuito, iniciou-se a programação do ESP8266 utilizando a Arduino IDE. A seguir, são apresentadas as principais etapas para estabelecer conexão segura com a nuvem, como mostrado na Figura 2.

FIGURA 2 – Fluxo de execução do firmware no ESP8266



Fonte: acervo pessoal, 2025.

O código do microcontrolador foi organizado de maneira clara e modular. A execução inicia-se na função setup, responsável por iniciar o monitor serial, carregar os certificados de segurança, configurar os pinos de entrada e saída, entre eles o pino do LED, estabelecer os parâmetros de conexão MQTT e chamar as demais funções do sistema.

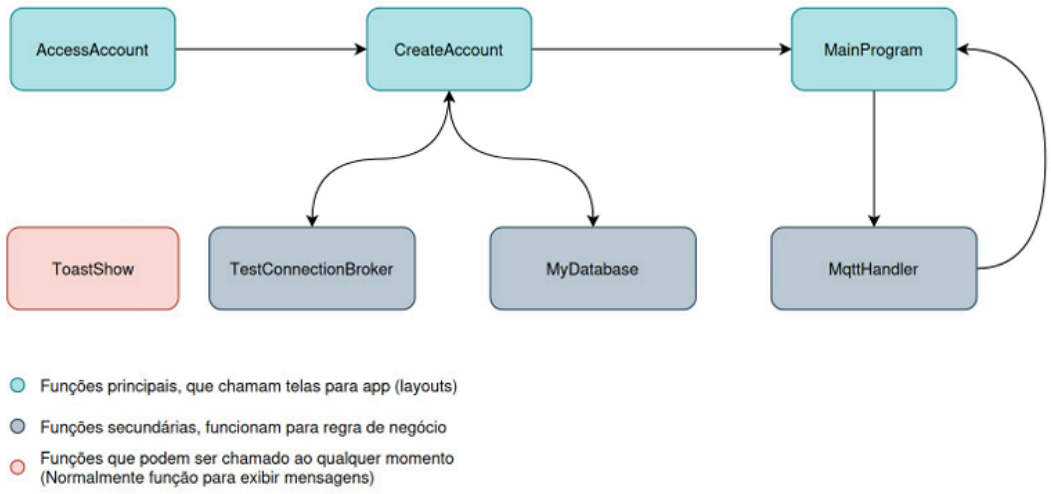
Entre essas funções está setup_wifi, encarregada de realizar tentativas de conexão à rede Wi-Fi até que o dispositivo obtenha sucesso. Em seguida, é executada a função setDateTime, que define o fuso horário e sincroniza o relógio interno com um servidor NTP, garantindo o funcionamento correto da camada de segurança. A função callback desempenha o papel de tratar as mensagens recebidas no tópico MQTT configurado. Sempre que o aplicativo envia dados em formato JSON, o callback interpreta o conteúdo e, se houver comandos para acionar os LEDs, aplica a lógica necessária.

Depois dessa fase de inicialização, o funcionamento contínuo fica a cargo da função loop, que verifica a todo instante se a conexão com o broker MQTT permanece ativa. Caso a conexão seja perdida, o dispositivo utiliza a função reconnect para tentar restabelecer o acesso ao servidor. Dentro do loop também ocorre a leitura do

potenciômetro, que simula a temperatura. Essa leitura é realizada a cada 2,5 segundos, convertida para um valor representativo e enviada como JSON ao broker.

Paralelamente ao firmware, o aplicativo Android foi desenvolvido no Android Studio, utilizando Java. Seu fluxo de funcionamento é mais complexo do que no ESP8266, pois envolve etapas de cadastro, validação de credenciais, persistência de dados no SQLite e comunicação MQTT segura. A figura a seguir apresenta o diagrama do software, destacando as funções principais e secundárias do aplicativo:

FIGURA 3 – Estrutura funcional do aplicativo Android



Fonte: acervo pessoal, 2025.

O sistema possui um mecanismo de cadastro que utiliza SQLite para armazenar localmente as credenciais MQTT. A função AccessAccount verifica se já existem dados salvos e direciona o usuário à tela apropriada. Caso seja necessário criar um novo cadastro, a função CreateAccount realiza a validação das informações fornecidas e permite testar a conexão com o broker utilizando a função TestConnectionBroker. As credenciais são gerenciadas pela classe MyDatabase, enquanto mensagens de feedback são exibidas pela classe ToastShow.

A interface principal do aplicativo é apresentada pela função MainProgram. Ela exibe em tempo real o valor de temperatura simulada enviado pelo ESP8266 e oferece botões para controlar os LEDs remotamente. Toda a comunicação MQTT no app é administrada pela classe MqttHandler, que realiza a conexão, inscrição nos tópicos, interpretação das mensagens JSON recebidas e envio dos comandos destinados ao microcontrolador.

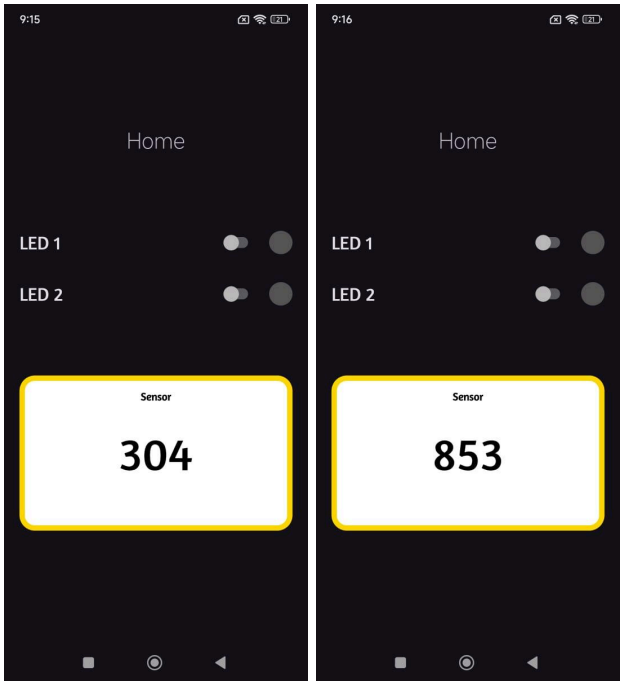
A combinação dessas etapas permitiu a criação de um protótipo funcional, integrando hardware, microcontrolador, comunicação segura via MQTT e um aplicativo Android, resultando em um sistema capaz de enviar leituras periódicas de temperatura simulada e responder aos comandos de controle de LEDs de forma remota.

4. RESULTADOS E TESTES REALIZADOS

A etapa de testes permitiu verificar o funcionamento do protótipo em todas as suas partes, avaliando a comunicação entre o ESP8266, o broker MQTT e o aplicativo Android. Os resultados obtidos demonstram que o sistema operou de forma estável, atendendo aos requisitos propostos.

Inicialmente, foram realizados testes da leitura de temperatura simulada por meio do potenciômetro. Ao ajustar o componente, o ESP8266 registrou variações no valor analógico e enviou os dados corretamente ao HiveMQ Cloud a cada 2,5 segundos. No aplicativo Android, os valores recebidos foram exibidos em tempo real, confirmando a comunicação contínua entre os dispositivos.

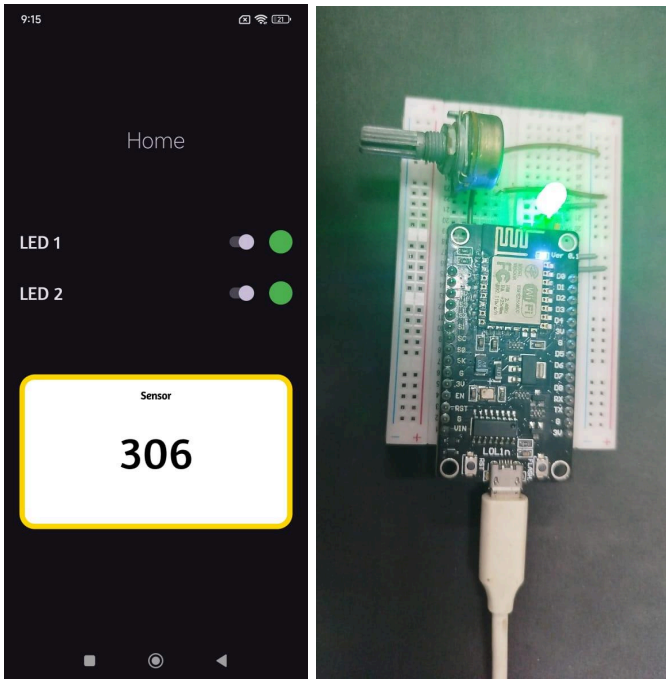
Figura 4 – Tela do aplicativo exibindo a leitura de temperatura simulada.



Fonte: acervo pessoal, 2025.

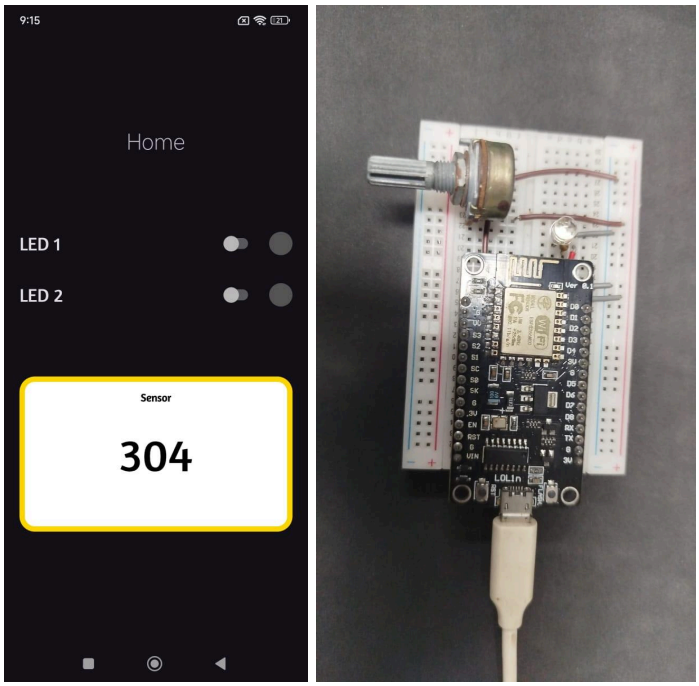
Também foram testados os comandos de controle dos LEDs. O aplicativo enviou mensagens MQTT para o broker, que foram recebidas pelo ESP8266 e interpretadas pela função callback. Os LEDs acenderam e apagaram conforme os comandos enviados, demonstrando o funcionamento correto da comunicação bidirecional.

Figura 5 – LEDs ligados durante o teste do comando enviado pelo aplicativo.



Fonte: acervo pessoal, 2025.

Figura 6 – LEDs desligado após comando remoto enviado pelo aplicativo.

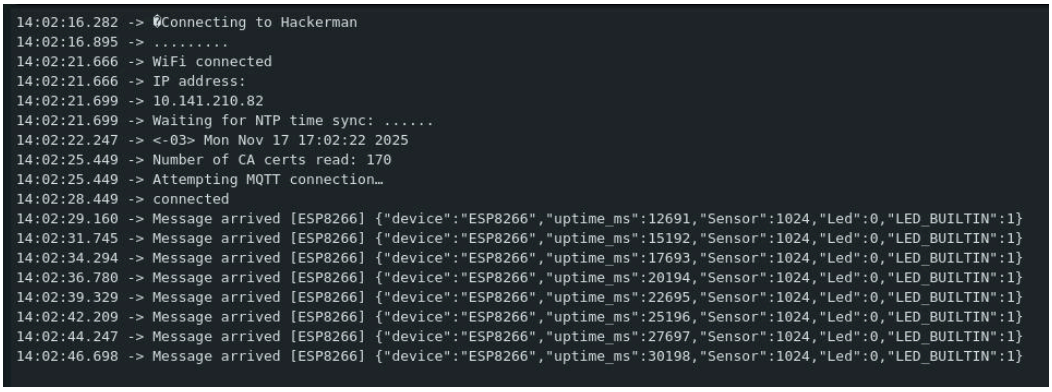


Fonte: acervo pessoal, 2025.

Em seguida, foram avaliadas a estabilidade da conexão e a reconexão automática. Durante os testes, o microcontrolador manteve a comunicação ativa com o broker, e, quando a conexão foi interrompida manualmente, a função reconnect

conseguiu restaurá-la e retomar a publicação dos dados.

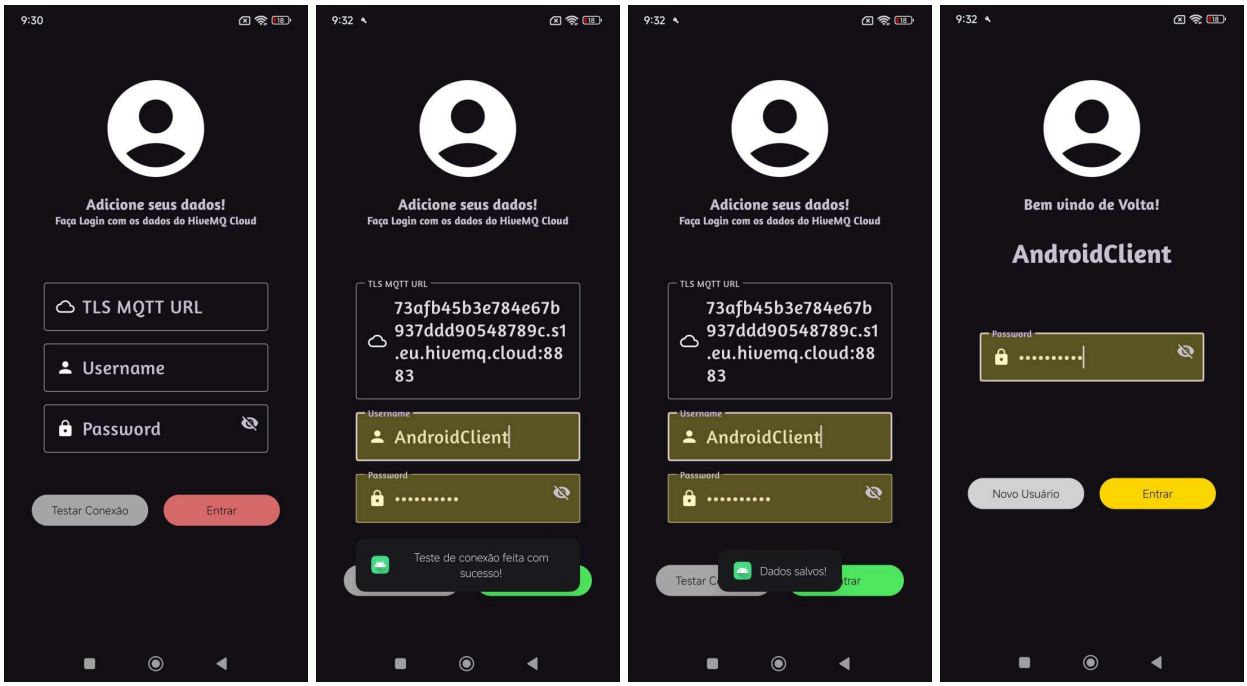
Figura 7 – Tela de confirmação de conexão do microcontroller com o MQTT.



Fonte: acervo pessoal, 2025.

Também foram realizados testes do fluxo interno do aplicativo, verificando o cadastro de usuário, validação de credenciais e armazenamento das informações no SQLite. A navegação entre telas ocorreu corretamente, e as credenciais salvas foram recuperadas sem falhas na função AccessAccount.

Figura 8 – Tela de cadastro ou login do aplicativo Android.



Fonte: acervo pessoal, 2025.

De modo geral, os testes confirmaram que o sistema é capaz de monitorar a temperatura simulada e controlar os LEDs de maneira remota, apresentando respostas rápidas, comunicação estável e funcionamento adequado das funções implementadas tanto no ESP8266 quanto no aplicativo Android.

5. CONCLUSÃO E POSSÍVEIS MELHORIAS

O projeto permitiu demonstrar, de forma prática, o funcionamento de um sistema IoT completo, integrando microcontrolador, comunicação em nuvem e desenvolvimento mobile. O protótipo atendeu plenamente aos objetivos estabelecidos, permitindo a leitura de temperatura simulada, o envio periódico desses dados a um broker MQTT seguro, bem como o controle remoto dos LEDs por meio do aplicativo Android desenvolvido. A integração entre hardware, firmware e software evidenciou a viabilidade e eficiência do uso de MQTT em aplicações de monitoramento e automação.

5.1. POSSÍVEIS MELHORIAS

Embora o protótipo tenha cumprido todos os requisitos propostos, existem algumas oportunidades de aprimoramento que podem tornar o sistema mais robusto, completo e próximo de uma aplicação real. Entre as melhorias possíveis, destacam-se:

- Substituir o potenciômetro por um sensor real de temperatura, como DHT11, DHT22 ou LM35.
- Implementar um dashboard web para monitoramento remoto em tempo real.
- Utilizar criptografia mais robusta e autenticação baseada em tokens para aumentar a segurança.
- Implementar gráficos no aplicativo para exibir histórico de dados coletados.
- Adicionar notificações push para alertas quando valores ultrapassarem limites pré-estabelecidos.

Os códigos referente à aplicação desenvolvida neste projeto estão disponíveis para consulta no seguinte repositório do GitHub: github.com/Hypothasis.

REFERÊNCIAS BIBLIOGRÁFICAS

BOCA, L. L. et al. An IoT System Proposed for Higher Education: Approaches and Challenges in Economics, Computational Linguistics, and Engineering. Sensors, v. 23, n. 14, p. 6272, 2023. Disponível em: mdpi.com/1424-8220/23/14/6272. Acesso em: 16 nov. 2025.

IBM. O que é IoT (internet das coisas)? Disponível em: ibm.com/internet-of-things. Acesso em: 16 nov. 2025.

CURVELLO, André. Apresentando o módulo ESP8266. Embarcados, 2015. Disponível em: modulo ESP8266. Acesso em: 16 nov. 2025.

ROISENBERG, Leandro. MQTT: A Linguagem Universal da Internet das Coisas (IoT). LRI, 2023. Disponível em: MQTT a linguagem universal da IoT. Acesso em: 16 nov. 2025.

Pacete, Luiz Gustavo. IoT: até 2025, mais de 27 bilhões de dispositivos estarão conectados. Forbes Brasil, 2022. Disponível em: IoT: até 2025, mais de 27 bilhões de dispositivos. Acesso em: 16 nov. 2025.