Instituto Federal de Educação, Ciência e Tecnologia de
Campus Betim
Engenharia de Controle e Automação

Lucas Henrique dos Anjos Oliveira

CONFIGURAÇÃO E INTEGRAÇÃO DO MÓDULO ESP-8285 COM MICROCONTROLADOR STM32 PARA COMUNICAÇÃO WI-FI DE DISPLAY LCD VIA INTERFACE WEB LOCAL

Betim

2026

Lucas Henrique dos Anjos Oliveira

CONFIGURAÇÃO E INTEGRAÇÃO DO MÓDULO ESP-8285 COM MICROCONTROLADOR STM32 PARA COMUNICAÇÃO WI-FI DE DISPLAY LCD VIA INTERFACE WEB LOCAL

Trabalho de conclusão de curso apresentado ao colegiado do curso de Engeharia de Controle e Automação do Instituto Federal de Educação, Ciência e Tecnologia de Minas Gerais, como requisito parcial para obtenção do título de Bacharel em Engenharia de Controle e Automação.

Orientador: Prof. Me. Helbert Ribeiro de Sá

Betim

2026

FICHA CATALOGRÁFICA

O48c Oliveira, Lucas Henrique dos Anjos

Configuração e integração do módulo ESP-8285 com microcontrolador STM32 para comunicação via *wi-fi* de display LCD via interface *web* local / Lucas Henrique dos Anjos Oliveira. – 2026.

57 f. : il.

Trabalho de conclusão de curso (Bacharelado em Engenharia de Controle e Automação) - Instituto Federal de Educação, Ciência e Tecnologia de Minas Gerais, Câmpus Betim, 2026.

Orientação: Prof. Me. Helbert Ribeiro de Sá

1. Microcontroladores. 2. Sistemas de comunicação sem fio. 3. Controle. 4. Sistemas embarcados . 5. Engenharia de Controle e Automação. I. Oliveira, Lucas Henrique dos Anjos. II. Título.

CDU: 681.51

Ficha catalográfica elaborada pelo Bibliotecário Denísio Pereira Marcos CRB-6/3142

MINISTÉRIO DA EDUCAÇÃO
SECRETARIA DE EDUCAÇÃO PROFISSIONAL E TECNOLÓGICA
INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DE MINAS GERAIS
Campus Betim
Diretoria de Ensino
Docentes Automação Industrial e Tecnologia da Informação
Rua Itamarati - CEP 32677-564 - Betim - MG
3135976360 - www.ifmg.edu.br

ATA DE DEFESA DE TRABALHO DE CONCLUSÃO DE CURSO

Aos 22 dias do mês de Junho do ano de 2026, às dezessete horas, nas dependências do IFMG – Campus Betim, reuniu-se a banca examinadora presidida por mim, **Helbert Ribeiro de Sá** e demais membros, **Deliene Costa Guimarães Barros** e **Alyson Geraldo Guimarães Cardoso**. Nesta ocasião o discente **Lucas Henrique dos Anjos Oliveira** do curso de Bacharelado em Engenharia de Controle e Automação, com registro acadêmico de número 0034868 do IFMG – Campus Betim, defendeu seu Trabalho de Conclusão de Curso (TCC) intitulado "**Configuração e Integração do Módulo ESP-8285 com Microcontrolador STM32 para Comunicação Wi-Fi de Display LCD via Interface Web Local**" e foi **APROVADO**, com **95 (Noventa e cinco)** pontos.

Este resultado reflete o cumprimento parcial dos critérios de avaliação estabelecidos pelo curso e reconhece os esforços e a dedicação do discente e seu orientador no desenvolvimento do seu TCC. O lançamento da nota e o consequente encerramento do respectivo processo está condicionado ao cumprimento dos procedimentos pós-defesa conforme previstos nos regulamentos vigentes. Tais procedimentos pós-defesa devem ser finalizados dentro do prazo limite de 10 dias, a contar da data desta ata. O descumprimento destes procedimentos até a data estipulada implicará em atribuição de nota 0 (zero) e consequente reprovação.

A sessão foi encerrada às dezenove horas. Para constar, eu,**Helbert Ribeiro de Sá**, redigi a presente ata, foi aprovada e assinada pelos membros da banca examinadora.

Betim, 02 de julho de 2026.

Documento assinado eletronicamente por **Helbert Ribeiro de Sa**, **Professor**, em 02/07/2026, às 12:57, conforme Decreto nº 10.543, de 13 de novembro de 2020.

Documento assinado eletronicamente por **Deliene Costa Guimaraes Barros**, **Testemunha**, em 02/07/2026, às 14:49, conforme Decreto nº 10.543, de 13 de novembro de 2020.

Documento assinado eletronicamente por **Alyson Geraldo Guimarães Cardoso**, **Professor Substituto**, em 02/07/2026, às 18:04, conforme Decreto nº 10.543, de 13 de novembro de 2020.

23792.000802/2026-93	2792649v1

Dedico este trabalho a todas as pessoas que me apoiaram e acreditaram em mim durante esta jornada acadêmica. Em especial, aos meus pais, que sempre me incentivaram a buscar o conhecimento e nunca desistir dos meus sonhos. Ao meu orientador, cuja orientação e paciência foram fundamentais para a realização deste projeto.

Agradecimentos

Agradeço esse trabalho em especial aos meus pais, que sempre me apoioaram na busca pelos estudos, o meu orientador Helbert Ribeiro de Sá, pela paciência e ensinamentos no desenvolvimento do trabalho, a todos os meus professores e meus amigos companheiros de curso, que passamos pelas dificuldades e aprovações sempre juntos.

"A morte do homem começa no instante em que ele desiste de aprender!"
Albino Teixeira

Resumo

Este trabalho tem como objetivo o desenvolvimento e a validação de um sistema de comunicação entre um microcontrolador STM32F103 e o módulo Wi-Fi ESP8285, utilizando comandos AT como forma de configuração e controle do dispositivo. A proposta visa compreender e aplicar os conceitos fundamentais de comunicação serial UART, amplamente utilizados em sistemas embarcados e aplicações de Internet das Coisas (IoT).

O projeto visa demonstrar que apesar da relevância dos módulos Wi-Fi no contexto atual de conectividade, observa-se que a abordagem por comandos AT ainda é pouco explorada no meio acadêmico. Os projetos contemporâneos tendem a privilegiar a programação interna direta de módulos, utilizando plataformas prototipadores como Arduino e ESP32 com SDKs nativos. Diante disso, é proposto uma solução acessível, documentada e replicável, que demonstra ser possível estabelecer conectividade Wi-Fi sem a necessidade de reprogramar o firmware do módulo, utilizando apenas a interface UART e comandos AT padronizados.

Os testes realizados demonstraram que o uso de comandos AT é uma solução viável para a configuração e o controle do módulo ESP8285, sendo possível estabelecer comunicação confiável entre os dispositivos e controlar remotamente um display LCD 16x2 por meio de uma interface web local. Os resultados obtidos confirmam a aplicabilidade do sistema proposto em projetos embarcados e servem como base para aplicações futuras voltadas à conectividade e automação.

Palavras-chave: UART. IoT, ESP8285, ESP-M2, Display Python, WebSocket ARM.

Abstract

This work aims to develop and validate a communication system between an STM32F103 microcontroller and an ESP8285 Wi-Fi module, using AT commands for device configuration and control. The proposal aims to understand and apply the fundamental concepts of UART serial communication, widely used in embedded systems and Internet of Things (IoT) applications.

The project aims to demonstrate that, despite the relevance of Wi-Fi modules in the current connectivity context, the AT command approach is still underexplored in academia. Contemporary projects tend to favor direct internal programming of modules, using prototyping platforms such as Arduino and ESP32 with native SDKs. Therefore, an accessible, documented, and replicable solution is proposed, demonstrating that it is possible to provide Wi-Fi connectivity without the need to reprogram the module's firmware, using only the UART interface and standardized AT commands.

The tests performed demonstrated that the use of AT commands is a viable solution for configuring and controlling the ESP8285 module, making it possible to establish reliable communication between devices and remotely control a 16x2 LCD display via a local web interface. The results obtained confirm the applicability of the proposed system in embedded projects and serve as a basis for future applications focused on connectivity and automation.

Keywords: UART. IoT, ESP8285, ESP-M2, Display, Python, WebSocket, ARM.

Lista de Figuras

Lista de abreviaturas e siglas

Trabalho Final de Curso - TFC
Trabalho de Conclusão de Curso - TCC
Comandos Attention - Comandos AT
Microcontrolador STM32F103 - STM32
Módulo Wi-Fi ESP-MS - ESP-M2

Sumário

1 INTRODUÇÃO

O avanço das tecnologias de comunicação e a crescente demanda por sistemas conectados têm impulsionado o desenvolvimento de soluções baseadas em sistemas embarcados, especialmente no contexto da Internet das Coisas (IoT). Nesse cenário, torna-se cada vez mais comum a necessidade de integração entre microcontroladores e módulos de comunicação sem fio, como os módulos Wi-Fi, permitindo que dispositivos embarcados se conectem a redes locais e à internet.

Entre os módulos Wi-Fi amplamente utilizados em aplicações acadêmicas e industriais, destaca-se o ESP-M2 com chip ESP8285, que reúne baixo custo, dimensões reduzidas e facilidade de integração. Esse módulo pode ser utilizado tanto de forma independente quanto em conjunto com microcontroladores externos, sendo frequentemente configurado por meio de comandos AT enviados via comunicação serial. Essa abordagem permite que a comunicação do módulo seja realizada sem a necessidade de reprogramação direta de seu firmware, o que simplifica o desenvolvimento inicial de projetos.

Entretanto, apesar da ampla disponibilidade de documentação e exemplos relacionados à comunicação entre microcontroladores e módulos Wi-Fi, muitos projetos ainda utilizam abordagens mais complexas de integração. Nesse contexto, a utilização do módulo ESP por meio de comandos AT apresenta uma alternativa mais prática e eficiente, permitindo realizar a parametrização da comunicação, configuração de rede e troca de dados de forma direta, sem a necessidade de desenvolvimento completo do firmware interno do módulo. Essa abordagem proporciona maior simplicidade de implementação, redução do tempo de desenvolvimento e facilidade na integração entre hardware e software.

Diante desse contexto, este trabalho propõe o desenvolvimento e aplicação de testes de funcionamento em um sistema embarcado baseado no microcontrolador STM32F103C8 e no módulo Wi-Fi ESP-M2 com chip ESP8285, utilizando comandos AT como meio de configuração e controle do módulo. O foco do projeto está na implementação da comunicação serial UART entre os dispositivos, na validação do envio e da recepção de comandos AT e na posterior configuração do módulo para operação em ambiente de rede, conforme descrito em um guia prático de referência.

Assim, o problema que orienta este trabalho pode ser definido da seguinte forma: como estabelecer, validar e documentar de maneira prática e confiável a comunicação entre um microcontrolador STM32F103 e o módulo ESP-M2 utilizando comandos AT e interface Web, considerando as limitações típicas encontradas em ambientes acadêmicos e de prototipagem?

A partir da resolução desse problema, busca-se fornecer uma base técnica sólida para a utilização do módulo ESP8285 em projetos embarcados, contribuindo para a formação acadêmica na área de sistemas embarcados e conectividade.indústria.

1.1 Colocação do problema

O avanço tecnológico nas áreas de automação e sistemas embarcados tem ampliado a utilização de dispositivos capazes de realizar transmissão e exibição de dados em tempo real de forma prática e acessível. Em diversos contextos acadêmicos e tecnológicos, torna-se necessário desenvolver soluções que integrem transmissão, processamento e exibição de informações, permitindo a interação entre microcontroladores, módulos Wi-Fi e interfaces de usuário. Nesse cenário, a utilização de módulos ESP configurados por comandos AT apresenta uma alternativa eficiente para implementação de sistemas de transmissão sem fio com baixo custo e relativa simplicidade de integração.

Entretanto, muitas soluções disponíveis comercialmente apresentam custo elevado, principalmente em aplicações que utilizam sistemas proprietários ou módulos industriais dedicados. Durante o período da pandemia de COVID-19, observou-se um aumento significativo nos preços de componentes eletrônicos em função da escassez global de semicondutores. Microcontroladores de 8 bits, que anteriormente eram encontrados na faixa de R$80,00, continuaram a ser comercializados por valores iguais ou superiores a R$ 80,00 em determinados períodos. Apesar desse cenário, microcontroladores de 32 bits da família STM32 mantiveram-se aproximadamente entre 5 e 10% os preços de microcontroladores de 8 bits, tendo uma boa relação custo-benefício e ampla disponibilidade no mercado, permanecendo posteriormente a faixas de preço mais acessíveis, atualmente entre aproximadamente R$ 5,00 e R$ 15,00 em plataformas de distribuição e lojas especializadas, dependendo do modelo e fornecedor. De forma semelhante, módulos Wi-Fi da família ESP, como o ESP-M2 baseado no ESP8285, continuam sendo comercializados por valores reduzidos, geralmente inferiores a R$ 30,00, tornando-se alternativas viáveis para aplicações embarcadas de baixo custo. Nesse contexto, a utilização de módulos

ESP configurados por comandos AT possibilita o desenvolvimento de sistemas voltados à transmissão e exibição de dados com relativa simplicidade de integração entre hardware, firmware e comunicação em rede.

1.2 Objetivos

1.2.1 Geral

Como alternativa a esse alto custo e à limitada adaptabilidade das soluções comerciais, o presente projeto propõe o desenvolvimento de um sistema embarcado microcontrolado para monitoramento e exibição de variáveis, integrando hardware dedicado, firmware embarcado e interface de visualização. A proposta busca demonstrar que é possível implementar uma solução funcional, eficiente e tecnicamente fundamentada, aplicando conceitos de sistemas embarcados, comunicação digital e projeto de placas de circuito impresso.

1.2.2 Específicos

- Apresentar o esquemático do sistema, definindo os componentes necessários para utilização, processamento e exibição das variáveis monitoradas.
- Utilização de um Kit ARM com microcontrolador STM32F103C8 onde foi desenvolvido um shield para ampliação de aplicações.
- Configurar e programar o microcontrolador utilizando ferramentas de desenvolvimento apropriado como o STMCube32 e o Keil, habilitando seus periféricos internos conforme a necessidade do projeto.
- Implementar a comunicação digital entre microcontrolador e o display por meio do protocolo bit a bit em paralelo.
- Desenvolver a apresentação da comunicação de um sistema Websocket para demonstração do funcionamento do sistema.

2 REVISÃO BIBLIOGRÁFICA

A fundamentação teórica para o desenvolvimento de um sistema embarcado microcontrolado para monitoramento e exibição de variáveis envolve a compreensão integrada de diferentes áreas da engenharia eletrônica e da computação embarcada. O projeto de sistemas desse tipo exige conhecimento sobre arquitetura de sistemas embarcados, funcionamento de microcontroladores, protocolos de comunicação digital, aquisição e processamento de sinais, além de técnicas de projeto de placas de circuito impresso. Nos últimos anos, os sistemas embarcados têm evoluído significativamente, tornando-se mais compactos, eficientes e acessíveis, possibilitando sua aplicação em contextos industriais, automotivos, médicos e acadêmicos. Nesse cenário, tecnologias como conversores analógico-digitais, interfaces seriais de comunicação (UART, SPI e I2C), displays digitais e ferramentas de configuração de periféricos desempenham papel fundamental na construção de soluções integradas e de baixo custo. Dessa forma, a compreensão desses conceitos é essencial para fundamentar as decisões técnicas adotadas no desenvolvimento do presente projeto.

2.1 Sistemas Embarcados

Os sistemas embarcados podem ser definidos como sistemas computacionais dedicados ao desempenho de funções específicas dentro de um equipamento ou aplicação maior. Diferentemente dos computadores de propósito geral, esses sistemas são desenvolvidos para executar tarefas determinadas, normalmente com requisitos de confiabilidade, baixo consumo de energia e integração direta com hardware eletrônico. Segundo Almeida (2019) [13], sistemas embarcados estão presentes em diversos dispositivos do cotidiano, como equipamentos industriais, sistemas automotivos, eletrodomésticos e dispositivos de comunicação, sendo fundamentais para aplicações de automação e controle.

A estrutura de um sistema embarcado é composta, geralmente, por um microcontrolador ou microprocessador, interfaces de entrada e saída, módulos de comunicação e firmware dedicado à aplicação proposta. De acordo com Valvano (2017)[15], a principal característica desses sistemas é a forte integração entre hardware e software, uma vez que

o firmware é desenvolvido especificamente para controlar os recursos físicos disponíveis no dispositivo. Essa integração permite maior eficiência operacional e melhor desempenho em aplicações específicas, principalmente em sistemas que utilizam comunicação serial, redes Wi-Fi e interfaces de exibição de dados.

Outro aspecto importante dos sistemas embarcados é a utilização crescente de microcontroladores de 32 bits em aplicações modernas. O avanço dessas arquiteturas permitiu o desenvolvimento de sistemas mais robustos e flexíveis, capazes de integrar múltiplos periféricos e protocolos de comunicação em um único dispositivo. Segundo Pereira (2021)[14], microcontroladores baseados em arquitetura ARM vêm sendo amplamente utilizados em projetos acadêmicos e industriais devido à sua capacidade de processamento, baixo custo e ampla disponibilidade de ferramentas de desenvolvimento, tornando-se alternativas eficientes para aplicações embarcadas conectadas em rede.

No contexto deste trabalho, o sistema embarcado desenvolvido utiliza o microcontrolador STM32F103C8T6 integrado ao módulo Wi-Fi ESP-M2 para realização da comunicação e exibição de mensagens em um display LCD 16x2. Dessa forma, o projeto demonstra a aplicação prática dos conceitos de sistemas embarcados, envolvendo integração entre hardware, firmware e comunicação em rede, além da utilização de ferramentas modernas para configuração e desenvolvimento do sistema.

2.2 Comunicação Serial, Comunicação Paralela e Comandos AT em Sistemas Embarcados

A comunicação entre dispositivos eletrônicos é um dos principais elementos dos sistemas embarcados, permitindo a troca de informações entre microcontroladores, módulos de comunicação e interfaces de exibição. Segundo Valvano (2017)[15], os sistemas embarcados modernos dependem diretamente de protocolos de comunicação para integração entre hardware e software, sendo a comunicação serial uma das mais utilizadas devido à simplicidade de implementação e à redução da quantidade de conexões físicas. Protocolos como UART, SPI e I2C são amplamente empregados em aplicações embarcadas, principalmente em sistemas que utilizam módulos de comunicação sem fio e periféricos externos.

A comunicação serial UART foi utilizada neste trabalho para integração entre o microcontrolador STM32 e o módulo Wi-Fi ESP-M2. Diferentemente da comunicação paralela, na qual múltiplos bits são transmitidos simultaneamente utilizando diversas linhas de

dados, a UART realiza a transmissão sequencial dos dados utilizando apenas linhas de transmissão (TX) e recepção (RX). De acordo com Pereira (2021)[14], essa abordagem reduz significativamente a complexidade do hardware e facilita a comunicação entre dispositivos embarcados. Já a comunicação paralela foi utilizada na interface com o display LCD 16x2 em modo de 4 bits, permitindo o envio de dados diretamente ao display por meio dos sinais e linhas de dados dedicadas.

Outro conceito importante aplicado neste projeto é a utilização de comandos AT (Attention Commands), amplamente empregados em módulos de comunicação como ESP8266 e ESP8285. Os comandos AT consistem em instruções textuais enviadas via comunicação serial com o objetivo de configurar, parametrizar e controlar dispositivos eletrônicos. Segundo a Espressif Systems (2023)[16], comandos como AT, utilizado para verificar comunicação com o módulo, AT+CWMODE, responsável pela configuração do modo de operação Wi-Fi, e AT+CWJAP, utilizado para conexão em uma rede sem fio, estão entre os principais comandos utilizados em aplicações com módulos ESP. A utilização desses comandos permite controlar o módulo sem necessidade de desenvolvimento de firmware próprio para o dispositivo Wi-Fi.

Apesar das inúmeras vantagens proporcionadas pelos comandos AT, muitos desenvolvedores e estudantes acabam não utilizando essa abordagem devido à predominância de conteúdos voltados à programação direta do firmware do ESP utilizando Arduino IDE ou ESP-IDF. Entretanto, a utilização de comandos AT simplifica significativamente a integração entre microcontroladores e módulos Wi-Fi, reduzindo a complexidade do desenvolvimento e permitindo maior foco na lógica principal do sistema embarcado. No contexto deste trabalho, essa abordagem possibilitou realizar a configuração da comunicação Wi-Fi e transmissão de mensagens de forma mais prática, eficiente e compatível com a arquitetura proposta. Abaixo, a Figura 1 detalha alguns exemplos de Comandos AT que podem ser utilizados para parametrização e configuração dos módulos ESP. Comandos desse tipo, são padronizados pelos fabricantes para facilitar tanto as configurações iniciais quanto confirmação de parâmetros estabelecidos de funcionamento para um sistema embarcado. O que possibilita um ganho em diagnóstico de falhas e configurações rápidas sem necessidade de acesso a programação interna do módulo.

Figura 1: Comandos AT

Direção dos dados	Comando (ASCII string)	Descrição
Checando status do modo STA.		
MCU➔Módulo Wifi	AT+STASTATUS	Checa o status do modo STA
Módulo Wifi➔MCU	STA:OK	O módulo respondeu que a conexão foi bem sucedida.
Módulo Wifi➔MCU	STA:DOWN	O módulo respondeu que a conexão não foi bem sucedida
Obter IP e MAC no modo STA		
MCU➔Módulo Wifi	AT+STAINFO	Obtém IP e MAC do módulo WiFi
Módulo Wifi➔MCU	Exemplo: Mac|IP, 5CCF7F116380|192.168.1. 125	O módulo responde com os dados de IP e MAC.
Verifica o status da conexão no modo cliente TCP.		
MCU➔Módulo Wifi	AT+TCPCLIENT	Checa o status da conexão no modo cliente TCP.
Módulo Wifi➔MCU	TCP:OK	O módulo respondeu que o cliente TCP está conectado.
Módulo Wifi➔MCU	TCP:OFF	O módulo respondeu que o cliente TCP não está conectado.
Reiniciar módulo		
MCU➔Módulo Wifi	AT+RST	Reiniciar módulo WiFi
Módulo Wifi➔MCU	RST:OK	O módulo respondeu que foi reiniciado corretamente.
Restaurar parâmetros de fábrica		
MCU➔Módulo Wifi	AT+RESTORE	Restaurar parâmetros de fábrica
Módulo Wifi➔MCU	RESTORE:OK	O módulo respondeu que os parâmetros foram restaurados para os de fábrica.

Fonte: Elaborado pelo autor, 2025

2.3 Interfaces de exibição de dados

Em sistemas embarcados, as interfaces de exibição de dados são responsáveis por apresentar informações ao usuário de forma clara e organizada, permitindo a visualização de mensagens, estados do sistema e dados processados pelo microcontrolador. Segundo Almeida (2019)[13], a utilização de interfaces visuais em sistemas embarcados contribui diretamente para a interação homem-máquina, tornando o sistema mais intuitivo e facilitando a interpretação das informações exibidas.

Entre os dispositivos mais utilizados para essa finalidade destacam-se os displays LCD alfanuméricos, principalmente o modelo LCD 16x2, amplamente empregado em aplicações

acadêmicas e industriais devido ao seu baixo custo, simplicidade de integração e reduzido consumo de energia. Esse display possui duas linhas com capacidade para dezesseis caracteres cada, permitindo a exibição de mensagens, valores numéricos e informações operacionais do sistema. De acordo com Pereira (2021), displays baseados no controlador HD44780 tornaram-se padrão em aplicações embarcadas devido à facilidade de comunicação com diferentes microcontroladores.

No presente trabalho, o display LCD 16x2 foi utilizado como principal interface de exibição de dados, sendo controlado diretamente pelo microcontrolador STM32 por meio de comunicação paralela em modo de 4 bits. Nesse método, os dados são transmitidos em duas etapas utilizando apenas quatro linhas de dados, reduzindo a quantidade de conexões necessárias em relação ao modo convencional de 8 bits. Além das linhas de dados, também são utilizados sinais de controle responsáveis pelo sincronismo e seleção das informações enviadas ao display.

A utilização do display LCD permitiu realizar a exibição das mensagens recebidas via comunicação serial e rede Wi-Fi, demonstrando a integração entre o microcontrolador STM32, o módulo ESP-M2 e a interface visual do sistema. Dessa forma, a interface de exibição contribuiu para validação prática da comunicação entre os dispositivos e para melhor interação do usuário com o sistema embarcado desenvolvido.

De acordo com o esquemático do sistema desenvolvido , a comunicação com o display é realizada por meio de sinais de controle e linhas de dados específicas. Os principais pinos utilizados são:

- RS (Register Select): responsável por definir se o dado enviado corresponde a um comando ou a um caractere a ser exibido;
- EN (Enable): utilizado para sincronizar o envio dos dados, indicando ao display o momento em que a informação deve ser lida;
- D4 a D7: linhas de dados utilizadas no modo de 4 bits, responsáveis pela transmissão dos dados de forma segmentada;
- VCC e GND: responsáveis pela alimentação do display;
- V0 (contraste): utilizado para ajuste do contraste dos caracteres exibidos.

2.4 Ferramentas de Desenvolvimento

O desenvolvimento de sistemas embarcados depende diretamente da utilização de softwares capazes de auxiliar na configuração, programação e validação do firmware. Segundo Valvano (2017)[15], essas ferramentas contribuem para reduzir erros de implementação, facilitar a integração entre hardware e software e aumentar a produtividade no desenvolvimento de aplicações embarcadas. No presente trabalho, foram utilizadas ferramentas específicas para configuração do microcontrolador STM32, desenvolvimento do firmware e validação da comunicação serial com o módulo ESP-M2.

A configuração inicial do microcontrolador foi realizada por meio do STM32CubeMX, cuja caracterização e funcionamento são detalhados na Seção 3.3. Após a parametrização e a geração automática do código-base, o firmware foi desenvolvido no ambiente Keil uVision5, utilizado para compilação, gravação e depuração do código, conforme descrito na Seção 3.5. Além dessas ferramentas, foram utilizados terminais seriais em especial o software Hercules para envio e monitoramento de dados durante os testes de comunicação entre o microcontrolador e o módulo Wi-Fi.

Outro aspecto importante deste projeto foi a utilização de comandos AT para configuração e comunicação com o módulo ESP-M2. Os comandos AT (Attention Commands) consistem em instruções enviadas via comunicação serial com o objetivo de controlar módulos de comunicação sem necessidade de programação direta do firmware interno do dispositivo. Entre os comandos utilizados, destacam-se AT, empregado para verificação da comunicação com o módulo, e AT+CWMODE, utilizado para definição do modo de operação Wi-Fi. Segundo a Espressif Systems (2023), o conjunto completo de comandos AT disponíveis para o ESP8285 abrange desde configurações de rede até a abertura de conexões TCP, sendo documentado no ESP8266 AT Instruction Set.

Dessa forma, a integração entre as ferramentas de desenvolvimento e os comandos AT permitiu maior agilidade na implementação do sistema proposto, contribuindo para a validação da comunicação serial, a integração com o módulo ESP-M2 e o desenvolvimento do firmware responsável pela exibição das mensagens no display LCD 16x2.

3 METODOLOGIA

3.1 Fluxograma dos Dados

Como ponto inicial importante para construção e execução do projeto, é necessário compreender o fluxo de informação, ou fluxo de dados. Entender esse processo torna-se imprescindível para um entendimento e execução clara de todas as etapas necessárias para conclusão do projeto da melhor maneira. Na figura abaixo vemos do ponto inicial ao ponto final da transmissão da informação, entendendo como o fluxo de dados se inicia no navegador ou site utilizado e chega até o display proporciona a capacidade de prever erros futuros ou análise para soluções de imprevistos que venham à ocorrer.

Figura 2: Microcontrolador ARM STM32 STM32F103C8T6

Fonte: Elaborado pelo autor, 2026

3.2 Recursos para desenvolvimento do Projeto.

Seguindo o fluxograma acima demonstrado, torna-se necessário toda uma gama ampla de componentes de hardware e softwares. Abaixo segue a tabela de recursos utilizados e suas respectivas disponibilidades de acesso. Através de licenças estudantis, o custo do projeto se resume ao hardware utilizado.

Figura 3: Disponibilidade de Recursos

Recursos	Disponibilidade
Microsoft Office	Disponível
STM32CubeMX	Versão Educacional (www.keil.com)
Keil uVision 5	Versão Educacional (www.st.com)
Microcontrolador ARM e PCB	Disponibilizada pelo Orientador
Display LCD 16x2 e Módulo ESP Wi-fi	Disponibilizado pelo autor

Fonte: Elaborado pelo autor, 2026

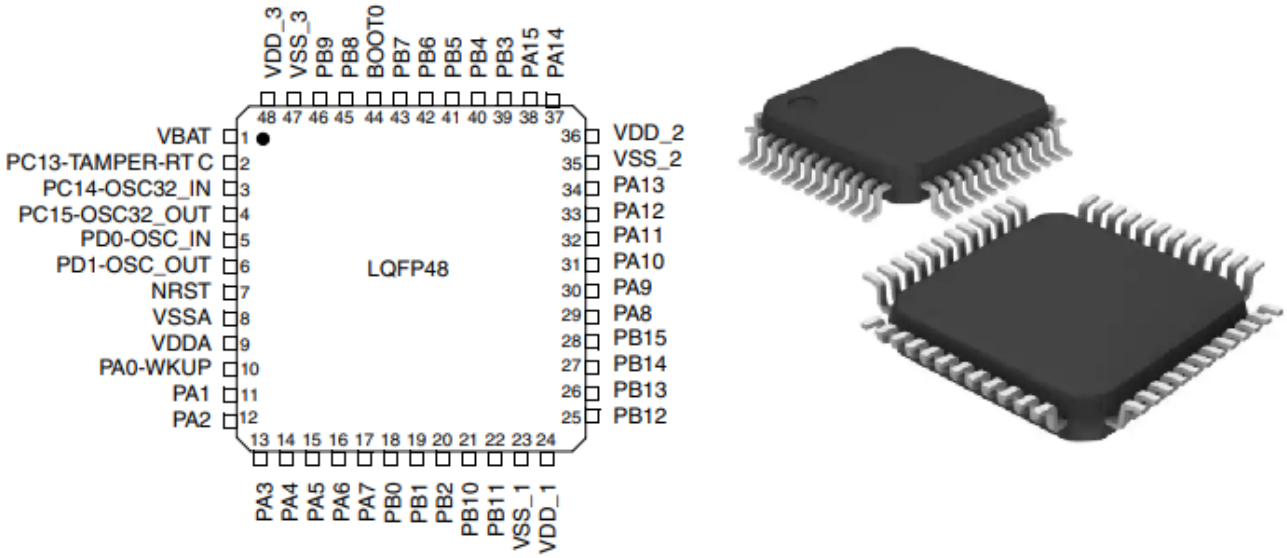
3.3 Microcontrolador STM32 Arquitetura ARM (STM32F103C8T6)

O microcontrolador utilizado no desenvolvimento deste projeto foi o STM32F103C8T6, baseado na arquitetura ARM Cortex-M3. A escolha de um microcontrolador de 32 bits ocorreu devido à crescente utilização dessa arquitetura em aplicações modernas de sistemas embarcados, principalmente em projetos que envolvem comunicação em rede, processamento de dados e integração com múltiplos periféricos. Embora microcontroladores de 8 bits sejam amplamente utilizados em ambientes acadêmicos por sua simplicidade didática, dispositivos de 32 bits oferecem maior capacidade de processamento, melhor gerenciamento de memória e maior flexibilidade para expansão do sistema.

O STM32F103C8T6 atua como unidade central de processamento do sistema, sendo responsável pelo gerenciamento da comunicação serial com o módulo ESP-M2, processamento das mensagens recebidas e controle da interface de exibição LCD em modo de 4 bits. Além disso, o microcontrolador realiza o controle dos sinais necessários para atualização das informações exibidas em tempo real.

A utilização da arquitetura ARM Cortex-M3 também contribui para uma maior eficiência na execução das tarefas e melhor integração com ambientes modernos de desenvolvimento, como STM32CubeMX e Keil uVision5. Dessa forma, a escolha do microcontrolador STM32 permitiu desenvolver um sistema mais robusto, flexível e alinhado às tecnologias atualmente utilizadas na área de sistemas embarcados.

Figura 4: Microcontrolador ARM STM32 STM32F103C8T6

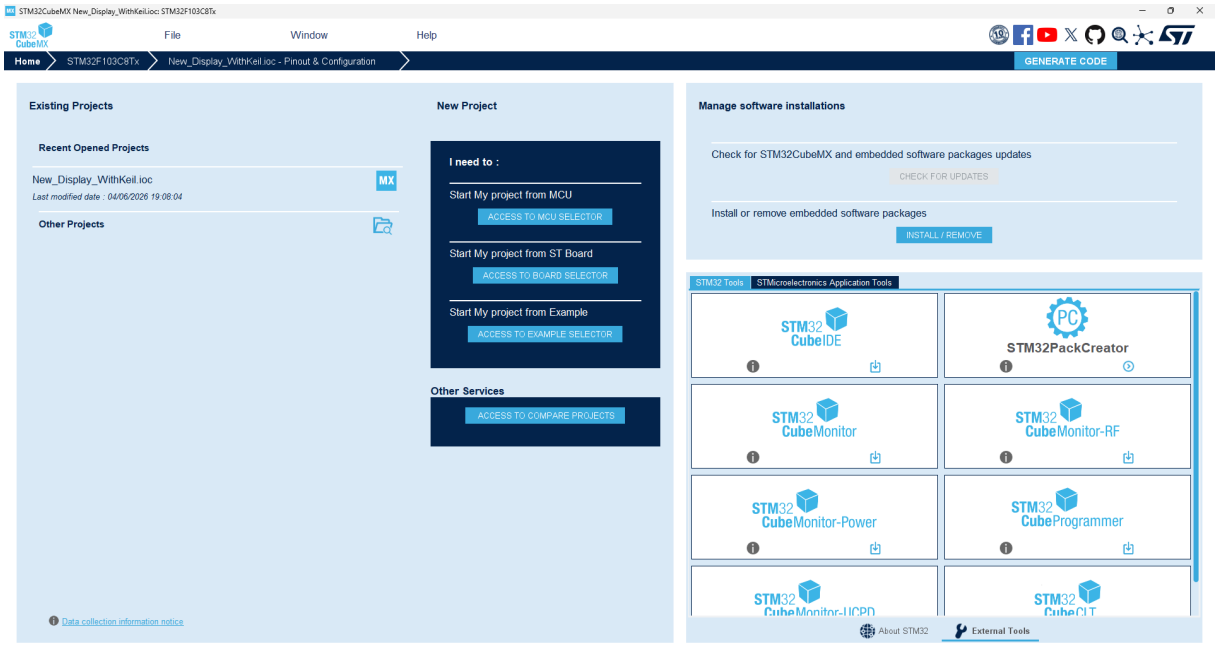


Fonte: (Utmel, 2026)

A etapa de desenvolvimento do sistema utilizou o STM32CubeMX, um software do tipo code generator desenvolvido pela STMicroelectronics para a configuração e inicialização de microcontroladores da família STM32. Esse software possui como principal objetivo simplificar o processo de desenvolvimento de sistemas embarcados, permitindo que o usuário realize a parametrização dos recursos internos do microcontrolador por meio de uma interface gráfica, reduzindo a necessidade de configuração manual diretamente em código de baixo nível.

No contexto deste trabalho, o STM32CubeMX foi utilizado como solução para facilitar a integração entre o microcontrolador STM32 e os demais módulos envolvidos no sistema, como o display LCD e o módulo Wi-Fi ESP-M2. Além de automatizar a geração da estrutura inicial do firmware, o software contribui para a redução de erros de configuração e torna o desenvolvimento mais organizado e eficiente, principalmente em aplicações que envolvem comunicação serial e múltiplas interfaces de hardware. Dessa forma, a utilização do STM32CubeMX permitiu maior agilidade na implementação do sistema proposto e melhor compatibilidade com o ambiente de desenvolvimento utilizado no projeto.

Figura 5: Ambiente STM32CubeMX



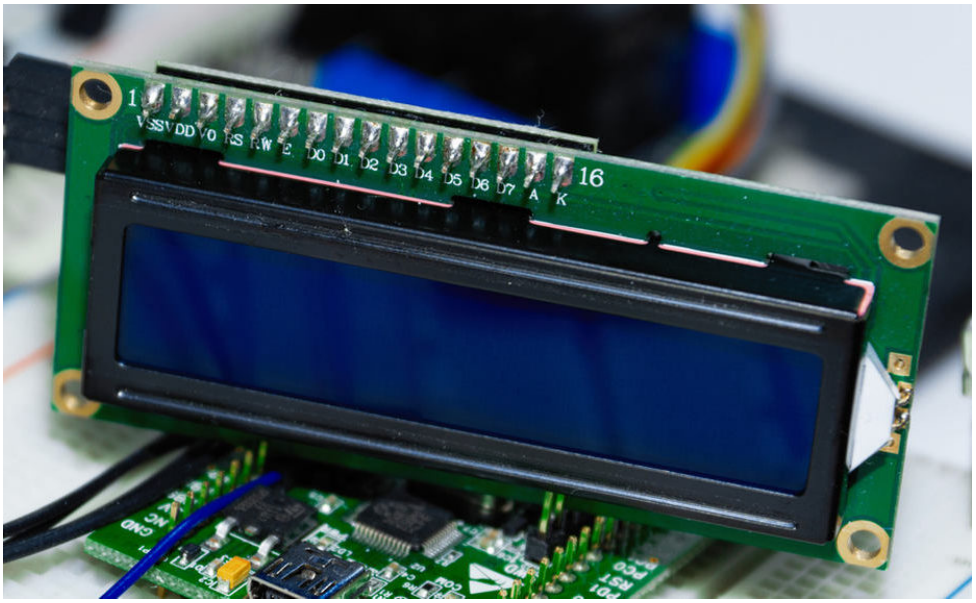
Fonte: Elaborado pelo Autor, 2026

Após a etapa de parametrização e configuração do microcontrolador, o STM32CubeMX possibilita a geração automática do projeto para diferentes ambientes de desenvolvimento. Entre as opções disponíveis, destacam-se o MDK-ARM, utilizado no ambiente Keil uVision5, e o STM32CubeIDE, permitindo ao desenvolvedor escolher a plataforma mais adequada para continuidade do desenvolvimento do firmware.

Para a exibição das variáveis monitoradas no sistema embarcado, será utilizado um display LCD 16x2, amplamente empregado em aplicações eletrônicas devido à sua simplicidade de uso, baixo custo e facilidade de integração com microcontroladores. Esse tipo de display possui duas linhas com capacidade para dezesseis caracteres cada, permitindo a apresentação clara e organizada das informações processadas pelo sistema.

Diferentemente de abordagens que utilizam comunicação serial, neste projeto o display possibilita a transmissão de informações por meio de comunicação paralela no modo de bit a bit. Nesse tipo de interface, os dados são enviados de forma segmentada, utilizando apenas quatro linhas de dados, o que reduz a quantidade de pinos necessários no microcontrolador em comparação ao modo de 8 bits. Essa escolha permitiu um melhor aproveitamento dos recursos do microcontrolador, mantendo a funcionalidade do sistema. A Figura 6 abaixo exemplifica o display tipo LCD 16x2 utilizado para a confirmação da transmissão de dados.

Figura 6: Display LCD 16x2

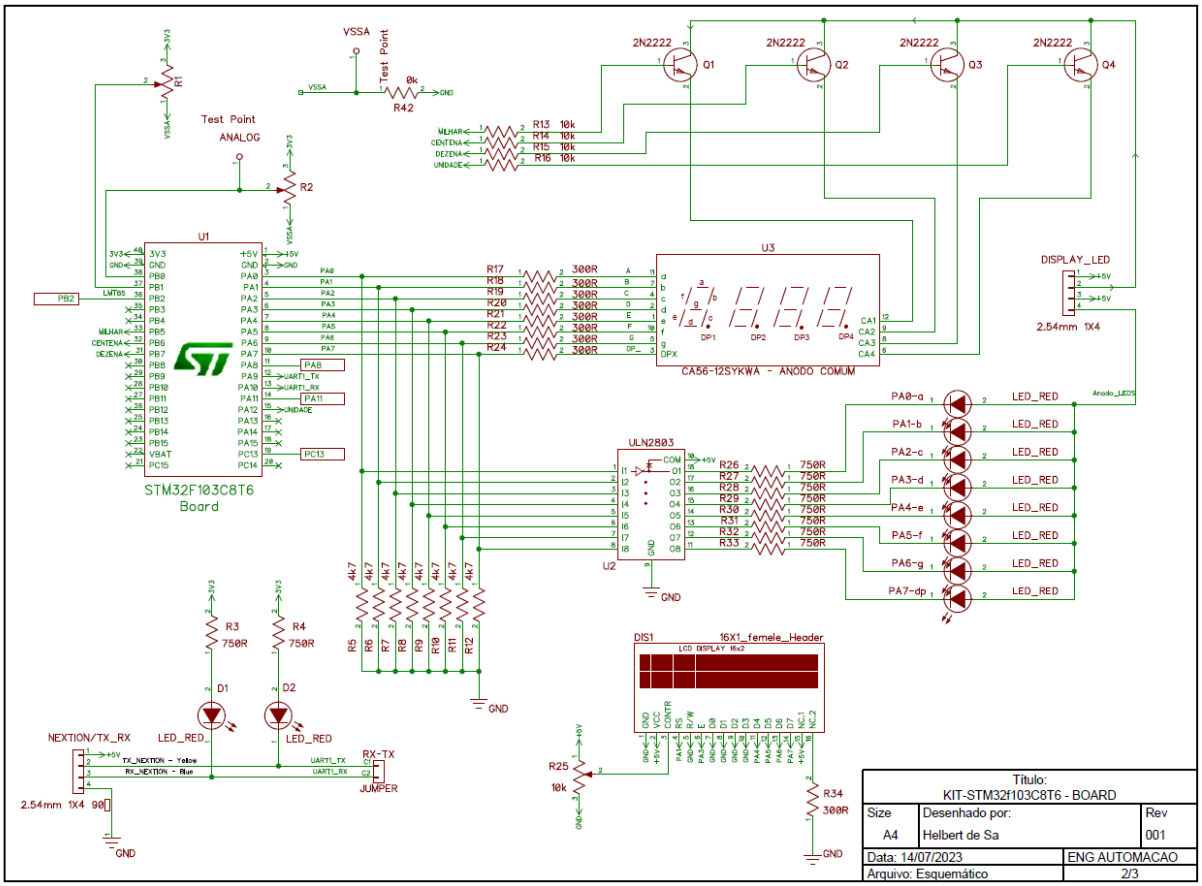


Fonte: (https://victorvision.com.br/blog/display-lcd-16x2/, 2026)

3.4 Kit de Desenvolvimento ARM

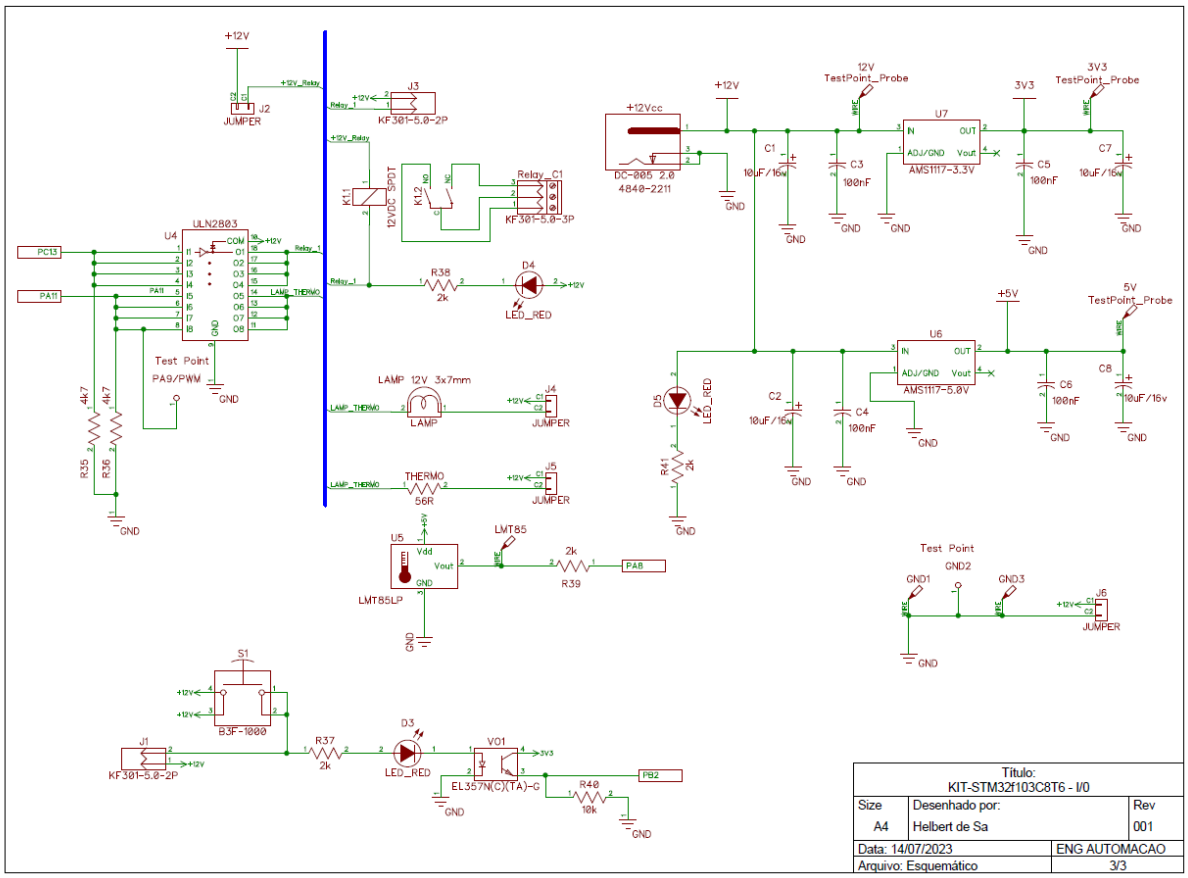
A placa de circuito impresso foi desenvolvida pelo Professor Orientador Helbert de Sá, para uso em sua disciplina de Microcontrolador com o objetivo de integrar todos os componentes eletrônicos do sistema em um único módulo físico. O uso desta shield permitiu organizar as conexões entre o microcontrolador, display e demais componentes, garantindo maior confiabilidade e estabilidade elétrica em comparação com montagens em protoboard.

Figura 7: Esquemático PCB Utilizada



Fonte: (Kit desenvolvido pelo Professor Orientador, 2025)

Figura 8: Esquemático PCB Utilizada



Fonte: (Kit desenvolvido pelo Professor Orientador, 2025)

3.5 Keil uVision 5

O desenvolvimento do firmware foi realizado no ambiente de programação Keil uVision5, um software profissional utilizado no desenvolvimento de sistemas embarcados baseados em arquitetura ARM. O ambiente oferece recursos avançados para edição de código, compilação, depuração e gravação de firmware em microcontroladores. Apesar de possuir elevado custo de licenciamento, com versões profissionais podendo atingir valores próximos a 10 mil dólares, o software é empregado em aplicações industriais e projetos profissionais devido à sua robustez, confiabilidade e elevado nível de integração com dispositivos da família STM32. No presente trabalho, a utilização do Keil uVision5 foi viabilizada por meio da licença acadêmica disponibilizada pelo IFMG.

A integração entre o STM32CubeMX e o Keil uVision5 ocorre por meio da geração automática do projeto no formato MDK-ARM, permitindo que toda a estrutura inicial do firmware configurada no STM32CubeMX seja importada diretamente para o ambi-

ente de desenvolvimento. Dessa forma, o desenvolvedor pode utilizar o código generator automaticamente e implementar as rotinas específicas do sistema, reduzindo o tempo de configuração inicial e facilitando a organização do projeto.

No ambiente Keil uVision5 foram implementadas as rotinas responsáveis pela comunicação serial com o módulo ESP-M2, processamento das mensagens recebidas e controle do display LCD 16x2 em modo de 4 bits. Além disso, o software oferece recursos avançados de depuração (debug), permitindo acompanhar simultaneamente variáveis em tempo real, monitorar registradores internos do microcontrolador, executar o código passo a passo e analisar o comportamento do firmware durante sua execução. Essas funcionalidades foram fundamentais para identificação de falhas, validação da comunicação serial e ajustes no funcionamento geral do sistema.

Além da implementação do firmware, o ambiente também foi utilizado para compilar o código e gerar o arquivo executável responsável pela gravação do programa no microcontrolador STM32.

3.6 Métodos de verificação de Funcionamento do Sistema

Durante o desenvolvimento do projeto foram utilizados métodos práticos de verificação de funcionamento com o objetivo de validar a comunicação entre os módulos e o correto comportamento do sistema embarcado. Diferentemente de aplicações industriais que exigem instrumentos específicos de medição ou processos formais de certificação, neste trabalho a comprovação do funcionamento ocorreu por meio da análise prática da execução do firmware, da comunicação serial e das respostas visuais apresentadas pelos dispositivos utilizados.

Entre os principais métodos empregados, destaca-se a utilização do recurso de debug disponível no ambiente Keil uVision5, permitindo acompanhar a execução do código em tempo real, analisar variáveis internas do microcontrolador e identificar possíveis falhas durante o desenvolvimento do firmware. Esse recurso contribuiu significativamente para validação da lógica implementada e verificação da comunicação entre o microcontrolador STM32 e o módulo ESP-M2.

Além disso, a observação dos LEDs indicadores presentes nas placas também foi utilizada como forma de comprovação do correto funcionamento do sistema. O acionamento contínuo dos LEDs de alimentação e comunicação do módulo ESP-M2 indicou que a parametrização da rede Wi-Fi e a inicialização dos módulos ocorreram corretamente, evi-

denciando estabilidade no funcionamento da comunicação serial e da conexão em rede.

4 DESENVOLVIMENTO DO PROJETO

4.1 Configuração do Microcontrolador no STM32CubeMX

A etapa de configuração do microcontrolador foi realizada utilizando a ferramenta STM32CubeMX, responsável por permitir a definição gráfica dos periféricos e pinos do dispositivo STM32F103C8T6. Essa ferramenta possibilita configurar de forma intuitiva os recursos internos do microcontrolador, além de gerar automaticamente o código base para inicialização do sistema.

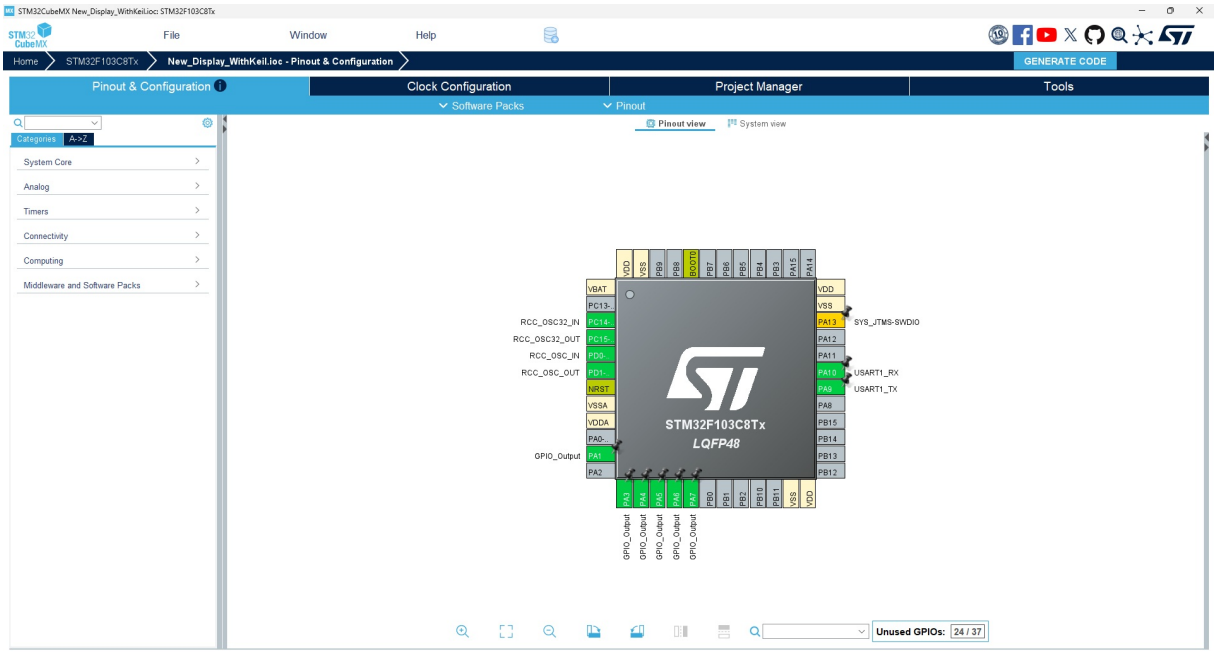
Inicialmente, foi realizada a configuração do clock do sistema, garantindo que o microcontrolador operasse na frequência adequada para execução das tarefas propostas. Em seguida, foram definidos os pinos de entrada e saída (GPIOs) utilizados no projeto, especialmente aqueles destinados à comunicação com o display LCD 16x2 no modo de 4 bits. Nessa configuração, foram atribuídos pinos específicos para os sinais de controle RS e EN, bem como para as linhas de dados D4, D5, D6 e D7, conforme o esquemático desenvolvido.

Esta ferramenta também permitiu configurar os modos de operação dos pinos como saída digital, assegurando que os sinais enviados ao display fossem corretamente interpretados. Além disso, foram ajustados parâmetros como velocidade de operação dos GPIOs, garantindo o funcionamento adequado da comunicação entre o microcontrolador e os periféricos do projeto.

Após a definição de todas as configurações necessárias, o STM32CubeMX foi utilizado para gerar automaticamente o código do projeto em linguagem C. Esse código serviu como base para o desenvolvimento do firmware no ambiente Keil uVision5, facilitando a implementação das funcionalidades do sistema e reduzindo a probabilidade de erros na configuração dos periféricos.

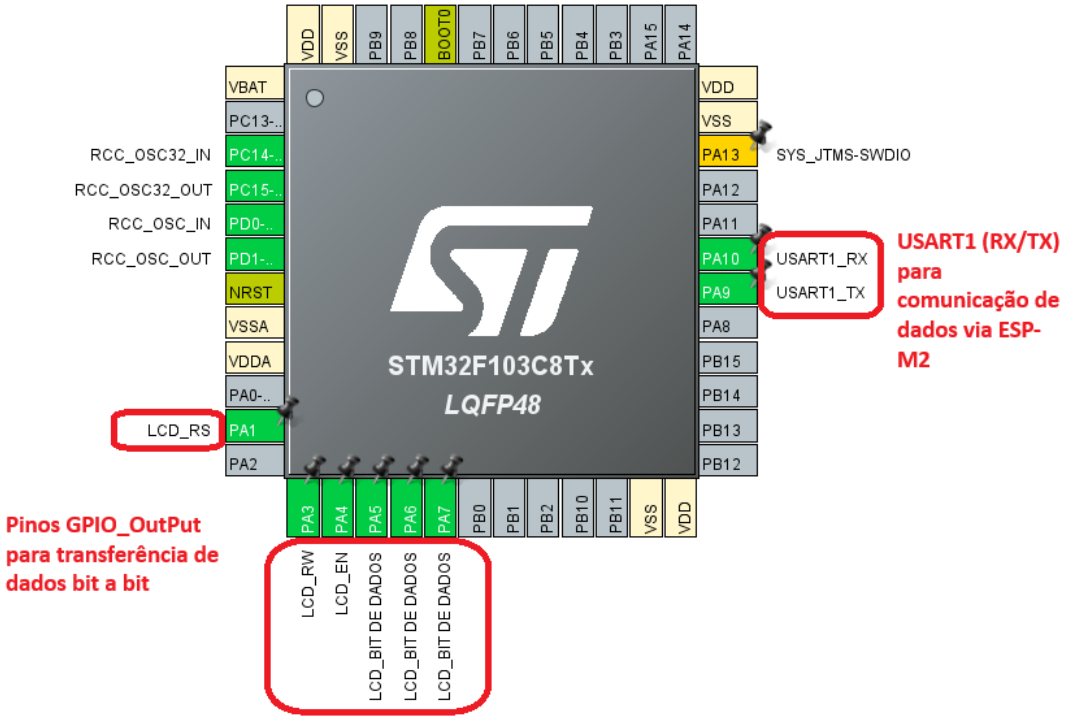
A Figura 10 abaixo apresenta a interface do Software STM32Cube MX e configuração dos pinos realizada no STM32CubeMX, evidenciando a atribuição das funções utilizadas na comunicação com o display e demais elementos do sistema.

Figura 9: Interface STM32CubeMX



Fonte: (IFMG Betim, 2025)

Figura 10: Configuração dos Pinos



Fonte: (IFMG Betim, 2025)

4.2 Desenvolvimento de Firmware no Keil uVision5

Após a configuração inicial do microcontrolador no STM32CubeMX, o desenvolvimento do firmware foi realizado no ambiente de programação Keil uVision5. Essa ferramenta foi utilizada para implementar as rotinas responsáveis pelo controle do sistema, incluindo a comunicação com o display LCD 16x2, o processamento das informações e a execução das funcionalidades principais do projeto.

O código gerado pelo STM32CubeMX foi importado para o Keil uVision5, serviu de base para o desenvolvimento do firmware. A partir dessa estrutura inicial, foram implementadas funções específicas para o controle do display em modo de 4 bits, incluindo rotinas de envio de comandos e dados. Essas funções foram responsáveis por manipular diretamente os pinos de controle (RS e EN) e as linhas de dados (D4 a D7), permitindo a comunicação bit a bit com o display.

No desenvolvimento da lógica do sistema, foi estruturado um fluxo de execução baseado em um loop principal, no qual o microcontrolador realiza continuamente as operações necessárias para o funcionamento do sistema. Nesse contexto, foram implementadas rotinas para inicialização do display, envio de mensagens e atualização das informações exibidas. A inicialização do LCD seguiu a sequência recomendada para operação em modo bit a bit, garantindo o correto funcionamento do dispositivo desde o início da execução.

Além disso, foram utilizados atrasos (delays) controlados no código, fundamentais para garantir a sincronização da comunicação com o display. Esses tempos de espera são necessários para que o LCD processe corretamente os comandos recebidos, evitando falhas na exibição das informações. A correta definição desses intervalos contribuiu para a estabilidade do sistema durante sua operação.

O software Keil uVision5 também foi utilizado para compilar o código e gerar o arquivo executável, que foi posteriormente gravado no microcontrolador. Durante o desenvolvimento, a ferramenta permitiu a identificação e correção de erros, além de facilitar a organização do código em módulos, tornando o firmware mais estruturado e de fácil manutenção.

A Figura 11, abaixo, demonstra as bibliotecas mínimas necessárias para que a programação do Firmware seja funcional e compatível com os objetivos do projeto. Bibliotecas como "stdio.h"e "stdlib.h"são mínimas e imprescindíveis em toda programação, onde já importam funções cotidianas e tratamento de dados alfanuméricos e númericos já como bits, o que facilita toda a programação. Sem elas, realizar cálculos, loop while e comandos

IF não funcionarão sem que cada operação seja escrita com o máximo de detalhes.

Figura 11: Bibliotecas utilizadas para desenvolvimento

```
/* USER CODE END Header */
/* Includes ------------------------------------------------------------------*/
#include "main.h"

/* Private includes ----------------------------------------------------------*/
/* USER CODE BEGIN Includes */

#include "stdio.h"
#include "stdlib.h"
#include "string.h"
#include "lcd_4bit.h"

/* USER CODE END Includes */
```

Fonte: Elaborado pelo Autor, 2026

Figura 12: Configuração da USART

```
static void MX_USART1_UART_Init(void)
{

  /* USER CODE BEGIN USART1_Init 0 */

  /* USER CODE END USART1_Init 0 */

  /* USER CODE BEGIN USART1_Init 1 */

  /* USER CODE END USART1_Init 1 */
  huart1.Instance = USART1;
  huart1.Init.BaudRate = 115200;
  huart1.Init.WordLength = UART_WORDLENGTH_8B;
  huart1.Init.StopBits = UART_STOPBITS_1;
  huart1.Init.Parity = UART_PARITY_NONE;
  huart1.Init.Mode = UART_MODE_TX_RX;
  huart1.Init.HwFlowCtl = UART_HWCONTROL_NONE;
  huart1.Init.OverSampling = UART_OVERSAMPLING_16;
  if (HAL_UART_Init(&huart1) != HAL_OK)
  {
    Error_Handler();
  }
  /* USER CODE BEGIN USART1_Init 2 */

  /* USER CODE END USART1_Init 2 */

}
```

Fonte: Elaborado pelo Autor, 2026

A Figura 12 demonstra que, mesmo com toda as definições e programações estabelecidas, sem uma faixa de Baudrate correspondente tanto na programação Keil, quanto na definição dos pinos dentro do STM32CubeMX, a comunicação não será estabelecida. Sendo necessário tanto a parametrização correta.

Figura 13: Inicialização periféricos periféricos e display

```
int main(void)
{

  /* USER CODE BEGIN 1 */

  /* USER CODE END 1 */

  /* MCU Configuration--------------------------------------------------------*/

  /* Reset of all peripherals, Initializes the Flash interface and the Systick. */
  HAL_Init();

  /* USER CODE BEGIN Init */

  /* USER CODE END Init */

  /* Configure the system clock */
  SystemClock_Config();

  /* USER CODE BEGIN SysInit */

  /* USER CODE END SysInit */

  /* Initialize all configured peripherals */
  MX_GPIO_Init();
  MX_USART1_UART_Init();
  /* USER CODE BEGIN 2 */

  LCD_Init();
  LCD_Clear();
  LCD_Put_Cur (0,0);
  LCD_Send_String("TCC II");
  LCD_Put_Cur (1,0);
  LCD_Send_String("Apresentando.");
  HAL_UART_Receive_IT(&huart1, &rxByte, 1);

  /* USER CODE END 2 */

  /* Infinite loop */
  /* USER CODE BEGIN WHILE */
```

Fonte: Elaborado pelo Autor, 2026

Figura 14: Estrutura para o envio de dados

```
  /* USER CODE BEGIN WHILE */

while (1)
{

  if (msgReady == 1)
  {
      msgReady = 0;
      LCD_Clear();
      LCD_Put_Cur(0, 0);

      if (strlen(rxBuffer) > 16)
      {
        char linha1[17] = {0};
        char linha2[17] = {0};
        strncpy(linha1, rxBuffer, 16);
        strncpy(linha2, rxBuffer + 16, 16);
        LCD_Send_String(linha1);
        LCD_Put_Cur(1, 0);
        LCD_Send_String(linha2);
      }
      else
      {
        LCD_Send_String(rxBuffer);
      }
    }

    if (rxIndex > 0)
    {
    HAL_Delay(50);
    if (rxIndex > 0)
    {
        rxBuffer[rxIndex] = '\0';
        rxIndex  = 0;
        msgReady = 1;
    }
    }

  }
}
    /* USER CODE END WHILE */
```

Fonte: Elaborado pelo Autor, 2026

As Figuras 13 e 14, acima, definem como a USART1 configurada no STM32CubeMX, através dos pinos PA9 e PA10, irá funcionar para transmissão dos dados enviados entre o ESP-M2 e o Microcontrolador de arquitetura ARM. Para projetos de maior complexidade com tratamento de dados, controle e transmissão em uma quantidade maior de fluxos, ou seja, projetos em que é necessário a utilização de mais de um canal USART, essa parametrização e programação deverá ser realizada de acordo com o especificado a cada projeto.

4.3 Implementação da comunicação com o Display.

A comunicação entre o microcontrolador e o display LCD 16x2 foi implementada utilizando o modo paralelo de bit a bit, no qual os dados são transmitidos de forma segmentada através de linhas digitais. Essa abordagem foi adotada com o objetivo de reduzir a quantidade de pinos utilizados no microcontrolador, mantendo a funcionalidade do display e garantindo uma comunicação eficiente.

No modo de 4 bits, cada byte de informação é dividido em duas partes, sendo inicialmente transmitidos os quatro bits mais significativos (MSB) e, em seguida, os quatro bits menos significativos (LSB). Esse processo exige que o microcontrolador realize o controle direto das linhas de dados (D4 a D7), bem como dos sinais de controle RS (Register Select) e EN (Enable), responsáveis por definir o tipo de dado enviado e sincronizar a comunicação com o display.

A Figura 15 apresenta onde no código é responsável pelo envio de dados ao display, evidenciando a manipulação das linhas de controle e dados durante a comunicação.

Figura 15: Callback para comunicação bit a bit com o display

```
/* USER CODE BEGIN 4 */

void HAL_UART_RxCpltCallback(UART_HandleTypeDef *huart)
{
  if (huart->Instance == USART1)
  {



    if (rxByte == '\r' || rxByte == '\n')
    {
      rxBuffer[rxIndex] = '\0';
      rxIndex  = 0;
      msgReady = 1;
    }
    else if (rxIndex < sizeof(rxBuffer) - 1)
    {
      rxBuffer[rxIndex++] = rxByte;
    }
    HAL_UART_Receive_IT(&huart1, &rxByte, 1);
  }
}
/* USER CODE END 4 */
```

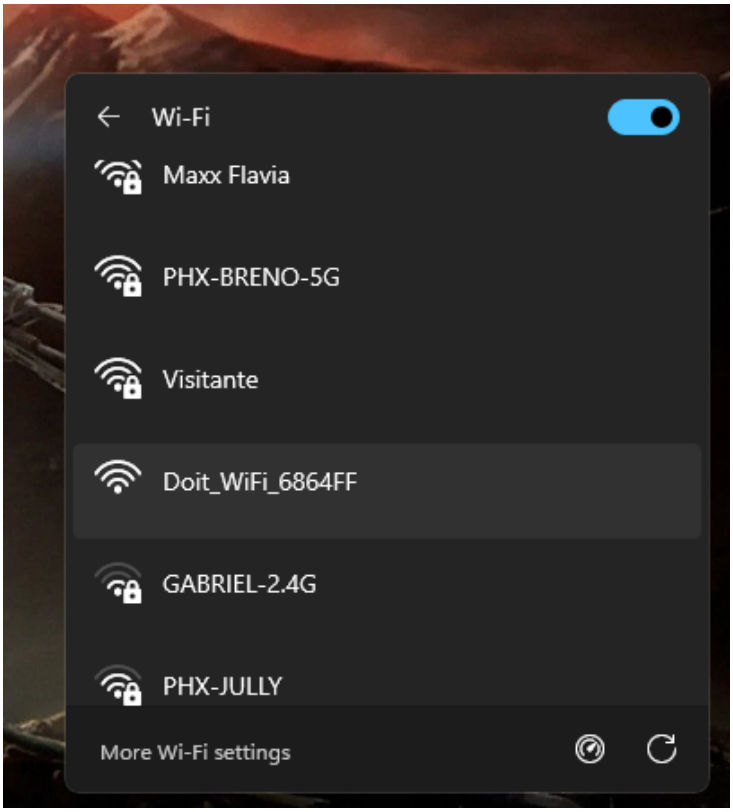
Fonte: Elaborado pelo Autor, 2026

A utilização do modo bit a bit juntamente com o callback programado no Keil definem como será feito o envio dos dados e quando alterá-los a partir da resposta obtida pelo Display, quando for "msgReady = 1", o mesmo fica responsável por pegar os dados enviados pelo microcontrolador e exibi-los em suas duas linhas LCD.

4.4 Configuração do Módulo ESP-M2

A configuração do módulo ESP-M2 foi realizada com o objetivo de permitir a comunicação sem fio entre o sistema embarcado e dispositivos externos, como computadores e smartphones. Para isso, o módulo foi configurado para operar no modo Station, no qual se conecta a uma rede Wi-Fi existente, possibilitando a troca de dados por meio de protocolos de rede.

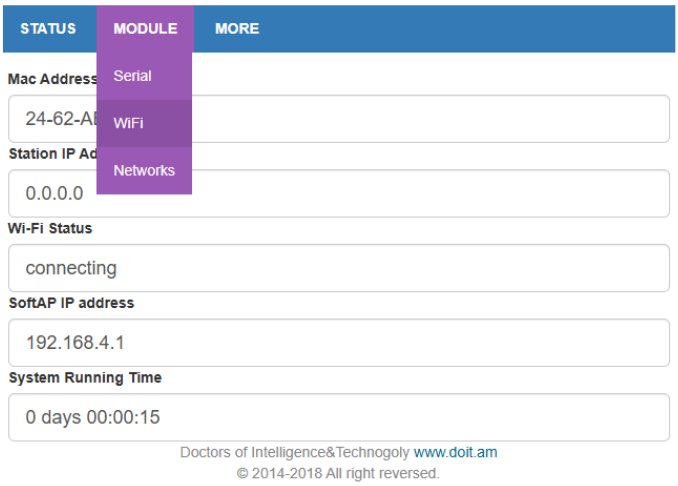
Figura 16: Conectando-se a rede própria do ESP-M2



Fonte: Elaborado pelo Autor, 2026

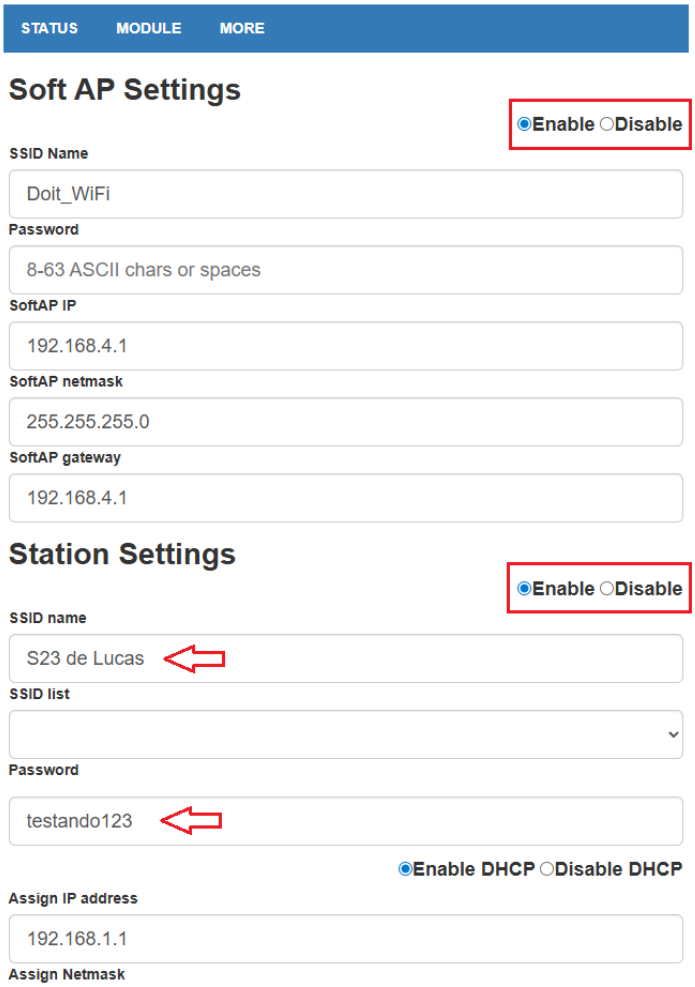
Inicialmente, foi realizado o acesso à interface de configuração do módulo por meio de um navegador web, utilizando o endereço IP padrão 192.168.4.1 disponibilizado pelo dispositivo. Através dessa interface, foi possível configurar os parâmetros de conexão, como o nome da rede (SSID) e a senha de acesso. No presente projeto, o módulo foi configurado para se conectar ao hotspot do dispositivo móvel, com nome de rede S23 de Lucas, permitindo a criação de um ambiente de comunicação local entre os dispositivos. Conforme demonstrado na Figura 16 acima e Figuras 17 e 18 abaixo, onde é realizada toda a configuração prévia para que o módulo ESP-M2 se conecte como cliente a um servidor central.

Figura 17: Acessando MODULE WI-FI para configuração do ESP-M2



Fonte: Elaborado pelo Autor, 2026

Figura 18: Parametrizando em qual rede o ESP irá se Conectar



Fonte: Elaborado pelo Autor, 2026

Após a conexão com a rede Wi-Fi, o módulo passou a operar como um servidor TCP, possibilitando o envio e recebimento de dados através de um endereço IP e porta previamente definidos. Essa configuração permite que dispositivos externos estabeleçam comunicação com o sistema embarcado por meio de sockets, conforme descrito no guia prático utilizado como referência . Dessa forma, tornou-se possível acessar o sistema remotamente e enviar comandos para o microcontrolador.

A integração entre o módulo ESP-M2 e o microcontrolador STM32 foi realizada por meio da interface serial USART, permitindo a troca de dados entre os dispositivos. Nesse contexto, o microcontrolador é responsável por interpretar os dados recebidos via Wi-Fi e executar as ações correspondentes, como a atualização das informações exibidas no display.

A configuração do módulo foi fundamental para viabilizar a comunicação remota do sistema, permitindo que comandos fossem enviados a partir de diferentes interfaces, como softwares de teste, aplicações em Python e páginas web. Dessa forma, o uso do ESP-M2 ampliou significativamente as funcionalidades do sistema embarcado, tornando-o mais flexível e adaptável a diferentes aplicações.

4.5 Utilização do Kit de Desenvolvimento

A etapa de montagem do protótipo consistiu na utilização de um DEVKIT desenvolvido pelo professor e orientador Helbet de Sá, onde nele é incluindo o microcontrolador STM32F103C8T6, o display LCD 16x2, o módulo ESP-M2 e os demais elementos eletrônicos utilizados no projeto. Essa fase teve como objetivo demonstrar, na prática, o funcionamento do sistema desenvolvido, garantindo a correta interligação entre hardware e software.

Inicialmente, foram realizadas as conexões elétricas entre os componentes, seguindo o esquemático previamente desenvolvido. O display LCD foi conectado ao microcontrolador por meio das linhas de dados D4 a D7 e dos sinais de controle RS e EN, conforme a comunicação paralela em modo de 4 bits. O módulo ESP-M2 foi integrado ao sistema utilizando comunicação serial, permitindo a troca de dados entre o microcontrolador e a interface de rede. Além disso, foram estabelecidas as conexões de alimentação e aterramento, garantindo o funcionamento adequado de todos os dispositivos.

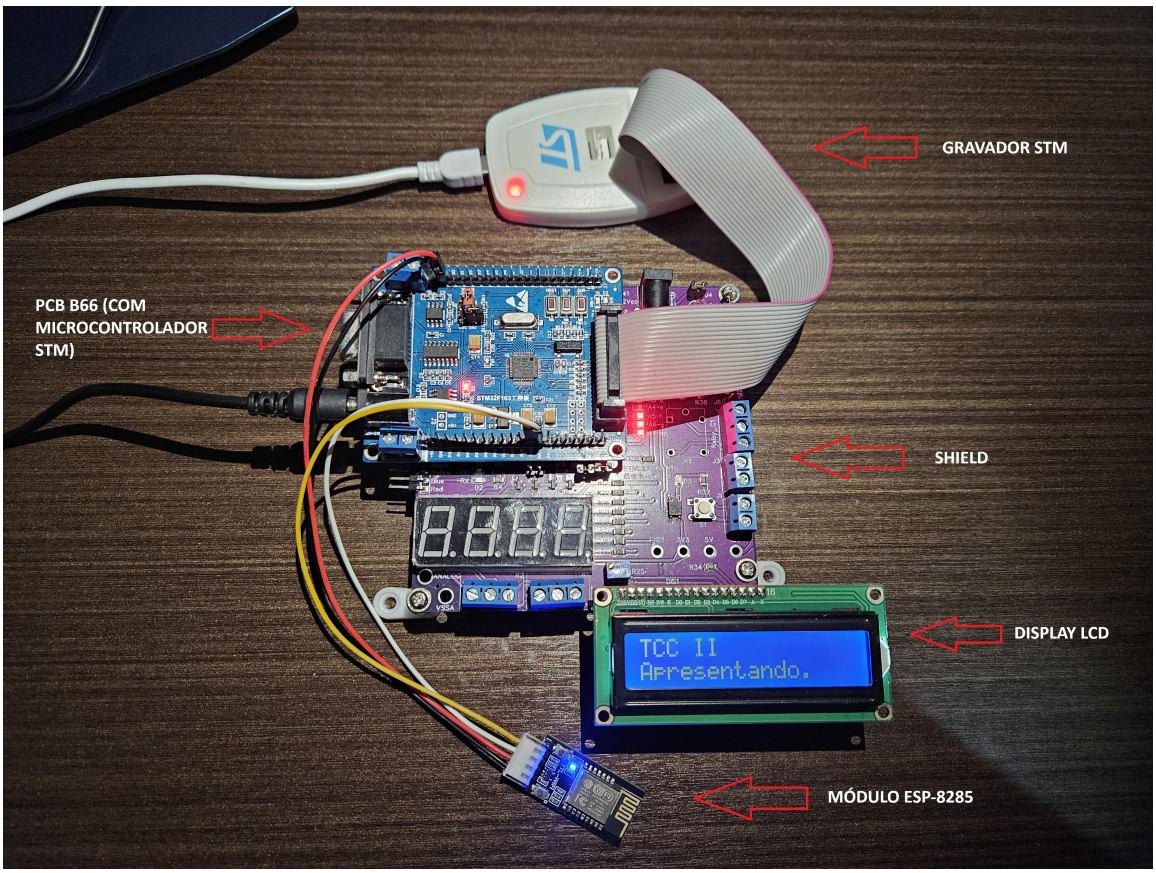
Após a validação inicial em bancada, a montagem foi consolidada na placa de circuito impresso desenvolvida para o projeto. A utilização da PCB permitiu maior organização

dos componentes, redução de interferências elétricas e maior confiabilidade do sistema, em comparação com montagens provisórias em protoboard. Essa etapa foi fundamental para a obtenção de um protótipo mais robusto e próximo de uma aplicação real.

Durante a montagem, foram realizados ajustes pontuais nas conexões e verificação de continuidade elétrica, assegurando que não houvesse falhas no circuito. Também foram observados aspectos como fixação dos componentes, organização dos cabos e qualidade das soldas, fatores que influenciam diretamente na estabilidade do sistema.

A Figura 19 apresenta o protótipo montado, evidenciando a disposição dos componentes e as conexões realizadas entre os módulos do sistema.

Figura 19: Sistema Embarcado conectado



Fonte: Elaborado pelo Autor, 2026

A montagem do protótipo permitiu validar a integração completa do sistema embarcado, possibilitando a realização dos testes de funcionamento e a verificação do comportamento do sistema em condições reais de operação. Note, que, o ESP-M2 apresenta o LED azul aceso e contínuo, demonstrando que o mesmo está conectado a rede WI-Fi configurada anteriormente, chamada de "S23 de Lucas".

4.6 Interface Web e Comunicação com o Sistema via ESP-M2

Com o objetivo de ampliar as formas de interação com o sistema embarcado, foi desenvolvida uma interface web para envio de comandos ao display LCD, utilizando comunicação em rede. Essa abordagem permite que o usuário envie mensagens remotamente por meio de um navegador, tornando o sistema mais flexível e acessível.

A interface foi implementada em linguagem HTML, com suporte a estilização e interatividade, permitindo ao usuário inserir mensagens de até 32 caracteres, correspondentes à capacidade do display LCD 16x2. Além disso, a aplicação apresenta recursos visuais como pré-visualização das mensagens divididas em duas linhas, indicação de status da conexão e registro de logs em tempo real, facilitando o acompanhamento da comunicação entre os dispositivos .

Para viabilizar a comunicação entre a interface web e o módulo ESP-M2, foi desenvolvido um script em Python responsável por atuar como uma ponte entre o protocolo WebSocket e a comunicação TCP. Essa abordagem foi necessária devido às limitações dos navegadores em estabelecer conexões TCP diretas, sendo o WebSocket utilizado como intermediário entre a aplicação web e o sistema embarcado.

O script em Python implementa um servidor WebSocket local, configurado para operar no endereço ws://localhost:8765, recebendo mensagens enviadas pela interface web. Ao receber uma mensagem, o servidor realiza o encaminhamento para o módulo ESP-M2 por meio de uma conexão TCP, utilizando o endereço IP e porta previamente configurados. No presente projeto, o módulo ESP foi configurado com o endereço 10.182.137.252 e porta 9000, permitindo a comunicação direta com o sistema embarcado .

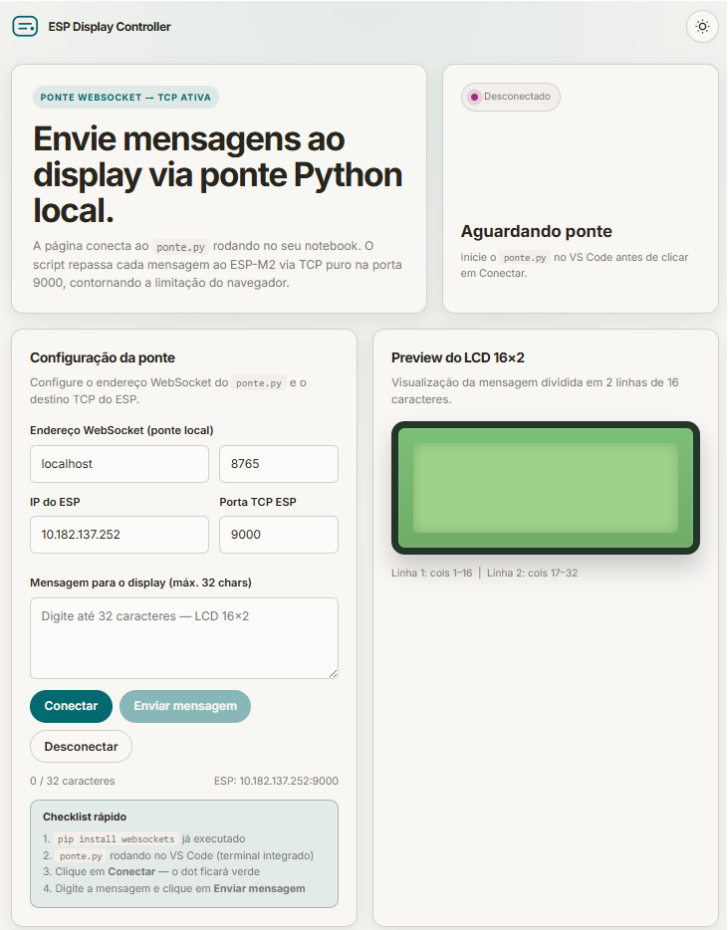
O processo de comunicação ocorre de forma sequencial: a mensagem é enviada pela interface web via WebSocket, recebida pelo servidor Python, transmitida ao ESP-M2 via TCP e, por fim, encaminhada ao microcontrolador para processamento e exibição no display. Após o envio, o script também pode retornar uma resposta para a interface web, permitindo a confirmação do envio ou a identificação de possíveis erros na comunicação.

Outro aspecto relevante da implementação é o controle do tamanho das mensagens enviadas. Considerando a limitação física do display LCD 16x2, o script em Python realiza automaticamente a restrição das mensagens para até 32 caracteres, garantindo compatibilidade com o sistema de exibição e evitando erros de processamento.

A Figura 20 apresenta a interface web desenvolvida para envio de comandos, enquanto

a Figura 21 ilustra a execução do script Python responsável pela ponte com websocket para comunicação entre os dispositivos.

Figura 20: Página Web Desenvolvida para comprovação dos envios das mensagens



Fonte: Elaborado pelo Autor, 2026

WebSocket é um protocolo de comunicação bidirecional e persistente que roda sobre HTTP. Diferente de uma requisição HTTP comum onde o cliente pergunta e o servidor responde e encerra a conexão o WebSocket mantém o canal aberto continuamente, permitindo troca de mensagens nos dois sentidos em tempo real. A utilização dessa arquitetura baseada em WebSocket e TCP proporcionou maior flexibilidade ao sistema, permitindo a integração contínua e possibilitando o controle remoto do display. Além disso, essa abordagem evidencia o uso de tecnologias modernas de comunicação, agregando valor ao projeto e demonstrando a capacidade de integração entre sistemas embarcados e aplicações em rede.

Figura 21: WebSocket programado e Python para exibição dos dados

```python
def tcp_send(message: str) -> str:
    """Abrindo uma conexão TCP, envia a mensagem e retorna a resposta (se houver)."""
    response = ""
    try:
        with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as s:
            s.settimeout(TIMEOUT)
            s.connect((ESP_IP, ESP_PORT))

            # Envia mensagem com terminador \r\n (ajuste se necessário)
            payload = message.encode("utf-8")
            s.sendall(payload)
            print(f"[TCP →] Enviado para {ESP_IP}:{ESP_PORT}  |  \"{message}\"")

            # Aguarda resposta do ESP (opcional — muitos ESPs não respondem)
            try:
                data = s.recv(256)
                if data:
                    response = data.decode("utf-8", errors="replace").strip()
                    print(f"[TCP ←] Resposta do ESP: \"{response}\"")
            except socket.timeout:
                response = "OK (sem resposta do ESP)"

    except ConnectionRefusedError:
        response = "ERRO: Conexão recusada — ESP offline ou porta incorreta."
        print(f"[ERRO] {response}")
    except socket.timeout:
        response = "ERRO: Timeout — ESP não respondeu em tempo hábil."
        print(f"[ERRO] {response}")
    except OSError as e:
        response = f"ERRO de rede: {e}"
        print(f"[ERRO] {response}")

    return response
```

Fonte: Elaborado pelo Autor, 2026

Figura 22: Registrando e exibindo o local de envio de dados

```python
async def handler(websocket):
    """Trata cada cliente WebSocket (página HTML)."""
    client = websocket.remote_address
    print(f"[WS] Novo cliente conectado: {client}")

    try:
        async for message in websocket:
            print(f"[WS ←] Recebido da página: \"{message}\"")

            # Limita a 32 caracteres (LCD 16x2 = 2 linhas × 16 cols)
            msg_limitado = message[:32]

            # Envia via TCP ao ESP (operação síncrona em thread separada)
            loop = asyncio.get_event_loop()
            response = await loop.run_in_executor(None, tcp_send, msg_limitado)

            # Devolve o resultado para a página HTML
            await websocket.send(response)

    except websockets.exceptions.ConnectionClosedOK:
        print(f"[WS] Cliente {client} desconectou normalmente.")
    except websockets.exceptions.ConnectionClosedError as e:
        print(f"[WS] Conexão com {client} encerrada com erro: {e}")
```

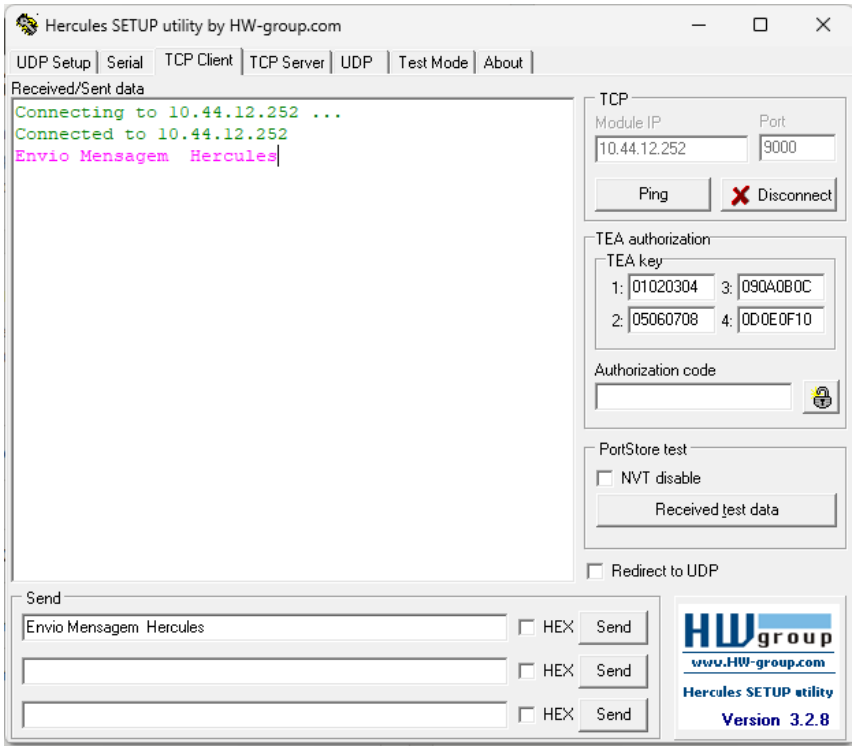
Fonte: Elaborado pelo Autor, 2026

4.7 Testes de Funcionamento e Resultados do Sistema

Após a montagem completa do protótipo e a integração entre os módulos de hardware e software, foram realizados testes com o objetivo de validar o funcionamento do sistema embarcado em condições reais de operação. Esses testes envolveram a verificação da comunicação entre o microcontrolador, o display LCD 16x2, o módulo ESP-M2 e a interface web desenvolvida, garantindo que todas as etapas do fluxo de dados estivessem operando corretamente.

Inicialmente, foram realizados testes de inicialização do sistema, verificando se o display LCD era corretamente configurado após o acionamento do microcontrolador, para isso foi utilizado o software Hércules, acessando a aba de "TCP Client"e comunicando com o IP gerado pelo ESP ao estar conectado em uma rede wi-fi. Nessa etapa, observou-se que o processo de inicialização em modo de 4 bits foi executado conforme esperado ao digitar mensagens e clicar no botão "send"do próprio aplicativo, permitindo a exibição de caracteres e mensagens sem falhas.

Figura 23: Enviando mensagem via TCP Cliente - Hercules



Fonte: Elaborado pelo Autor, 2026

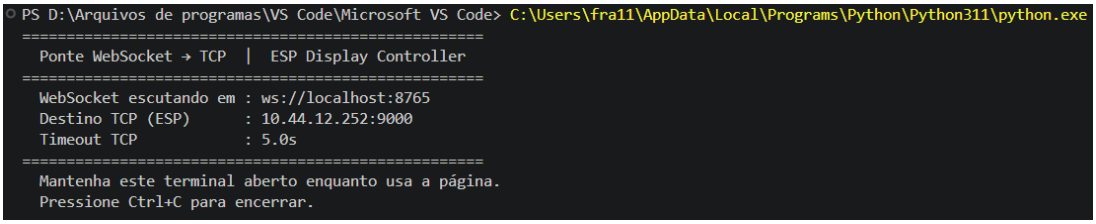
Figura 24: Mensagem recebida via TCP Client pelo Hércules

Fonte: Elaborado pelo Autor, 2026

Em seguida, foram conduzidos testes envolvendo a interface web e o sistema de comunicação via WebSocket e TCP. As mensagens inseridas na página HTML foram transmiti-

das com sucesso a ponte elaborada em Python, que por sua vez realizou o encaminhamento ao módulo ESP-M2.

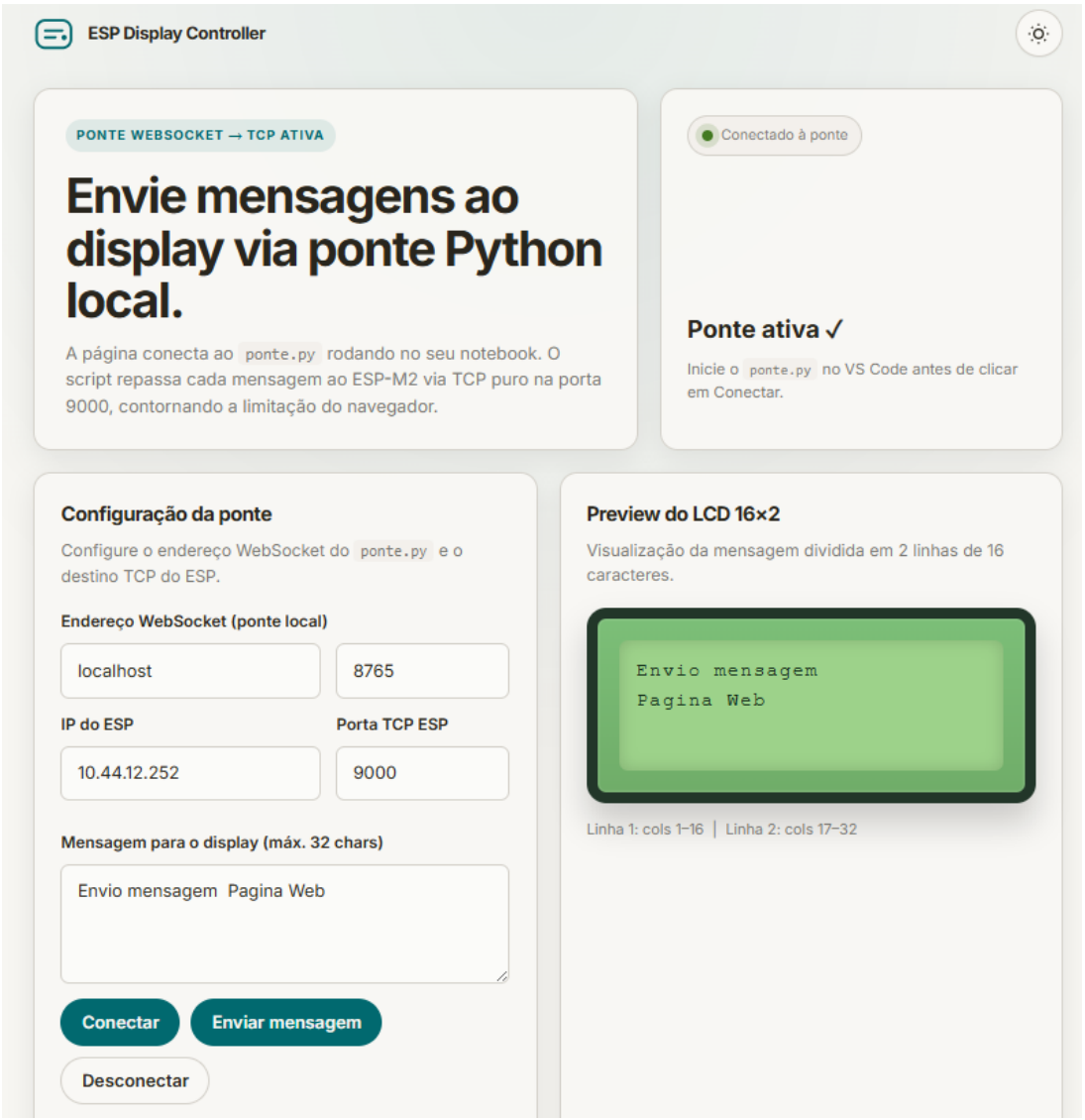
Figura 25: WebSocket no Python conectando corretamente com o STM32



Fonte: Elaborado pelo Autor, 2026

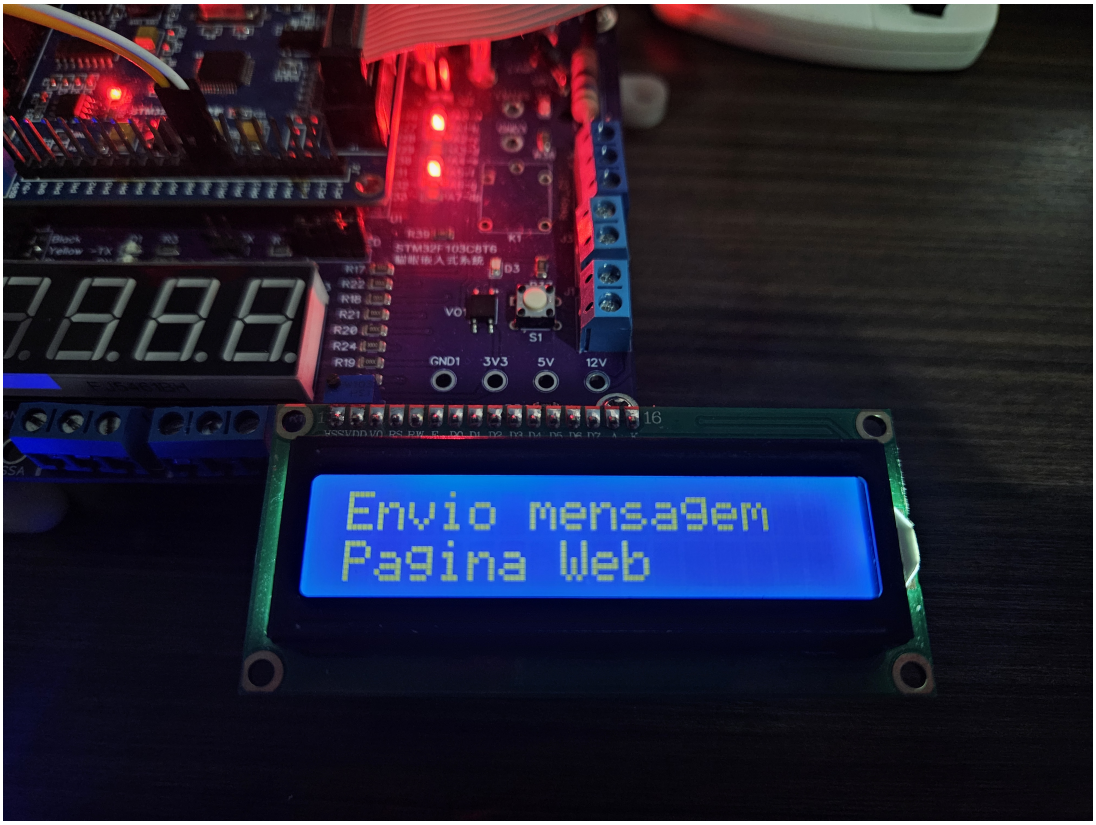
O microcontrolador recebeu os dados corretamente e realizou a exibição no display, confirmando o funcionamento completo do fluxo de comunicação. Durante esses testes, verificou-se a consistência na entrega das mensagens e a ausência de falhas na comunicação.

Figura 26: Enviando mensagem via WebSocket - Página HTML



Fonte: Elaborado pelo Autor, 2026

Figura 27: Mensagem recebida via WebSocket



Fonte: Elaborado pelo Autor, 2026

Outro aspecto analisado foi o tempo de resposta do sistema. Observou-se que a exibição das mensagens no display ocorria de forma imediata após o envio pela interface web, indicando que o sistema possui desempenho adequado para aplicações de monitoramento em tempo real. A utilização de delays no controle do display não comprometeu significativamente a velocidade de atualização das informações.

No que se refere à apresentação dos dados, o display LCD 16x2 demonstrou eficiência na exibição das mensagens, permitindo a organização das informações em duas linhas de forma clara e legível. A limitação de 32 caracteres foi respeitada durante os testes, garantindo que as mensagens fossem exibidas corretamente sem sobreposição ou perda de informação. Para projetos de um nível maior de complexidade, que necessite de uma transmissão de maior quantidade de dados, o uso de um display que aceite um buffer maior de caracteres para exibição se torna o ideal. Com a utilização de modelos de 3 a 4 linhas ou até modelos onde a capacidade de caracteres por linha seja maior, torna a exibição mais eficiente de acordo com as premissas do projeto ou exigência de clientes.

5 CONCLUSÃO

O projeto em questão teve como objetivo o desenvolvimento de um sistema embarcado microcontrolado para monitoramento e exibição de variáveis, utilizando como base o microcontrolador STM32F103C8T6 e diferentes tecnologias de comunicação e interface. A proposta surgiu da necessidade de soluções acessíveis e eficientes para aquisição e visualização de dados, especialmente em aplicações que demandam monitoramento em tempo real.

Ao longo do desenvolvimento, foi possível projetar e implementar um sistema completo, envolvendo desde a configuração do hardware até a criação do firmware e das interfaces de comunicação. A utilização do ambiente STM32CubeMX e do Keil uVision5 permitiu a correta configuração e programação do microcontrolador, enquanto o uso do display LCD 16x2 em modo de 4 bits possibilitou a exibição clara e organizada das informações, com controle direto dos sinais e maior domínio da comunicação.

Além disso, a integração do módulo ESP-M2 trouxe ao projeto a capacidade de comunicação remota, permitindo o envio de dados por meio de rede Wi-Fi. A implementação de uma interface web, associada a um servidor WebSocket desenvolvido em Python, possibilitou a criação de uma arquitetura de comunicação moderna, na qual o usuário pode interagir com o sistema de forma remota e em tempo real. Essa abordagem ampliou significativamente as funcionalidades do sistema, tornando-o mais versátil e aplicável a diferentes cenários.

Os testes realizados demonstraram que o sistema apresentou funcionamento estável e confiável, com resposta rápida na exibição das mensagens no display após o envio dos comandos. A comunicação entre os módulos ocorreu de forma consistente, sem falhas significativas, evidenciando a eficiência da integração entre hardware e software. A limitação física do display foi devidamente considerada, garantindo a correta exibição das informações.

Dessa forma, conclui-se que o sistema desenvolvido atendeu aos objetivos propostos, demonstrando a viabilidade da utilização de sistemas embarcados aliados a tecnologias de comunicação em rede para monitoramento e exibição de dados. O projeto também contri-

buiu para o aprofundamento dos conhecimentos em eletrônica, programação de microcontroladores e integração de sistemas, evidenciando a importância da interdisciplinaridade na área de engenharia.

Como sugestão para trabalhos futuros, destaca-se a possibilidade de integração com sensores reais para aquisição automática de dados, a utilização de interfaces gráficas mais avançadas e a implementação de armazenamento em nuvem, permitindo o acesso remoto às informações de forma contínua e escalável.

Referências

[1] STMICROELECTRONICS. *STM32F103x8/STM32F103xB: Medium-density performance line ARM-based 32-bit MCU with 64 or 128 KB Flash, USB, CAN, 7 timers, 2 ADCs, 9 com. interfaces Datasheet*. Genebra: STMicroelectronics, 2023. Disponível em: `https://www.st.com/resource/en/datasheet/stm32f103c8.pdf`. Acesso em: 05 dez. 2025.

[2] STMICROELECTRONICS. *STM32CubeMX for STM32 configuration and initialization C code generation User Manual UM1718*. Genebra: STMicroelectronics, 2023. Disponível em: `https://www.st.com/resource/en/user_manual/um1718-stm32cubemx-for-stm32-configuration-and-initialization-c-code.pdf`. Acesso em: 10 dez. 2025.

[3] ESPRESSIF SYSTEMS. *ESP8266 AT Instruction Set Version 3.2.0.0*. Xangai: Espressif Systems, 2023. Disponível em: `https://www.espressif.com/sites/default/files/documentation/4a-esp8266_at_instruction_set_en.pdf`. Acesso em: 10 dez. 2025.

[4] ESPRESSIF SYSTEMS. *ESP8285 Datasheet: Single-chip 1 Mbytes Flash Wi-Fi Enabled Microcontroller*. Xangai: Espressif Systems, 2022. Disponível em: `https://www.espressif.com/sites/default/files/documentation/0a-esp8285_datasheet_en.pdf`. Acesso em: 20 fev. 2026.

[5] HITACHI SEMICONDUCTOR. *HD44780U (LCD-II): Dot Matrix Liquid Crystal Display Controller/Driver Datasheet*. Tóquio: Hitachi, 1998. Disponível em: `https://www.sparkfun.com/datasheets/LCD/HD44780.pdf`. Acesso em: 25 fev. 2026.

[6] FETTE, I.; MELNIKOV, A. *The WebSocket Protocol RFC 6455*. Internet Engineering Task Force (IETF), 2011. Disponível em: `https://datatracker.ietf.org/doc/html/rfc6455`. Acesso em: 12 mar. 2026.

[7] TANENBAUM, Andrew S.; WETHERALL, David. *Redes de Computadores*. 5. ed. São Paulo: Pearson Prentice Hall, 2011.

[8] KALVERBOER, A. *Web2Tcp: bridging Websockets and Tcp Sockets*. GitHub, 2018. Disponível em: `https://github.com/akalverboer/web2tcp_bridge`. Acesso em: 12 mar. 2026.

[9] DRPRESS. *A Comparative Analysis of Synchronous USART And Asynchronous UART in STM32F103ZET6*. HSET Highlights in Science, Engineering and Technology, v. 78, 2024. Disponível em: `https://drpress.org/ojs/index.php/HSET/article/view/16508`. Acesso em: 01 abr. 2026.

[10] WHITE, Elecia. *Making Embedded Systems: Design Patterns for Great Software*. Sebastopol: O'Reilly Media, 2011.

[11] NGUYEN, Thi Thu Hang. *Design of an Embedded System Using STM32F103 Microcontroller*. Dissertação (Mestrado em Engenharia). Finlândia: Universidade de Ciências Aplicadas de Turku, 2015. Disponível em: `https://www.diva-portal.org/smash/get/diva2:829934/FULLTEXT01.pdf`. Acesso em: 01 abr. 2026.

[12] PENG, Y. et al. *Application of the upgraded design of the STM32F103 in battery monitoring systems for hospital power*. Heliyon, v. 10, 2024. Disponível em: `https://www.sciencedirect.com/science/article/pii/S2405844024150165`. Acesso em: 01 abr. 2026.

[13] ALMEIDA, Rodrigo Maximiano. *Sistemas Embarcados*. São Paulo: Érica, 2019.

[14] PEREIRA, Fábio. *Microcontroladores ARM Cortex-M em Sistemas Embarcados*. Rio de Janeiro: Ciência Moderna, 2021.

[15] VALVANO, Jonathan W. *Embedded Systems: Real-Time Interfacing to ARM Cortex-M Microcontrollers*. 3. ed. Estados Unidos: CreateSpace, 2017.

[16] ESPRESSIF SYSTEMS. *ESP8266 AT Instruction Set*. Shanghai: Espressif Systems, 2023.