Universidade do Vale do Itajaí
Internet das Coisas

Trabalho Final IoT

Monitoramento Ambiental via MQTT e Node-RED

Aluno: Fábio Volkmann Coelho
Professor: Jordan P. Sausen

Itajaí - SC
2025

1 Definição do Problema e Objetivo

1.1 Contextualização e Problema

A Internet das Coisas (IoT) tem transformado a gestão de ambientes físicos, permitindo que objetos cotidianos coletem e troquem dados em tempo real. Em diversos cenários, como, salas de servidores (Data Centers), estufas agrícolas, laboratórios farmacêuticos ou residências, o controle rigoroso das condições climáticas é essencial para a preservação de alguns equipamentos e produtos.

1.2 Objetivo do Projeto

O objetivo principal deste trabalho é desenvolver e implementar um protótipo funcional de um sistema IoT para o monitoramento remoto e contínuo de temperatura e umidade. Para atingir este objetivo, o projeto propõe:

1. **Coleta de Dados:** Utilizar o microcontrolador ESP8266 NodeMCU integrado ao sensor digital DHT11 para a leitura das grandezas físicas.
2. **Conectividade e Transmissão:** Estabelecer a comunicação via Wi-Fi e transmitir os dados coletados utilizando o protocolo MQTT, escolhido por sua leveza e eficiência (MQTT..., 2014).
3. **Visualização:** Integrar o sistema à plataforma Node-RED para a criação de um *dashboard* gráfico, permitindo a leitura e interpretação dos dados em tempo real.

2 Detalhamento da Plataforma de Hardware

Para a implementação do sistema de monitoramento ambiental proposto, selecionou-se um conjunto de hardware capazes de suprir a necessidade do projeto. A plataforma é composta por três elementos principais: uma unidade de processamento com conectividade integrada e um sensor digital de grandezas físicas. Para a implementação do sistema de monitoramento ambiental proposto, selecionou-se um conjunto de hardware composto por uma unidade de processamento de 32-bits com conectividade integrada e um sensor digital de grandezas físicas.

2.1 Microcontrolador: ESP8266 NodeMCU V3

A unidade central de processamento é o módulo de desenvolvimento NodeMCU V3, baseado no *System-on-Chip*(SoC) ESP8266EX. Esta plataforma foi escolhida por integrar processamento e Wi-Fi em um único encapsulamento, eliminando a necessidade de módulos de rede externos e simplificando o circuito.

Especificações Técnicas

- **Processador**: Núcleo RISC Tensilica L106 de 32 bits.
- **Clock**: Opera nativamente a 80 MHz.
- **Memória Flash**: 4 MB (interface SPI), utilizada para armazenamento do firmware, sistema de arquivos (LittleFS) e bibliotecas.
- **RAM**: Aproximadamente 50 KB de RAM disponível para instruções e dados do usuário durante a execução.

Características Elétricas e GPIOs

A placa opera estritamente com nível lógico de 3.3V. O pino utilizado para leitura de dados neste projeto, **D5**, corresponde ao **GPIO14** no mapeamento interno (NodeMCU Team, 2025).

- **Tensão de Alimentação**: 3.3V (regulada internamente a partir da entrada USB de 5V).
- **Mapeamento de Pinos**: Os pinos externos utilizam uma nomenclatura própria (D1, D2, etc.) que difere do mapeamento interno do SoC. O pino utilizado para leitura de dados neste projeto, D5, corresponde ao GPIO14.
- **Corrente Máxima**: Os pinos de I/O suportam uma drenagem máxima de 12mA.

Interface de Gravação

O módulo possui um conversor USB-Serial (CH340 ou CP2102) integrado ao PCB. Este circuito é responsável por:

- **Conversão de Sinal**: Transpor os níveis de tensão do USB (5V) para UART (3.3V).
- **Modo de Boot**: Controlar os pinos RTS e DTR para alternar automaticamente o GPIO0 e o RESET, colocando o microcontrolador em modo de gravação (Flash Mode) sem intervenção manual.

2.2 Sensor de Temperatura e Umidade: DHT11

A coleta de dados é realizada pelo sensor digital **DHT11**. O componente integra internamente um sensor de umidade capacitivo e um termistor (NTC) para temperatura, conectados a um microcontrolador de 8 bits de alto desempenho (Aosong Electronics, 2025).

Protocolo de Comunicação

A comunicação ocorre via protocolo serial *Single-Wire*. O ESP8266 envia um sinal de *Start* (pulso baixo de 18ms) e o sensor responde com uma sequência de 40 bits contendo umidade, temperatura e *checksum*. . O processo de leitura segue as seguintes etapas:

- **Sinal de Start**: O ESP8266 envia um pulso baixo por pelo menos 18ms para "acordar"o sensor.
- **Resposta**: O DHT11 responde com um pulso de presença.
- **Transmissão de Dados**: O sensor envia uma sequência de 40 bits de dados.

  O nível lógico '0' ou '1' é determinado pela duração do pulso de tensão alta (26µs para '0', 70µs para '1').

Estrutura dos Dados (40 Bits)

O pacote de dados recebido é estruturado da seguinte forma:

- **8 bits**: Parte inteira da Umidade.
- **8 bits**: Parte decimal da Umidade.
- **8 bits**: Parte inteira da Temperatura.
- **8 bits**: Parte decimal da Temperatura.
- **8 bits**: Checksum (Soma de verificação).
- **Nota**: O Checksum é a soma dos 4 primeiros bytes. Se o cálculo coincidir com o último byte recebido, a leitura é considerada válida.

Especificações Operacionais

:

- **Faixa de Umidade**: 20% a 90% RH (Precisão ±5%).
- **Faixa de Temperatura**: 0°C a 50°C (Precisão ±2°C).
- **Tempo de Amostragem**: O sensor requer um intervalo mínimo de 1 a 2 segundos entre leituras para estabilização térmica.

Recursos de Conectividade

A comunicação de dados é suportada pelo transceptor de rádio integrado ao SoC, operando no padrão IEEE 802.11 b/g/n na frequência de 2.4 GHz. A implementação utiliza a pilha de protocolos TCP/IP (IPv4) embarcada no SDK do microcontrolador. Esta infraestrutura permite que o dispositivo gerencie a autenticação WPA2 na rede sem fio e mantenha conexões persistentes (Sockets TCP), requisito fundamental para o funcionamento estável do protocolo de aplicação MQTT utilizado na transmissão da telemetria.

3 Configuração do Ambiente de Desenvolvimento

Para garantir a reprodutibilidade e a eficiência no desenvolvimento do firmware, optou-se por um ambiente de desenvolvimento moderno baseado no ecossistema PlatformIO, integrado ao editor de código Visual Studio Code (VS Code).(PlatformIO, 2025).

3.1 IDE e Ferramentas

Diferente da Arduino IDE tradicional, o PlatformIO oferece um gerenciamento de dependências declarativo e automatizado. Toda a configuração do projeto — incluindo a placa alvo, velocidade do monitor serial e versões das bibliotecas — é definida no arquivo platformio.ini. Isso assegura que o projeto possa ser compilado em qualquer computador sem a necessidade de instalar bibliotecas manualmente.

3.2 Configurações do Projeto

As dependências foram definidas no arquivo `platformio.ini`:

- **Plataforma:** espressif8266
- **Framework:** Arduino
- **Velocidade Serial:** 115200 baud
- **Gerenciamento de Bibliotecas**: Instalação automática via registro do PlatformIO.

3.3 Bibliotecas Utilizadas

O desenvolvimento do código baseou-se em bibliotecas de código aberto (*Open Source*) consolidadas na comunidade, reduzindo o tempo de implementação de protocolos complexos.

1. **Adafruit DHT Sensor Library (v1.4.4) e Adafruit Unified Sensor (v1.1.9)**
   - Responsáveis pela abstração do protocolo *Single-Wire* do sensor DHT11.
   - Realizam a leitura dos pulsos digitais de 40 bits e a conversão para valores de ponto flutuante (*float*) de temperatura e umidade.
2. **PubSubClient (Nick O'Leary, v2.8):**
   - Implementa um cliente MQTT leve compatível com o padrão Arduino.
   - Gerencia a conexão TCP com o *broker*, o *keep-alive* (ping) para manter a conexão ativa e a serialização das mensagens para publicação nos tópicos.
3. **ESP8266WiFi:**
   - Biblioteca nativa do *framework* Arduino para ESP8266.
   - Gerencia a conexão com o *Access Point* (Roteador).

3.4 Configurações Específicas do Projeto

O firmware foi parametrizado para atender aos requisitos de conectividade e topologia de rede do projeto.

- **Conexão Wi-Fi:** O dispositivo opera em modo Station (STA), conectando-se a uma rede WPA2-Personal (2.4 GHz).
- **Protocolo MQTT:**
  - **Broker:** Utilizou-se o servidor público `test.mosquitto.org` ou `broker.hivemq.com` na porta padrão 1883 (sem criptografia TLS, visando menor latência para este protótipo).
  - **Tópicos:** A estrutura de tópicos foi hierarquizada para separar os dados por grupo e tipo de grandeza:
    - `mestrado/iot/aluno/coelho/temperatura`
    - `mestrado/iot/aluno/coelho/umidade`
- **Estratégia de Leitura:** Implementou-se um loop de controle não bloqueante (ou baseado em *delay* para simplificação didática) que respeita o tempo de integração mínimo de 10 segundos, garantindo a estabilidade do sensor DHT11 e evitando o congestionamento do broker.

4 Desenvolvimento de Software

O algoritmo foi projetado para ser resiliente a falhas de conexão. Antes de qualquer leitura de sensor, o sistema verifica a conectividade com a rede Wi-Fi e com o broker MQTT. Caso a conexão caia, uma rotina de reconexão é acionada automaticamente, garantindo que o dispositivo não precise ser reiniciado manualmente. A lógica de controle do firmware foi estruturada seguindo o paradigma de programação orientada a eventos, típica de sistemas embarcados baseados no framework Arduino. O código divide-se em duas rotinas principais: a configuração inicial (Setup) e o laço de repetição infinito (Loop).

Para atender ao requisito de tempo de integração, o sistema permanece em espera (delay) por 10 segundos após cada transmissão bem-sucedida.

4.1 Pseudocódigo

```
ALGORITMO Monitoramento_IoT
    CONSTANTES
        WIFI_SSID = "Nome_da_Rede"
        WIFI_PASS = "Senha_da_Rede"
        MQTT_SERVER = "test.mosquitto.org"
        MQTT_PORT = 1883
        TOPICO_TEMP = "graduacao/iot/{grupo}/temperatura"
        TOPICO_UMID = "graduacao/iot/{grupo}/umidade"


    INICIO
        // Configuracao Inicial (Executado uma vez)
        FUNCAO Setup()
            INICIAR comunicacao serial (115200 baud)
            INICIAR sensor DHT11

            // Conexao Wi-Fi
            INICIAR conexao Wi-Fi (WIFI_SSID, WIFI_PASS)
            ENQUANTO (Estado Wi-Fi != CONECTADO) FACA
                AGUARDAR 500ms
                IMPRIMIR "."
            FIM_ENQUANTO
            IMPRIMIR "Wi-Fi Conectado"

            CONFIGURAR servidor MQTT (MQTT_SERVER, MQTT_PORT)
        FIM_FUNCAO
        // Loop Principal (Executado repetidamente)
        FUNCAO Loop()
            // Verificacao de Conexao MQTT
            SE (Cliente MQTT NAO conectado) ENTAO
                CHAMAR Reconectar_MQTT()
            FIM_SE

            MANTER cliente MQTT ativo (Loop)
            // Leitura do Sensor
            VAR umidade = LER_UMIDADE(DHT11)
            VAR temperatura = LER_TEMPERATURA(DHT11)
            // Validacao de Erros
            SE (temperatura EH Nulo OU umidade EH Nulo) ENTAO
                IMPRIMIR "Falha na leitura do sensor"
                RETORNAR // Pula para o inicio do loop
            FIM_SE

            // Publicacao de Dados (Telemetria)
            PUBLICAR(TOPICO_TEMP, converter_string(temperatura))
            PUBLICAR(TOPICO_UMID, converter_string(umidade))
            IMPRIMIR "Dados enviados com sucesso"
            // Tempo de Integracao (Requisito: Minimo 10s)
            AGUARDAR 10000ms
        FIM_FUNCAO
    FIM
```

Listing 1: Lógica Simplificada do Firmware

4.2 Disponibilidade do Código Fonte

Conforme solicitado nas instruções do projeto, o código fonte completo em C++ (PlatformIO) encontra-se disponível no repositório digital abaixo:

- **Link:** `https://github.com/FabioVCoelho/mestrado/tree/main/IoT/Trabalho_Final_IoT/Board_ESP8266_IoT`

5 Integração com Node-RED para Monitoramento

A etapa final do fluxo de dados consiste na visualização das informações coletadas em uma interface gráfica amigável (*Dashboard*). Para isso, utilizou-se a plataforma Node-RED, uma ferramenta de programação baseada em fluxo (*Flow-based programming*) desenvolvida originalmente pela IBM.

Arquitetura de Solução do Node-RED

O Node-RED atua neste projeto como um cliente Assinante (*Subscriber*) no protocolo MQTT. A lógica de visualização foi implementada através de um fluxo linear composto por três etapas principais:

Nó de Entrada (MQTT In):

- Configurado para conectar-se ao broker `test.mosquitto.org` ou `broker.hivemq.com` na porta 1883.
- Assina os tópicos específicos do grupo:
  - `mestrado/iot/aluno/coelho/temperatura`
  - `mestrado/iot/aluno/coelho/umidade`
- Recebe o *payload* (carga útil) contendo o valor numérico bruto enviado pelo ESP8266.

Processamento (Opcional):
Como os dados são enviados em formato de texto (*String*) contendo apenas números, o Node-RED realiza a conversão automática de tipos para a plotagem gráfica, dispensando nós complexos de tratamento de dados (*parsers*).

Nó de Saída (Dashboard/Gauge):

- Utilizou-se o pacote `node-red-dashboard` para a criação da interface.
- Foram configurados dois widgets do tipo *Gauge* (Manômetro):
  - **Temperatura:** Escala de 0°C a 50°C.
  - **Umidade:** Escala de 0% a 100%.

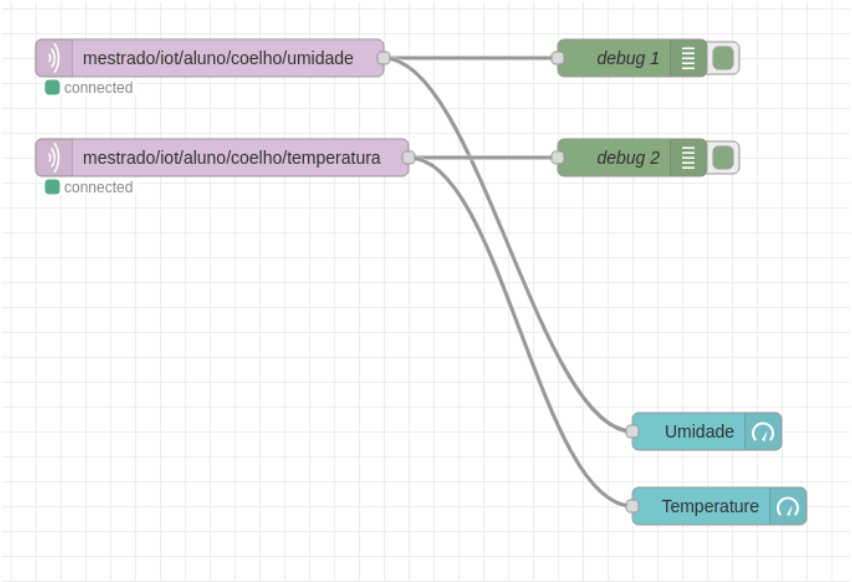

Figura 1: FlowChart do Node-RED (Foto do Autor)

5.1 Validação dos Resultados

A integração foi validada observando-se a atualização síncrona dos indicadores no painel de controle a cada 10 segundos, correspondendo ao intervalo de envio programado no firmware. A solução permite o monitoramento remoto de qualquer dispositivo com acesso à internet, centralizando as informações ambientais em uma única tela.(Node-RED, 2025)

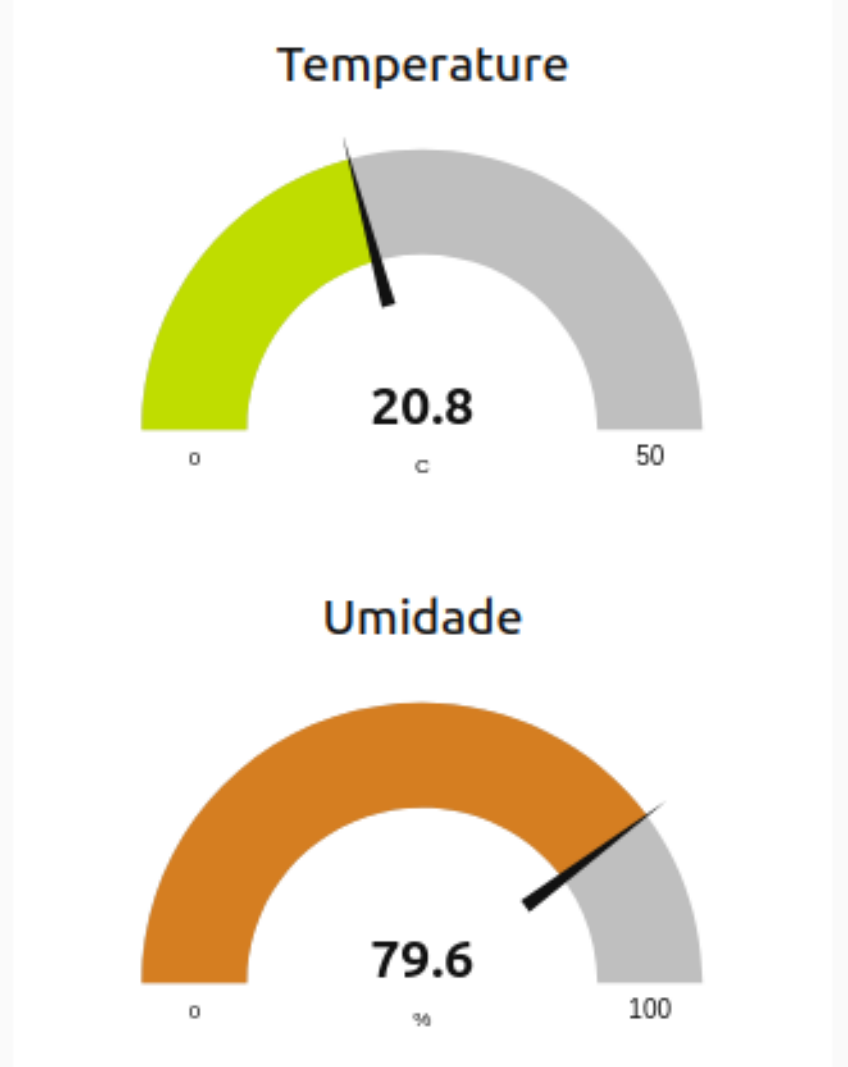

Figura 2: FlowChart do Node-Dashboard (Foto do Autor)

6 Esquema de Conexão do Hardware

Para atender ao requisito de documentação das ligações elétricas, apresenta-se abaixo o esquema de conexão desenvolvido. Devido às dimensões do módulo NodeMCU (que ocupa toda a largura útil da protoboard), optou-se por uma montagem utilizando apenas o barramento lateral direito do microcontrolador.

6.1 Diagrama de Ligações

A figura a seguir ilustra o fluxo de sinais e alimentação entre o microcontrolador ESP8266 e o sensor DHT11.

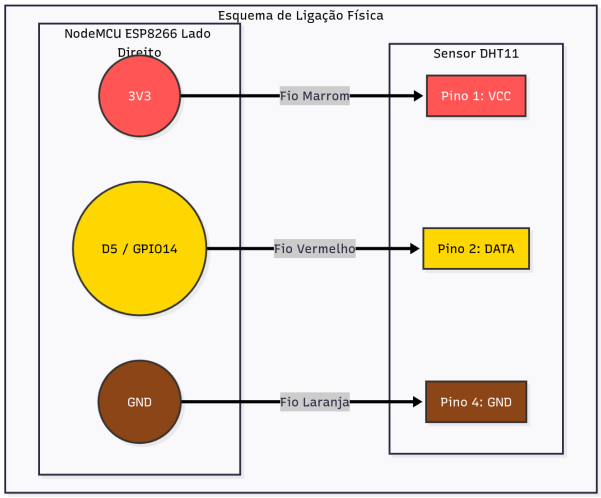

Figura 3: Montagem Física (Foto do Autor)

6.2 Montagem Experimental

A concretização física do circuito pode ser visualizada na imagem abaixo. Nota-se a utilização de jumpers fêmea-macho para a conexão direta do sensor, contornando a limitação de espaço físico da protoboard.

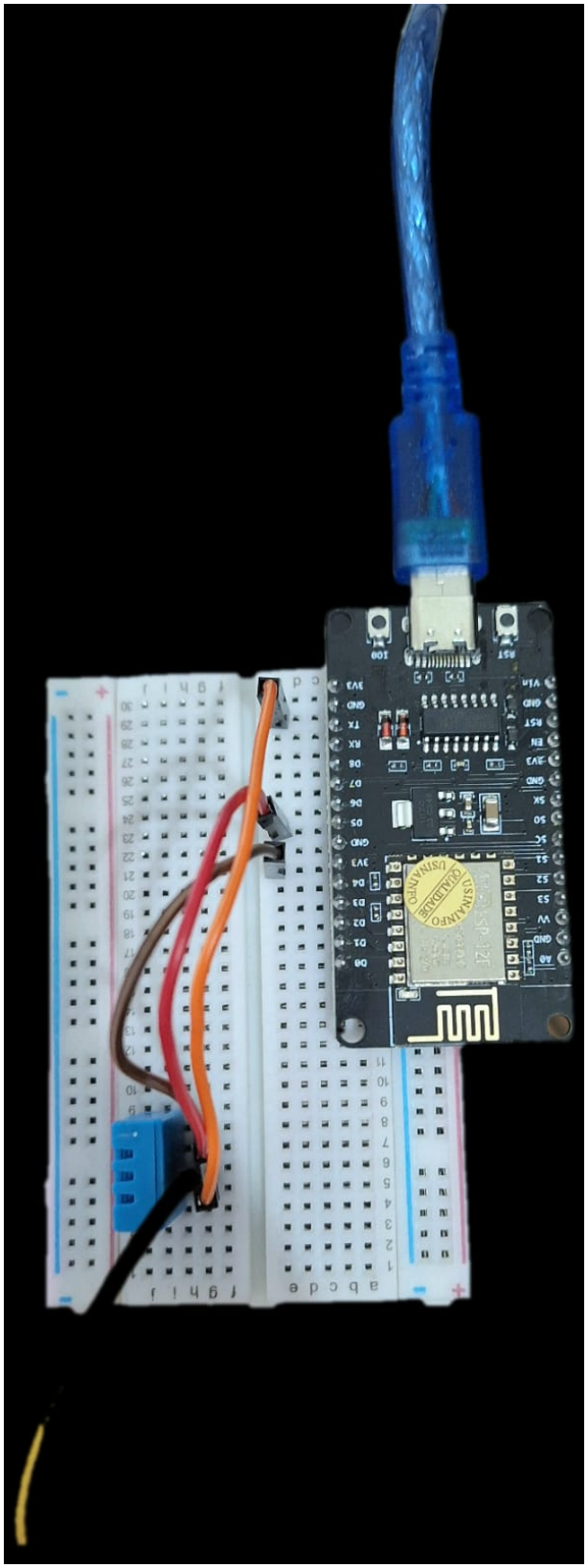

Figura 4: Montagem Experimental do Protótipo (Foto do Autor)

Referências Bibliográficas

AOSONG ELECTRONICS. **DHT11 Humidity & Temperature Sensor Datasheet**. Guangzhou. Disponível em: `https://www.aosong.com/`. Acesso em: 24 nov. 2025.

MQTT.ORG. **MQTT Version 3.1.1 OASIS Standard**. [*S. l.: s. n.*], 2014. Disponível em: `http://mqtt.org/documentation`. Acesso em: 24 nov. 2025.

NODE-RED. **Node-RED Documentation: Flow-based programming for the Internet of Things**. Disponível em: `https://nodered.org/`. Acesso em: 24 nov. 2025.

NODEMCU TEAM. **NodeMCU Documentation**. Disponível em: `https://nodemcu.readthedocs.io/en/master/`. Acesso em: 24 nov. 2025.

PLATFORMIO. **PlatformIO Documentation: A professional collaborative platform for embedded development**. Disponível em: `https://docs.platformio.org/`. Acesso em: 24 nov. 2025.