Estação Meteorológica com ESP32

Integrantes:

- Caio Filipe
- Juliana Aparecida Vecchi
- Leonardo Siemens
- Rafael Augusto de Oliveira Souza

Objetivo do Projeto

Este projeto tem como objetivo o desenvolvimento de uma Estação Meteorológica Inteligente, utilizando sensores ambientais conectados a um microcontrolador **ESP32**. A estação será capaz de monitorar em tempo real diversos parâmetros climáticos, como temperatura, umidade relativa do ar, pressão atmosférica e qualidade do ar, além de calcular índices derivados como o ponto de orvalho e o índice de calor.
Para isso, serão utilizados os sensores **DHT22**, que coletam dados de temperatura e umidade, **BMP280 / BME280** para medir a pressão atmosférica, e **MQ135** para detectar a qualidade do ar (gases nocivos). O microcontrolador **ESP32** será responsável por processar as informações coletadas pelos sensores e transmiti-las via Wi-Fi para uma plataforma em nuvem.
As informações serão armazenadas na nuvem e exibidas em uma interface web interativa com gráficos e indicadores visuais.
A aplicação principal da estação meteorológica é fornecer dados ambientais precisos e acessíveis para estudos meteorológicos, alertas preventivos, apoio à agricultura, controle de ambientes internos e ações voltadas à saúde e qualidade de vida.
Além disso, o sistema contará com um módulo de alertas, capaz de identificar situações críticas como calor extremo ou qualidade do ar prejudicial, notificando o usuário na plataforma.

Justificativa

Este projeto é altamente relevante para sistemas ciberfísicos porque **integra o mundo físico ao mundo digital de forma inteligente e interconectada**. A estação meteorológica utiliza sensores para coletar dados ambientais, que são processados por um microcontrolador (ESP32) e enviados a uma plataforma web onde são armazenados, analisados e visualizados em tempo real.

Essa comunicação entre os **dispositivos físicos e os sistemas computacionais** é a base de um sistema ciberfísico. Além disso, o projeto:

- Auxilia na **tomada de decisão** com base nos dados coletados (ex: alertas climáticos);
- Promove a **aplicação real de conceitos de IoT, computação embarcada e automação inteligente**.

Tecnologias Utilizadas

Hardware:

- ESP32
- Sensor DHT22 (Temperatura e Umidade)
- Sensor BMP280 ou BME280 (Pressão Atmosférica)
- Sensor MQ135 (Qualidade do Ar)
- Jumpers
- Protoboard

Bibliotecas:

- **WiFi.h** – para conexão do ESP32 à rede Wi-Fi
- **Adafruit_Sensor.h** – biblioteca base da Adafruit, usada junto com sensores como o BMP280 e DHT11
- **Adafruit_BMP280.h** – biblioteca específica do sensor BMP280
- **DHT.h** – biblioteca usada para ler dados do sensor DHT11

Protocolos:

- Protocolo MQTT– Protocolo de comunicação utilizado para enviar os dados dos sensores para a plataforma na nuvem.

Arquitetura geral do sistema

1. Camada de Sensoriamento (Sensores)

- DHT22 – coleta temperatura e umidade relativa do ar
- BMP280 / BME280 – mede pressão atmosférica
- MQ135 – detecta qualidade do ar (gases nocivos)

2. Camada de Processamento e Comunicação (ESP32)

- Lê os dados dos sensores
- Processa e envia via Wi-Fi para a plataforma web
- Pode ter lógica para envio periódico ou por variação significativa dos dados

3. Camada de Nuvem / Servidor Web

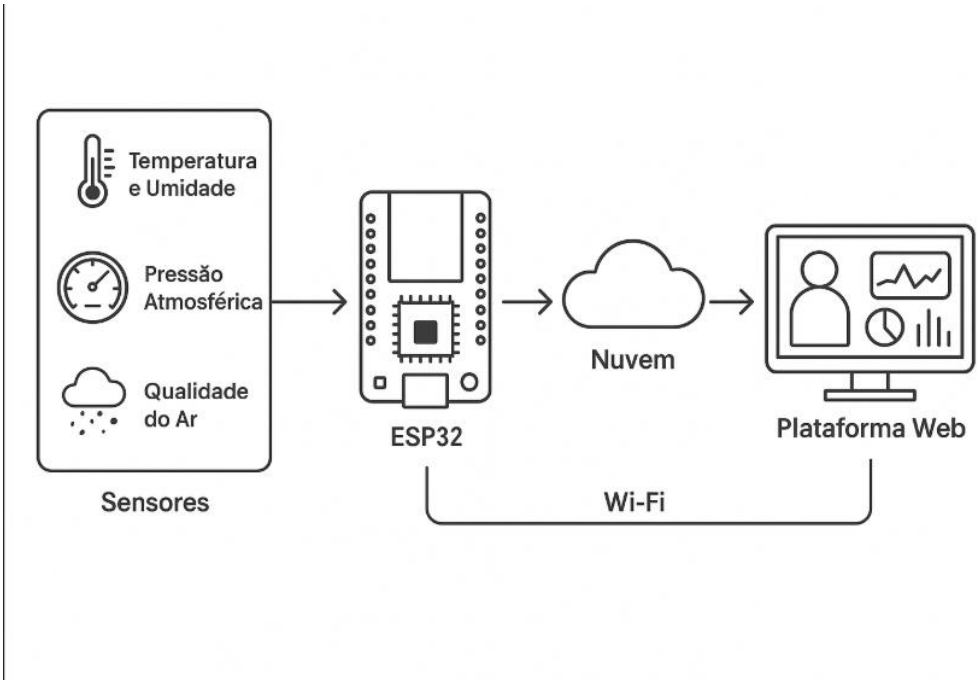
- Recebe os dados do ESP32 (via protocolo MQTT)
- Armazena em banco de dados (Firebase)

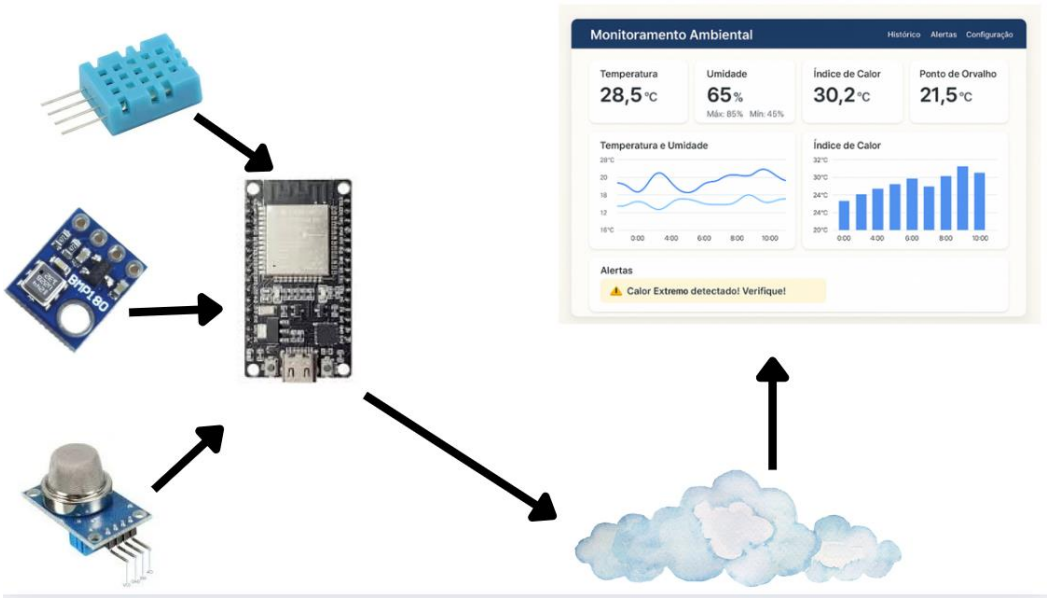
- Disponibiliza uma interface web com dashboards gráficos (Chart.js)

4. Camada de Visualização (Navegador do Usuário)

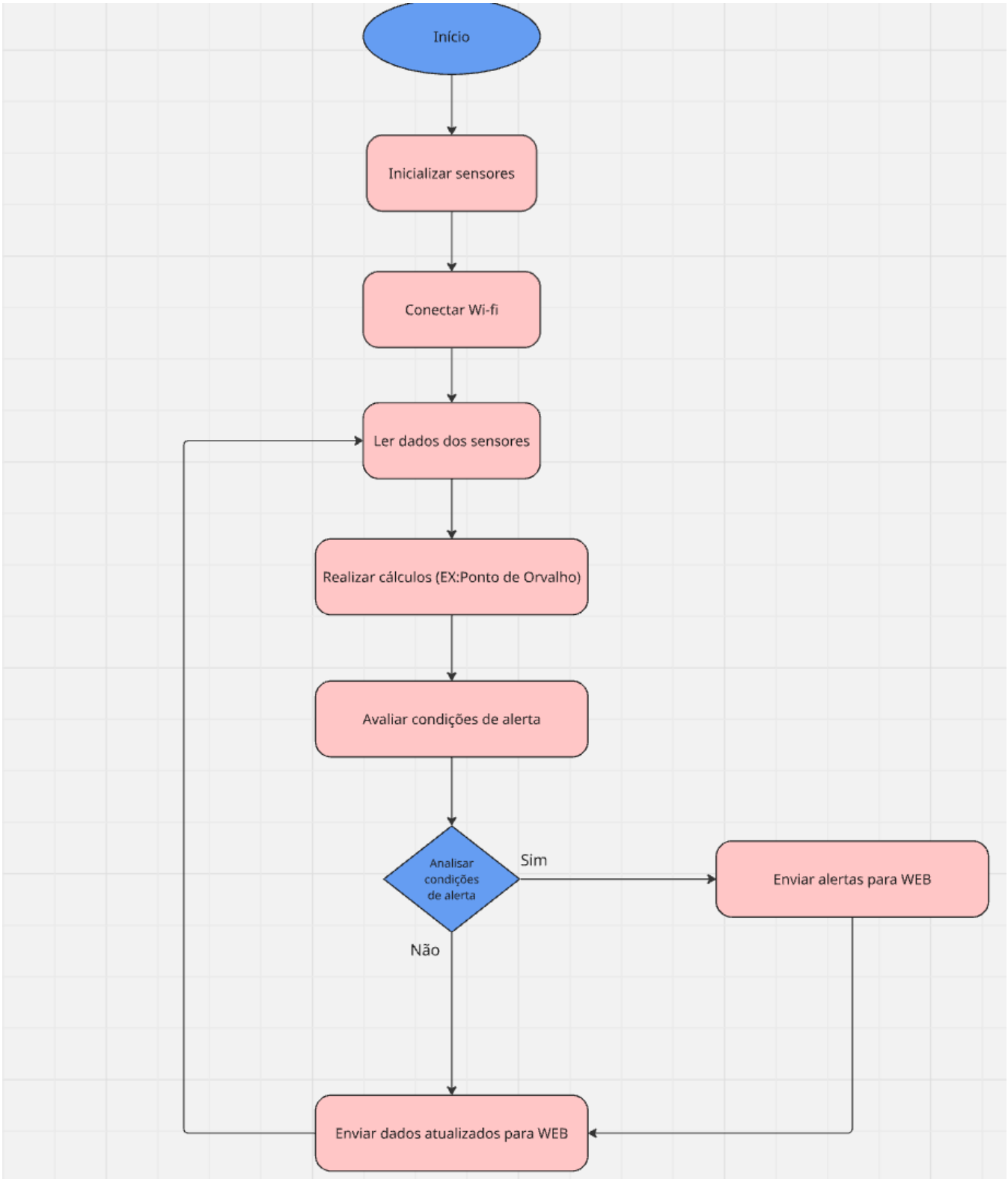
- Interface Web
- Exibe os dados coletados em tempo real (com gráficos, indicadores)

Diagramas do sistema





Fluxo Geral do Sistema



Funcionalidades

1. Exibição em tempo real das leituras atuais:

- Temperatura (°C)
- Umidade (%)
- Pressão Atmosférica (hPa)
- Qualidade do ar (índice ou estado textual: bom, moderado, ruim)

2. Gráficos históricos (linhas ou barras) com:

- Temperatura vs. tempo
- Umidade vs. tempo
- Pressão Atmosférica vs. tempo
- Qualidade do ar vs. tempo

3. Ponto de Orvalho (Dew Point)

- É a temperatura em que o vapor de água começa a condensar no ar.
- Se a temperatura ambiente se aproxima do ponto de orvalho, pode haver formação de neblina, orvalho ou até chuva.

4. Índice de Calor (Heat Index)

- Mostra como o corpo humano percebe a temperatura considerando a umidade do ar.
- Exemplo: 30 °C com alta umidade pode parecer 35 °C para o corpo.

5. Índice de Qualidade do Ar (IQA)

- Mede o quão limpo ou poluído está o ar em determinado local.
- Pode incluir: Material particulado (PM2.5 e PM10), CO_2, ozônio, monóxido de carbono, entre outros.
- Gráficos e alertas ajudam a indicar quando a qualidade do ar está prejudicial à saúde.

6. Sistema de Alertas Inteligentes

- **Alerta de Calor Extremo:** Temperatura ou índice de calor acima de níveis seguros
- **Alerta de Formação de Neblina:** Temperatura ≈ ponto de orvalho e alta umidade
- **Alerta de Umidade Elevada:** Umidade > 90% por tempo contínuo.
- **Alerta de Umidade relativa do ar muito elevada:** Quando se tem um úmido por muito tempo, pode acabar surgindo mofos ou bolor.
- **Alerta de Condensação:** Risco de formação de gotículas em superfícies frias

Cronograma de execução:

Atividade	Descrição da Atividade	Participantes	Data de Início	Data de Término	Status
Definição e Justificativa do Projeto	Definir o problema a ser resolvido e a importância do projeto para sistemas ciberfísicos.	Caio, Juliana, Leonardo, Rafael	31/03/2025	01/04/2025	Concluído
Objetivos e Tecnologias Utilizadas	Estabelecer os objetivos do projeto e listar as tecnologias (hardware, bibliotecas, protocolos) utilizadas.	Caio, Juliana, Leonardo, Rafael	31/03/2025	01/04/2025	Concluído
Arquitetura Geral do Sistema	Criar o diagrama de arquitetura, incluindo fluxos de comunicação entre módulos e a estrutura do sistema.	Caio, Juliana, Leonardo, Rafael	07/04/2025	07/04/2025	Concluído
Pesquisa sobre os sensores	Realizar pesquisa para entender as especificações e funcionamento dos sensores a serem utilizados.	Caio, Juliana, Leonardo, Rafael	07/04/2025	07/04/2025	Concluído
Configuração do ambiente de desenvolvimento	Configurar o ambiente de desenvolvimento necessário para os testes.	Caio, Juliana, Leonardo, Rafael	17/04/2025	17/04/2025	Concluído
Teste do sensor BMP280	Realizar o teste isolado do sensor BMP280 para	Caio, Juliana, Leonardo, Rafael	17/04/2025	17/04/2025	Concluído

	verificar seu funcionamento.				
Teste do sensor DHT11	Realizar o teste isolado do sensor DHT11 para verificar seu funcionamento.	Caio, Juliana, Leonardo, Rafael	17/04/2025	17/04/2025	Concluído
Teste do sensor MQ135	Realizar o teste isolado do sensor MQ135 para verificar seu funcionamento.	Caio, Juliana, Leonardo, Rafael	17/04/2025	17/04/2025	Concluído
Elaboração do relatório final	Elaborar o relatório final do projeto, contendo objetivos, resultados dos testes e conclusões.	Caio, Juliana, Leonardo, Rafael	31/03/2025	21/04/2025	Concluído
Criação do Git para Organização da Equipe	Criar repositório Git, registrar todas as atividades realizadas e promover a colaboração entre a equipe.	Caio, Juliana, Leonardo, Rafael	18/04/2025	28/04/2025	Concluído
Desenvolvimento do Sistema de Comunicação (MQTT)	Implementar e testar a comunicação entre o ESP32 e a plataforma de nuvem via MQTT.	Caio, Juliana, Leonardo, Rafael	05/05/2025	*	A fazer
Desenvolvimento da Interface Web	Criar a interface web com os gráficos e indicadores.	Caio, Juliana, Leonardo, Rafael	05/05/2025	*	A fazer
Implementação dos Alertas Inteligentes	Desenvolver a lógica de alertas para condições críticas (calor extremo, baixa qualidade do ar, etc.).	Caio, Juliana, Leonardo, Rafael	05/05/2025	*	A fazer
Testes de Integração do Sistema	Realizar testes de integração de todo o sistema	Caio, Juliana, Leonardo, Rafael	05/05/2025	*	A fazer

	(sensores, ESP32, plataforma na nuvem, interface web).				
Ajustes Finais e Testes de Estabilidade	Realizar ajustes finais com base nos testes e garantir que o sistema seja estável e confiável.	Caio, Juliana, Leonardo, Rafael	12/05/2025	*	A fazer

Testes Isolados:

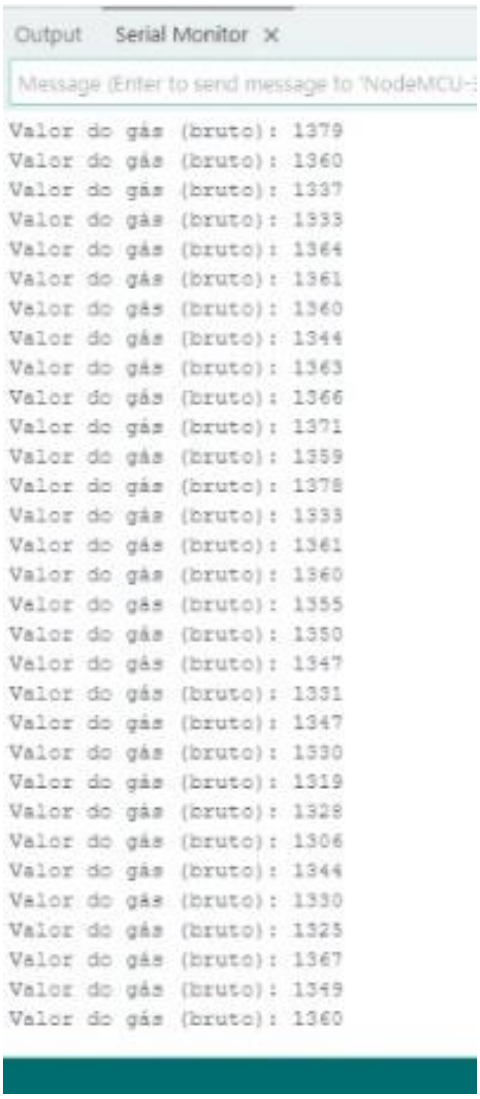
RELATÓRIO DE TESTE - SENSOR MQ135

- **Data:** 17/04/2025
- **Hora inicio do teste:** 17:21
- **Plataforma:** ESP32
- **Linguagem:** C (via Arduino IDE)
- **Objetivo do teste:**
  Verificar se o sensor MQ135 está funcionando corretamente ao ser conectado ao ESP32.
- **Componentes utilizados:**
  - ESP32
  - Sensor MQ135
  - Jumper MxF de conexão
  - Protoboard
- **Código utilizado:**

```
#define MQ135_PIN 34


void setup() {
 Serial.begin(115200);
 Serial.println("Teste MQ135 - Qualidade do Ar");
}


void loop() {
 int analogValue = analogRead(MQ135_PIN);
 Serial.print("Valor do gás (bruto): ");
 Serial.println(analogValue);


 delay(2000);
```

}

- **Resultados obtidos:**



- **Conclusão:**
    - O sensor MQ135 está operando normalmente. As leituras apresentaram variações coerentes, com uma diferença aproximada de 4 dígitos entre cada medição do valor bruto de gás. O sensor ainda não foi calibrado para fornecer valores em PPM (partes por milhão), mas os dados brutos indicam que ele está funcional e respondendo adequadamente às variações do ambiente.

RELATÓRIO DE TESTE - SENSOR BMP280

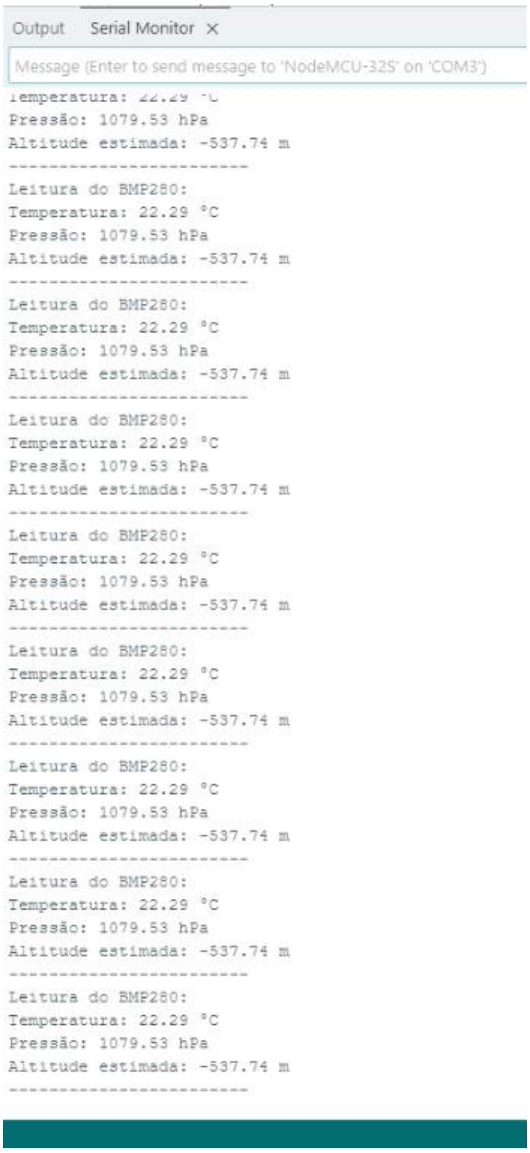
- **Data:** 17/04/2025

- **Hora inicio do teste:** 17:41
- **Plataforma:** ESP32
- **Linguagem:** C (via Arduino IDE)
- **Objetivo do teste:**
  Verificar se o sensor BMP280 está funcionando corretamente ao ser conectado ao ESP32.
- **Componentes utilizados:**
  - ESP32
  - Sensor BMP280
  - Jumper MxF de conexão
  - Protoboard
- **Código utilizado:**

```
#include <Wire.h>
#include <Adafruit_Sensor.h>
#include <Adafruit_BMP280.h>

Adafruit_BMP280 bmp;

void setup() {
 Serial.begin(115200);
 delay(1000);

 if (!bmp.begin(0x77)) {
  Serial.println("Erro ao encontrar o BMP280! Verifique as conexões.");
  while (1);
 }

 Serial.println("BMP280 iniciado com sucesso!");
}
```

```
void loop() {
  float temperatura = bmp.readTemperature();
  float pressao = bmp.readPressure() / 100.0F;
  float altitude = bmp.readAltitude(1013.25);  do mar (ajuste conforme necessário)

  Serial.println("Leitura do BMP280:");
  Serial.print("Temperatura: ");
  Serial.print(temperatura);
  Serial.println(" °C");

  Serial.print("Pressão: ");
  Serial.print(pressao);
  Serial.println(" hPa");

  Serial.print("Altitude estimada: ");
  Serial.print(altitude);
  Serial.println(" m");

  Serial.println("-------------------------");
  delay(2000);
}
```

- **Resultados obtidos:**



- **Conclusão:**
    - O sensor BMP280 apresentou funcionamento adequado nas medições de temperatura e pressão atmosférica, com valores coerentes ao ambiente de teste. No entanto, a leitura da altitude apresentou um valor negativo aproximado de -500 metros, o que não condiz com a altitude real do local. Esse comportamento pode estar relacionado à ausência de calibração baseada na pressão ao nível do mar (pressão de referência), necessária para que o cálculo da altitude seja preciso. Apesar disso, o sensor demonstrou estar operando corretamente, sendo recomendada apenas a configuração adequada da pressão de referência para obter valores realistas de altitude.

RELATÓRIO DE TESTE - SENSOR DHT22

- **Data:** 17/04/2025
- **Hora inicio do teste:** 18:24
- **Plataforma:** ESP32
- **Linguagem:** C (via Arduino IDE)
- **Objetivo do teste:**
  Verificar se o sensor DHT22 está funcionando corretamente ao ser conectado ao ESP32.
- **Componentes utilizados:**
  - ESP32
  - Sensor DHT22
  - Jumper MxF de conexão
  - Protoboard
- **Código utilizado:**

```
#include "DHT.h"

#define DHTPIN 4
#define DHTTYPE DHT11

DHT dht(DHTPIN, DHTTYPE);

void setup() {
 Serial.begin(115200);
 dht.begin();
 Serial.println("Teste de Sensor DHT11 - ESP32");
}

void loop() {
 delay(2000);
```

```
  float h = dht.readHumidity();
  float t = dht.readTemperature();

  if (isnan(h) || isnan(t)) {
   Serial.println("Falha na leitura do sensor DHT11!");
   return;
  }

  Serial.print("Umidade: ");
  Serial.print(h);
  Serial.print(" %  |  Temperatura: ");
  Serial.print(t);
  Serial.println(" °C");
 }
```

- **Resultados obtidos:**

- **Conclusão:**
    - O sensor DHT11 apresentou medições coerentes de temperatura e umidade relativa do ar. Durante o teste, a leitura da umidade variou aproximadamente 1%, o que está dentro do esperado para pequenas flutuações ambientais. A temperatura manteve-se praticamente constante, demonstrando estabilidade e precisão nas condições em que o teste foi realizado. Os resultados indicam que o sensor está operando corretamente.

Link do GitHub

https://github.com/EstacaoMeteorologica3c/EstacaoMeteorologica