



Desenvolvimento de sensores e atuadores IoT – Parte 2

Aula síncrona (15/07/2025)

Prof. Wilton Lacerda Silva

Executores:



Coordenação:



Iniciativa:



Sumário

- Objetivos
- Sensores de pressão, umidade e temperatura
- Exemplos de aplicação na BitDogLab.

Objetivos

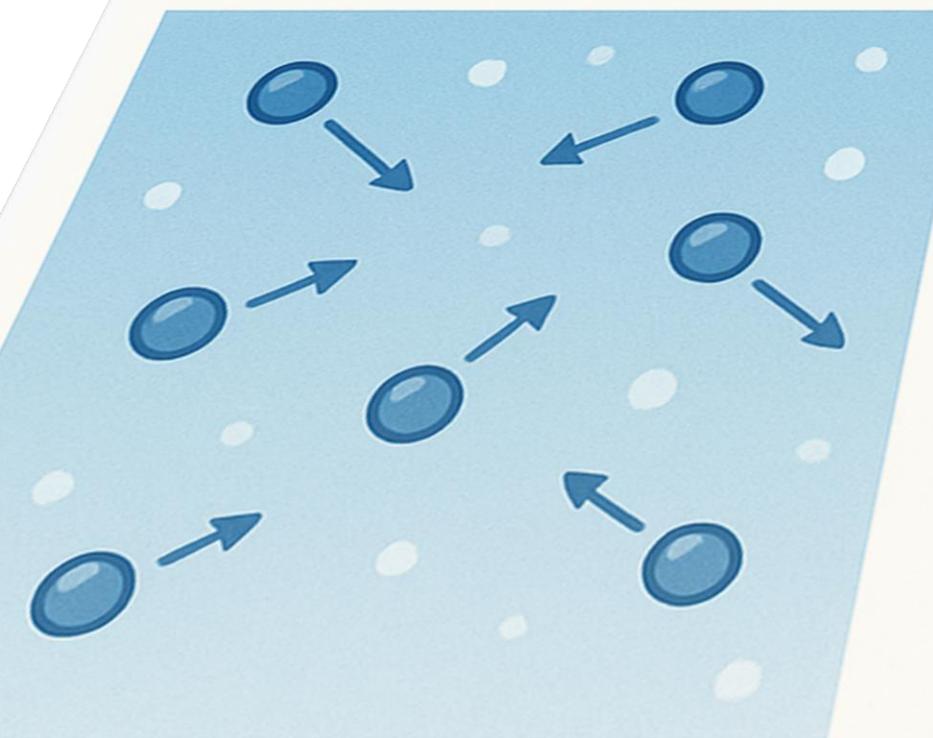
- Conceituar algumas grandezas físicas.
- Integrar sensores específicos na BitDogLab.
- Revisão do protocolo de comunicação I2C.
- Desenvolver alguns exemplos para fixação dos conceitos e realizar atividades práticas com sensores.



Conceitos: Temperatura

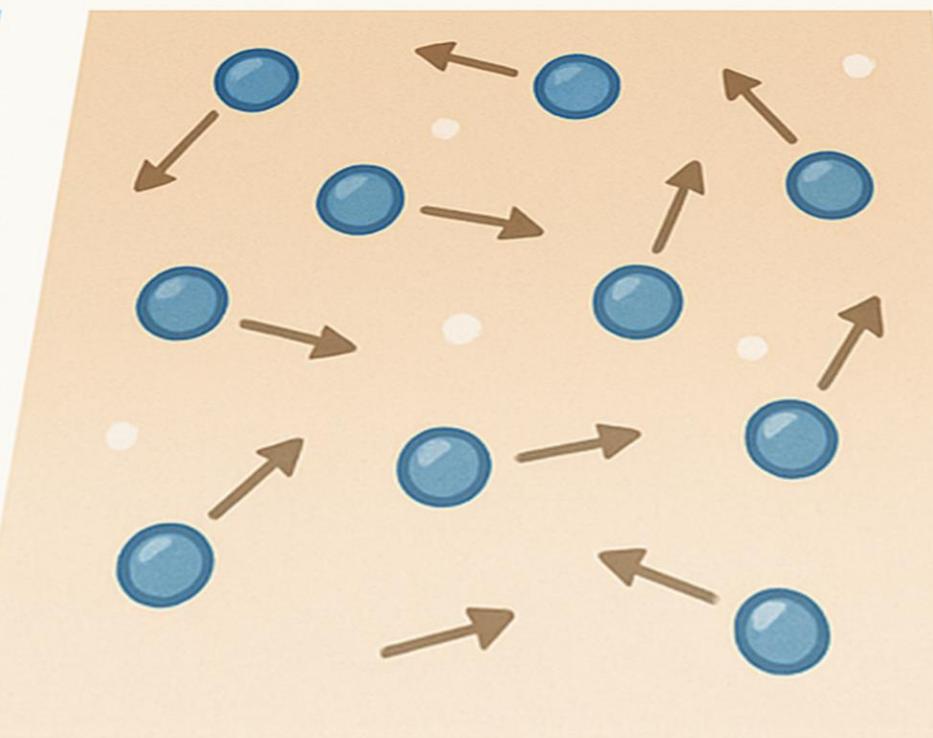
temperatura → agitação térmica de moléculas, átomos ou partículas de um corpo

AR FRIO



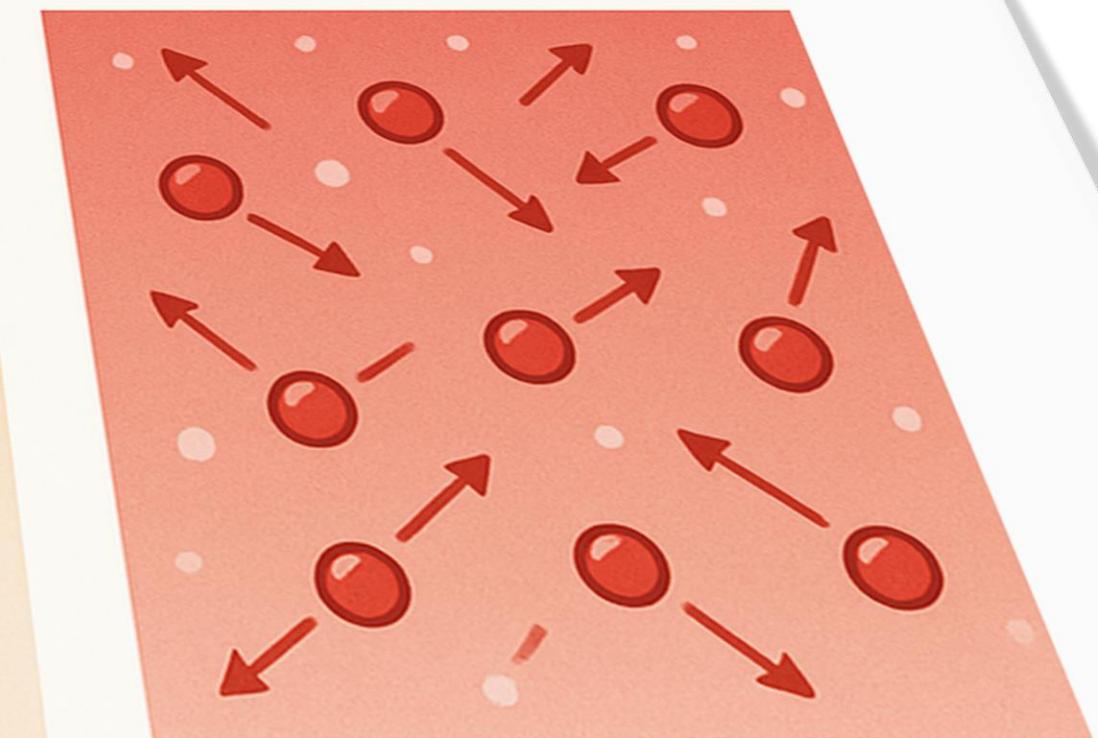
0 °C

AR MORNO



25 °C

AR QUENTE



100 °C

Conceitos: Temperatura

A temperatura é uma das variáveis mais usadas na indústria de controle e processos.

A medição de temperatura é ponto de interesse da ciência há muitos anos. O corpo humano é um péssimo termômetro, pois só consegue diferenciar o que está frio ou quente em relação à sua própria temperatura. Portanto com o passar dos tempos o homem começou a criar aparelhos que o auxiliassesem nesta tarefa.

Veja algumas referências históricas sobre o estudo de medição de temperatura em:

<https://www.instrumatic.com.br/medicao-da-temperatura>



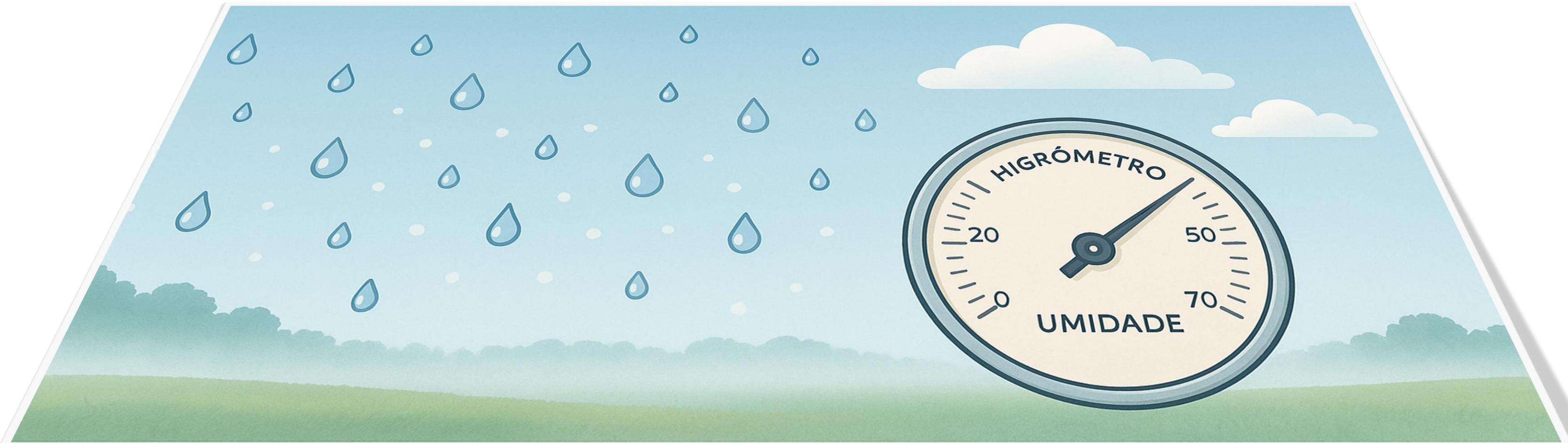
Temperatura é uma medida da energia cinética média das partículas de um sistema, indicando o quanto quente ou frio ele está. É uma propriedade intensiva, medida em escalas como Celsius ($^{\circ}\text{C}$), Kelvin (K) ou Fahrenheit ($^{\circ}\text{F}$).

Escalas de temperatura:

- **Celsius:** 0°C (gelo derretendo), 100°C (água fervendo ao nível do mar).
- **Kelvin:** Escala absoluta, onde $0\text{ K} = -273,15^{\circ}\text{C}$ (zero absoluto).
- **Fahrenheit:** Menos comum em ciências, mas usada em alguns países (ex.: $32^{\circ}\text{F} = 0^{\circ}\text{C}$, $212^{\circ}\text{F} = 100^{\circ}\text{C}$).

A temperatura influencia processos físicos, químicos e biológicos, como evaporação, reações químicas e conforto humano.

Conceitos: umidade



Umidade relativa: quantidade de vapor d'água no ar em relação ao máximo possível.

Conceitos: umidade

Umidade é a quantidade de **vapor de água presente no ar**. É uma medida essencial em muitos contextos industriais e ambientais.

Medições precisas são essenciais em indústrias como:
Alimentos, medicamentos, madeira, papel, eletrônicos.

Umidade excessiva pode causar:
Corrosão, mofo, perda de eficácia em produtos sensíveis.

Umidade relativa (%UR ou %RH) é a relação entre a quantidade atual de vapor de água no ar e a quantidade máxima que o ar poderia conter a uma dada temperatura, sem que ocorra condensação.

$$\%RH = \left(\frac{\text{pressão de vapor atual}}{\text{pressão de saturação}} \right) \times 100$$



Tipo de Umidade	Definição
Umidade absoluta	Massa de vapor de água por volume de ar (g/m³).
Umidade específica	Massa de vapor de água por massa total de ar úmido (g/kg).
Umidade relativa (%RH)	Relação entre a pressão parcial de vapor e a pressão de saturação a uma dada temperatura.

Conceitos: umidade observações:

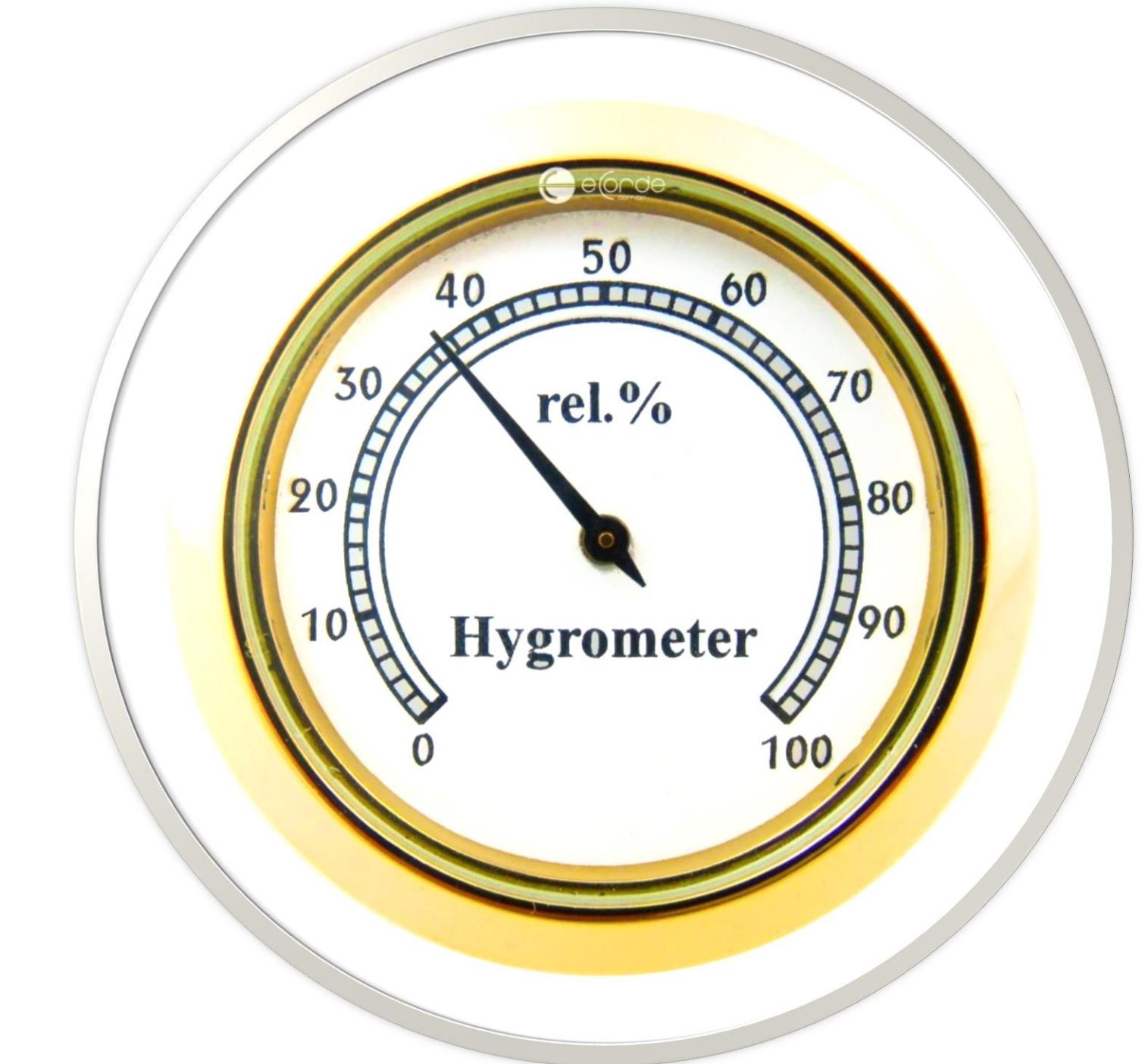
Variação com temperatura: O ar quente pode conter mais vapor d'água, enquanto o ar frio tem menor capacidade, afetando a umidade relativa. Isto implica que a temperatura interfere diretamente na medição da umidade.

Exemplo prático:

Se o ar a 30°C contém 15 g/m^3 de vapor d'água e a capacidade máxima é 30 g/m^3 , a UR é 50%.

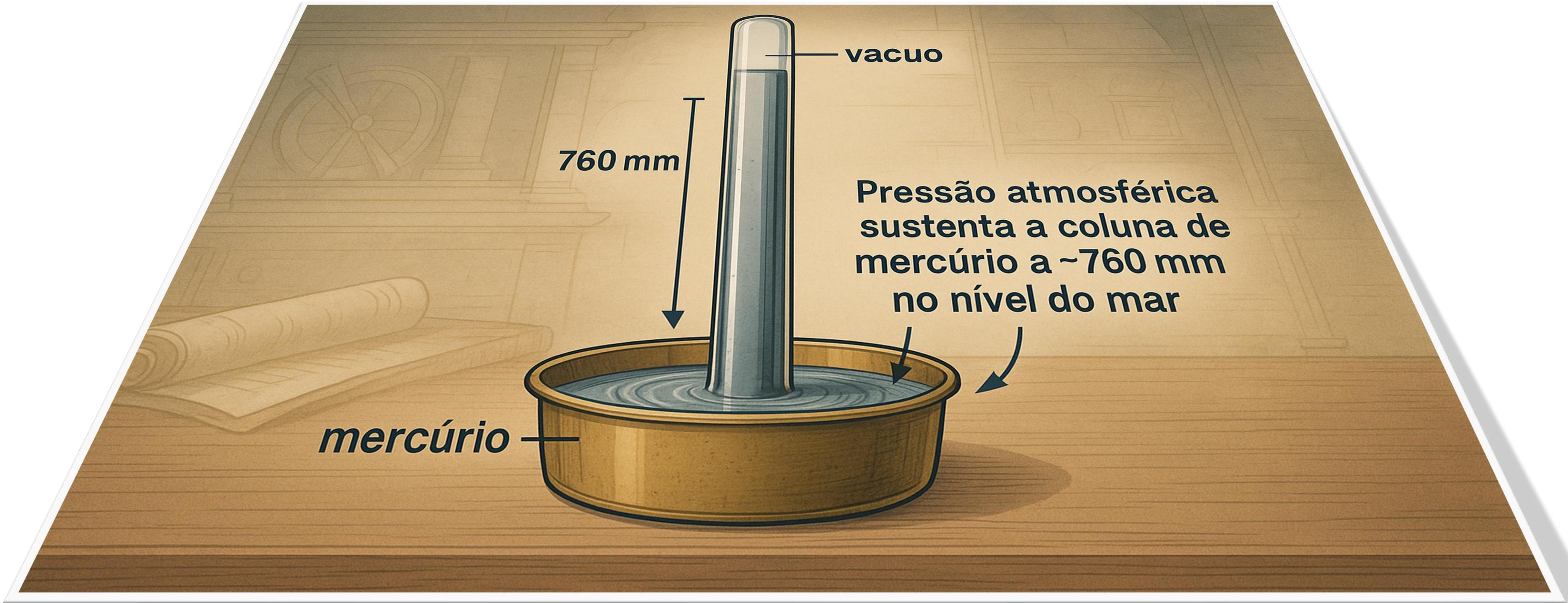
Se a temperatura cair para 20°C , a capacidade máxima pode diminuir para, digamos, 20 g/m^3 .

Com os mesmos 15 g/m^3 de vapor, a UR aumenta para 75%.



Instrumentos como higrômetros (análogicos ou digitais) são usados para medir **umidade relativa**.

Conceitos: Pressão atmosférica



Conceitos: Pressão atmosférica

Pressão atmosférica é a força exercida pela coluna de ar da atmosfera sobre uma unidade de área na superfície terrestre. É causada pelo peso do ar acima de um determinado ponto.

Ela é medida em Pascal (**Pa**), mas outras unidades comuns incluem **hPa** (hectopascal), **mmHg** (milímetros de mercúrio) e **atm** (atmosferas).

$$1 \text{ atm} \approx 1013,25 \text{ hPa} \approx 101325 \text{ Pa}$$

A **pressão atmosférica** diminui com o aumento da altitude, pois há menos ar acima. Essa relação é fundamental para estimar altitudes com base em medições de pressão.



Conceitos: Pressão atmosférica

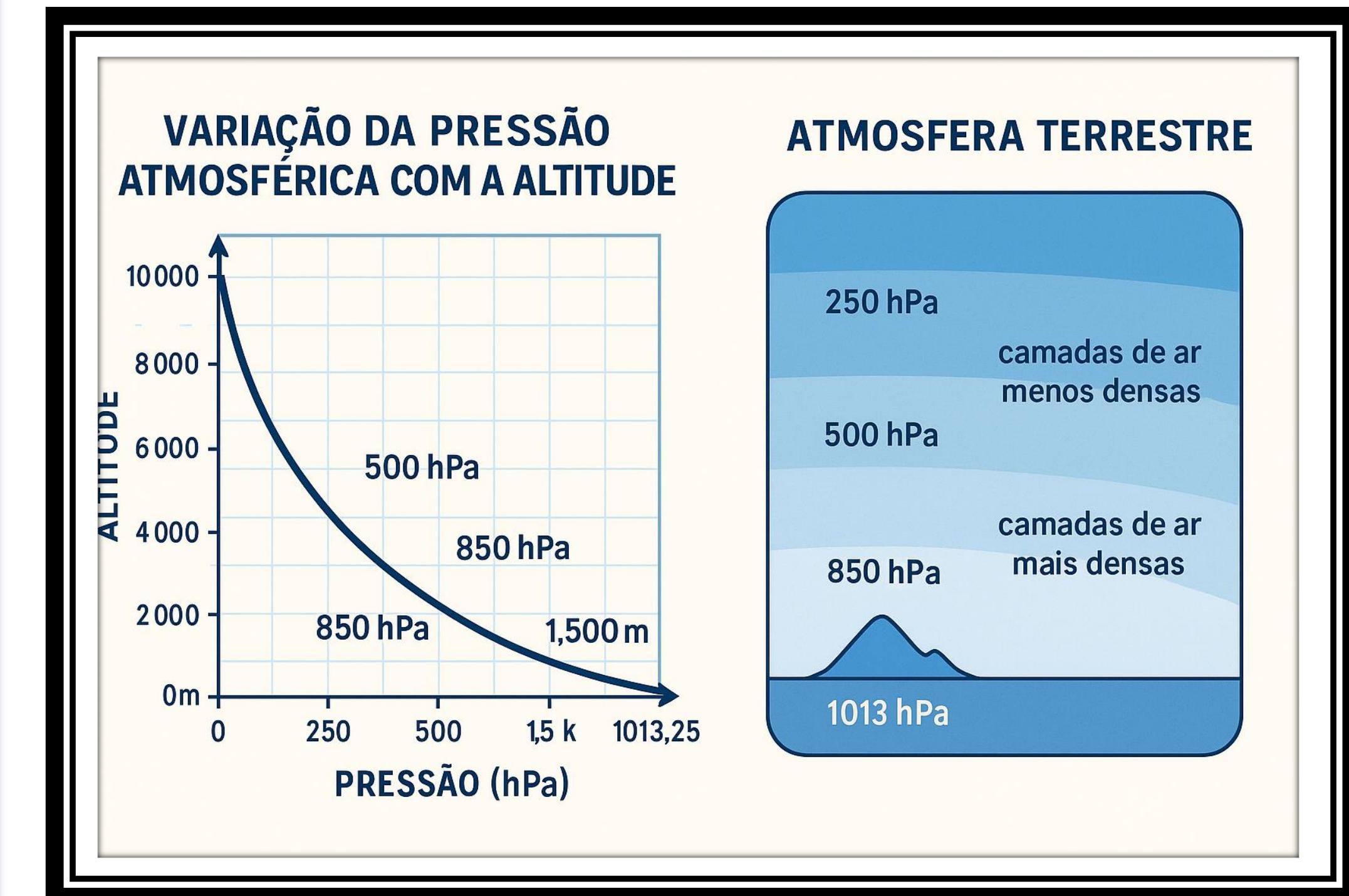
Relação pressão-altitude:

A pressão atmosférica diminui de forma aproximadamente exponencial com a altitude. A equação barométrica é usada para estimar a altitude com base na pressão medida:

$$h = \left(1 - \left(\frac{P}{P_0} \right)^{0,1903} \right) \times 44330$$

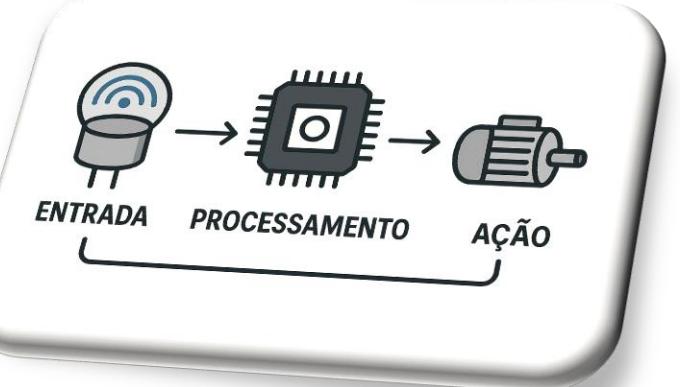
Onde:

- h : altitude em metros.
- P : pressão medida pelo sensor (em hPa).
- P_0 : pressão de referência no nível do mar.

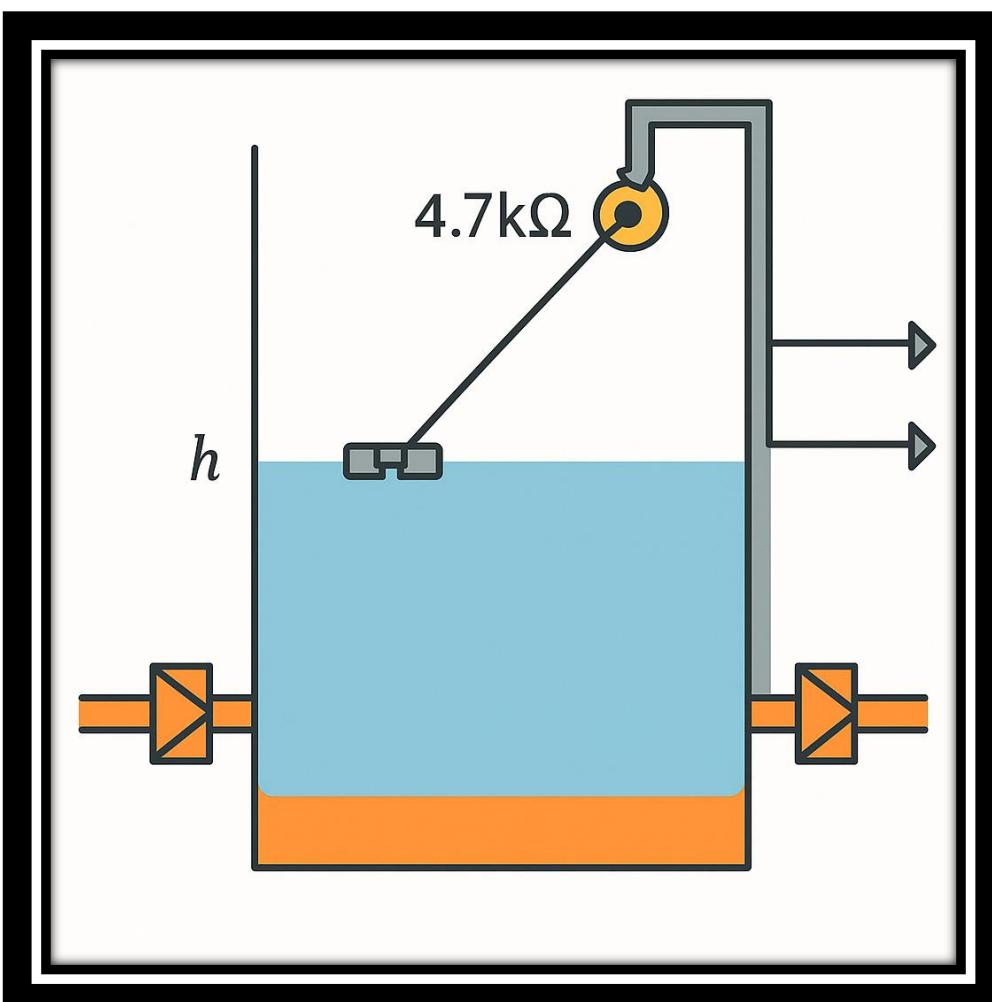
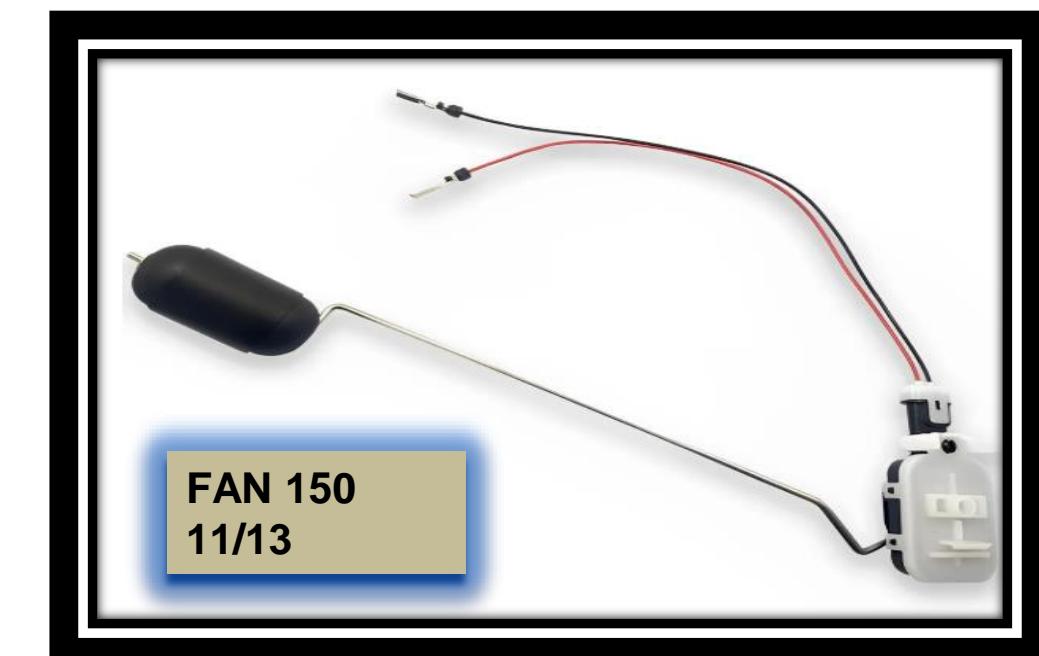
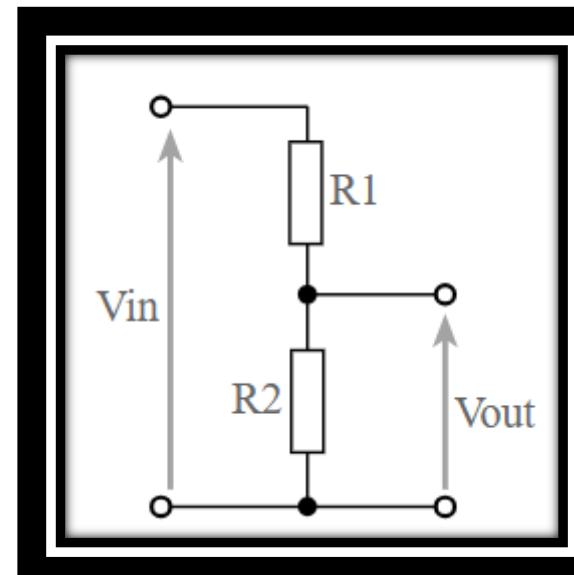
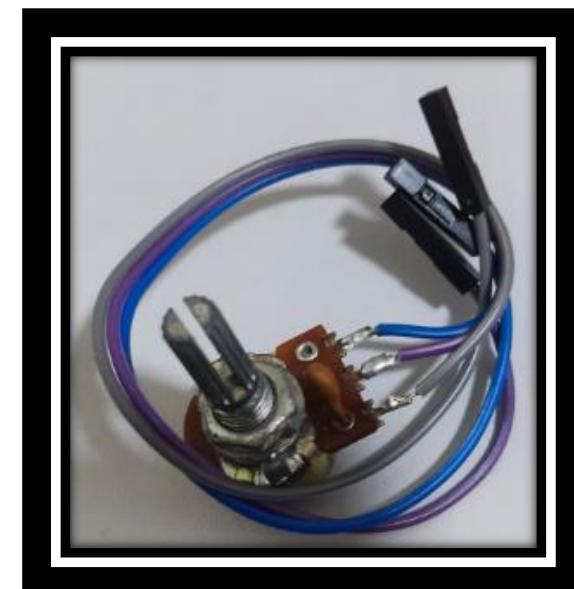
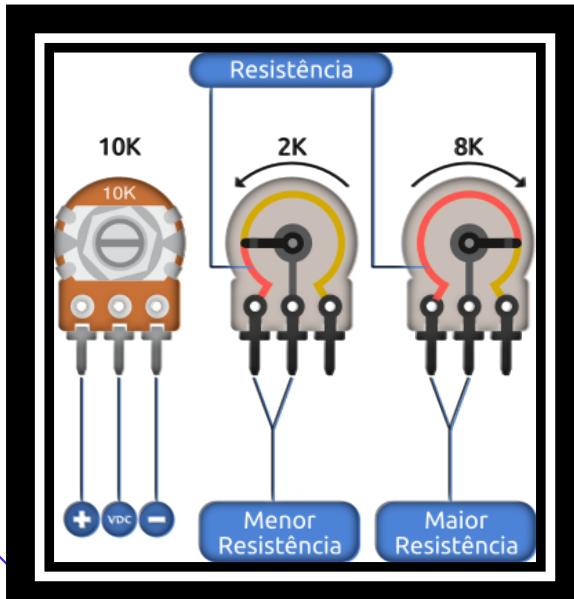


Revisão: Conceito de sensor

Do ponto de vista da **eletrônica, instrumentação e automação**, um sensor é um dispositivo que detecta uma grandeza física ou química e a converte em um sinal elétrico que pode ser medido, interpretado e processado por sistemas eletrônicos, como **microcontroladores, CLPs ou sistemas de aquisição de dados**.



Sensor é um **transdutor** que converte uma variável do ambiente (como temperatura, pressão, luz, umidade, força, movimento, etc.) em um sinal elétrico utilizável por um sistema de controle ou monitoramento.



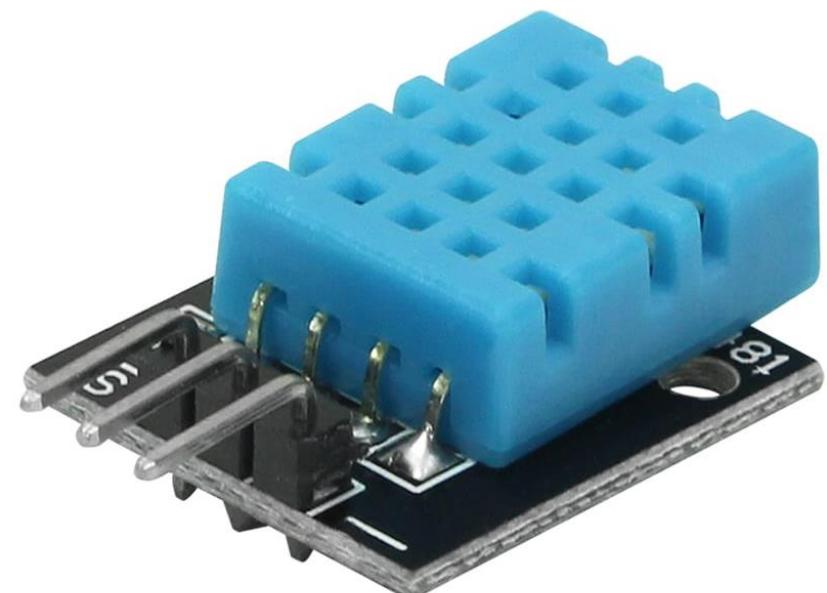
Sensor: umidade

O que é um sensor de umidade?

Um sensor de umidade é um dispositivo eletrônico que mede a quantidade de **umidade presente no ar** (umidade relativa) ou **em materiais** (umidade do solo, por exemplo).

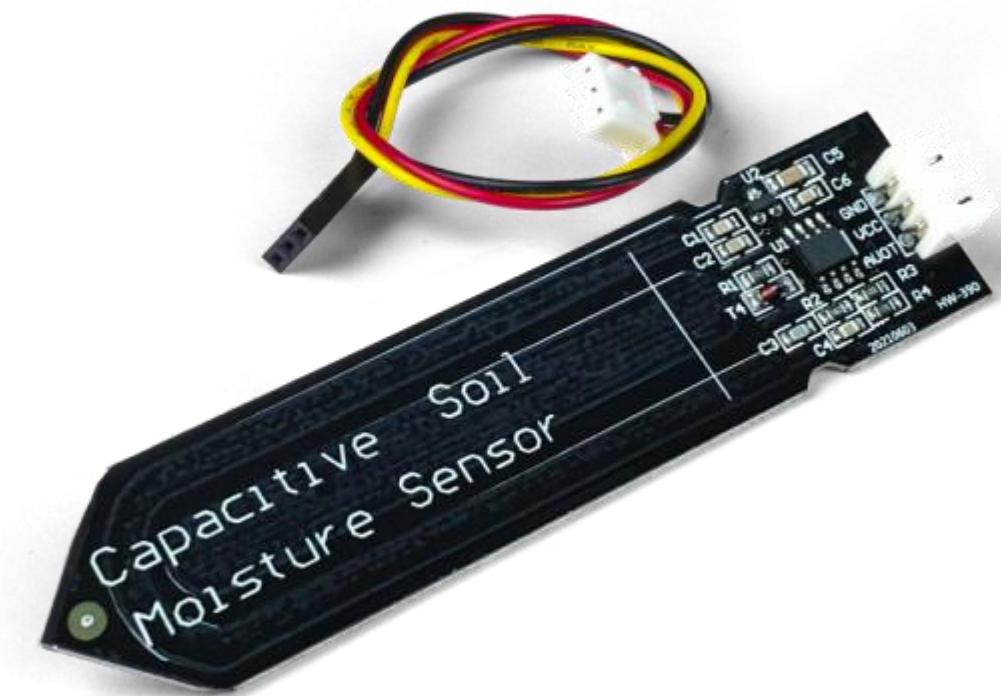
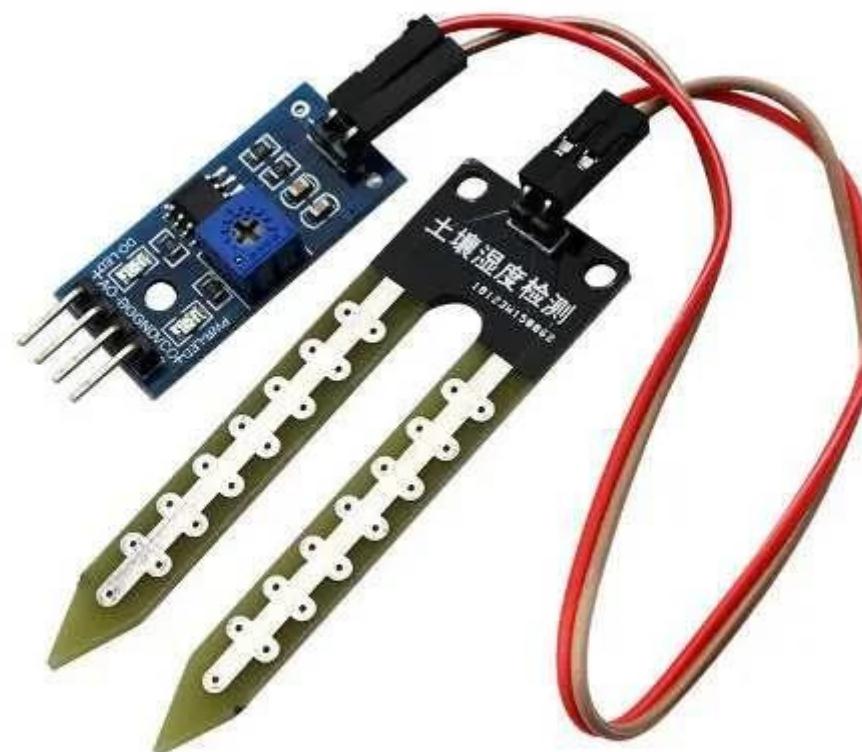


<https://brasilescola.uol.com.br/geografia/umidade-ar.htm>



Sensores de Umidade do Solo:

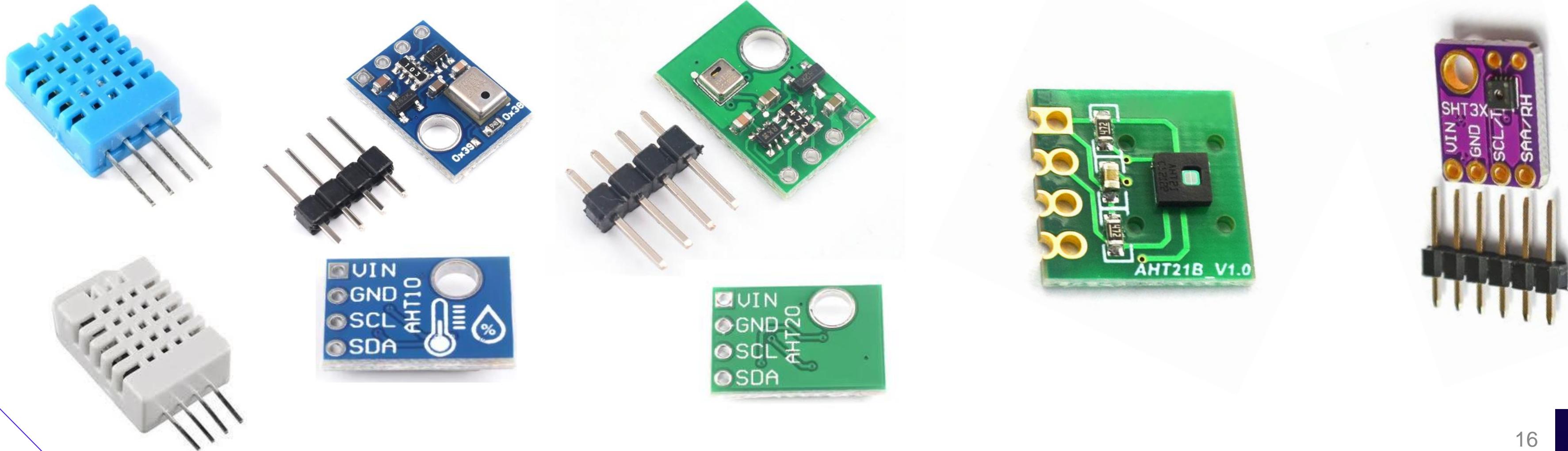
- Detectam a **presença de água** no solo.
- Úteis em projetos de irrigação automática.
- Podem ser resistivos (mais simples) ou capacitivos (mais duráveis).
- Exemplos: **Sensor YL-69** (resistivo), **Sensor capacitivo v2.0.**, e **HD-38**



Sensor: umidade

Sensores de umidade Umidade Relativa (HR ou RH - Relative Humidity):

1. Medem a **quantidade de vapor de água no ar** em relação à quantidade máxima que o ar pode conter a uma certa temperatura.
2. Expressa em %.
3. Exemplos: **DHT11, DHT22, AHT10, AHT20, AHT21, SHT31, etc.**

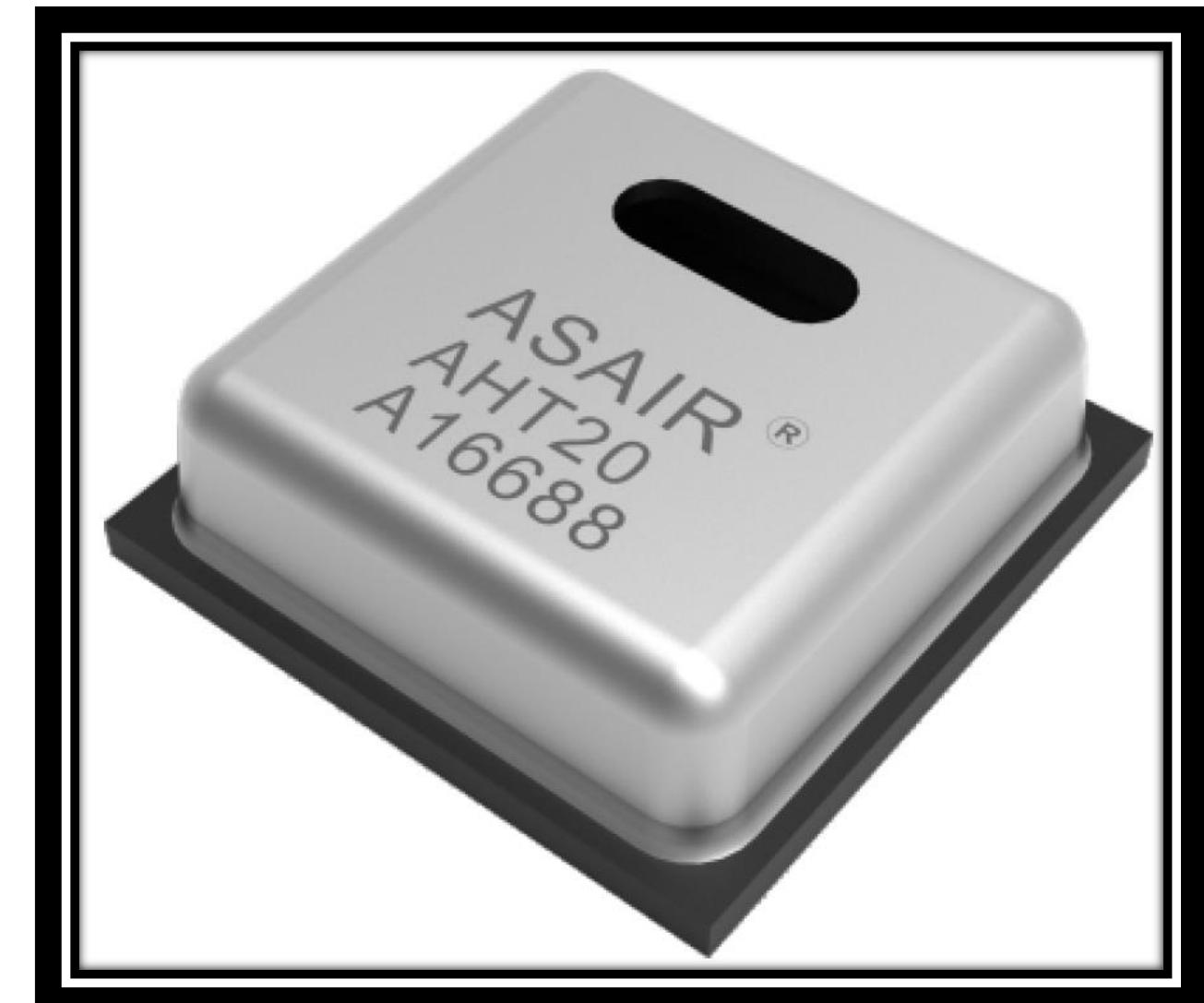


Informações do datasheet.

O **AHT20**, como uma nova geração de sensores de temperatura e umidade, estabeleceu um novo padrão em termos de tamanho e inteligência. Ele está encapsulado em um invólucro plano sem terminais de duas fileiras, adequado para soldagem por refusão, com uma base de **3 x 3 mm** e altura de **1,0 mm**.

O sensor fornece sinais digitais calibrados no formato **padrão I2C**. O AHT20 é equipado com um **chip ASIC** de novo design, um elemento sensor de umidade capacitivo em semicondutor **tipo MEMS** aprimorado e um elemento sensor de temperatura padrão integrado ao chip.

Cada sensor é calibrado e testado, com o número do lote impresso na superfície do produto.



Explicação técnica sobre ASIC

ASIC é a sigla para **Application-Specific Integrated Circuit**, ou seja, **Circuito Integrado de Aplicação Específica**.

É um chip projetado para **executar uma função específica**, com alta eficiência e baixo consumo de energia.

No caso do **AHT20**, o **ASIC** é responsável por:

- Controlar internamente os sensores de temperatura e umidade.
- Realizar a **leitura precisa dos sinais** dos sensores.
- **Calibrar automaticamente** os dados antes de enviá-los.
- Formatar a resposta em **padrão digital I2C**, pronta para o microcontrolador ler.



O **ASIC** é o "cérebro" do sensor: ele recebe os dados internos, trata tudo automaticamente, e já manda as medições prontas Raspberry Pi Pico.

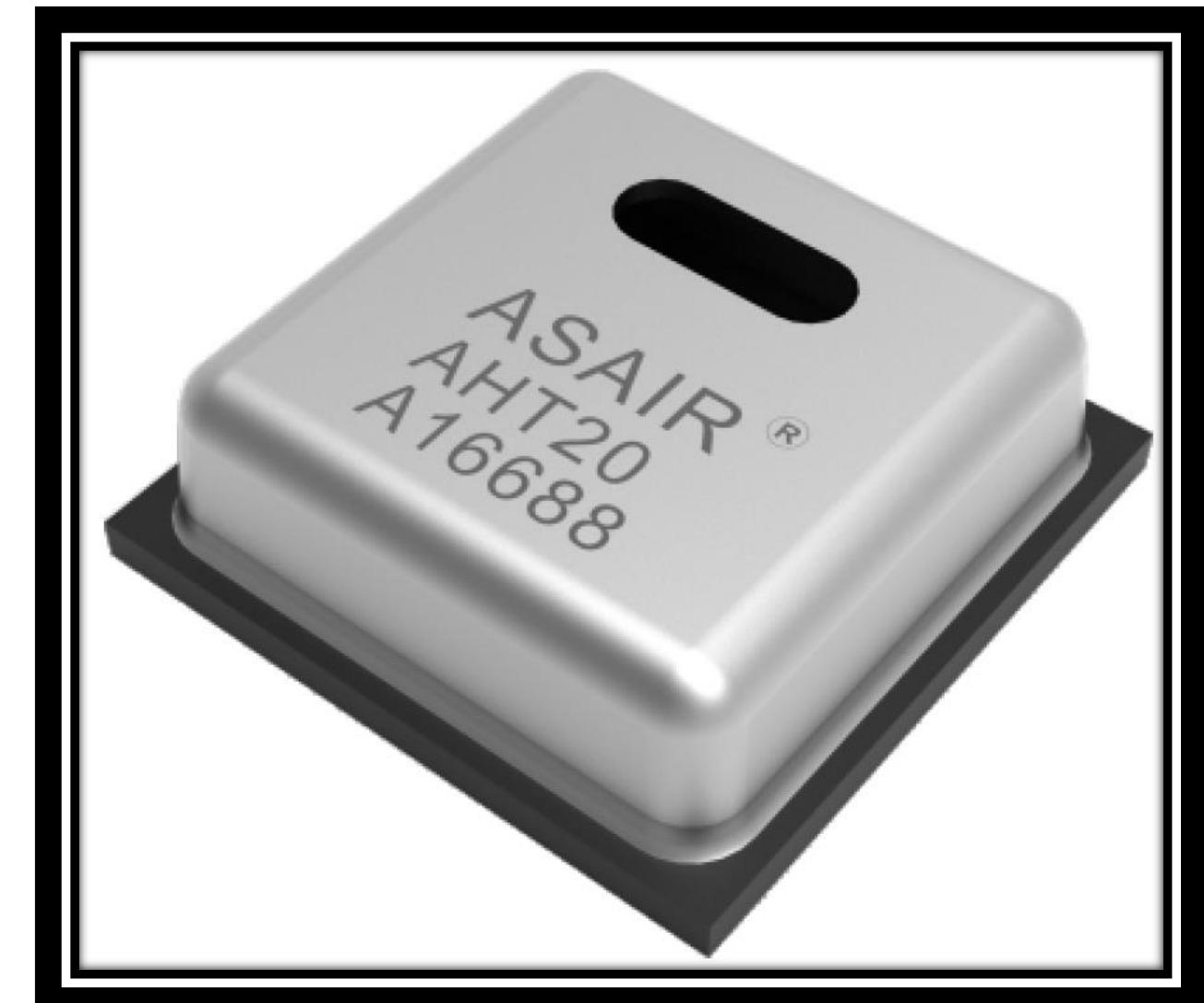
Tipo de chip	Característica principal	Exemplo de uso
ASIC	Projetado para uma aplicação única	Sensor AHT20 (umidade/temperatura), BMP280 (pressão/temperatura), LED RGB (WS2812B)
FPGA	Reprogramável após fabricação	Prototipagem de hardware digital, sistemas embarcados
MCU (microcontrolador)	Programável, multifuncional	RP2 em automação residencial

Explicação técnica sobre MEMS

Um **semicondutor MEMS** (*Micro-Electro-Mechanical Systems*) combina **circuitos eletrônicos em silício** (como num chip) com **partes móveis microscópicas** (como placas, pontes ou sensores).

No caso do sensor **AHT20**, o elemento de **umidade** é **capacitivo** e feito usando tecnologia **MEMS**:

- O sensor possui duas microplacas condutoras.
- Entre elas, há um material sensível à umidade (como um polímero).
- Quando a umidade do ar muda, esse material muda suas propriedades elétricas, alterando a capacidade, e isso é lido pelo sensor.



MEMS (*Micro-Electro-Mechanical Systems*) é uma tecnologia usada para criar sensores **muito pequenos e precisos**, com **peças microscópicas** que reagem ao ambiente.

Vantagens dos Sensores MEMS

1. Compactos: Devido à miniaturização, os sensores de umidade MEMS são extremamente pequenos, tornando-os ideais para aplicações onde o espaço é uma restrição.

2. Baixo consumo de energia: Esses sensores requerem muito pouca energia para operar, tornando-os ideais para dispositivos alimentados por bateria ou aplicações de baixo consumo.

3. Rápida resposta: Os sensores MEMS reagem rapidamente às mudanças na umidade, fornecendo leituras em tempo real.

4. Durabilidade: Por serem construídos usando técnicas de micro fabricação, os sensores MEMS são robustos e podem operar em uma variedade de ambientes desafiadores.

Aplicações Típicas dos Sensores de Umidade

Os sensores de umidade MEMS encontraram uso em uma ampla gama de aplicações devido à sua precisão e tamanho compacto. Alguns exemplos incluem:

- **Dispositivos Wearables:** Em relógios inteligentes e rastreadores de fitness, para monitorar condições ambientais e ajudar a otimizar o conforto do usuário.

- **Eletrodomésticos:** Em geladeiras, ar-condicionados e umidificadores para garantir condições ideais e economia de energia.

- **Agricultura:** Para monitorar as condições do solo e otimizar a irrigação.

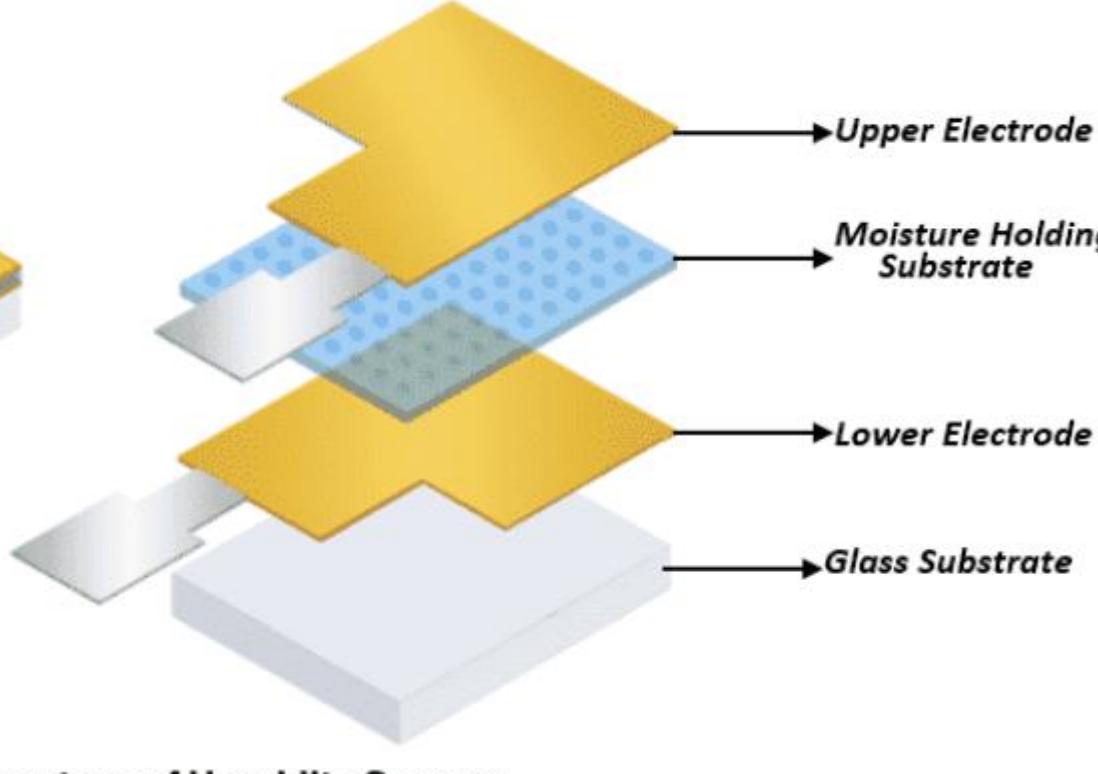
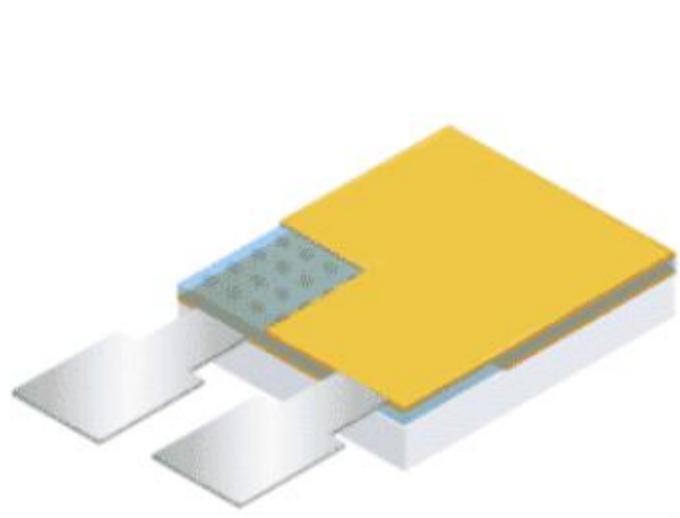
- **Indústria:** No controle de processos, armazenamento de produtos e monitoramento de ambientes críticos.

- **Telecomunicações:** Em centrais de servidores para monitorar e controlar a umidade e evitar danos aos equipamentos.

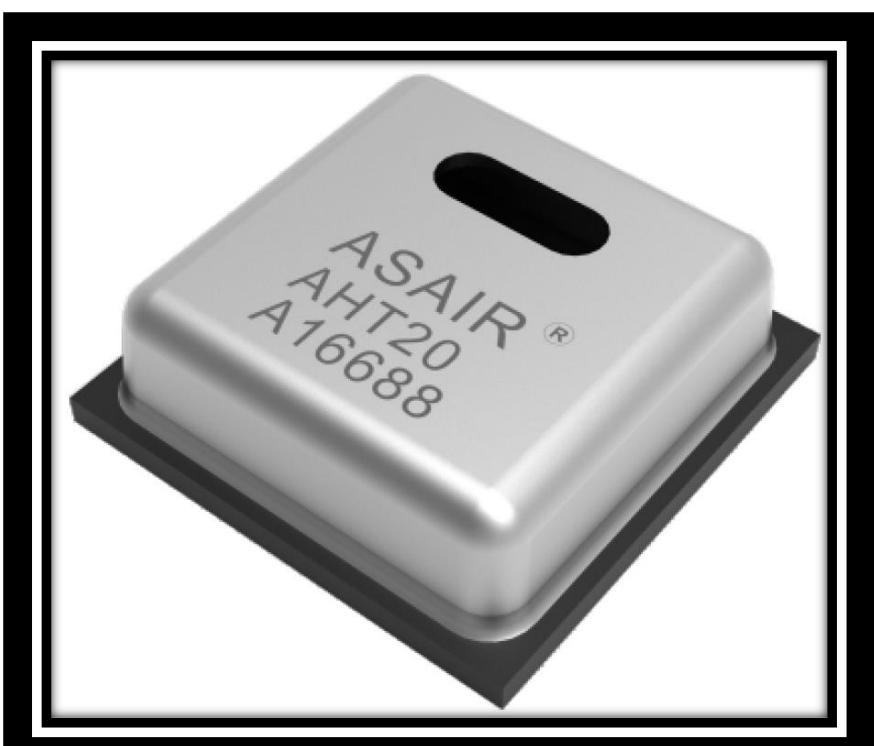
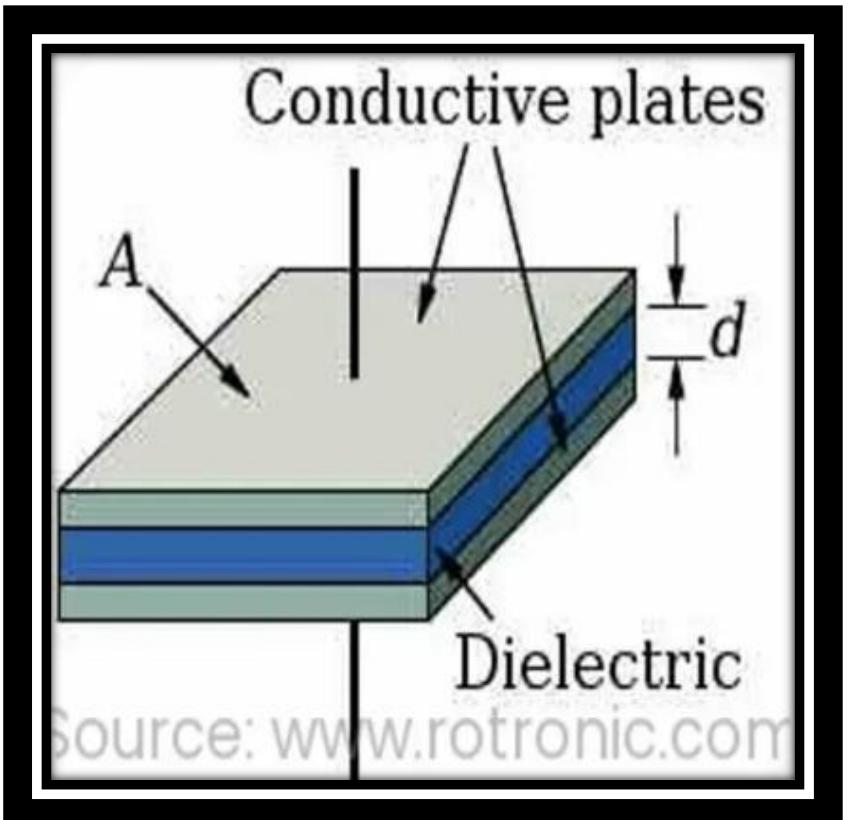
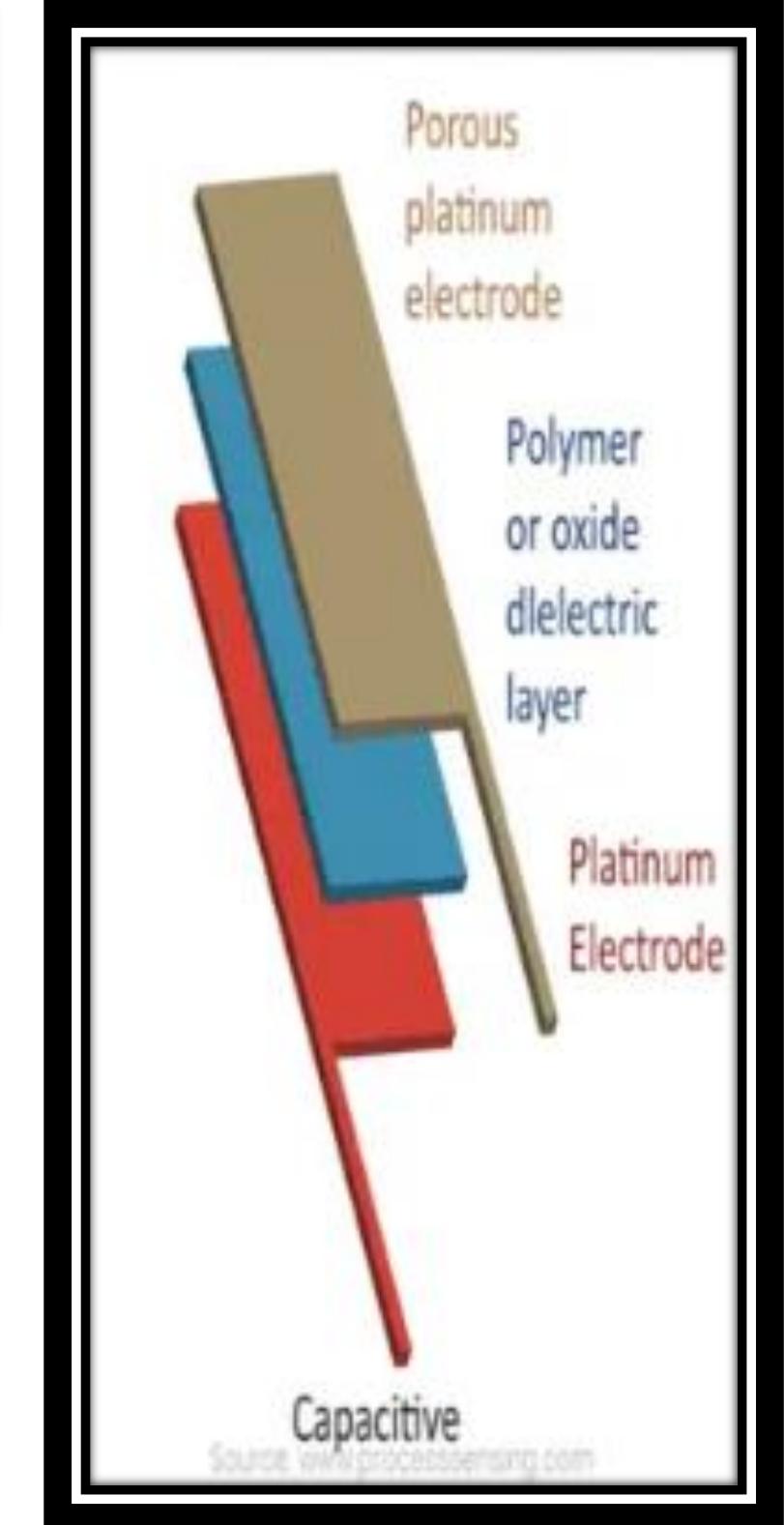
Sensor: umidade

No sensor AHT20, o elemento de umidade é capacitivo

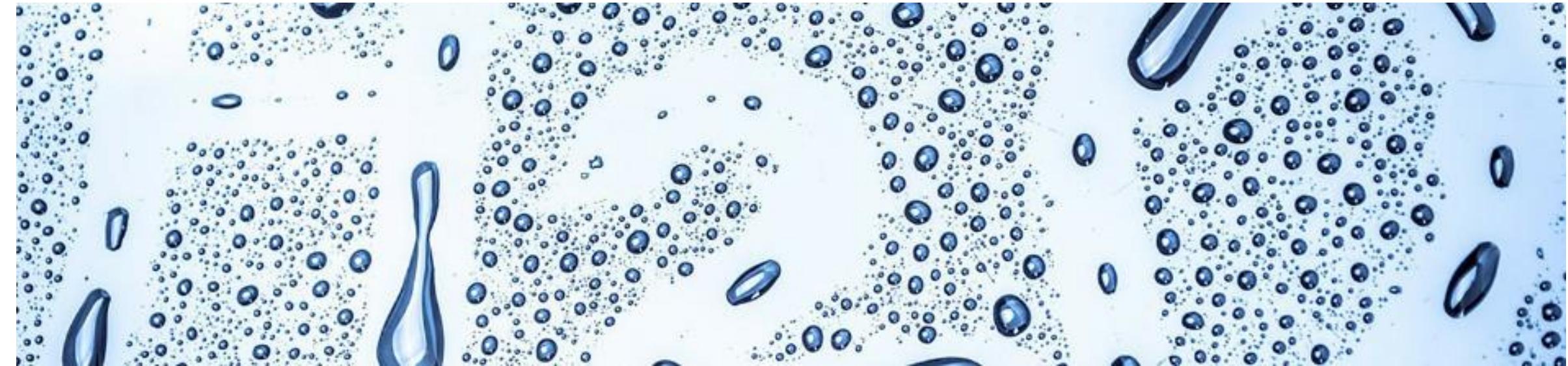
Capacitivo: A umidade altera a constante dielétrica entre duas placas, modificando a capacidade. Muito comum em sensores modernos.



Internal Structure of Humidity Sensor



Sensor de umidade: Aplicações práticas



Setor	Por que a umidade relativa é importante?
Farmacêutico	Estabilidade de medicamentos
Laboratórios	Precisão em reações químicas e biológicas
Museus e arquivos	Preservação de livros, instrumentos, obras de arte
Construção civil	Prevenção de mofo, bolor e deterioração de materiais
Agricultura / Estufas	Condições ideais para crescimento de plantas
Data centers	Prevenção de eletricidade estática e corrosão de circuitos

Sensor de umidade: Aplicações rotronic

Foco em Monitoramento em Tempo Real e Confiabilidade

- **Novartis:**

Co-desenvolvimento de sistema de monitoramento em tempo real com a Rotronic (RMS) para ambientes farmacêuticos.

- **Igreja histórica em Mönchaltdorf (Suíça):**

RMS (Rotronic Monitoring System) usado para prevenir mofo no teto e órgão de uma igreja com mais de 500 anos.

Agricultura, Indústria 4.0 e Logística

- **Bioline Agrosciences (Reino Unido):**

Controle ideal de temperatura e umidade para a criação de organismos biológicos de controle.

- **Setor automotivo (Reino Unido):**

Uso do RMS em laboratórios de teste automotivo para atender às demandas da Indústria 4.0 e veículos elétricos.

- **World Cargo Center (Alemanha):**

Mapeamento térmico e qualificação de armazém conforme GDP para cargas aéreas sensíveis à temperatura.

Museus, Escolas e Alimentos

- **Centro de Sementes em British Columbia (Canadá):**

Medição de atividade de água para preservar sementes raras.

- **“Meine Raumluft” (Suíça):**

Projeto de qualidade do ar em 100 salas de aula, medindo CO₂, temperatura e umidade.

- **Museu Laténium (Suíça):**

Preservação de artefatos arqueológicos com sensores de umidade e temperatura atualizados.

- **Exposição “Ladies Only”:**

Monitoramento climático ideal para acervos históricos.



Sensores pressão atmosférica ou barométrica.

O **BMP280** é um **sensor digital de pressão barométrica e temperatura**, desenvolvido para aplicações móveis e IoT com **baixo consumo de energia, alta precisão e dimensões reduzidas**.

Aplicações típicas

- Dispositivos móveis (smartphones, wearables)
- Estações meteorológicas portáteis
- GPS com correção de altitude
- Drones (controle de altitude)
- Dispositivos IoT
- Monitoramento de ambientes



Sensores pressão atmosférica ou barométrica.

O BMP280 é um **sensor digital de pressão barométrica e temperatura**, desenvolvido pela empresa **Bosch Sensortec**. Ele é uma evolução do BMP180, com **melhor precisão, menor consumo de energia e tamanho reduzido**.

Característica	Detalhes
Medições	Pressão atmosférica (barométrica) e temperatura
Interface	I ² C (até 3,4 MHz) ou SPI (até 10 MHz)
Faixa de pressão	300 a 1100 hPa (equivale a altitudes de -500 m a 9000 m)
Precisão da pressão	±1 hPa (±8 m de altitude)
Precisão da temperatura	±1 °C
Tensão de operação	1.71 V a 3.6 V
Consumo	~2.74 µA Muito baixo (ideal para dispositivos móveis e baterias)
Tempo de conversão	~7,5 ms (modo normal)
Tamanho	2.0 × 2.5 × 0.95 mm

Sensores pressão atmosférica ou barométrica.

Funcionamento

O **BMP280** utiliza uma tecnologia baseada em **MEMS** (*Micro-Electro-Mechanical Systems*) para medir pressão atmosférica com alta precisão. O elemento sensor de pressão no BMP280 funciona através de um princípio **piezorresistivo**, que converte deformações mecânicas em sinais elétricos.

O coração do BMP280 é um **diafragma micromecânico** (membrana em escala microscópica) fabricado em silício. Esse diafragma se flexiona devido à pressão atmosférica aplicada sobre ele.

Componentes-chave:

- **Membrana de Silício**: Flexiona sob pressão.
- **Resistores Piezorresistivos**: Integrados na superfície da membrana.
- **Circuito ASIC**: Condiciona o sinal e faz a compensação térmica.

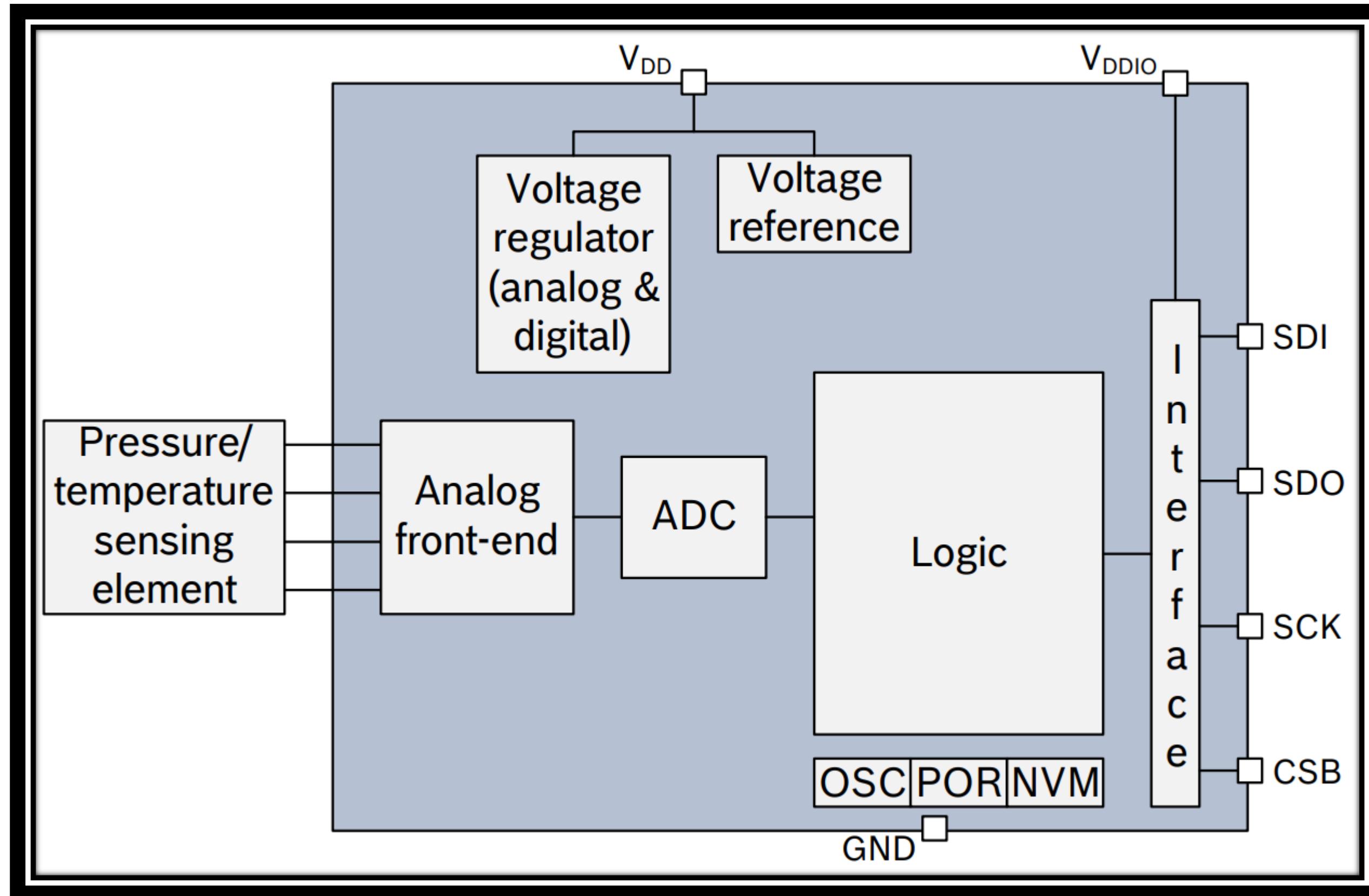
O BMP280 inclui um sensor de **temperatura** interno.



O **princípio piezorresistivo** é um fenômeno físico observado em certos materiais onde a **resistência elétrica varia em resposta a uma deformação mecânica**.

Sensores pressão atmosférica ou barométrica.

Diagrama de blocos de um BMP280 do datasheet da Bosch



Sensores pressão atmosférica ou barométrica.

Medida de temperatura no BMP280. Retirado do datasheet da Bosch

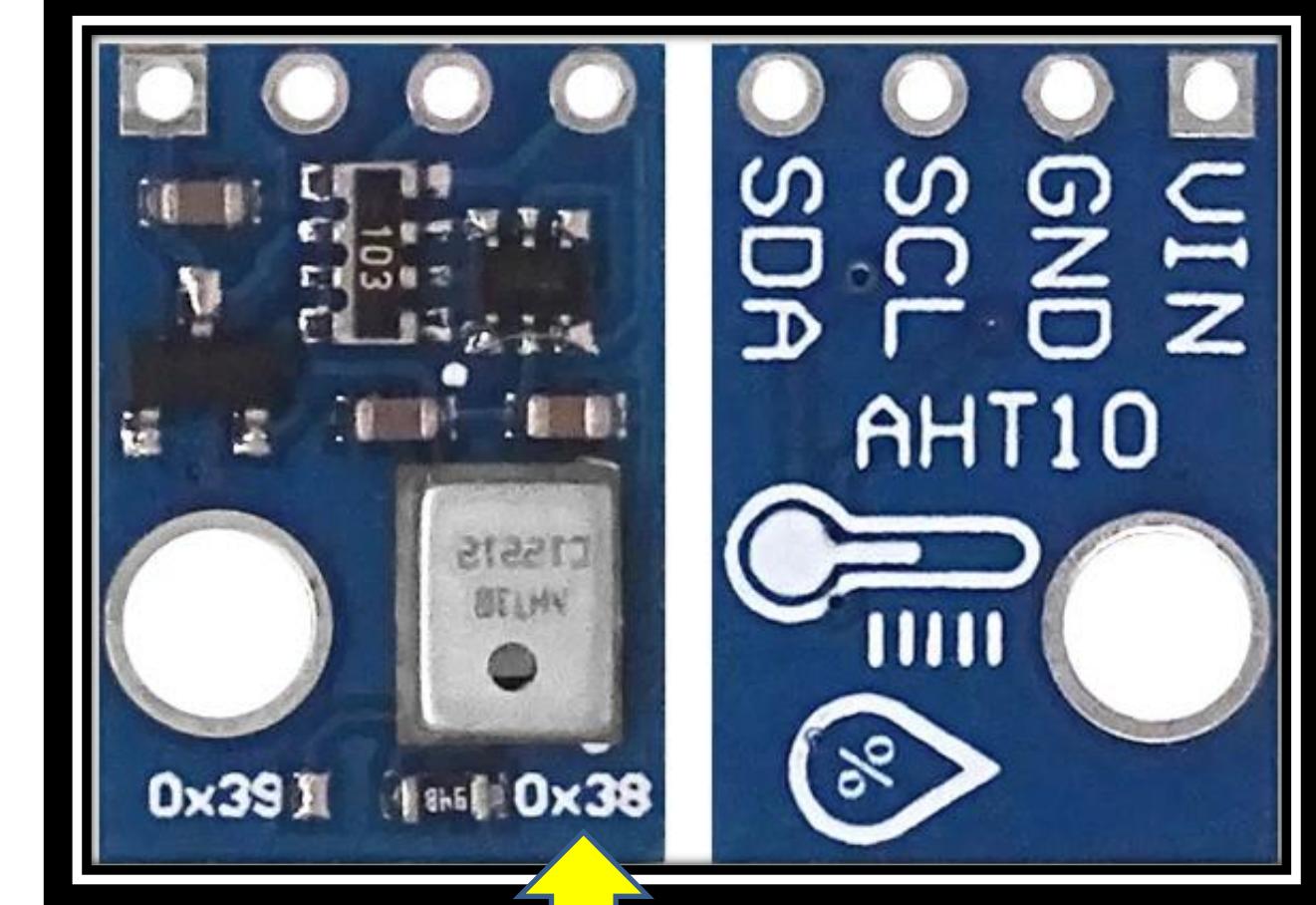
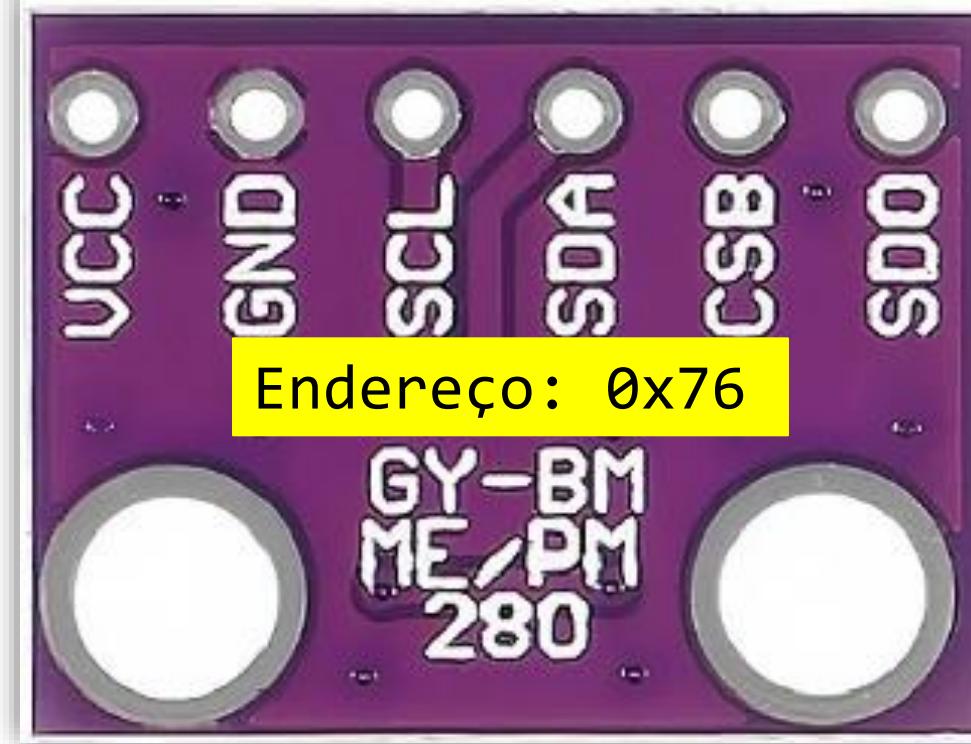
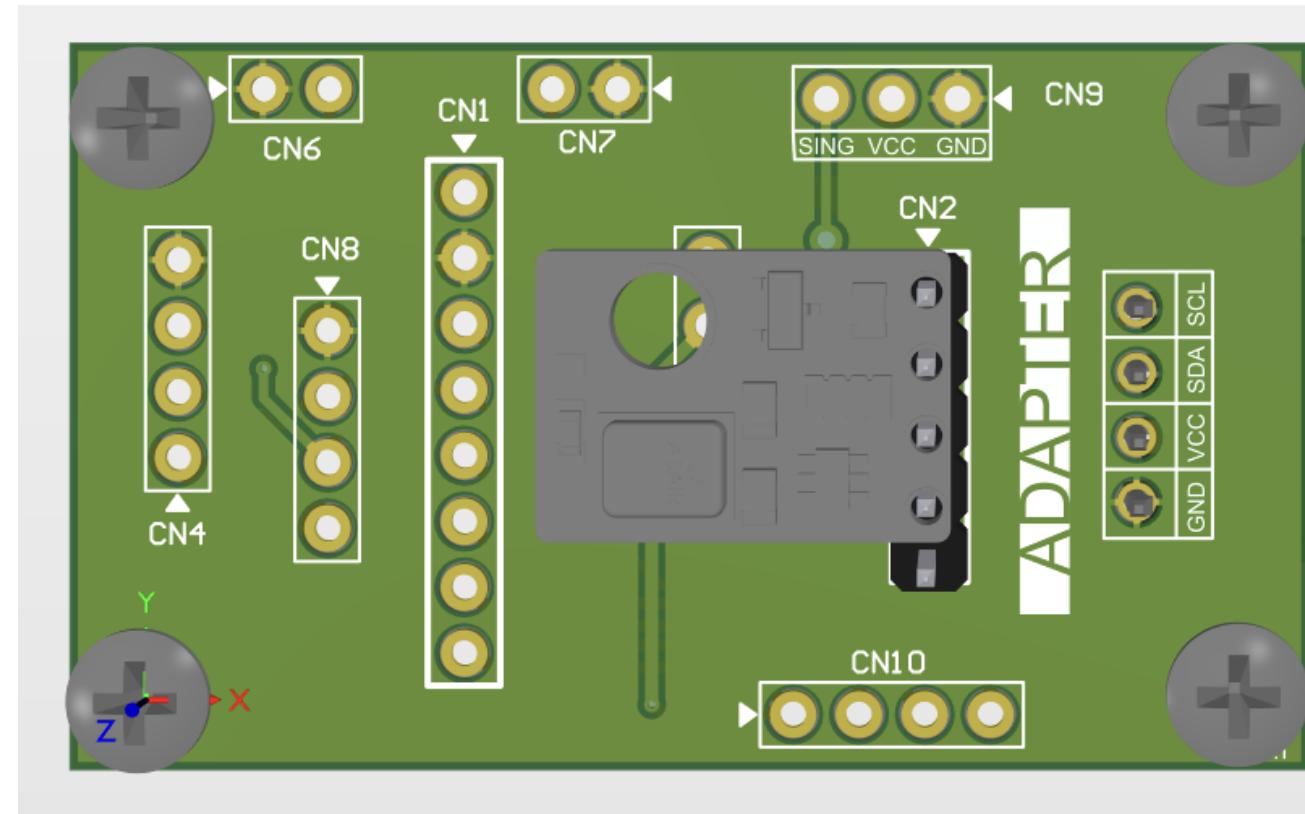
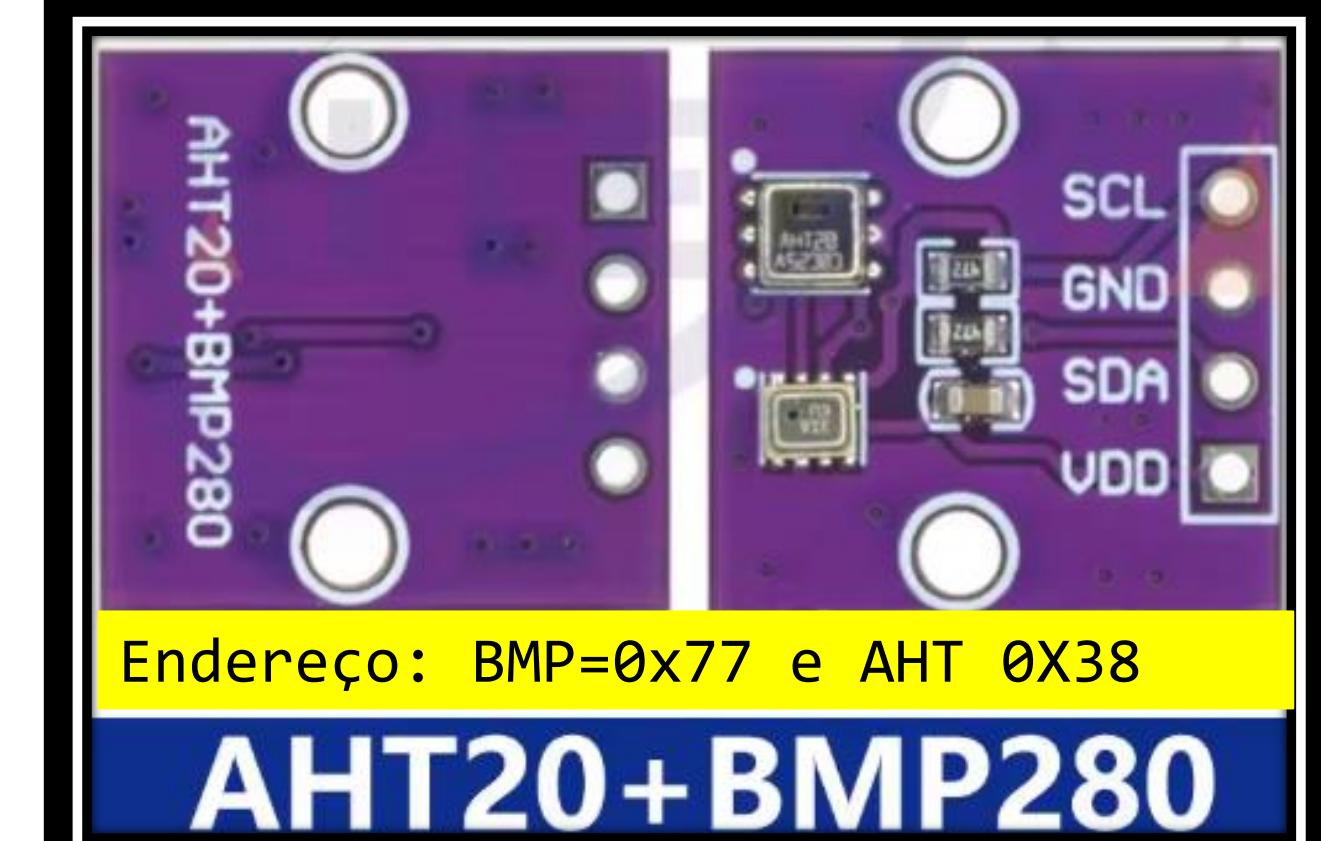
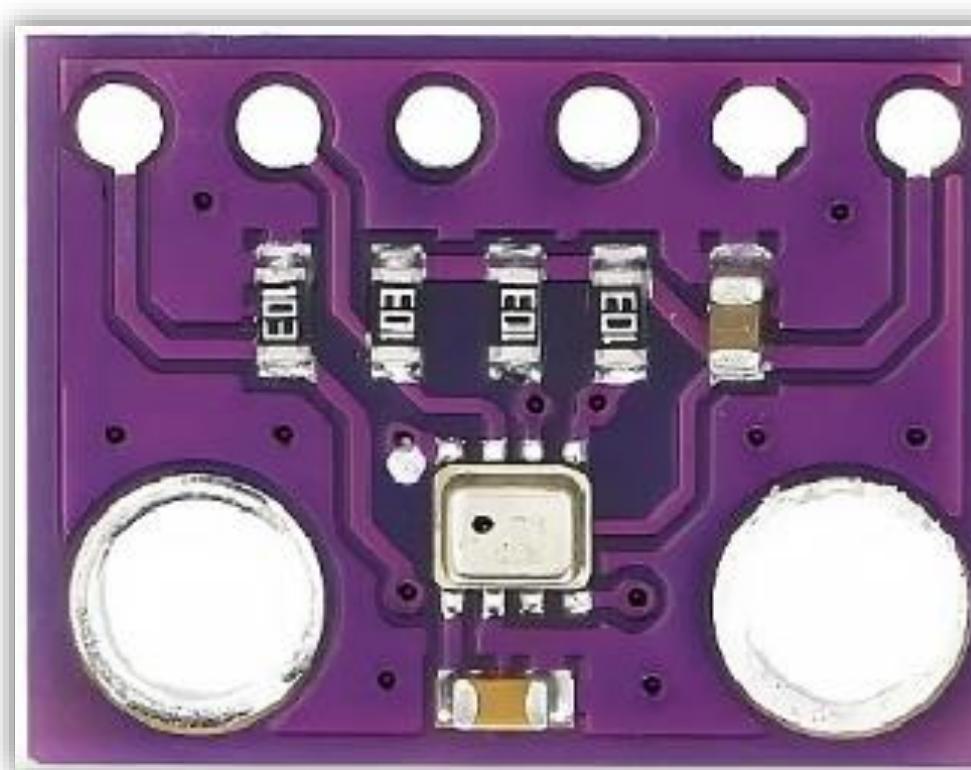
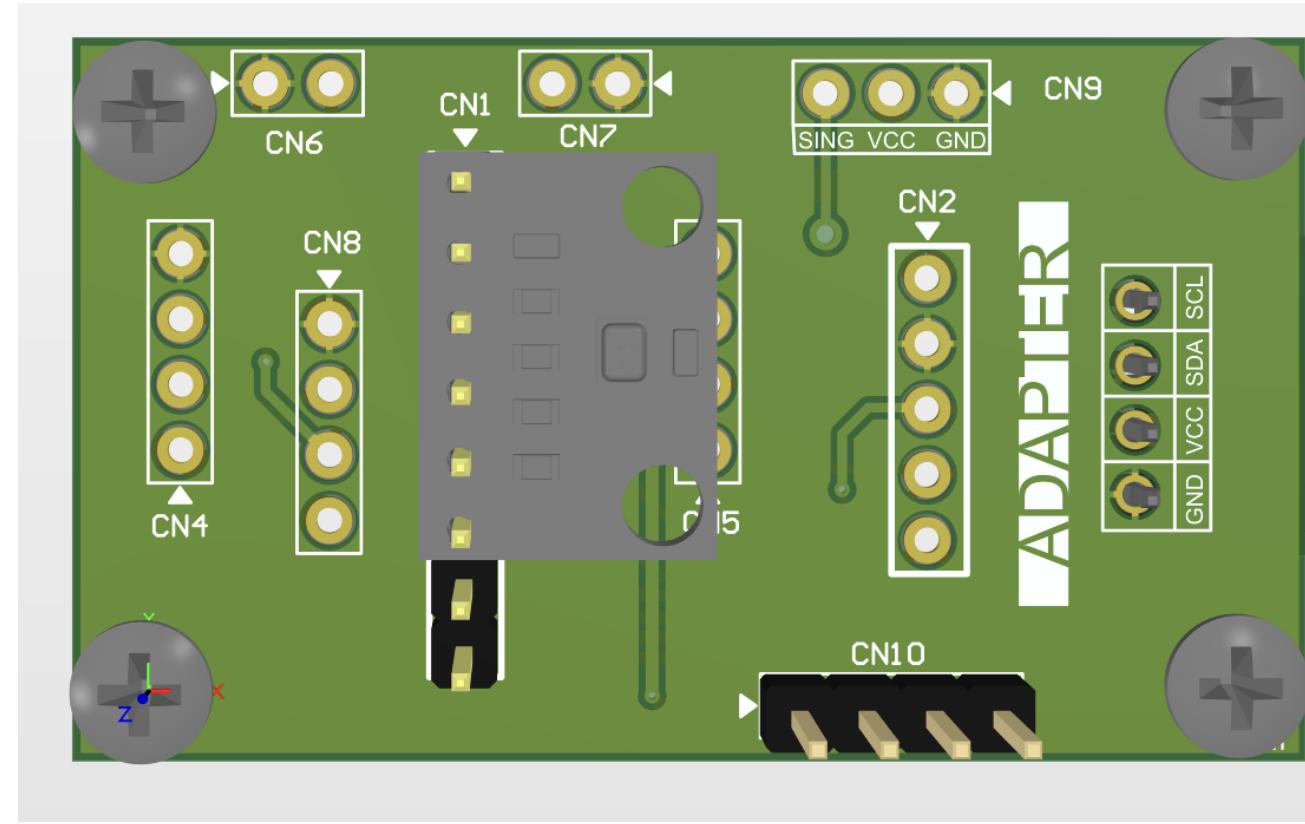
The data type “BMP280_S64_t” should define a 64 bit signed integer variable type, which on most supporting platforms can be defined as “long long signed int”. The revision of the code is rev.1.1.

```
// Returns temperature in DegC, resolution is 0.01 DegC. Output value of "5123" equals 51.23
DegC.
// t_fine carries fine temperature as global value
BMP280_S32_t t_fine;
BMP280_S32_t bmp280_compensate_T_int32(BMP280_S32_t adc_T)
{
    BMP280_S32_t var1, var2, T;
    var1 = (((adc_T>>3) - ((BMP280_S32_t)dig_T1<<1))) * ((BMP280_S32_t)dig_T2)) >> 11;
    var2 = (((((adc_T>>4) - ((BMP280_S32_t)dig_T1)) * ((adc_T>>4) - ((BMP280_S32_t)dig_T1)))
>> 12) *
        ((BMP280_S32_t)dig_T3)) >> 14;
    t_fine = var1 + var2;
    T = (t_fine * 5 + 128) >> 8;
    return T;
}
```
// Returns pressure in Pa as unsigned 32 bit integer in Q24.8 format (24 integer bits and 8
fractional bits).
// Output value of "24674867" represents 24674867/256 = 96386.2 Pa = 963.862 hPa
BMP280_U32_t bmp280_compensate_P_int64(BMP280_S32_t adc_P)
{
 BMP280_S64_t var1, var2, p;
 var1 = ((BMP280_S64_t)t_fine) - 128000;
 var2 = var1 * var1 * (BMP280_S64_t)dig_P6;
 var2 = var2 + ((var1*(BMP280_S64_t)dig_P5)<<17);
 var2 = var2 + (((BMP280_S64_t)dig_P4)<<35);
 var1 = ((var1 * var1 * (BMP280_S64_t)dig_P3)>>8) + ((var1 * (BMP280_S64_t)dig_P2)<<12);
 var1 = (((((BMP280_S64_t)1)<<47)+var1))*((BMP280_S64_t)dig_P1)>>33;
 if (var1 == 0)
 {
 return 0; // avoid exception caused by division by zero
 }
 p = 1048576-adc_P;
 p = (((p<<31)-var2)*3125)/var1;
 var1 = (((BMP280_S64_t)dig_P9) * (p>>13) * (p>>13)) >> 25;
 var2 = (((BMP280_S64_t)dig_P8) * p) >> 19;
 p = ((p + var1 + var2) >> 8) + (((BMP280_S64_t)dig_P7)<<4);
 return (BMP280_U32_t)p;
}
```



# Sensores: AHT10, AHT20 e BMP280

## BitDogLab



The 7-bit device address is 111011x. The 6 MSB bits are fixed. The last bit is changeable by SDO value and can be changed during operation. Connecting SDO to GND results in slave address 1110110 (0x76); connecting it to V<sub>DDIO</sub> results in slave address 1110111 (0x77), which

# Aplicação: Meteorologia

A **pressão atmosférica**, a **umidade** e a **temperatura** são variáveis fundamentais para a formação de uma vasta gama de fenômenos meteorológicos (como a formação de nuvens, ventos, frentes, tempestades) e que, em escala de tempo maior, contribuem para as características do clima de uma região.

É a interação contínua entre esses fatores que determina o clima que vivenciamos todos os dias.

Pressão atmosférica, umidade e temperatura interagem na atmosfera, determinando o tempo e o clima.

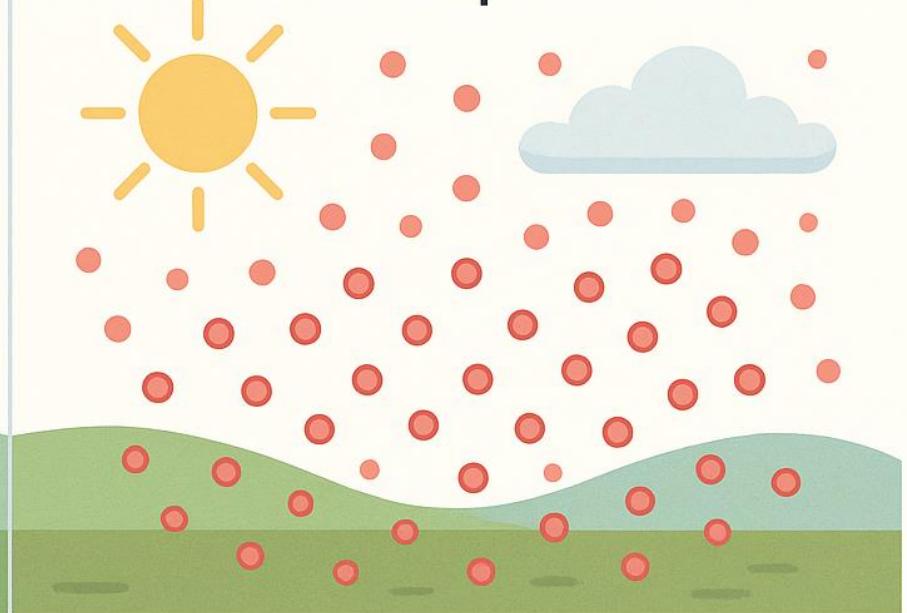
Tempo instável



Pressão baixa:  
1005 hPa

Temperatura quente:  
30 °C

Tempo estável



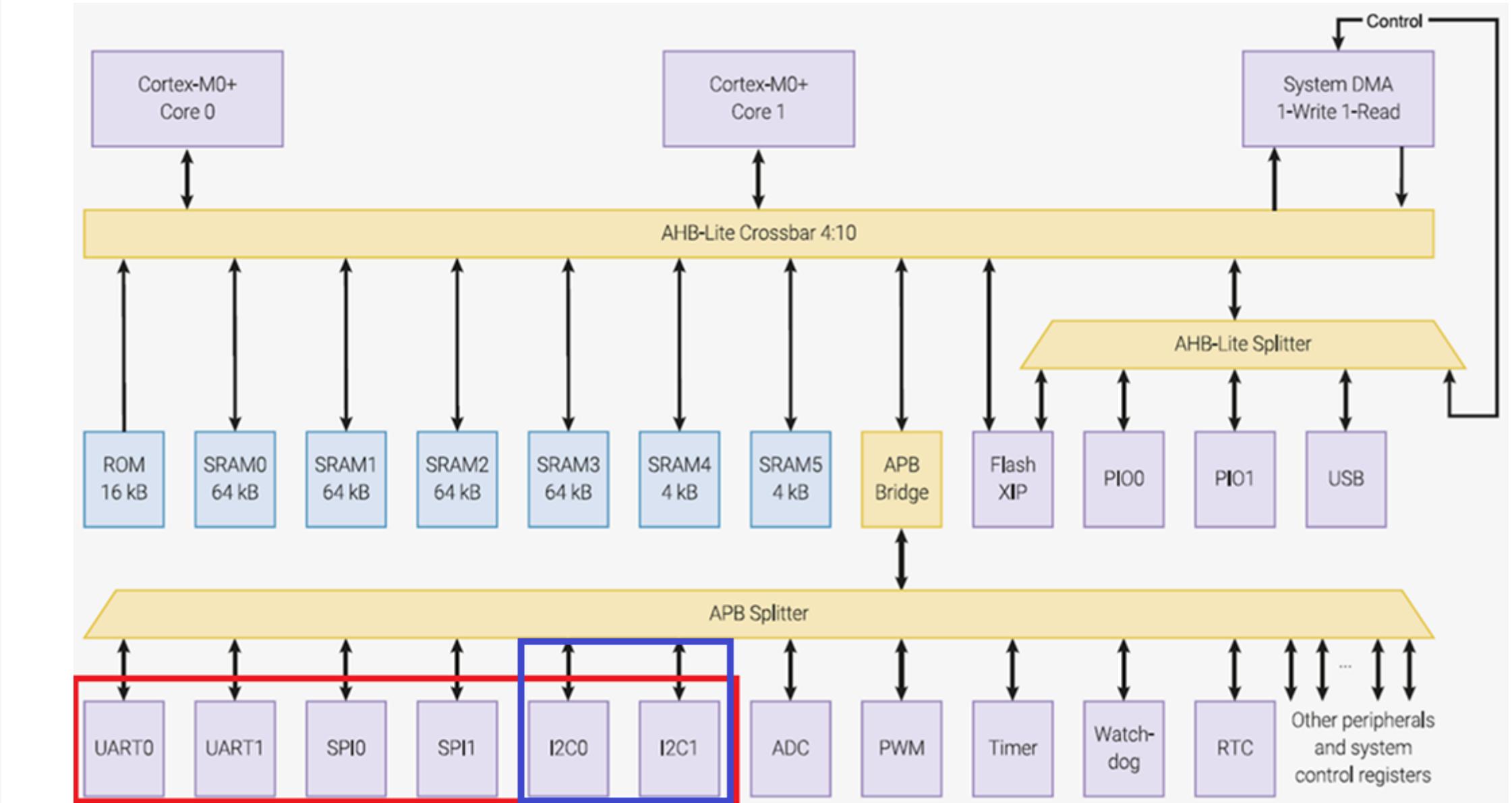
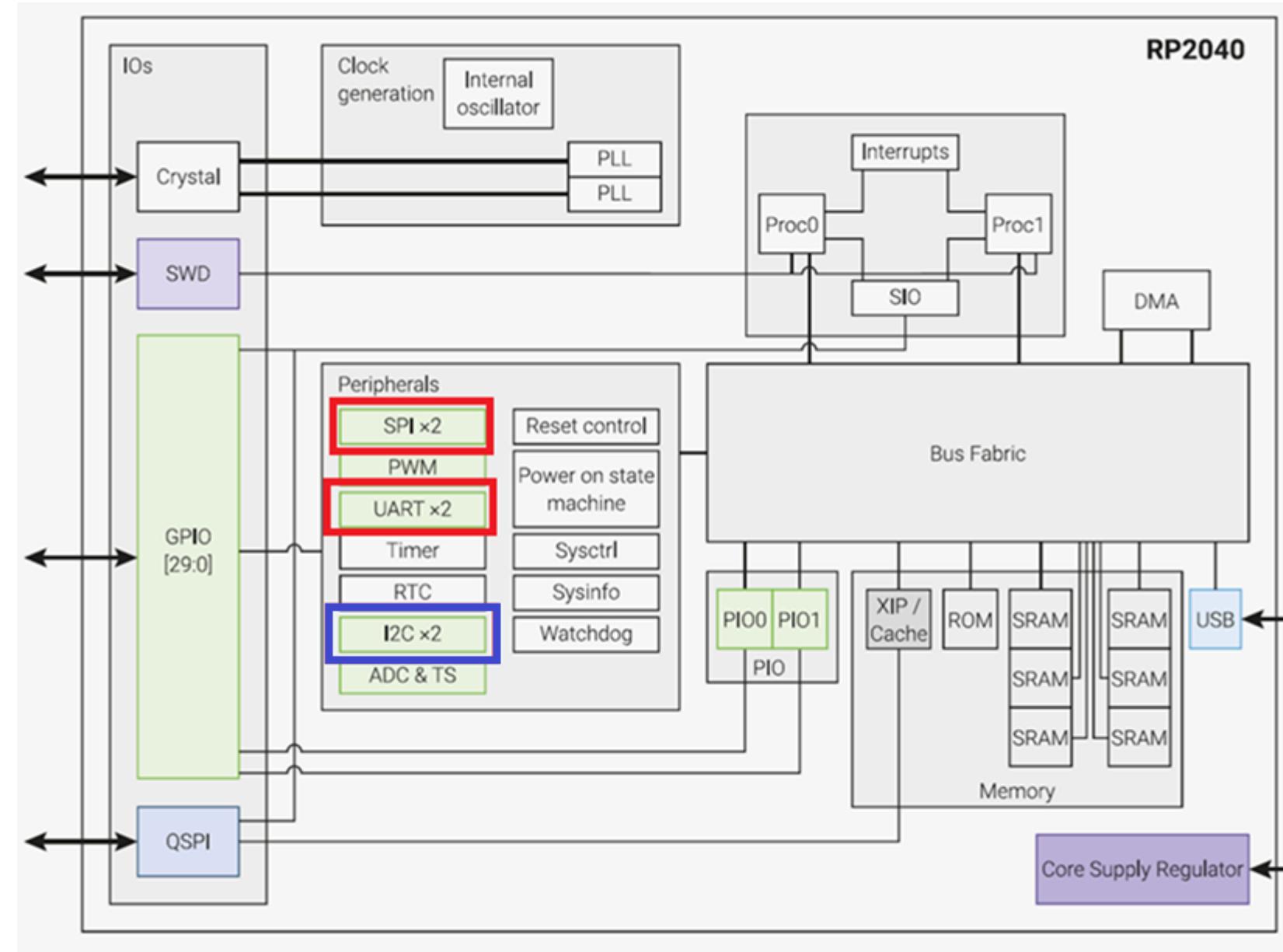
Pressão alta:  
1020 hPa

Temperatura fria:  
15 °C

# Revisão do protocolo I<sup>2</sup>C

## Unidade 4 | Capítulo 6 - Aula síncrona (03/02/2024)

## I<sup>2</sup>C no RP2040.



### I<sup>2</sup>C (Inter-Integrated Circuit)

O I<sup>2</sup>C é um **protocolo serial síncrono** usado para a troca de dados entre MCUs e periféricos, tais como: sensores e displays.

Criado pela Philips Semiconductors em 1982. O protocolo suporta **múltiplos dispositivos** alvo em um barramento de comunicação e também pode suportar **múltiplos controladores** que enviam e recebem comandos e dados. A comunicação é **enviada em pacotes** de bytes com um **endereço exclusivo** para cada dispositivo alvo.

### Características em destaque do I<sup>2</sup>C :

#### Barramento de dois fios:

- **SDA** (Serial Data Line): linha de dados bidirecional.
- **SCL** (Serial Clock Line): linha de clock gerada pelo dispositivo mestre.

#### Topologia mestre-escravo:

- O dispositivo mestre controla o clock e inicia as comunicações.
- Escravos respondem a comandos do mestre.

#### Endereçamento:

- Cada dispositivo escravo tem um endereço único de 7 bits.

#### Velocidade:

**Standard Mode:** até 100 kbps.

**Fast Mode Plus:** até 1 Mbps.

**Ultra-Fast Mode:** até 5 Mbps.

**Fast Mode:** até 400 kbps.

**High-Speed Mode:** até 3.4 Mbps.

#### Pull-up resistors:

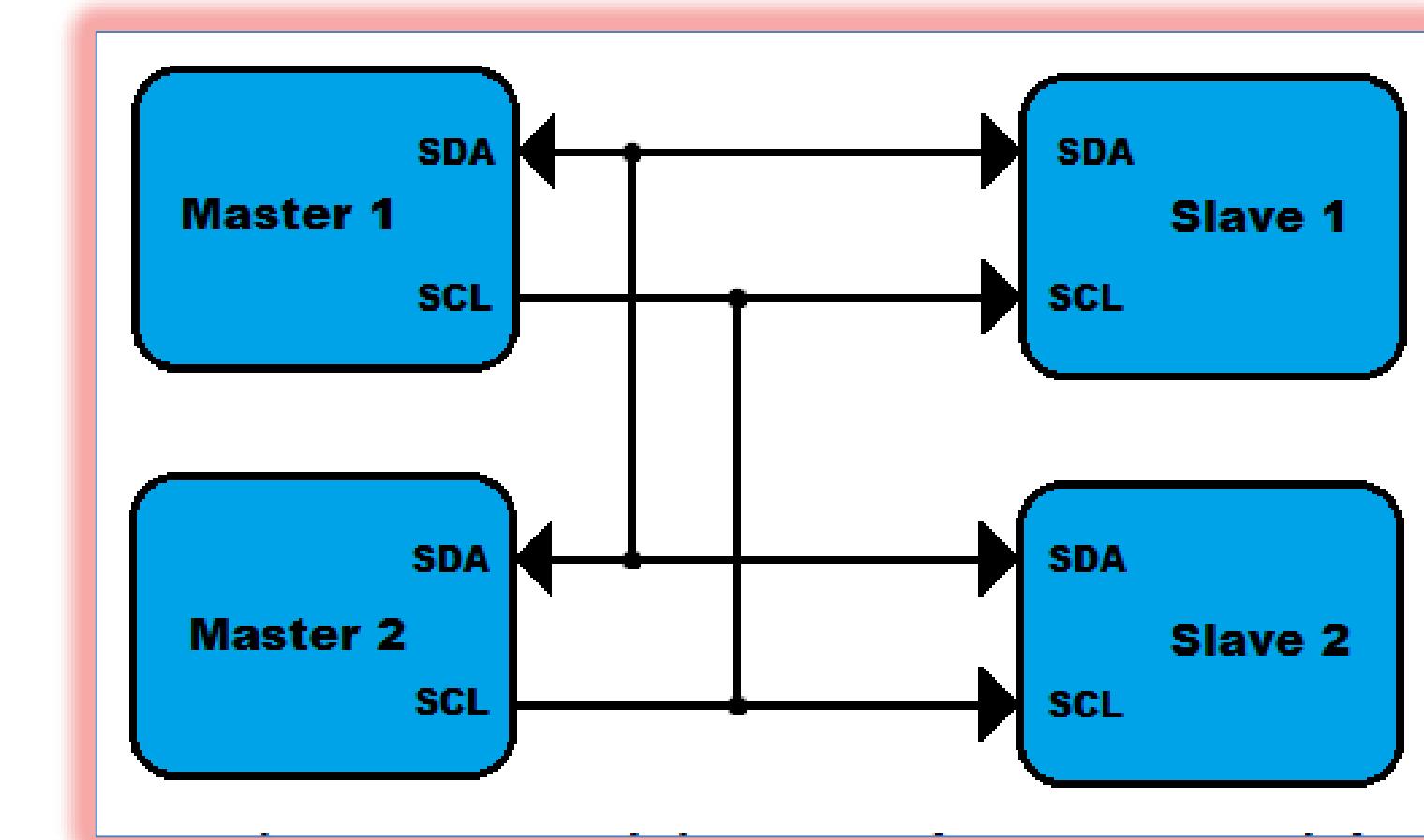
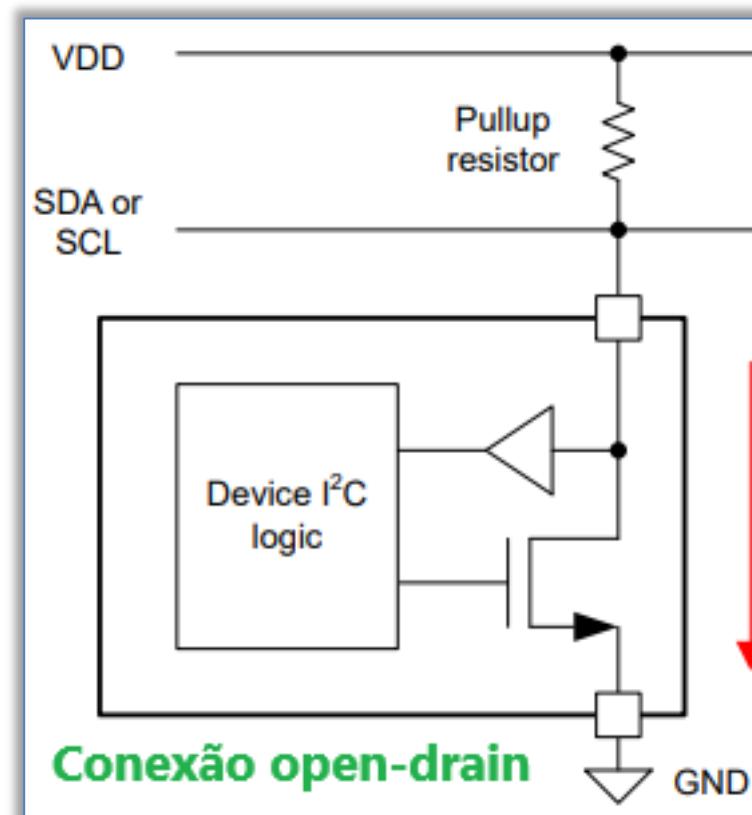
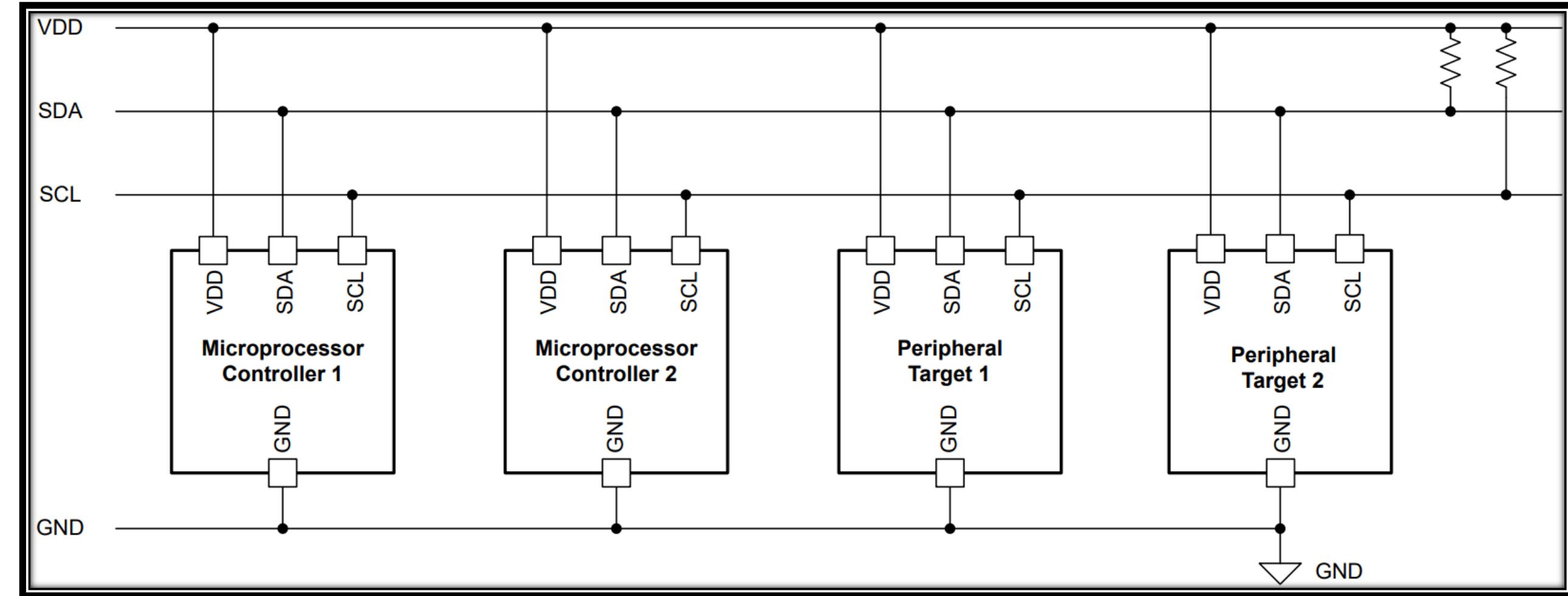
As linhas SDA e SCL são conectadas a **resistores pull-up** para manter o estado alto quando não estão em uso.

# Revisão do protocolo I<sup>2</sup>C

## Unidade 4 | Capítulo 6 - Aula síncrona (03/02/2024)

### Linhas de comunicação RP2

- **SDA** (Serial Data)
- **SCL** (Serial Clock)

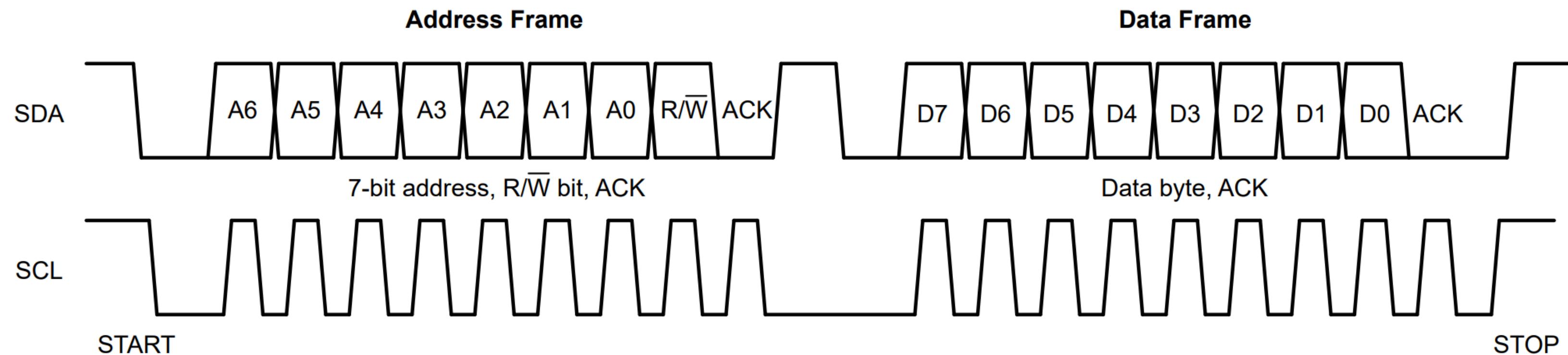


## Quadro de endereços e Dados do protocolo I<sup>2</sup>C

As mensagens são divididas em dois tipos de quadro (frame):

- Um **quadro de endereço**, onde o mestre indica o escravo para o qual a mensagem está sendo enviada. (7 bits o que resulta em no máximo **127 dispositivos** endereçáveis).
- Um ou mais **quadros de dados**, que são mensagens de dados de 8 bits passadas de mestre para escravo ou vice-versa .

Os dados são colocados na linha SDA depois que o SCL fica baixo e são amostrados depois que a linha SCL fica alta. O tempo entre a transição do clock e a leitura/gravação dos dados é definido pelos dispositivos no barramento e varia de chip para chip.

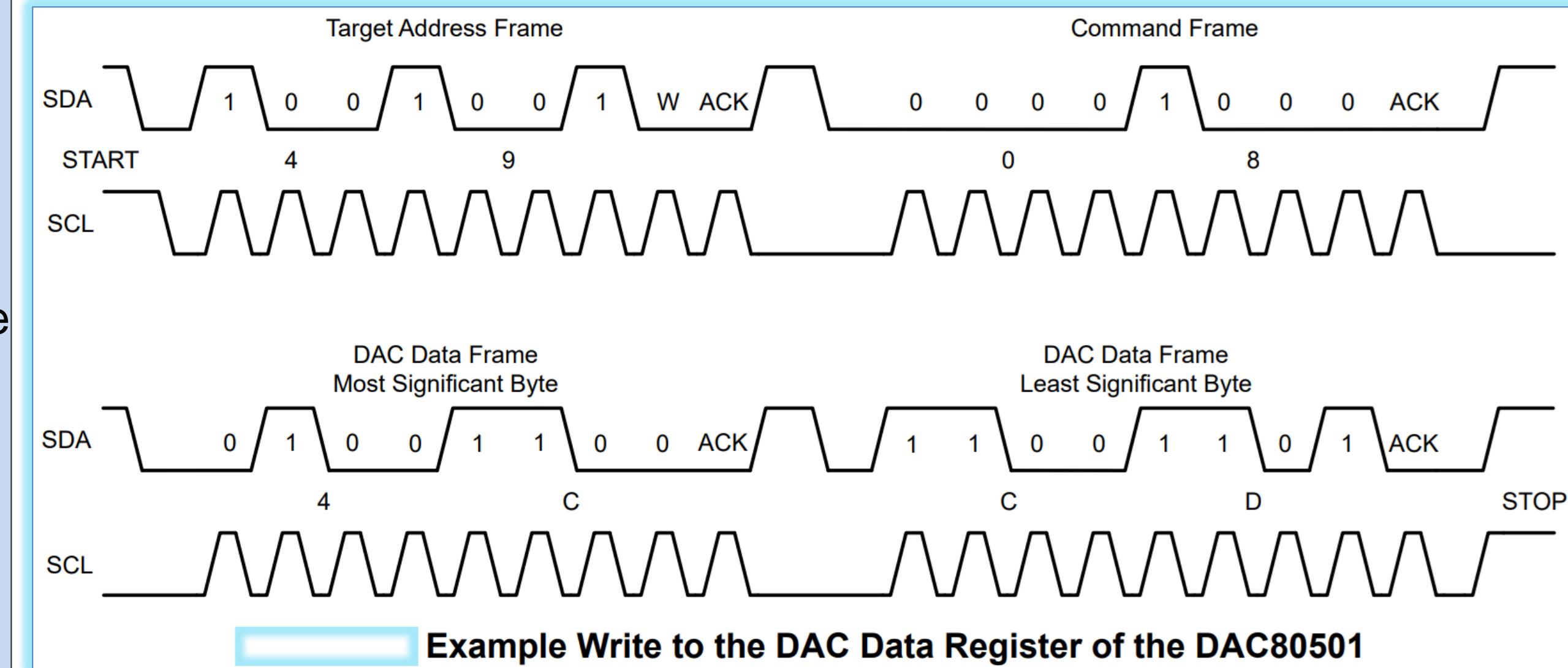
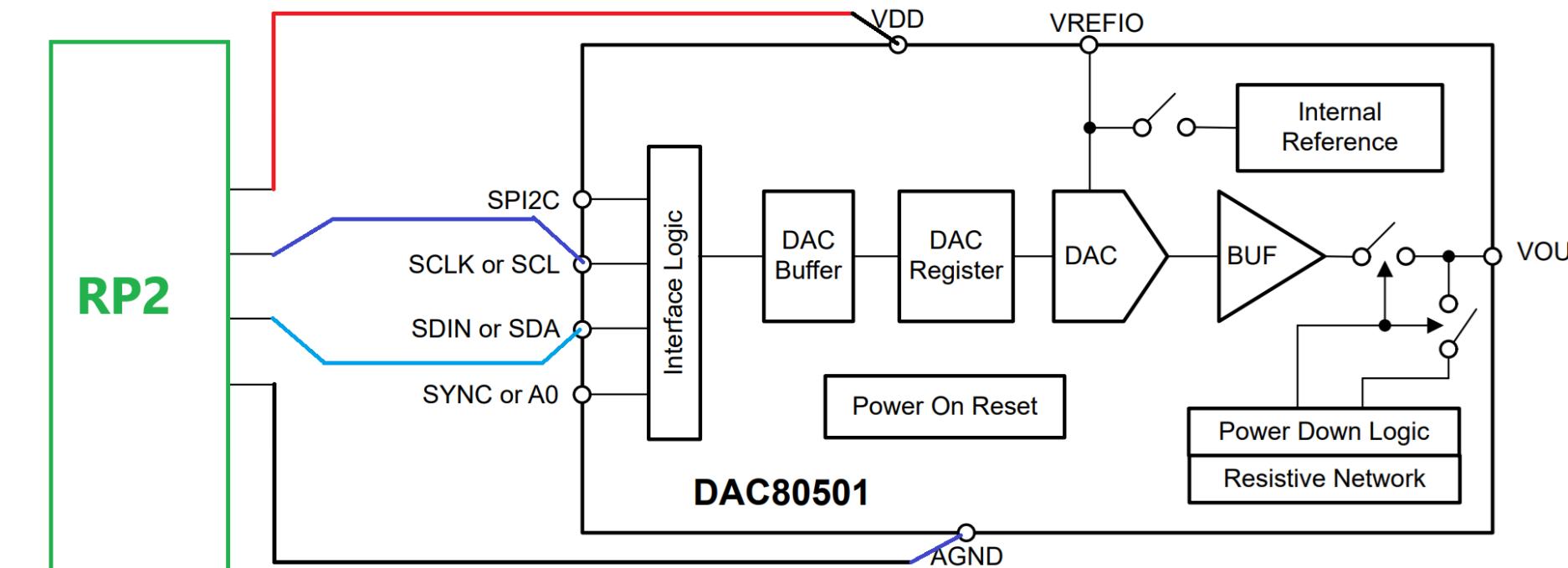


# Revisão do protocolo I<sup>2</sup>C

Unidade 4 | Capítulo 6 - Aula síncrona (03/02/2024)

## Funcionamento Básico:

- Início (Start Condition):** O mestre envia um pulso de início puxando SDA para baixo enquanto SCL está alto.
- Endereçamento:** O mestre envia o endereço do escravo desejado, seguido de um bit indicando se a operação será de leitura ou escrita.
- Transferência de Dados:**  
**Escrita:** O mestre envia dados ao escravo.  
**Leitura:** O escravo envia dados ao mestre.
- Confirmação (ACK/NACK):** Após cada byte transferido, o receptor (mestre ou escravo) envia um sinal de confirmação (ACK) ou negação (NACK).
- Término (Stop Condition):** O mestre sinaliza o fim da comunicação liberando SDA enquanto SCL está alto.

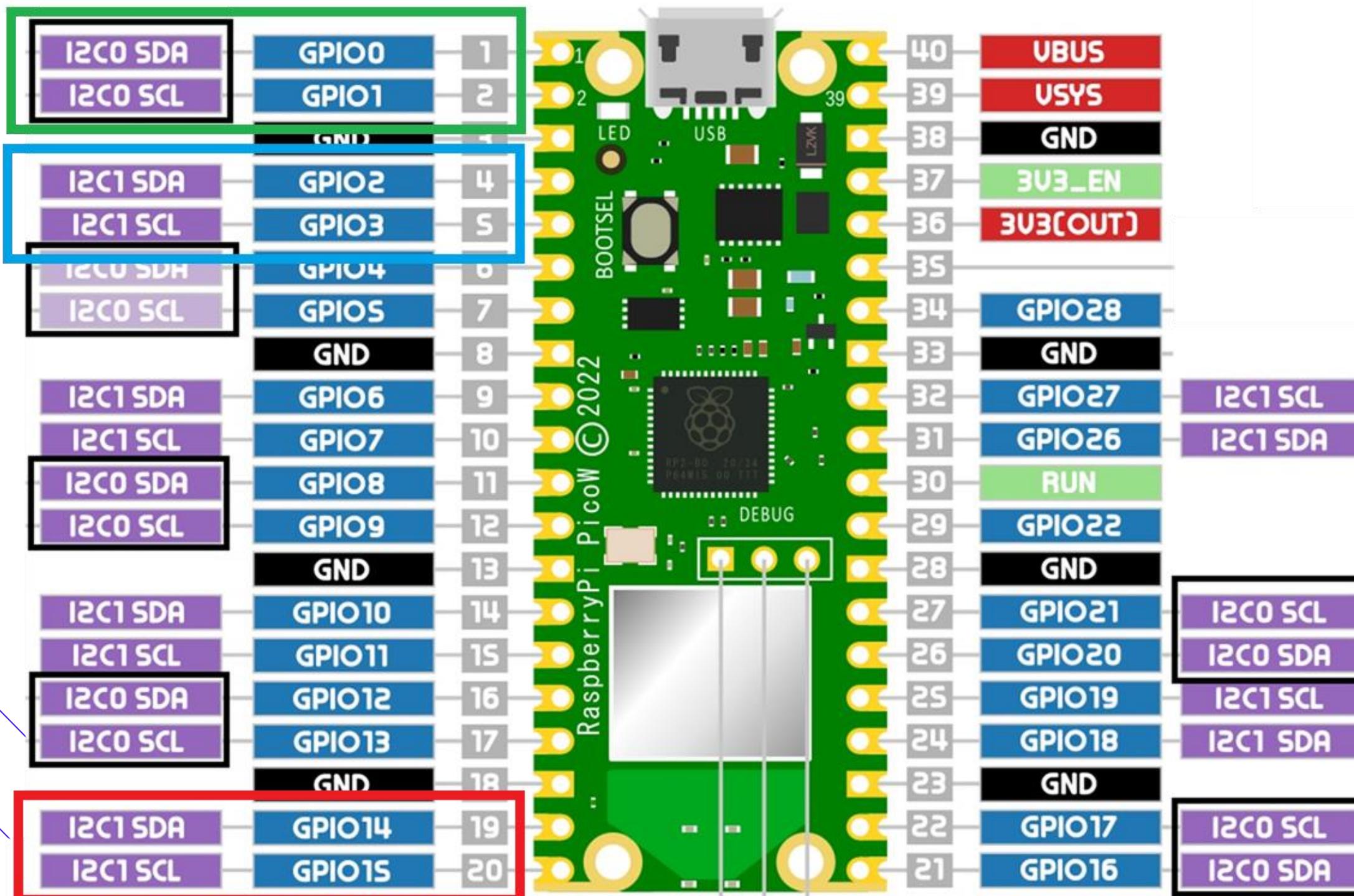


# Revisão do protocolo I<sup>2</sup>C

Unidade 4 | Capítulo 6 - Aula síncrona (03/02/2024)

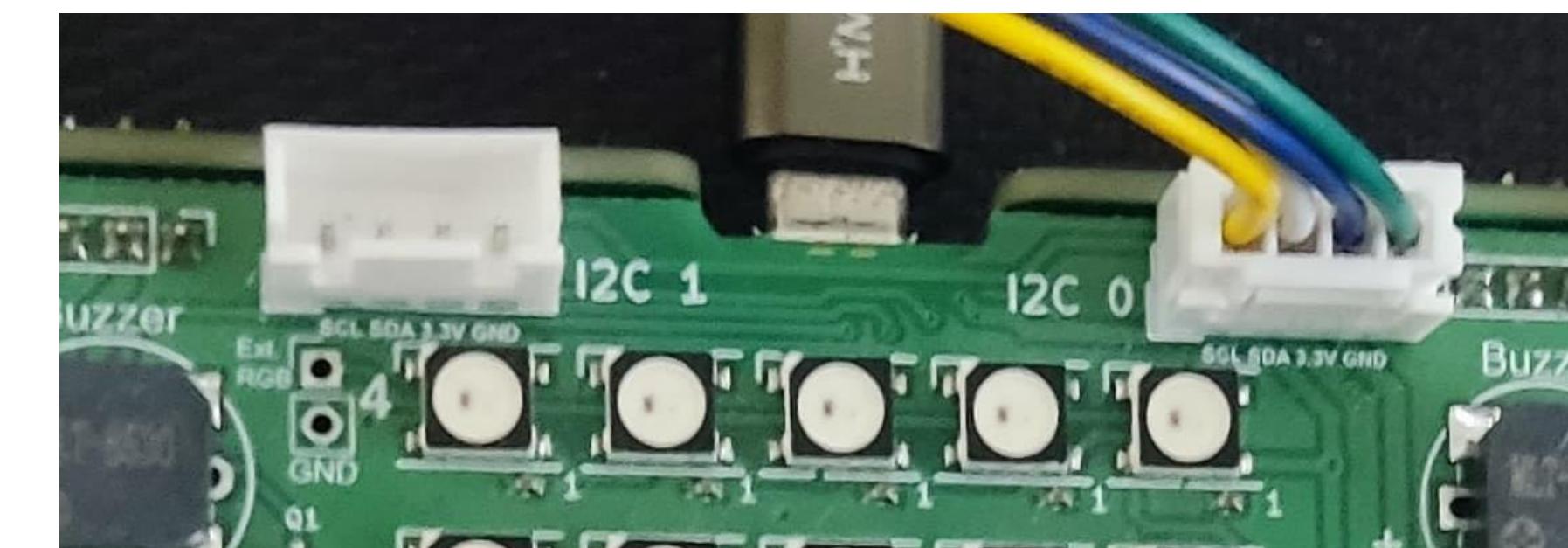
## Observação:

O display SSD1306 de 128x64 pixels, encontra-se interligado aos pinos relativos à GPIO14 e GPIO15 na BitDogLab. Os conectores de extensão da BitDogLab estão nos pinos relativos à GPIO0 e GPIO1 (I2C0) e GPIO1/GPIO2 (I2C1). Portanto, caso esteja utilizando o Display na I2C1 o conector de extensão I2C1 não funcionará como I2C válido.



# Linhos de I2C no RP2

- **SDA** (Serial Data Line)
  - **SCL** (Serial Clock Line)



**Exemplos de programas desenvolvidos para verificação de  
funcionamento dos sensores BMP280, AHT10 e AHT20.**



Procura por endereços de dispositivos I2C conectados nos barramentos I2C (I2C0 e I2C1). Ou seja: detectar dispositivos I2C conectados na BitDogLab.

### Funcionalidades

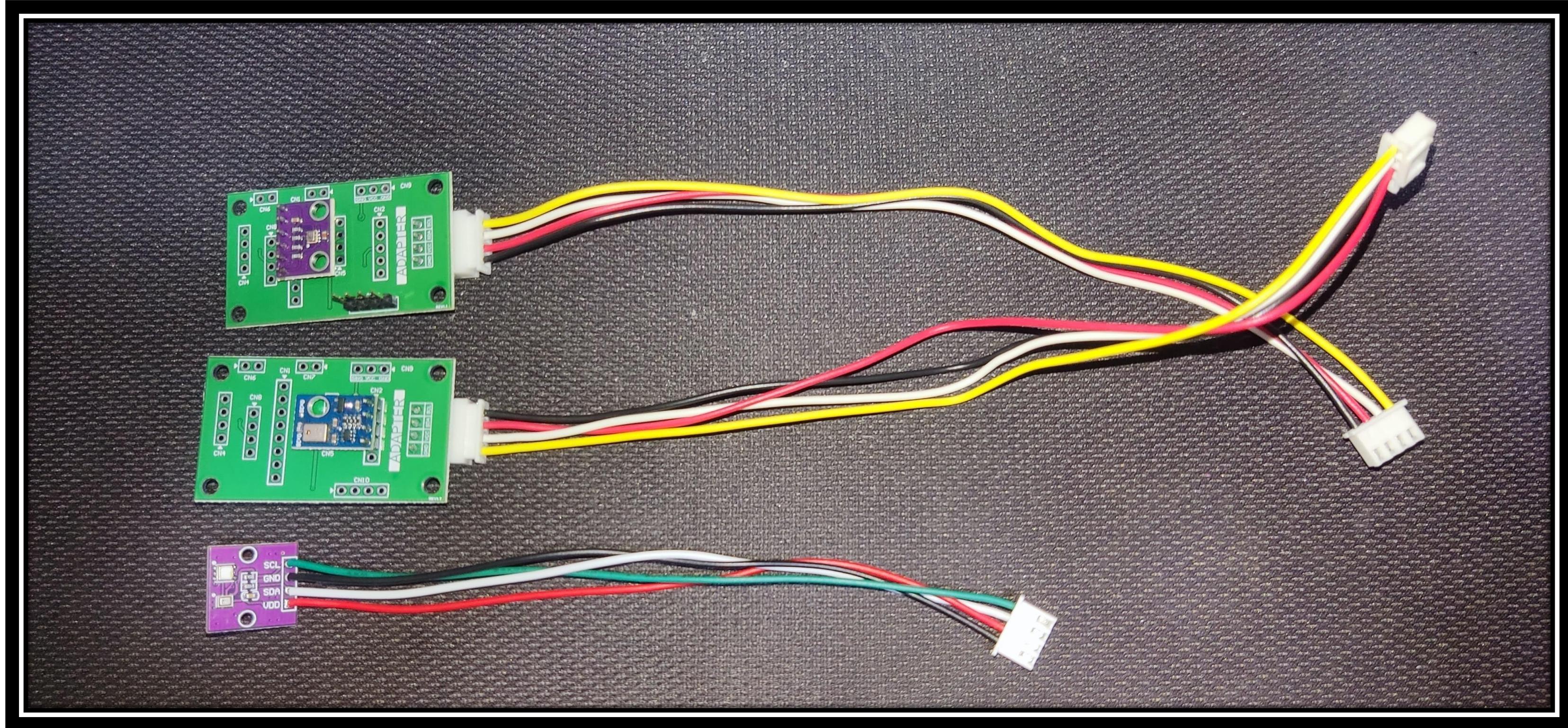
O programa escaneia os barramentos I2C0 (GPIO0 para SDA, GPIO1 para SCL) e I2C1 (GPIO14 para SDA, GPIO15 para SCL) do Raspberry Pi Pico, procurando dispositivos I2C nos endereços de 0x08 a 0x77 (**8 a 119 em decimal**) . Os resultados são exibidos via UART em um terminal serial (115200 baud). O escaneamento é repetido a cada 5 segundos.

### Observações:

Nem todos os endereços de 0x00 a 0x7F são usados para dispositivos I2C comuns. Alguns endereços são reservados pela especificação I2C para propósitos especiais.

- **0x00 a 0x07:** Reservados para propósitos especiais, como o endereço de "chamada geral".
- **0x78 a 0x7F:** Reservados para endereços de 10 bits (quando usado o modo de endereçamento de 10 bits) ou outros usos específicos, como endereços de gerenciamento de energia.

# Sensores: Umidade, pressão e temperatura

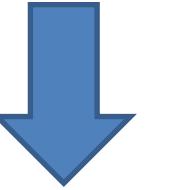


# Sensores

Exemplo 1:



```
1 int main()
2 {
```



```
36
37 // Trecho para modo BOOTSEL com botão B
38 #include "pico/bootrom.h"
39 #define botaoB 6
40 void gpio_irq_handler(uint gpio, uint32_t events)
41 {
42 reset_usb_boot(0, 0);
43 }
44
45 int main()
46 {
47 gpio_init(botaoB);
48 gpio_set_dir(botaoB, GPIO_IN);
49 gpio_pull_up(botaoB);
50 gpio_set_irq_enabled_with_callback(botaoB, GPIO_IRQ_EDGE_FALL, true, &gpio_irq_handler);
51 //Fim do modo BOOTSEL
```



```
1 #include <stdio.h>
2 #include "pico/stdlib.h"
3 #include "hardware/i2c.h"
4
5 // Definições dos pinos I2C
6 #define I2C0_SDA_PIN 0
7 #define I2C0_SCL_PIN 1
8 #define I2C1_SDA_PIN 14
9 #define I2C1_SCL_PIN 15
10
11 // Velocidade do I2C (100 kHz)
12 #define I2C_BAUDRATE 100000
```

# Sensores

## Exemplo 1:

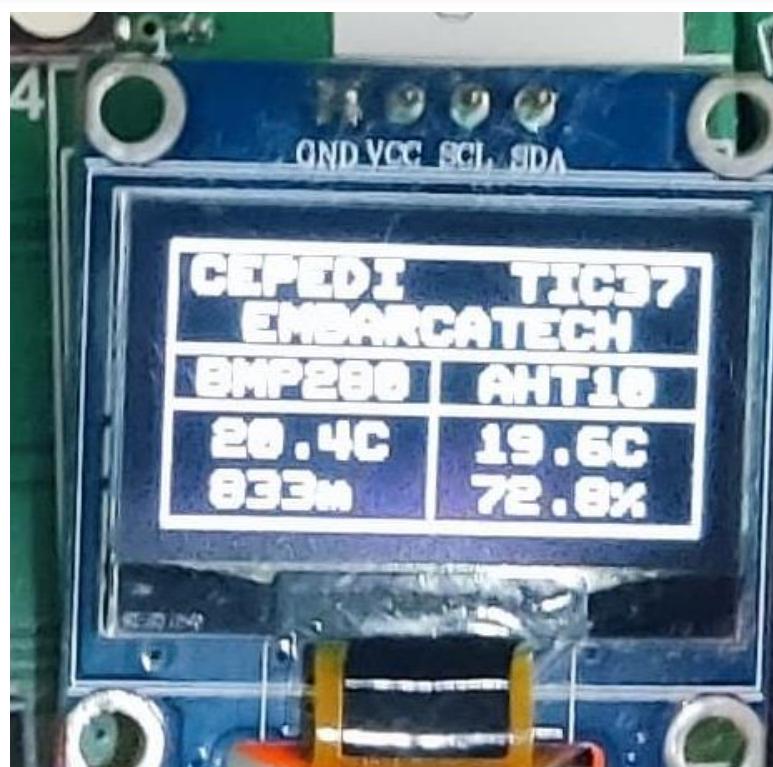
```
60 // Aguarda um momento para garantir que a UART esteja pronta
61 sleep_ms(2000);
62 printf("Iniciando scanner I2C para Raspberry Pi Pico\n");
63
64 // Inicializa I2C0
65 i2c_init(i2c0, I2C_BAUDRATE);
66 gpio_set_function(I2C0_SDA_PIN, GPIO_FUNC_I2C);
67 gpio_set_function(I2C0_SCL_PIN, GPIO_FUNC_I2C);
68 gpio_pull_up(I2C0_SDA_PIN);
69 gpio_pull_up(I2C0_SCL_PIN);
70
71 // Inicializa I2C1
72 i2c_init(i2c1, I2C_BAUDRATE);
73 gpio_set_function(I2C1_SDA_PIN, GPIO_FUNC_I2C);
74 gpio_set_function(I2C1_SCL_PIN, GPIO_FUNC_I2C);
75 gpio_pull_up(I2C1_SDA_PIN);
76 gpio_pull_up(I2C1_SCL_PIN);
77
78 while (true) {
79 // Escaneia ambos os barramentos I2C
80 scan_i2c_bus(i2c0, "I2C0");
81 scan_i2c_bus(i2c1, "I2C1");
82
83 printf("\nEscaneamento concluído. Aguardando 5 segundos
84 sleep_ms(5000);
85 }
86
87 return 0;
88 }
```

```
11 // Velocidade do I2C (100 kHz)
12 #define I2C_BAUDRATE 100000
13
14 void scan_i2c_bus(i2c_inst_t *i2c, const char *bus_name) {
15 printf("\nEscaneando barramento %s...\n", bus_name);
16 bool found = false;
17
18 // Escaneia endereços de 0x08 a 0x77 (endereços válidos de 7 bits)
19 for (uint8_t addr = 8; addr <= 0x77; addr++) {
20 uint8_t buffer;
21 // Tenta ler um byte do endereço atual
22 int ret = i2c_read_blocking(i2c, addr, &buffer, 1, false);
23
24 // Se a leitura for bem-sucedida (ou seja, dispositivo responde),
25 if (ret >= 0) {
26 printf("Dispositivo encontrado no endereço: 0x%02X\n", addr);
27 found = true;
28 }
29 }
30
31 if (!found) {
32 printf("Nenhum dispositivo I2C encontrado no %s.\n", bus_name);
33 }
34 }
```

Este projeto utiliza a Bitdoglab ligada à sensores digitais de umidade/temperatura **AHT20**, pressão/temperatura/altitude **BMP280** e um display OLED I2C **SSD1306**.

### Funcionalidades

- Leitura simultânea de temperatura e umidade (AHT10 ou AHT20)
- Leitura de temperatura, pressão e cálculo de altitude (BMP280)
- Exibição das informações em tempo real no display OLED SSD1306
- Comunicação via barramento I2C com múltiplos dispositivos em diferentes barramentos (i2c0 e i2c1)
- Saída dos dados para o terminal serial (debug/log)
- Função BOOTSEL no botão B (GPIO 6) para facilitar a regravação do firmware



```
lib > C bmp280.c > bmp280_init(i2c_inst_t *)
1 #include "bmp280.h"
2 #include "hardware/i2c.h"
3
4 #define ADDR _u(0x76)
5

lib > C bmp280.c > ...
1 #include "bmp280.h"
2 #include "hardware/i2c.h"
3
4 #define ADDR _u(0x77)
5
```

```
lib > C bmp280.h > ...
1 #ifndef BMP280_H
2 #define BMP280_H
3
4 #include "hardware/i2c.h"
5
6 // Defina os endereços e registros
7 #define ADDR _u(0x76)
8
```

```
lib > C bmp280.h > ...
1 #ifndef BMP280_H
2 #define BMP280_H
3
4 #include "hardware/i2c.h"
5
6 // Defina os endereços e registros
7 #define ADDR _u(0x77)
```

<https://github.com/wiltonlacerda/EmbarcaTechResU3Ex05.git>

# Sensores

## Exemplo 2:

```
#include <stdio.h>
#include "pico/stdc.h"
#include "hardware/i2c.h"
#include "aht20.h"
#include "bmp280.h"
#include "ssd1306.h"
#include "font.h"
#include <math.h>
```

```
#define I2C_PORT i2c0
#define I2C_SDA 0
#define I2C_SCL 1
#define SEA_LEVEL_PRESSURE 101325.0
// Display na I2C
#define I2C_PORT_DISP i2c1
#define I2C_SDA_DISP 14
#define I2C_SCL_DISP 15
#define endereco 0x3C
```

```
// Função para calcular a altitude a partir da pressão atmosférica
double calculate_altitude(double pressure)
{
 return 44330.0 * (1.0 - pow(pressure / SEA_LEVEL_PRESSURE, 0.1903));
}

// Trecho para modo BOOTSEL com botão B
#include "pico/bootrom.h"
#define botaoB 6
void gpio_irq_handler(uint gpio, uint32_t events)
{
 reset_usb_boot(0, 0);
}

int main()
{
 // Para ser utilizado o modo BOOTSEL com botão B
 gpio_init(botaoB);
 gpio_set_dir(botaoB, GPIO_IN);
 gpio_pull_up(botaoB);
 gpio_set_irq_enabled_with_callback(botaoB, GPIO_IRQ_EDGE_FALL, true, &gpio_irq_handler);
 // Fim do trecho para modo BOOTSEL com botão B
}
```

# Sensores

## Exemplo 2:

```
stdio_init_all();

// I2C do Display funcionando em 400Khz.
i2c_init(I2C_PORT_DISP, 400 * 1000);

gpio_set_function(I2C_SDA_DISP, GPIO_FUNC_I2C);
gpio_set_function(I2C_SCL_DISP, GPIO_FUNC_I2C);
gpio_pull_up(I2C_SDA_DISP);
gpio_pull_up(I2C_SCL_DISP);
ssd1306_t ssd;
ssd1306_init(&ssd, WIDTH, HEIGHT, false, endereco, I2C_PORT_DISP);
ssd1306_config(&ssd);
ssd1306_send_data(&ssd);

// Limpa o display. O display inicia com todos os pixels apagados.
ssd1306_fill(&ssd, false);
ssd1306_send_data(&ssd);

// Inicializa o I2C
i2c_init(I2C_PORT, 400 * 1000);
gpio_set_function(I2C_SDA, GPIO_FUNC_I2C);
gpio_set_function(I2C_SCL, GPIO_FUNC_I2C);
gpio_pull_up(I2C_SDA);
gpio_pull_up(I2C_SCL);
```

```
// Inicializa o BMP280
bmp280_init(I2C_PORT);
struct bmp280_calib_params params;
bmp280_get_calib_params(I2C_PORT, ¶ms);

// Inicializa o AHT20
aht20_reset(I2C_PORT);
aht20_init(I2C_PORT);

// Estrutura para armazenar os dados do sensor
AHT20_Data data;
int32_t raw_temp_bmp;
int32_t raw_pressure;

char str_tmp1[5]; // Buffer para armazenar a string
char str_alt[5]; // Buffer para armazenar a string
char str_tmp2[5]; // Buffer para armazenar a string
char str_umi[5]; // Buffer para armazenar a string

bool cor = true;
while (1)
{
```

# Sensores

## Exemplo 2:

```
bool cor = true;
while (1)
{
 // Leitura do BMP280
 bmp280_read_raw(I2C_PORT, &raw_temp_bmp, &raw_pressure);
 int32_t temperature = bmp280_convert_temp(raw_temp_bmp, ¶ms);
 int32_t pressure = bmp280_convert_pressure(raw_pressure, raw_temp_bmp, ¶ms);

 // Cálculo da altitude
 double altitude = calculate_altitude(pressure);

 printf("Pressao = %.3f kPa\n", pressure / 1000.0);
 printf("Temperatura BMP: = %.2f C\n", temperature / 100.0);
 printf("Altitude estimada: %.2f m\n", altitude);

 // Leitura do AHT20
 if (aht20_read(I2C_PORT, &data))
 {
 printf("Temperatura AHT: %.2f C\n", data.temperature);
 printf("Umidade: %.2f %%\n\n\n", data.humidity);
 }
 else
 {
 printf("Erro na leitura do AHT10!\n\n\n");
 }
}
```

```
sprintf(str_tmp1, "%.1fC", temperature / 100.0); /
sprintf(str_alt, "%.0fm", altitude); // Converte o
sprintf(str_tmp2, "%.1fC", data.temperature); // C
sprintf(str_umi, "%.1f%%", data.humidity); // Conv

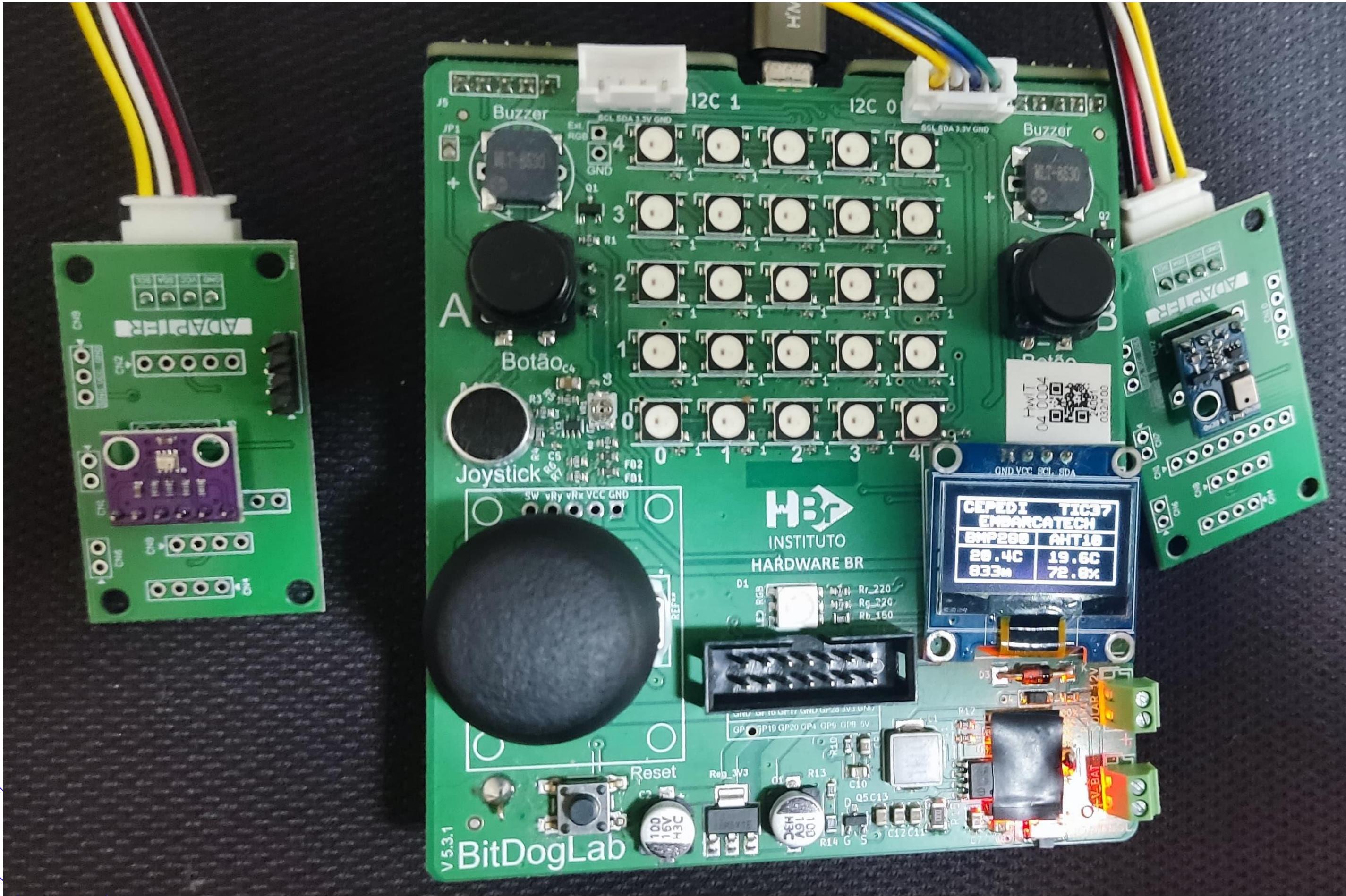
// Atualiza o conteúdo do display com animações
ssd1306_fill(&ssd, !cor);
ssd1306_rect(&ssd, 3, 3, 122, 60, cor, !cor);
ssd1306_line(&ssd, 3, 25, 123, 25, cor);
ssd1306_line(&ssd, 3, 37, 123, 37, cor);
ssd1306_draw_string(&ssd, "CEPEDI TIC37", 8, 6);
ssd1306_draw_string(&ssd, "EMBARCATECH", 20, 16);
ssd1306_draw_string(&ssd, "BMP280 AHT10", 10, 28);
ssd1306_line(&ssd, 63, 25, 63, 60, cor);
ssd1306_draw_string(&ssd, str_tmp1, 14, 41);
ssd1306_draw_string(&ssd, str_alt, 14, 52);
ssd1306_draw_string(&ssd, str_tmp2, 73, 41);
ssd1306_draw_string(&ssd, str_umi, 73, 52);
ssd1306_send_data(&ssd);

sleep_ms(500);

}

return 0;
```

# BitDogLab + AHT10 + BMP280



## Atenção:

Evite posicionar as placas com sensores sobre superfícies metálicas ou diretamente sobre a placa BitDogLab, pois isso pode causar curto-circuito e danificar os componentes.

# Tarefa



## Tarefa:

### Estação Meteorológica com Interface Web

#### Enunciado

O objetivo deste projeto é consolidar os conhecimentos sobre **sensores I<sup>2</sup>C** integrados à plataforma BitDogLab e sobre o desenvolvimento de interfaces web, por meio da criação de uma estação meteorológica capaz de monitorar **temperatura, umidade e pressão atmosférica**.

#### Descrição do Projeto

Cada residente desenvolverá **individualmente** um sistema embarcado que lê dados de dois sensores: AHT20/AHT10 (temperatura e umidade relativa) e BMP280 (pressão barométrica e temperatura).

O sistema deverá:

- Capturar continuamente os valores de temperatura, umidade e pressão.
- Exibir esses dados localmente em um display OLED SSD1306.
- Servir uma página HTML responsiva via Wi-Fi, exibindo em tempo real os valores lidos e permitindo configurar limites e calibrações. **Gerar Gráficos**.
- Emitir alertas sonoros (buzzer) e visuais (LED RGB e matriz de LEDs) caso algum parâmetro ultrapasse limites pré-definidos pelo usuário.

O vídeo deve estar na **horizontal**, como se estivesse segurando o celular deitado.





# Obrigado!

Executores:



Coordenação:



Iniciativa:

