Liga Acadêmica de Sistemas Embarcados
Centro de Informática · UFPE

FORMAÇÃO EM ESP32 · MÓDULO 2

Estação de **temperatura e umidade**

Um DHT22, um display LCD 16×2 e um ESP32-S3, primeiro no simulador Wokwi e depois na bancada. Dois protocolos de comunicação, um programa curto e os cuidados elétricos que fazem o circuito funcionar fora da tela também.

Lucas Rayan Guerra
Ciência Embarcada · ESPDocs linkedin.com/in/lucasrguerra

Material de apoio
Formação presencial · uso livre para fins educacionais

Apresentação

O que vamos construir, o que você precisa e como usar esta apostila.

No módulo 1 lemos um sensor e imprimimos o resultado no monitor serial, que só existe enquanto há um computador ligado à placa. Neste módulo o projeto ganha **uma tela própria**: um display LCD de 16 colunas por 2 linhas mostra a temperatura e a umidade medidas por um **DHT22**, e o conjunto passa a funcionar sozinho, ligado em qualquer carregador USB.

Toda a formação acontece no **Wokwi**, um simulador de microcontroladores que roda no navegador. Ele compila o mesmo código que você gravaria na placa, simula o ESP32-S3, o sensor e o display, e deixa você mudar a temperatura com um controle deslizante. Assim cada pessoa monta e testa o circuito sem precisar de componentes, e quem quiser repetir em casa com peças de verdade encontra no capítulo 7 o que muda fora do simulador.

O que você vai levar deste módulo

- Montar e simular um projeto completo de ESP32 no Wokwi, do zero.
- Entender como o DHT22 transmite uma medição por um único fio.
- Entender o barramento I2C: dois fios, endereços e resistores de pull-up.
- Escrever no LCD 16×2, inclusive com um caractere que você mesmo desenha.
- Saber o que precisa mudar para o mesmo circuito funcionar na bancada.

Pré-requisitos

Ter acompanhado o módulo 1 ajuda, mas não é obrigatório. Basta saber o que são `setup()` e `loop()` num programa Arduino. Para a simulação você precisa só de um navegador atualizado; não é preciso instalar nada nem criar conta.

Os arquivos deste módulo

Tudo o que aparece nesta apostila foi compilado e executado no Wokwi, e os arquivos estão na mesma pasta da formação:

ARQUIVO	CONTEÚDO
sketch.ino	O programa da estação
diagram.json	O circuito: peças, posições e fios
libraries.txt	As bibliotecas que o Wokwi deve instalar
scanner-i2c.ino	Programa de diagnóstico do barramento I2C
modulo-2-wokwi.zip	Os quatro arquivos acima num só pacote
Apostila em PDF	Esta apostila, pronta para imprimir
Slides em PDF	Os slides da formação, em 16:9

Ao longo do texto você encontra quatro tipos de caixa:

DICA

Atalho, boa prática ou recomendação que economiza tempo.

ATENÇÃO

Ponto que costuma causar problema se passar despercebido.

ERRO COMUM

Armadilha concreta, com o sintoma que ela produz e como escapar.

CURIOSIDADE

Contexto ou detalhe técnico que ajuda a fixar o conceito.

Sumário

Oito capítulos, da peça solta ao circuito funcionando.

1 O projeto 5

2 O sensor DHT22 7

3 O display LCD 16×2 e o barramento I2C 9

4 O simulador Wokwi 10

5 Montagem 12

6 O programa 14

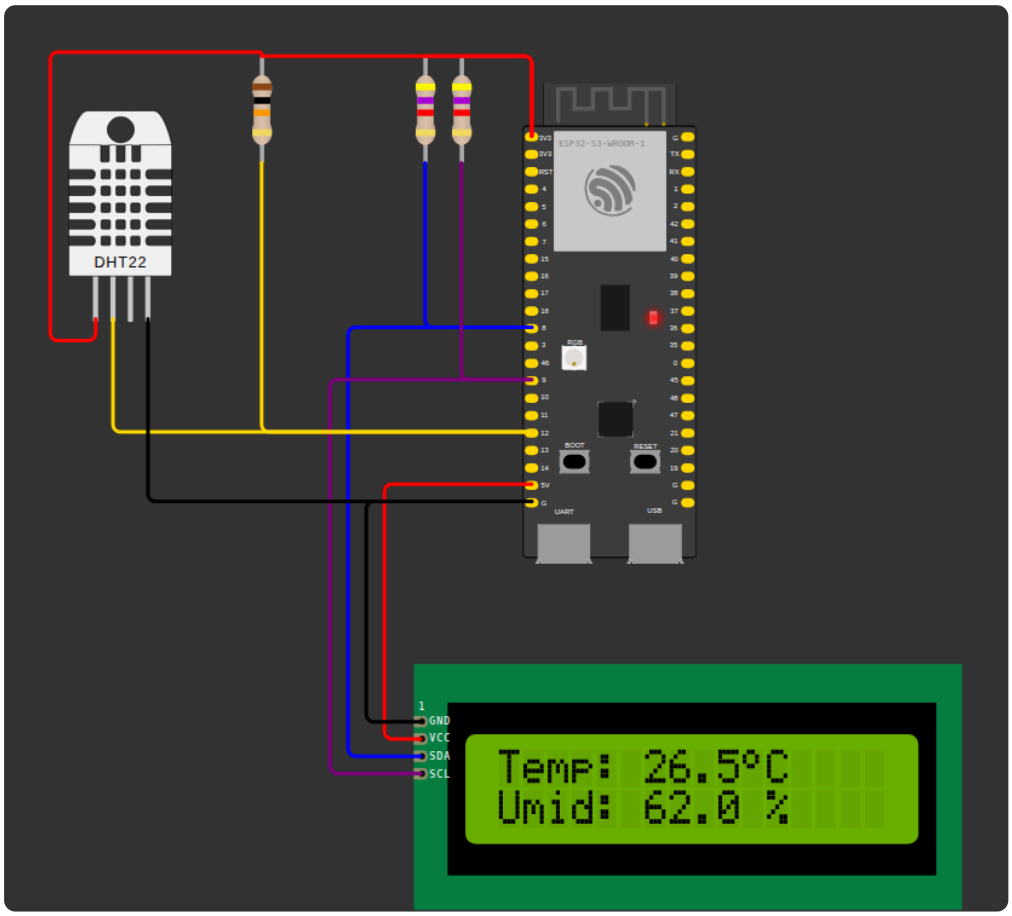
7 Da simulação para a bancada 17

8 Problemas comuns e desafios 19

1 O projeto

Três peças, dois protocolos e uma tela que mostra o ambiente.

A estação é composta por três peças. O **DHT22** mede temperatura e umidade relativa do ar. O **ESP32-S3** pede uma medição a cada dois segundos, formata os números e os envia ao **LCD 16×2**, que os mantém na tela até a próxima atualização.



O circuito deste módulo rodando no Wokwi. O display mostra a leitura que o DHT22 simulado está informando: 26,5 °C e 62 % de umidade.

Cada peça conversa com o ESP32 de um jeito diferente, e é isso que torna o projeto bom para aprender. O DHT22 usa um **protocolo próprio de um fio**, em que o valor de cada bit está na duração de um pulso. O LCD usa o **I2C**, um barramento padronizado de dois fios em que cada dispositivo tem um endereço. Quem entende esses dois protocolos já entende a maior parte dos sensores e módulos baratos do mercado.

Lista de material

No Wokwi as peças já existem. Para montar em bancada você vai precisar de:

ITEM	QTD.	OBSERVAÇÃO
ESP32-S3-DevKitC-1	1	Qualquer variante de memória serve
Sensor DHT22 (AM2302)	1	O componente branco de 4 pinos
Display LCD 16×2 com módulo I2C	1	A plaquinha PCF8574 vem soldada atrás
Conversor de nível I2C (BSS138)	1	Explicado no capítulo 7
Resistor de 10 kΩ	1	Pull-up do DHT22 (marrom, preto, laranja)
Resistor de 4,7 kΩ	2	Pull-ups do I2C (amarelo, violeta, vermelho)
Protoboard, jumpers e cabo USB de dados	—	Cabo só de carga não grava a placa

2 O sensor DHT22

Dois sensores numa caixinha e um protocolo de um fio só.

Dentro do DHT22 existem dois elementos de medição e um pequeno microcontrolador. A umidade é medida por um **sensor capacitivo**: um material que absorve vapor d'água e, com isso, muda a capacitância. A temperatura vem de um **termistor**, um resistor cujo valor varia com o calor. O microcontrolador interno lê os dois, aplica a calibração gravada de fábrica e entrega os números já prontos, em formato digital.

DHT22 ou DHT11

No módulo 1 usamos o DHT11. O DHT22 tem a mesma pinagem e o mesmo protocolo, mas mede melhor:

CARACTERÍSTICA	DHT11	DHT22
Faixa de temperatura	0 a 50 °C	**−40 a 80 °C**
Precisão de temperatura	±2 °C	**±0,5 °C**
Faixa de umidade	20 a 90 %	**0 a 100 %**
Precisão de umidade	±5 %	**±2 a 5 %**
Resolução	1 °C e 1 %	**0,1 °C e 0,1 %**
Intervalo mínimo entre leituras	**1 s**	2 s
Alimentação	3,3 a 5 V	3,3 a 6 V

Valores típicos dos datasheets da Aosong. Confira sempre o do seu fornecedor.

O preço é a única desvantagem real do DHT22, e o intervalo de dois segundos entre leituras raramente importa: temperatura e umidade de um ambiente não mudam mais rápido do que isso.

Os quatro pinos

PINO	NOME	LIGAÇÃO
1	VCC	3V3 do ESP32
2	DATA	GPIO12, com pull-up de 10 kΩ para o 3V3
3	NC	Não conectado
4	GND	GND do ESP32

Pinos contados da esquerda para a direita, com a grade do sensor virada para você.

Como uma medição atravessa um único fio

O ESP32 e o sensor dividem o mesmo fio de dados, e por isso nenhum dos dois "força" nível alto: os dois apenas puxam a linha para o terra quando precisam de um zero e a soltam no resto do tempo. Quem traz a linha de volta para o nível alto é o resistor de **pull-up**. Uma leitura acontece assim:

1 O ESP32 pede a medição

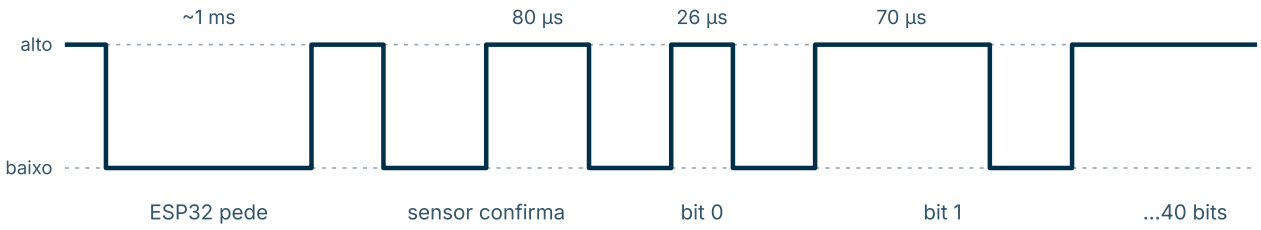
Puxa a linha para baixo por cerca de 1 ms e depois a solta.

2 O sensor confirma

Responde com 80 µs em nível baixo seguidos de 80 µs em nível alto.

3 Chegam 40 bits

Cada bit começa com 50 µs em baixo. Depois vem um pulso alto: se ele dura cerca de 26 µs é um **0**; se dura cerca de 70 µs é um **1**. São 16 bits de umidade, 16 de temperatura e 8 de verificação.



O valor de cada bit está na **largura** do pulso alto. Sem o pull-up a linha sobe devagar demais, os tempos saem errados e a leitura falha.

SENSOR SOLTO OU MÓDULO EM PLAQUINHA

O DHT22 também é vendido soldado numa plaquinha de três pinos, que já traz o resistor de pull-up. Se o seu for assim, o resistor externo de 10 kΩ é dispensável. O sensor solto, de quatro pinos, **precisa** dele.

3 O display LCD 16x2 e o barramento I2C

Trinta e dois caracteres, quatro fios e um endereço.

O LCD 16×2 mostra duas linhas de dezesseis caracteres. Cada caractere é uma matriz de **5 por 8 pontos**, e o controlador do display, o HD44780 ou um compatível, já sabe desenhar letras, números e símbolos comuns. Ele também guarda até **oito caracteres personalizados**, que você desenha ponto a ponto. É assim que este projeto mostra o símbolo de grau.

Sozinho, o display precisa de pelo menos seis fios de dados e controle. Por isso ele quase sempre vem com uma plaquinha soldada atrás, baseada no chip **PCF8574**, que converte esses sinais para o barramento **I2C**: com ela, bastam quatro fios para alimentar e controlar o display. A plaquinha também tem um **trimpot** para ajustar o contraste.

Como o I2C funciona

O I2C usa dois fios: **SDA**, por onde passam os dados, e **SCL**, o relógio que marca o ritmo da conversa. O ESP32 é o *controlador*: é sempre ele quem inicia a comunicação. Os outros dispositivos são *alvos*, e cada um tem um endereço de 7 bits. Vários dispositivos podem dividir os mesmos dois fios, porque só responde quem tiver o endereço chamado.

PLAQUINHA	ENDEREÇO	COMO SABER
PCF8574 ou PCF8574T	0x27	A mais comum, e a usada no Wokwi
PCF8574A	0x3F	Mesma placa, outro chip

Na dúvida, o programa de scanner do capítulo 7 descobre o endereço por você.

Assim como no DHT22, as linhas do I2C são de **dreno aberto**: os dispositivos só puxam para baixo, e são resistores de pull-up que trazem SDA e SCL de volta ao nível alto. Valores entre 2,2 kΩ e 10 kΩ funcionam na velocidade padrão; aqui usamos 4,7 kΩ, o valor mais comum.

O DISPLAY LIGA MAS NÃO MOSTRA NADA

Com o contraste no mínimo, a tela acende e parece vazia; com ele no máximo, aparecem só retângulos cheios. Antes de desconfiar do código, gire o trimpot azul atrás do display com uma chave de fenda até os caracteres aparecerem. No Wokwi isso não existe, porque o contraste simulado já vem ajustado.

4 O simulador Wokwi

Um laboratório de eletrônica que abre no navegador.

O Wokwi simula o processador do ESP32-S3 instrução por instrução e simula também cada peça ligada a ele. O programa é compilado com o mesmo **núcleo Arduino da Espressif** que você instalaria na IDE, então o código que funciona aqui é o mesmo que vai para a placa.

Os três arquivos de um projeto

ARQUIVO	PARA QUE SERVE
sketch.ino	O programa, exatamente como na IDE do Arduino.
diagram.json	As peças, a posição de cada uma, os fios e as propriedades, como a temperatura inicial do DHT22.
libraries.txt	Uma biblioteca por linha. O Wokwi baixa todas antes de compilar.

Abrindo o projeto deste módulo

1 Crie um projeto vazio

Acesse **wokwi.com/projects/new/esp32-s3**. Não é preciso criar conta para simular.

2 Substitua o programa e o circuito

Cole o conteúdo de `sketch.ino` na aba de mesmo nome e o de `diagram.json` na aba *diagram.json*, apagando o que já estava lá.

3 Adicione as bibliotecas

Abra a aba *Library Manager*, clique em **+** e adicione **DHT sensor library** e **LiquidCrystal I2C**. O Wokwi cria o `libraries.txt` sozinho.

4 Simule

Clique no botão verde. A primeira compilação leva alguns segundos; depois o display acende e o monitor serial aparece embaixo.

MUDE O CLIMA COM UM CLIQUE

Com a simulação rodando, clique no DHT22: aparecem dois controles deslizantes, um de temperatura e outro de umidade. Mova-os e veja o display acompanhar na leitura seguinte. É o jeito mais rápido de testar o que o programa faz com valores extremos.

SALVAR EXIGE CONTA

Sem login a simulação funciona normalmente, mas o projeto se perde ao fechar a aba. Para guardar o seu trabalho, entre com uma conta gratuita e clique em **Save**, ou mantenha os três arquivos no seu computador.

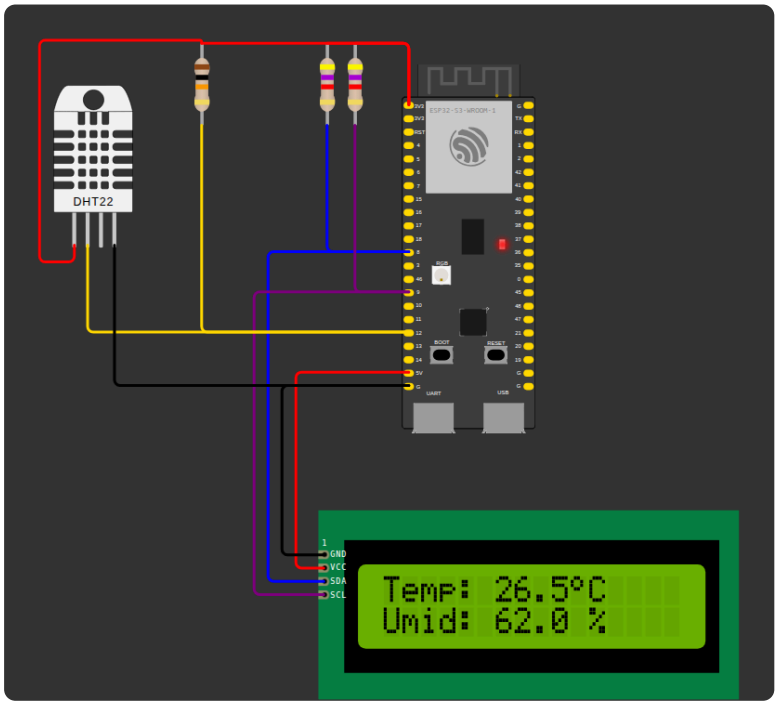
5 Montagem

Nove ligações e três resistores.

Todos os pinos usados ficam na **coluna esquerda** da placa, o que deixa a fiação curta e sem cruzamentos. As cores seguem uma convenção que vale a pena adotar também na bancada: vermelho para alimentação, preto para terra, e uma cor por sinal.

DE	PARA	COR E FUNÇÃO
DHT22 VCC	3V3	Vermelho · alimentação
DHT22 DATA	GPIO12	Amarelo · dados do sensor
DHT22 GND	GND	Preto · terra
Resistor 10 kΩ	GPIO12 ↔ 3V3	Pull-up do DHT22
LCD VCC	5V	Vermelho · o display precisa de 5 V
LCD GND	GND	Preto · terra
LCD SDA	GPIO8	Azul · dados do I2C
LCD SCL	GPIO9	Roxo · relógio do I2C
2 resistores de 4,7 kΩ	GPIO8 e GPIO9 ↔ 3V3	Pull-ups do I2C

GPIO8 e GPIO9 são os pinos I2C padrão do ESP32-S3 no núcleo Arduino.



O fio vermelho no alto funciona como um barramento de 3V3: dele saem a alimentação do DHT22 e o lado de cima dos três resistores. O lado de baixo de cada resistor encontra a linha de sinal correspondente.

POR QUE PULL-UP PARA 3V3, E NÃO PARA 5V

Os pinos do ESP32-S3 trabalham em 3,3 V e **não toleram 5 V**. Um resistor de pull-up define a tensão de repouso da linha, então ligá-lo ao 5V colocaria 5 V direto no GPIO. O display é alimentado em 5 V, mas as linhas de sinal precisam ficar em 3,3 V. Na bancada isso exige um cuidado a mais, explicado no capítulo 7.

6 O programa

Um programa curto, lido em três partes.

O programa é lido aqui em três partes, exatamente como foi compilado e executado no Wokwi. O arquivo completo, pronto para colar, está em wokwi/sketch.ino.

1. Configuração

```
#include <Wire.h>
#include <DHT.h>
#include <LiquidCrystal_I2C.h>

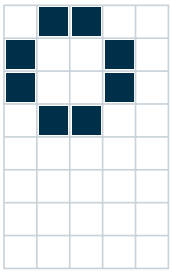
#define PINO_DHT  12        // fio amarelo
#define TIPO_DHT  DHT22
#define PINO_SDA  8         // I2C padrão do ESP32-S3
#define PINO_SCL  9

DHT dht(PINO_DHT, TIPO_DHT);
LiquidCrystal_I2C lcd(0x27, 16, 2);   // endereço, colunas, linhas

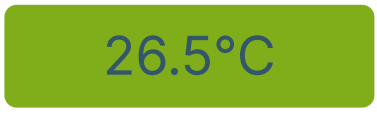
// Símbolo de grau: 8 linhas de 5 pixels
byte grau[8] = { 0b01100, 0b10010, 0b10010, 0b01100, 0, 0, 0, 0 };
```

As três bibliotecas cuidam do trabalho pesado: `Wire` é o I2C do ESP32, `DHT` fala o protocolo do sensor e `LiquidCrystal_I2C` traduz comandos como "escreva na linha 2" para o que a plaquinha PCF8574 entende. Os `#define` reúnem no topo tudo o que depende da montagem: se você mudar um fio de pino, só precisa mudar uma linha.

O vetor `grau` desenha o símbolo de grau. Cada número é uma linha do caractere, e cada bit `1` acende um dos cinco pontos daquela linha:



0b01100
0b10010
0b10010
0b01100
0
0
0
0



o que aparece no display

`lcd.createChar(0, grau)` guarda o desenho na posição 0 da memória do display, e `lcd.write(0)` o imprime como qualquer outro caractere.

2. setup()

```
void setup() {
  Serial.begin(115200);
  Wire.begin(PINO_SDA, PINO_SCL);
  dht.begin();

  lcd.init();
  lcd.backlight();
  lcd.createChar(0, grau);
  lcd.setCursor(0, 0);
  lcd.print("LASER  ESP32-S3");
  lcd.setCursor(0, 1);
  lcd.print("Iniciando...");
  delay(2000);            // o DHT22 precisa de ~2 s para a primeira leitura
  lcd.clear();
}
```

`Wire.begin(8, 9)` inicia o I2C nos pinos da montagem. Em seguida o display é inicializado, a luz de fundo é ligada e o caractere de grau é registrado. A mensagem "Iniciando..." não é só enfeite: ela cobre os **dois segundos** que o DHT22 precisa depois de ligado para entregar a primeira medição válida.

3. loop()

```
void loop() {
  float umidade     = dht.readHumidity();
  float temperatura = dht.readTemperature();

  if (isnan(umidade) || isnan(temperatura)) {
    lcd.setCursor(0, 0);
    lcd.print("Falha no sensor ");
    Serial.println("Falha na leitura do DHT22");
  } else {
    lcd.setCursor(0, 0);
    lcd.print("Temp: ");
    lcd.print(temperatura, 1);
    lcd.write(0);           // o caractere de grau criado acima
    lcd.print("C   ");

    lcd.setCursor(0, 1);
    lcd.print("Umid: ");
    lcd.print(umidade, 1);
    lcd.print(" %   ");

    Serial.printf("Temperatura: %.1f C | Umidade: %.1f %%\n", temperatura, umidade);
  }

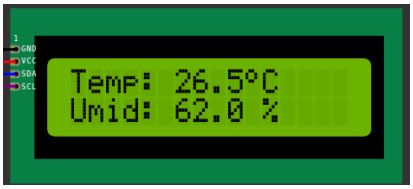
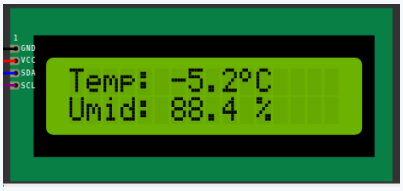
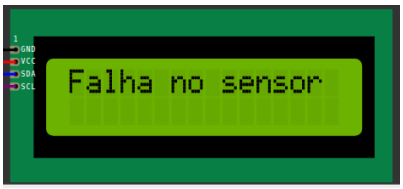
  delay(2000);              // o DHT22 aceita no máximo uma leitura a cada 2 s
}
```

A cada volta, o programa lê umidade e temperatura. Se qualquer uma vier como `NaN`, a sigla de *not a number* com que a biblioteca avisa que a leitura falhou, o display mostra "Falha no sensor". Caso contrário, os valores são escritos com uma casa decimal. O `delay(2000)` respeita o intervalo mínimo do DHT22.

RESTOS DE TEXTO NA TELA

O LCD só sobrescreve as posições em que você escreve; o resto continua como estava. Se a temperatura cair de "26.5" para "-5.2" e depois para "8.0", o texto fica mais curto e sobram caracteres da leitura anterior. Os espaços em `"C "` e `" % "` existem para apagar essas sobras. Chamar `lcd.clear()` a cada leitura também resolveria, mas faz a tela piscar.

Os três testes feitos no Wokwi

SITUAÇÃO	O QUE O DISPLAY MOSTROU
Sala, 26,5 °C e 62 %	
Frio, −5,2 °C e 88,4 %	
Fio de dados desconectado	

O terceiro teste confirma que o programa detecta a falha em vez de mostrar números inventados.

7 Da simulação para a bancada

O que o Wokwi perdoa e o hardware de verdade não.

O simulador é ideal para aprender e testar a lógica, mas ele modela os componentes como peças perfeitas. Na bancada, três detalhes fazem diferença.

1. Os pull-ups escondidos na plaquinha I2C

Quase toda plaquinha PCF8574 vendida no Brasil já traz **resistores de pull-up ligados ao VCC dela**. Com o display alimentado em 5 V, esses resistores puxam SDA e SCL para 5 V. Somar os nossos 4,7 kΩ para 3V3 não resolve: os dois resistores formam um divisor, e a linha fica em torno de 4 V, ainda acima do que o ESP32-S3 suporta.

Há duas saídas corretas:

SOLUÇÃO	COMO FAZER
Conversor de nível (recomendado)	Um módulo bidirecional com BSS138, de quatro canais. O lado **LV** vai no 3V3 e nos GPIO8/9; o lado **HV** vai no 5V e no SDA/SCL do display. O módulo já tem pull-ups dos dois lados, então os 4,7 kΩ externos saem do circuito.
Remover os pull-ups da plaquinha	Dessoldar os dois resistores ligados a SDA e SCL na plaquinha e manter os 4,7 kΩ para 3V3, exatamente como no Wokwi. O PCF8574 reconhece 3,3 V como nível alto mesmo alimentado em 5 V.

COMO SABER SE A SUA PLAQUINHA TEM PULL-UPS

Com tudo desligado, meça com o multímetro a resistência entre o pino **SDA** e o **VCC** da plaquinha. Algo entre 2 kΩ e 10 kΩ indica um pull-up. Se não houver continuidade, ela não tem, e os 4,7 kΩ para 3V3 bastam.

"MAS EU LIGUEI DIRETO E FUNCIONOU"

É comum funcionar: o resistor limita a corrente e o ESP32 tem diodos de proteção que desviam o excesso. Funcionar não é o mesmo que estar dentro da especificação. A Espressif não garante os pinos acima de 3,6 V, e um circuito fora da especificação pode falhar com o tempo ou em outra placa. Em projeto que vai ficar ligado, use o conversor.

2. Descubra o endereço do display

Se o display acende mas não mostra nada mesmo com o contraste ajustado, o endereço pode ser `0x3F` em vez de `0x27`. Este programa varre o barramento e imprime o endereço de cada dispositivo que responder. No Wokwi ele encontra o display em `0x27`:

```ino
// Scanner I2C: lista o endereço de cada dispositivo no barramento.
// Use antes do programa principal se o LCD não mostrar nada.

#include <Wire.h>

void setup() {
  Serial.begin(115200);
  Wire.begin(8, 9);                 // SDA, SCL
  delay(500);

  Serial.println("Procurando dispositivos I2C...");
  int encontrados = 0;
  for (byte endereco = 1; endereco < 127; endereco++) {
    Wire.beginTransmission(endereco);
    if (Wire.endTransmission() == 0) {
      Serial.printf("Dispositivo em 0x%02X\n", endereco);
      encontrados++;
    }
  }
  if (encontrados == 0) Serial.println("Nenhum dispositivo. Confira SDA, SCL e alimentacao.");
}

void loop() {}
```

```
Procurando dispositivos I2C...
Dispositivo em 0x27
```

3. Gravando na placa

Na IDE do Arduino, instale o suporte ao ESP32 e as bibliotecas **DHT sensor library** (aceitando a dependência **Adafruit Unified Sensor**) e **LiquidCrystal I2C**, de Frank de Brabander. Selecione a placa **ESP32S3 Dev Module** e lembre do detalhe do módulo 1: na porta marcada **USB**, a opção *USB CDC On Boot* precisa estar em **Enabled** para o monitor serial funcionar; na porta **UART**, em **Disabled**.

8 Problemas comuns e desafios

O que fazer quando não funciona, e para onde ir quando funcionar.

SINTOMA	CAUSA PROVÁVEL
"Falha no sensor" o tempo todo	Fio de dados fora do GPIO12, falta do pull-up de 10 kΩ ou VCC e GND invertidos.
Display aceso e sem texto	Contraste desajustado, endereço errado ou SDA e SCL trocados.
Retângulos cheios na linha de cima	O display recebeu energia mas não foi inicializado: o I2C não está chegando.
Display apagado	Falta o 5V no VCC do display ou o `lcd.backlight()`.
Caracteres estranhos no fim da linha	Sobras da leitura anterior; veja a caixa "Restos de texto na tela".
Erro de compilação com `DHT.h`	Biblioteca não instalada, ou instalada sem a Adafruit Unified Sensor.

Desafios

1 Mínima e máxima

Guarde a menor e a maior temperatura desde que a placa ligou e alterne a segunda linha entre a umidade e esses dois valores.

2 Alerta de calor

Ligue um LED, ou use o LED RGB que já existe no GPIO38 da placa, quando a temperatura passar de 30 °C. Faça a luz de fundo do display piscar junto.

3 Sensação térmica

A biblioteca DHT tem a função `computeHeatIndex()`. Mostre o índice de calor no lugar da umidade quando o ar estiver quente e úmido.

4 Sem `delay()`

Reescreva o `loop()` usando `millis()`, para que o programa possa fazer outras coisas, como ler um botão, entre uma leitura e outra.

PRÓXIMO MÓDULO

A estação já mede e mostra. O passo natural é publicar esses dados: o ESP32-S3 tem Wi-Fi, e com poucas linhas a mais a mesma leitura pode ir para uma página web ou para um servidor MQTT.

DÚVIDAS, CORREÇÕES E CONTATO

Correções e sugestões são bem-vindas e ajudam as próximas turmas. Fale direto com o autor por contato@lucasrguerra.dev.br ou pelo LinkedIn, em linkedin.com/in/lucasrguerra.

LASER · Liga Acadêmica de Sistemas Embarcados · Centro de Informática · UFPE
Formação em ESP32 · Módulo 2: Estação de Temperatura e Umidade