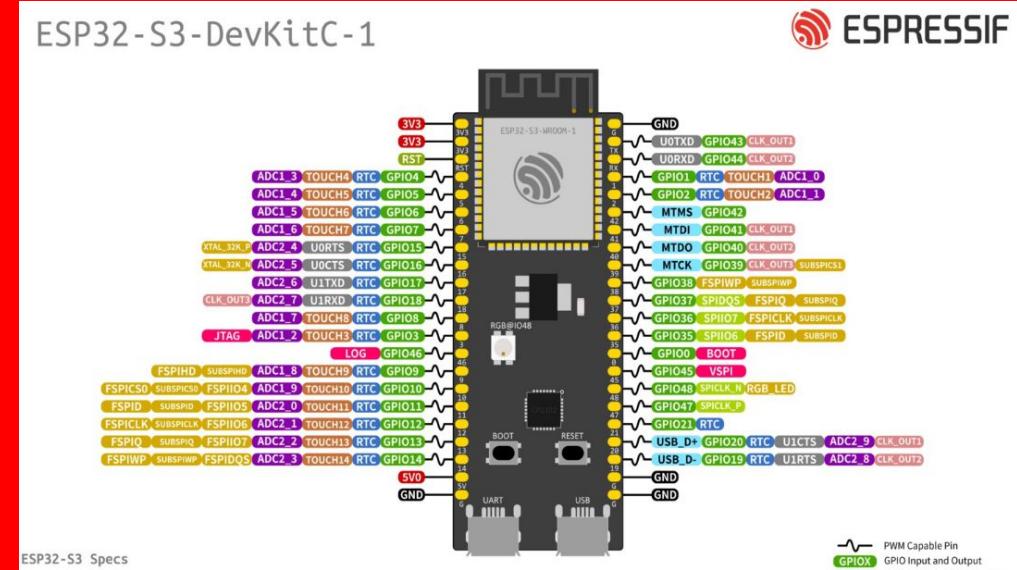


Entenda os pinos do ESP32-S3



3 Formas de ver o esp32

1ª

CHIP NO ENCAPSULAMENTO



2ª

CHIP DENTRO DE UM MÓDULO COM
COMPONENTES INCLUIDOS DENTRO

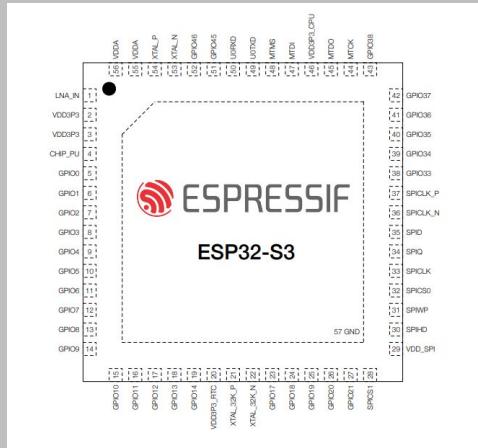


Espressif e outras marcas

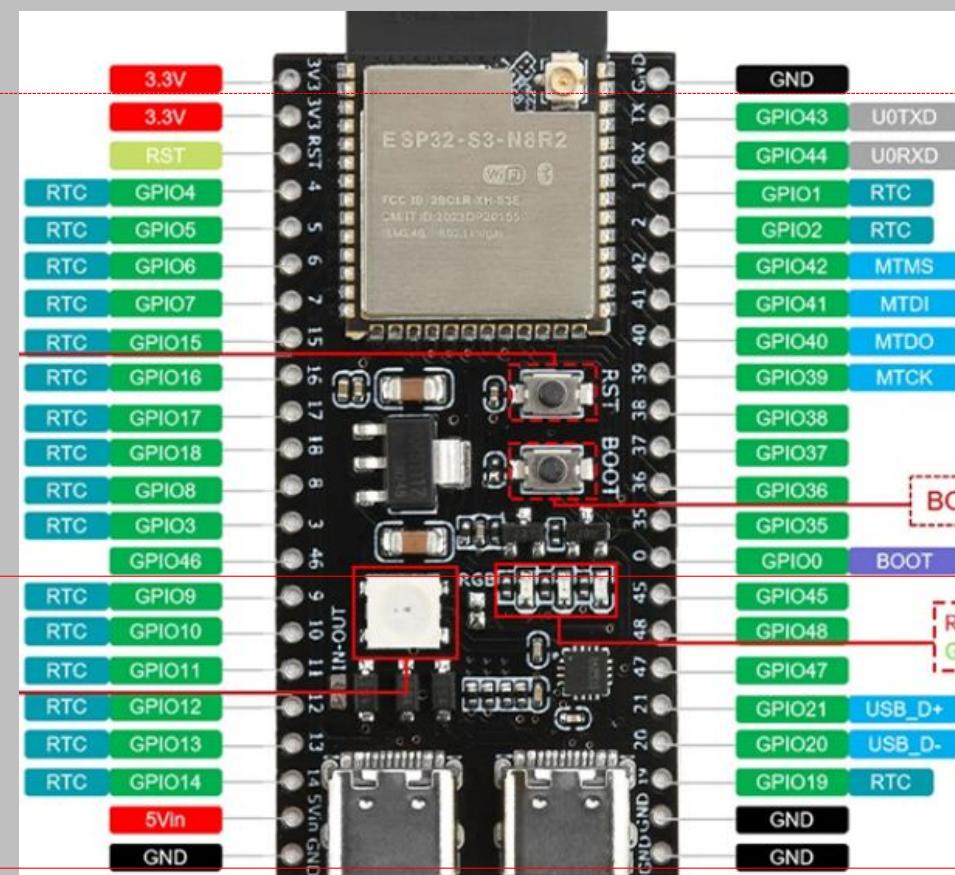
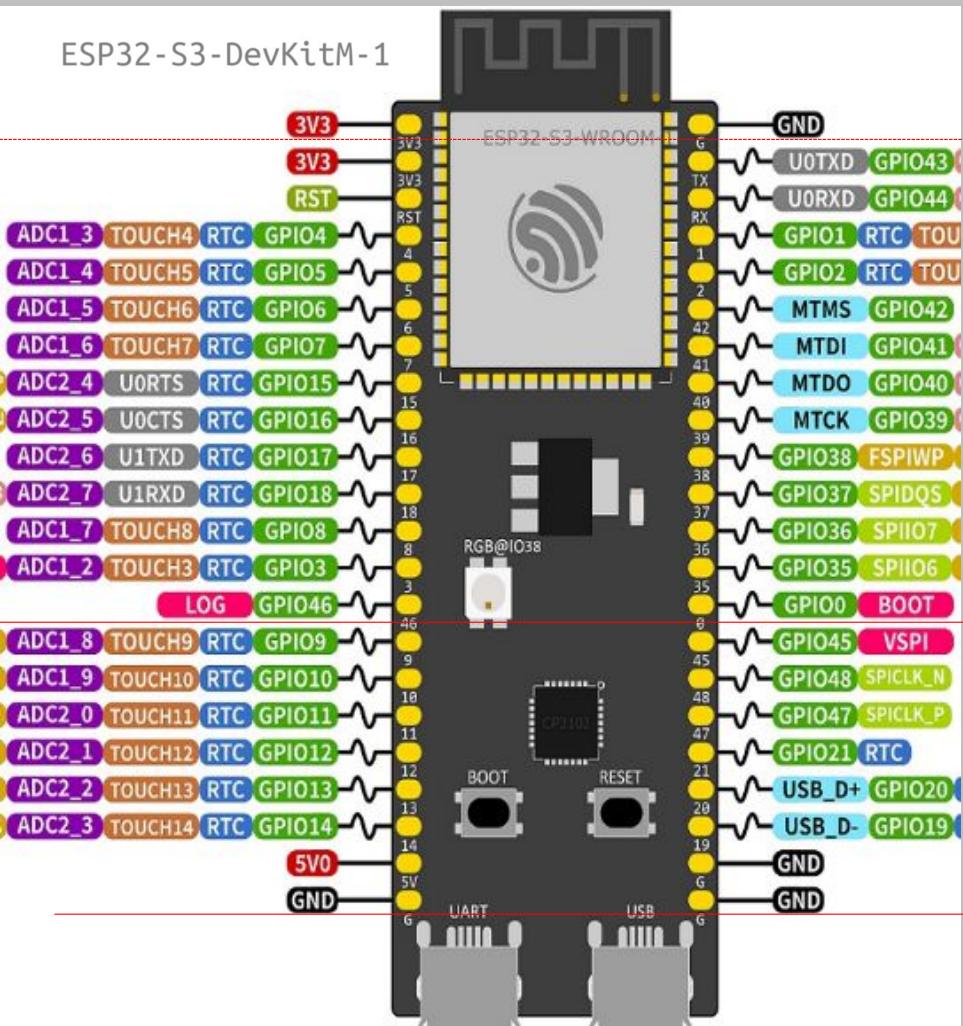
3ª

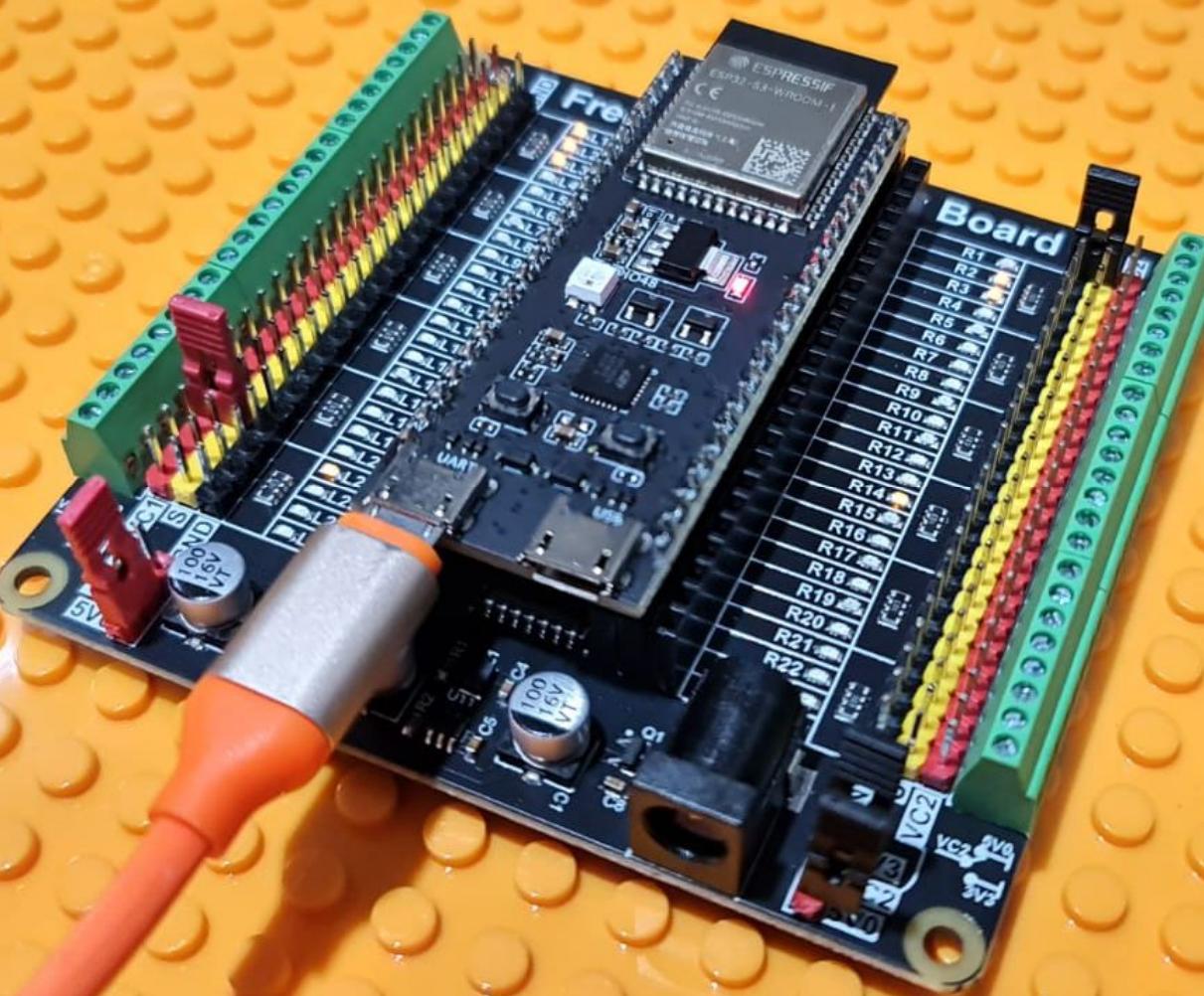


Espressif e outras marcas



ESP32-S3-DevKitM-1





```
=====
ESP32-S3 DevKitC-1 LED Sequencial
Versão Ultra Segura
=====
Pinos evitados: GPIO0, GPIO1, GPIO2, GPIO3
Total de pinos utilizados: 29
=====
Configurando GPIO42 -> OUTPUT (Pino 1/29)
Configurando GPIO41 -> OUTPUT (Pino 2/29)
Configurando GPIO40 -> OUTPUT (Pino 3/29)
Configurando GPIO39 -> OUTPUT (Pino 4/29)
Configurando GPIO38 -> OUTPUT (Pino 5/29)
Configurando GPIO37 -> OUTPUT (Pino 6/29)
Configurando GPIO36 -> OUTPUT (Pino 7/29)
Configurando GPIO35 -> OUTPUT (Pino 8/29)
Configurando GPIO45 -> OUTPUT (Pino 9/29)
Configurando GPIO48 -> OUTPUT (Pino 10/29)
Configurando GPIO47 -> OUTPUT (Pino 11/29)
Configurando GPIO21 -> OUTPUT (Pino 12/29)
Configurando GPIO20 -> OUTPUT (Pino 13/29)
Configurando GPIO19 -> OUTPUT (Pino 14/29)
Configurando GPIO18 -> OUTPUT (Pino 15/29)
Configurando GPIO17 -> OUTPUT (Pino 16/29)
Configurando GPIO16 -> OUTPUT (Pino 17/29)
Configurando GPIO15 -> OUTPUT (Pino 18/29)
Configurando GPIO17 -> OUTPUT (Pino 19/29)
Configurando GPIO06 -> OUTPUT (Pino 20/29)
Configurando GPIO05 -> OUTPUT (Pino 21/29)
Configurando GPIO04 -> OUTPUT (Pino 22/29)
Configurando GPIO09 -> OUTPUT (Pino 23/29)
Configurando GPIO10 -> OUTPUT (Pino 24/29)
Configurando GPIO11 -> OUTPUT (Pino 25/29)
Configurando GPIO12 -> OUTPUT (Pino 26/29)
Configurando GPIO13 -> OUTPUT (Pino 27/29)
Configurando GPIO14 -> OUTPUT (Pino 28/29)
Configurando GPIO08 -> OUTPUT (Pino 29/29)
=====
✓ Todos os pinos configurados com sucesso!
✓ Sistema pronto para iniciar sequência
✓ Aguardando 3 segundos para começar...
=====
```

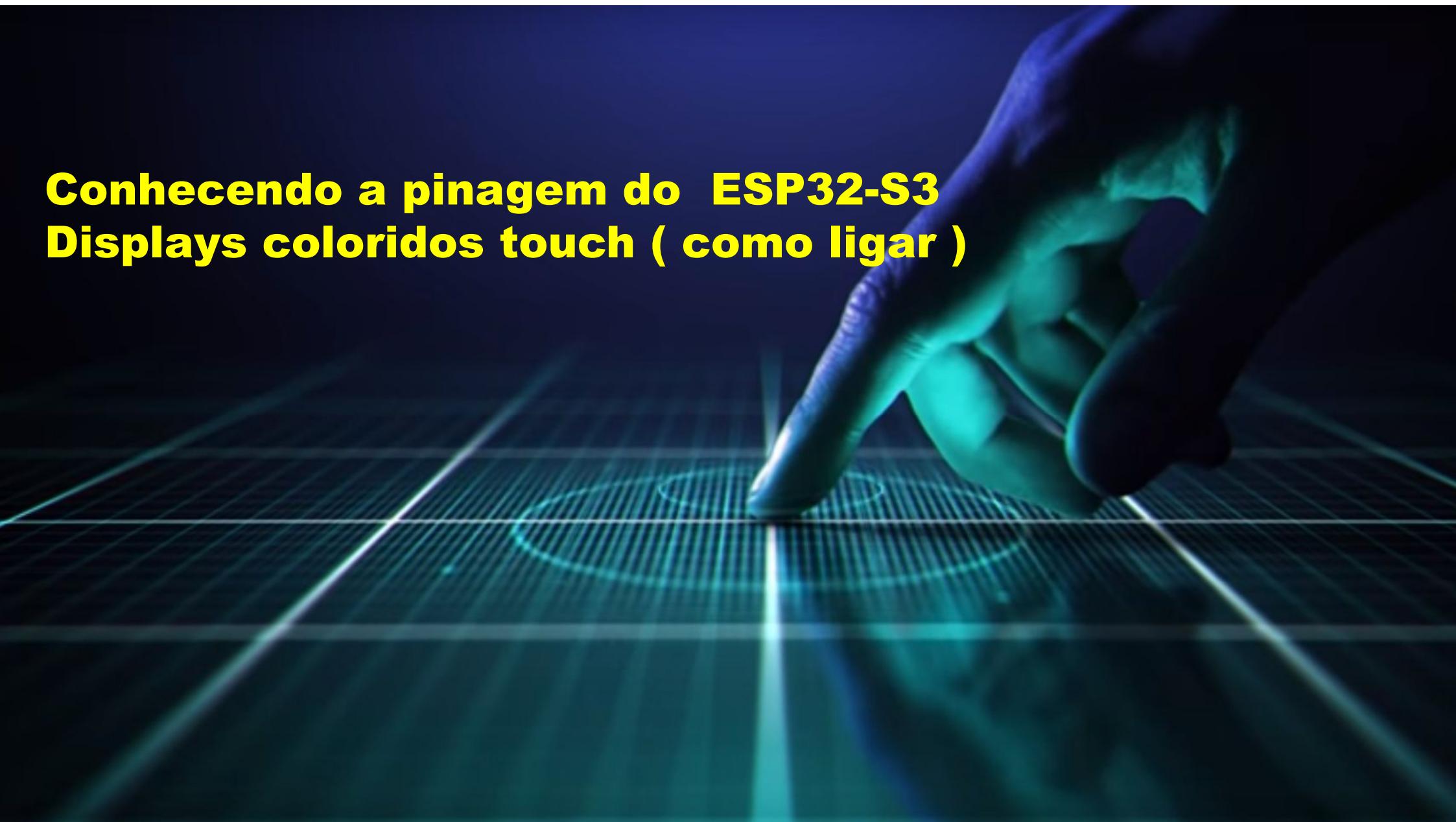
INICIANDO NOVO CICLO

- ▶ Passo 01/29 | GPIO42 = HIGH → LED ACESO •
Apagando GPIO42 = LOW → LED APAGADO ○
- ▶ Passo 02/29 | GPIO41 = HIGH → LED ACESO •
Apagando GPIO41 = LOW → LED APAGADO ○
- ▶ Passo 03/29 | GPIO40 = HIGH → LED ACESO •
Apagando GPIO40 = LOW → LED APAGADO ○
- ▶ Passo 04/29 | GPIO39 = HIGH → LED ACESO •
Apagando GPIO39 = LOW → LED APAGADO ○
- ▶ Passo 05/29 | GPIO38 = HIGH → LED ACESO •
Apagando GPIO38 = LOW → LED APAGADO ○
- ▶ Passo 06/29 | GPIO37 = HIGH → LED ACESO •
Apagando GPIO37 = LOW → LED APAGADO ○
- ▶ Passo 07/29 | GPIO36 = HIGH → LED ACESO •
Apagando GPIO36 = LOW → LED APAGADO ○
- ▶ Passo 08/29 | GPIO35 = HIGH → LED ACESO •
Apagando GPIO35 = LOW → LED APAGADO ○
- ▶ Passo 09/29 | GPIO45 = HIGH → LED ACESO •
Apagando GPIO45 = LOW → LED APAGADO ○
- ▶ Passo 10/29 | GPIO48 = HIGH → LED ACESO •
Apagando GPIO48 = LOW → LED APAGADO ○
- ▶ Passo 11/29 | GPIO47 = HIGH → LED ACESO •
Apagando GPIO47 = LOW → LED APAGADO ○
- ▶ Passo 12/29 | GPIO21 = HIGH → LED ACESO •
Apagando GPIO21 = LOW → LED APAGADO ○
- ▶ Passo 13/29 | GPIO20 = HIGH → LED ACESO •
Apagando GPIO20 = LOW → LED APAGADO ○

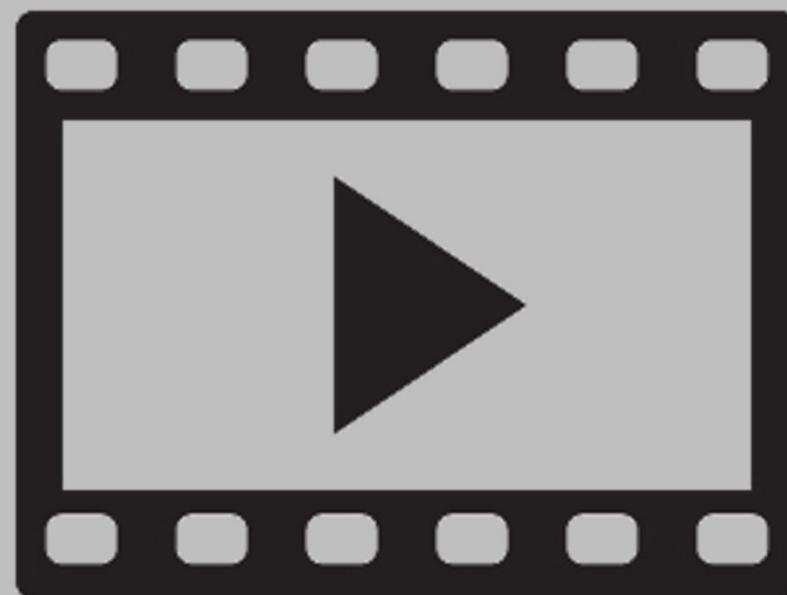
- ▶ Passo 14/29 | GPIO19 = HIGH → LED ACESO •
Apagando GPIO19 = LOW → LED APAGADO ○
- ▶ Passo 15/29 | GPIO18 = HIGH → LED ACESO •
Apagando GPIO18 = LOW → LED APAGADO ○
- ▶ Passo 16/29 | GPIO17 = HIGH → LED ACESO •
Apagando GPIO17 = LOW → LED APAGADO ○
- ▶ Passo 17/29 | GPIO16 = HIGH → LED ACESO •
Apagando GPIO16 = LOW → LED APAGADO ○
- ▶ Passo 18/29 | GPIO15 = HIGH → LED ACESO •
Apagando GPIO15 = LOW → LED APAGADO ○
- ▶ Passo 19/29 | GPIO07 = HIGH → LED ACESO •
Apagando GPIO07 = LOW → LED APAGADO ○
- ▶ Passo 20/29 | GPIO06 = HIGH → LED ACESO •
Apagando GPIO06 = LOW → LED APAGADO ○
- ▶ Passo 21/29 | GPIO05 = HIGH → LED ACESO •
Apagando GPIO05 = LOW → LED APAGADO ○
- ▶ Passo 22/29 | GPIO04 = HIGH → LED ACESO •
Apagando GPIO04 = LOW → LED APAGADO ○
- ▶ Passo 23/29 | GPIO09 = HIGH → LED ACESO •
Apagando GPIO09 = LOW → LED APAGADO ○
- ▶ Passo 24/29 | GPIO10 = HIGH → LED ACESO •
Apagando GPIO10 = LOW → LED APAGADO ○
- ▶ Passo 25/29 | GPIO11 = HIGH → LED ACESO •
Apagando GPIO11 = LOW → LED APAGADO ○
- ▶ Passo 26/29 | GPIO12 = HIGH → LED ACESO •
Apagando GPIO12 = LOW → LED APAGADO ○
- ▶ Passo 27/29 | GPIO13 = HIGH → LED ACESO •
Apagando GPIO13 = LOW → LED APAGADO ○
- ▶ Passo 28/29 | GPIO14 = HIGH → LED ACESO •
Apagando GPIO14 = LOW → LED APAGADO ○
- ▶ Passo 29/29 | GPIO08 = HIGH → LED ACESO •
Apagando GPIO08 = LOW → LED APAGADO ○

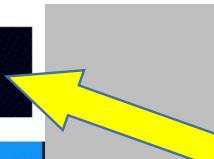
CICLO FINALIZADO
Aguardando para reiniciar...

Conhecendo a pinagem do ESP32-S3 Displays coloridos touch (como ligar)



Demonstração






Receba o meu conteúdo
GRATUITAMENTE

Insira aqui seu melhor email...

QUERO RECEBER GRÁTIS



Se eu soubesse disso antes!

by Fernando K - 12 novembro

Voltamos a falar de instrumentação e, hoje, vamos falar de um módulo de obtenção de parâmetros elétricos , que é o RIDEN 100V/10A ...

Leia mais

Supergrupo de colaboração entre meus seguidores.

Conteúdo exclusivo, que não tem no Instagram / Youtube!

INSTAGRAM @FERNANDOK_OFICIAL

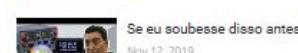
FACEBOOK



INSCREVA-SE NO YOUTUBE



RECENTE POPULAR COMENTÁRIOS



SEJA MEMBRO



Links onde comprei os componentes

<https://bit.ly/2VM1ZdQ>

Em www.fernandok.com



Instagram
fernandok_oficial



Telegram
fernandok_oficial



3 Esp32 mais comuns no Ali....

ESP32 30 PINOS



Choice Promo Placa de desenvolvimento E...
R\$23,25 R\$46,5 -50%

ESP32 38 PINOS



R\$26,99

6% de Cashback

Compra internacional, +R\$12,07 em impostos estimados ⓘ

ESP32-S3 44 PINOS



R\$28,51

R\$46,77 39% desc.

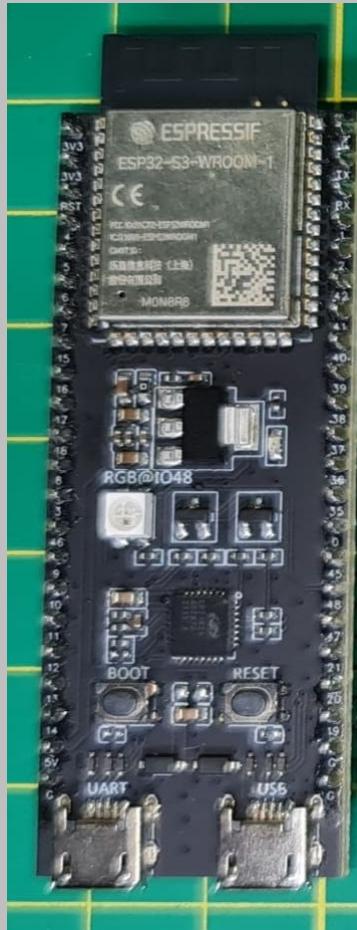
6% de Cashback

Compra internacional, +R\$12,79 em impostos estimados ⓘ

Preços estimados em 13/07/2025 sem impostos

ESP32-S3 DEVKITC-1

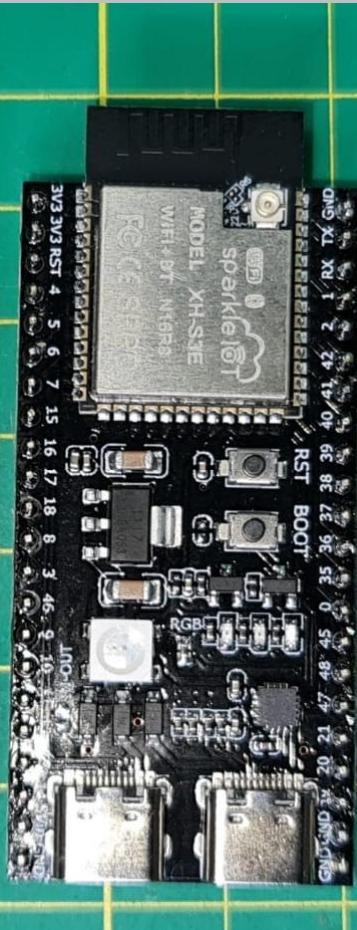
Chip Espressif
Módulo Espr.
Placa Espressif



Chip Espressif
Módulo Espr.
Placa genérica



Chip Espressif
Módulo gen.
Placa genérica



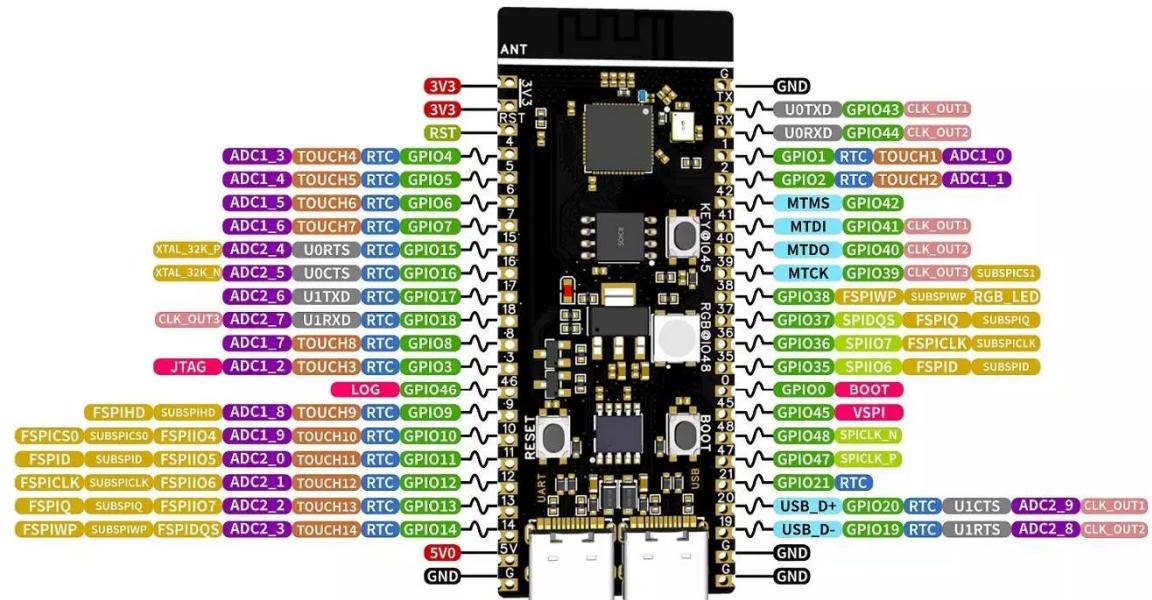
Chip Espressif
Não usa Módulo
Placa Weact Studio



<https://www.universal-solder.ca/product/weact-esp32-s3-wifi-bluetooth-mesh-development-board-16mb/>

ESP32-S3

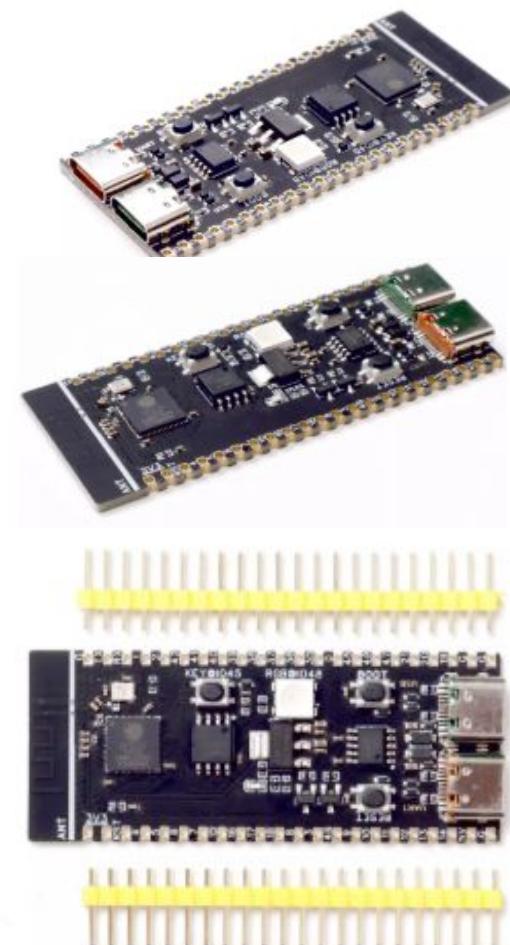
WeAct Studio



ESP32-S3 Specs

32-bit Xtensa® dual-core @240MHz
Wi-Fi IEEE 802.11 b/g/n 2.4GHz + BLE 5 Mesh
512 KB SRAM (16 KB SRAM in RTC)
384 KB ROM
45 GPIOs, 4x SPI, 3x UART, 2x I2C,
14x Touch, 2x I2S, RMT, LED PWM, USB-OTG,
TWAI®, 2x 12-bit ADC, 1x LCD interface, DVP

MISC Miscellaneous/SPI functions
CLK_OUTx Clock Output



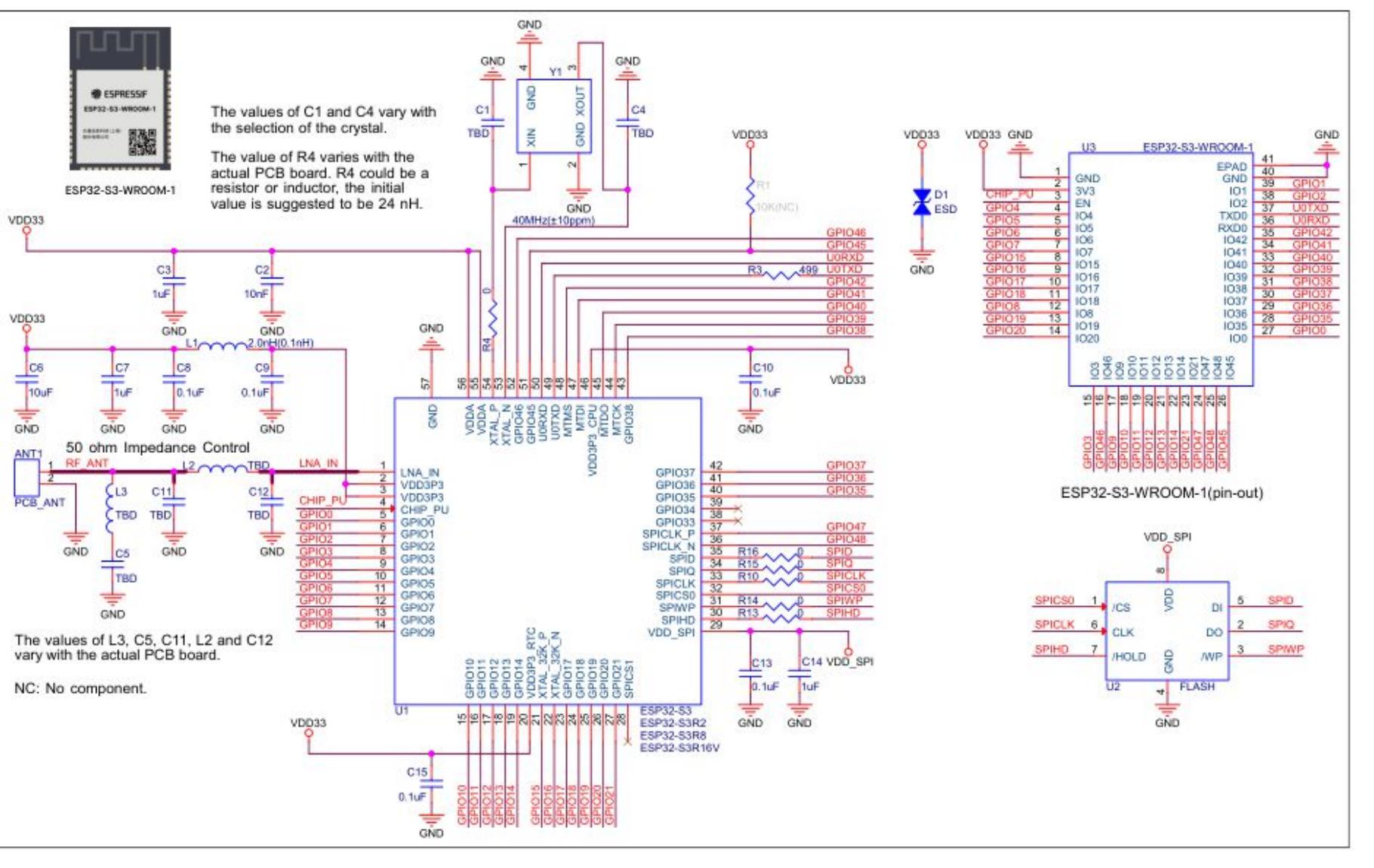


Figure 5: ESP32-S3-WROOM-1 Schematics

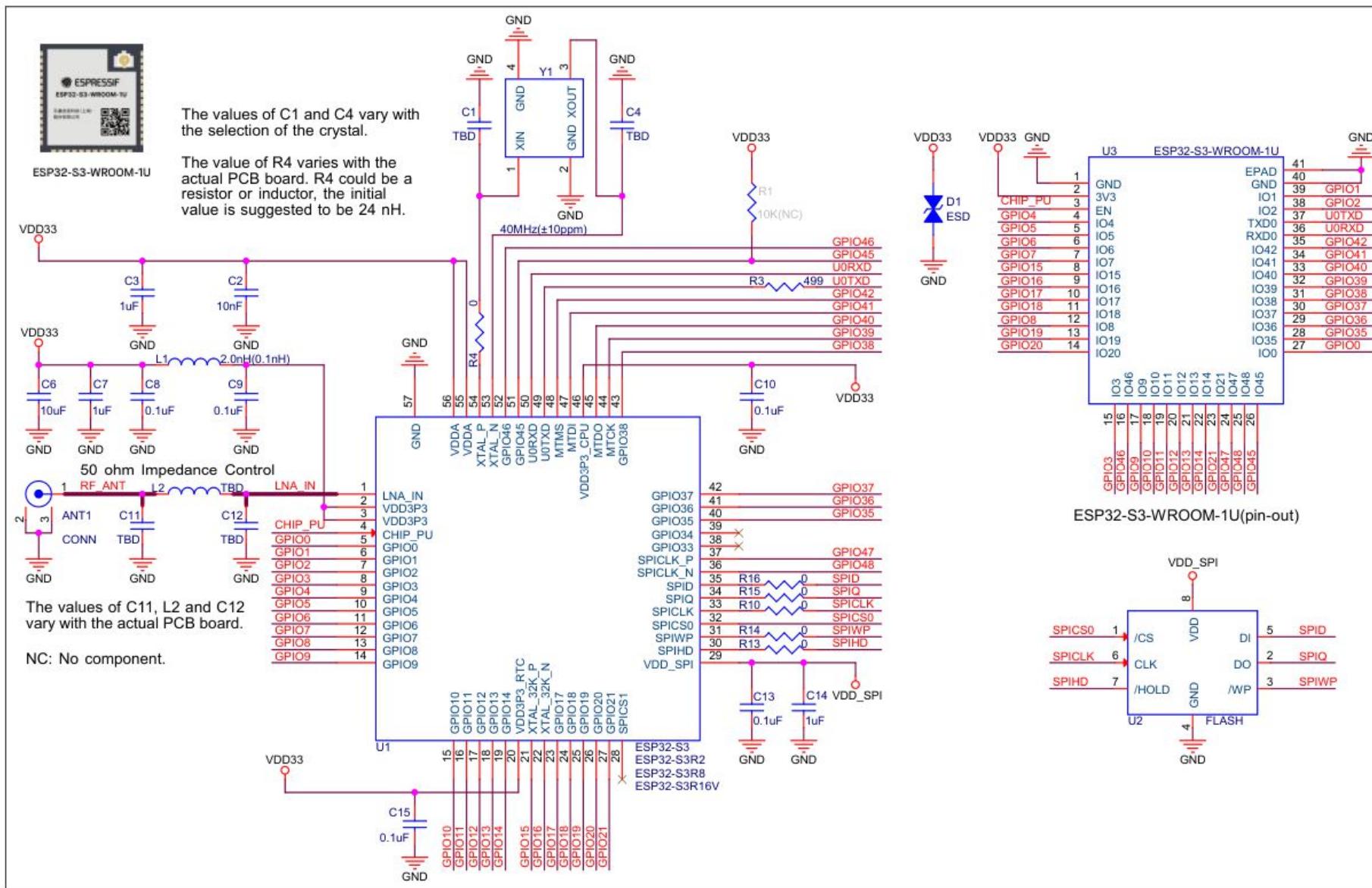


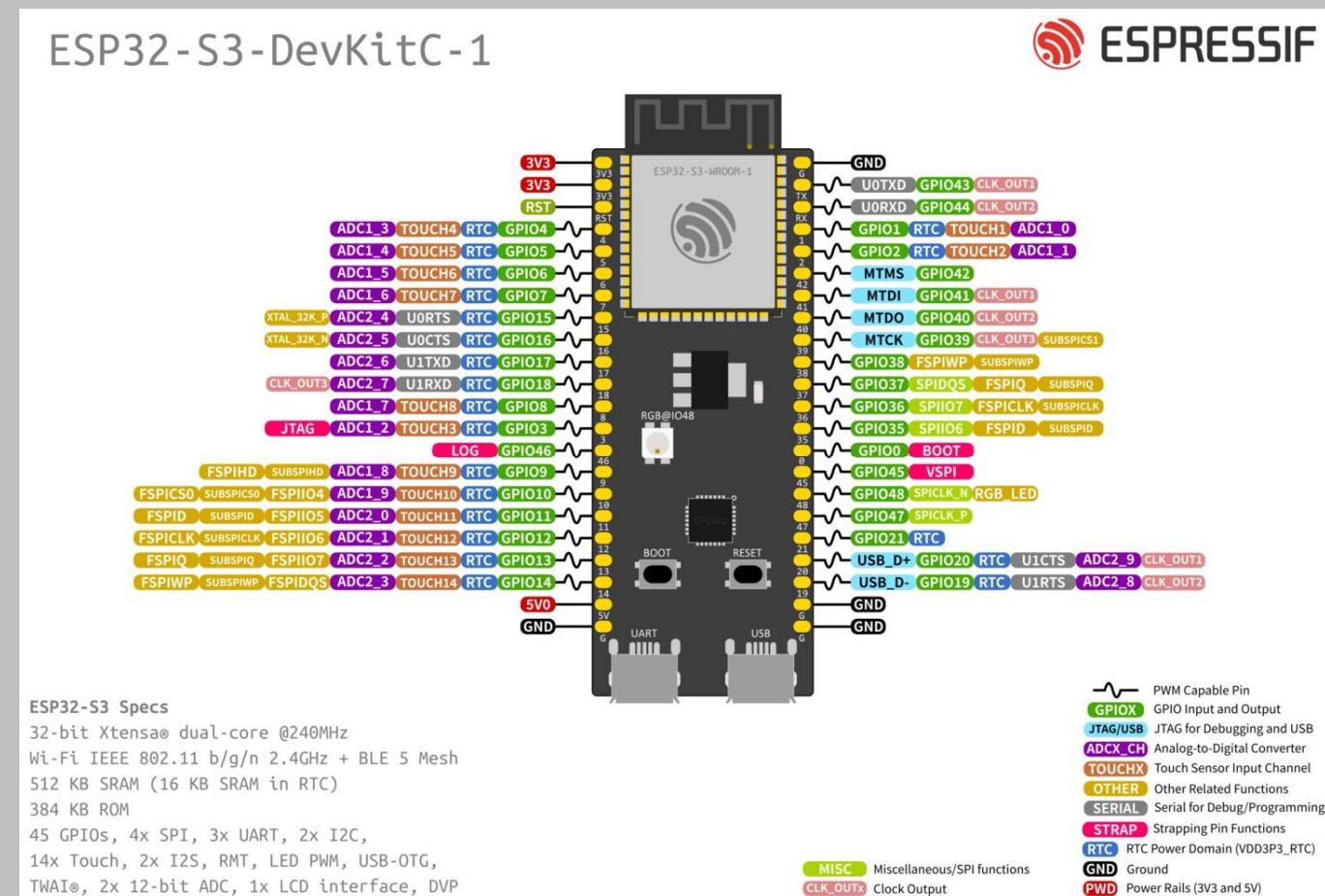
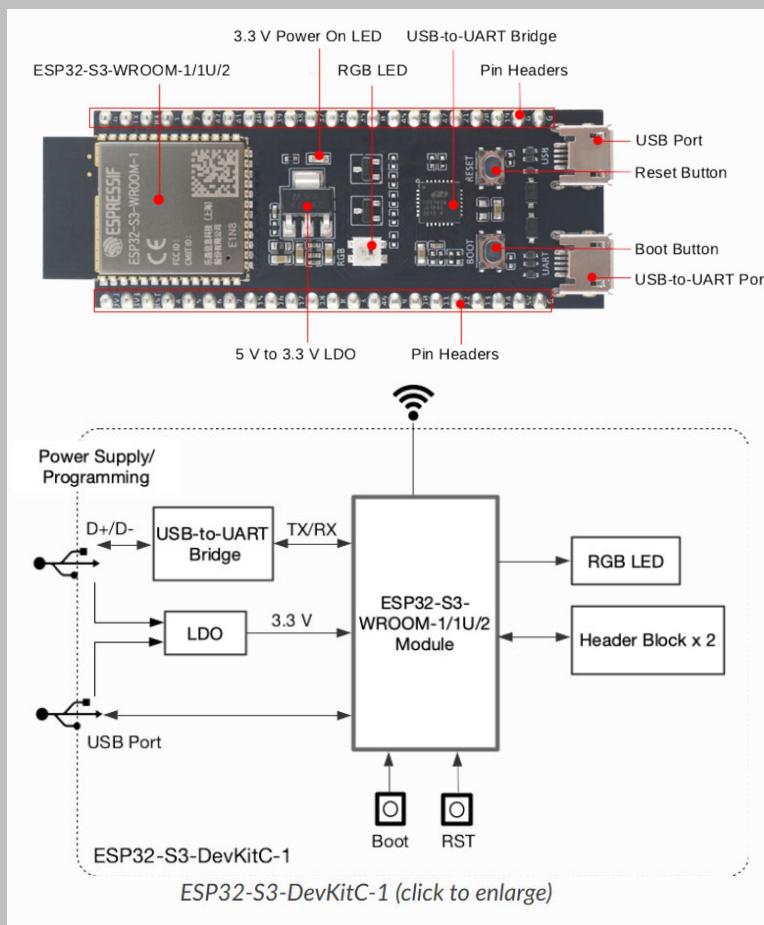
Figure 6: ESP32-S3-WROOM-1U Schematics

Em www.fernandok.com

Download arquivo PDF e INO

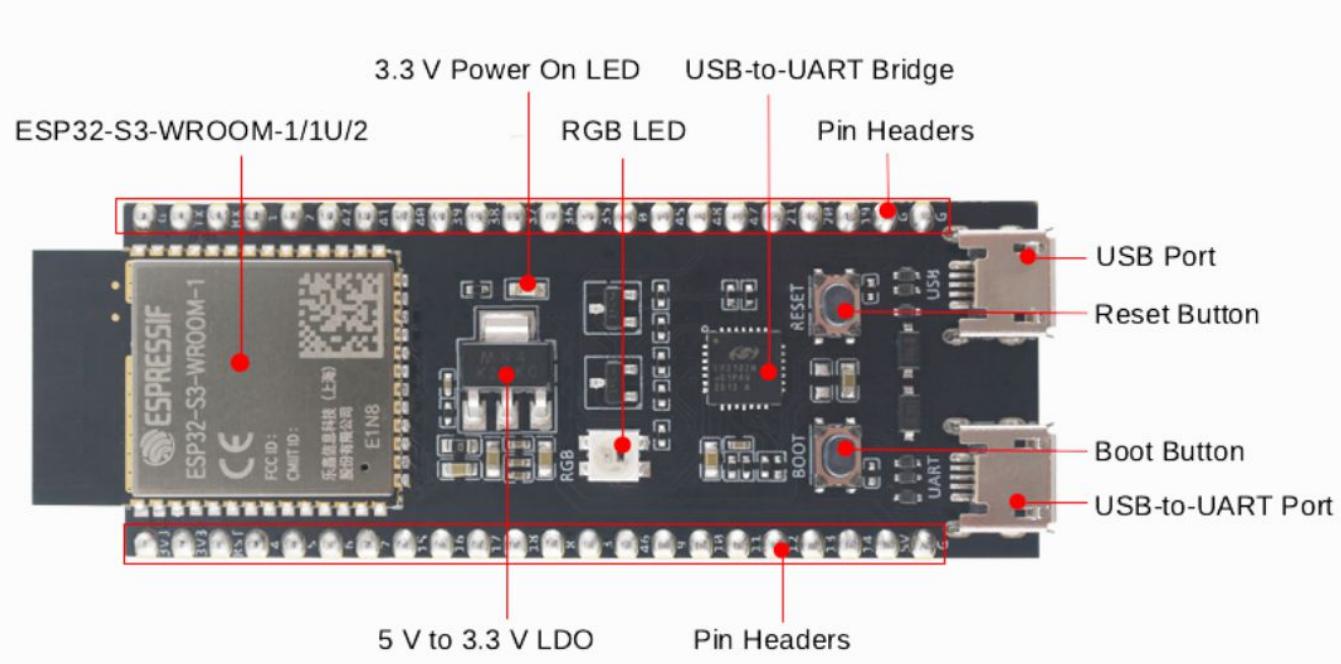


Placa da Espressif ESP32-S3 DEVKITC-1



Nº	Nome/ GPIO	Tipo	Funções (principais)	Observações
1	3V3	P	3.3 V power	Saída regulada
2	3V3	P	3.3 V power	Saída regulada
3	RST (EN)	I	Reset enable	Botão RESET ativo em LOW
4	GPIO4	I/O/T	RTC_GPIO4, TOUCH4, ADC1_CH3	GPIO seguro
5	GPIO5	I/O/T	RTC_GPIO5, TOUCH5, ADC1_CH4	GPIO seguro
6	GPIO6	I/O/T	RTC_GPIO6, TOUCH6, ADC1_CH5	GPIO seguro
7	GPIO7	I/O/T	RTC_GPIO7, TOUCH7, ADC1_CH6	GPIO seguro
8	GPIO15	I/O/T	RTC_GPIO15, U0RTS, ADC2_CH4, XTAL_32K_P	bootstrap
9	GPIO16	I/O/T	RTC_GPIO16, U0CTS, ADC2_CH5, XTAL_32K_N	bootstrap
10	GPIO17	I/O/T	RTC_GPIO17, U1TXD, ADC2_CH6	Serial1 TX
11	GPIO18	I/O/T	RTC_GPIO18, U1RXD, ADC2_CH7, CLK_OUT3	Serial1 RX
12	GPIO8	I/O/T	RTC_GPIO8, TOUCH8, ADC1_CH7, SUBSPICS1	GPIO seguro
13	GPIO3	I/O/T	RTC_GPIO3, TOUCH3, ADC1_CH2	GPIO seguro
14	GPIO46	I/O/T	GPIO46	GPIO seguro
15	GPIO9	I/O/T	RTC_GPIO9, TOUCH9, ADC1_CH8, FSPIHD	GPIO seguro
16	GPIO10	I/O/T	RTC_GPIO10, TOUCH10, ADC1_CH9, FSPICS0	GPIO seguro
17	GPIO11	I/O/T	RTC_GPIO11, TOUCH11, ADC2_CH0, FSPIID	GPIO seguro
18	GPIO12	I/O/T	RTC_GPIO12, TOUCH12, ADC2_CH1, FSPICLK	GPIO seguro
19	GPIO13	I/O/T	RTC_GPIO13, TOUCH13, ADC2_CH2, FSPIQ	GPIO seguro
20	GPIO14	I/O/T	RTC_GPIO14, TOUCH14, ADC2_CH3, FSPIWP	GPIO seguro
21	5V0	P	5 V input (from USB)	Entrada 5 V da USB
22	GND	G	Ground	Terra

Nº	Nome/ GPIO	Tipo	Funções (principais)	Observações
23	GND	G	Ground	Terra
24	TX / GPIO43	I/O/T	U0TXD, CLK_OUT1	Serial0 TX → CP2102
25	RX / GPIO44	I/O/T	U0RXD, CLK_OUT2	Serial0 RX → CP2102
26	GPIO1	I/O/T	RTC_GPIO1, TOUCH1, ADC1_CH0	bootstrap
27	GPIO2	I/O/T	RTC_GPIO2, TOUCH2, ADC1_CH1	bootstrap
28	GPIO42	I/O/T	MTMS, GPIO42	JTAG TMS / GPIO geral
29	GPIO41	I/O/T	MTDI, GPIO41, CLK_OUT1	JTAG TDI / GPIO
30	GPIO40	I/O/T	MTDO, GPIO40, CLK_OUT2	JTAG TDO / GPIO
31	GPIO39	I/O/T	MTCK, GPIO39, CLK_OUT3, SUBSPICS1	JTAG TCK / GPIO
32	GPIO38	I/O/T	FSPIWP, SUBSPIWP, GPIO38, RGB LED (opcional)	GPIO seguro
33	GPIO37	I/O/T	SPIDQS, FSPIQ, GPIO37	GPIO seguro
34	GPIO36	I/O/T	SPIIO7, FSPICLK, GPIO36	Livre no WROOM-1, reservado no WROOM-2
35	GPIO35	I/O/T	SPIIO6, FSPIID, GPIO35	Livre no WROOM-1, reservado no WROOM-2
36	GPIO0	I/O/T	RTC_GPIO0, GPIO0	BOOT button strap
37	GPIO45	I/O/T	GPIO45	GPIO seguro
38	GPIO48	I/O/T	SPICLK_N, RGB LED	LED endereçável onboard
39	GPIO47	I/O/T	SPICLK_P, GPIO47	GPIO seguro
40	GPIO21	I/O/T	RTC_GPIO21, GPIO21	GPIO seguro
41	GPIO20	I/O/T	RTC_GPIO20, U1CTS, ADC2_CH9, USB_D+	⚠️ USB nativo
42	GPIO19	I/O/T	RTC_GPIO19, U1RTS, ADC2_CH8, USB_D-	⚠️ USB nativo
43	GND	G	Ground	Terra
44	GND	G	Ground	Terra



Para placas com módulos ESP32-S3-WROOM-1/1U de memória flash/PSRAM Octal SPI incorporados e placas com módulos ESP32-S3-WROOM-2, os pinos **GPIO35**, **GPIO36** e **GPIO37** são usados para comunicação interna entre o ESP32-S3 e a memória flash/PSRAM SPI, não estando, portanto, disponíveis para uso externo.

“Tanto a versão inicial quanto a versão v1.1 do ESP32-S3-DevKitC-1 estão disponíveis no mercado. A principal diferença está na atribuição do GPIO para o LED RGB: a versão inicial usa o **GPIO48**, enquanto a v1.1 usa o **GPIO38**.”

📌 ESP32-S3-DevKitC-1 — Pinos comprometidos

A placa **ESP32-S3-DevKitC-1** vem com alguns pinos já usados para funções específicas **onboard**, que **não podem ser usados livremente** sem alterar o hardware ou o firmware base.

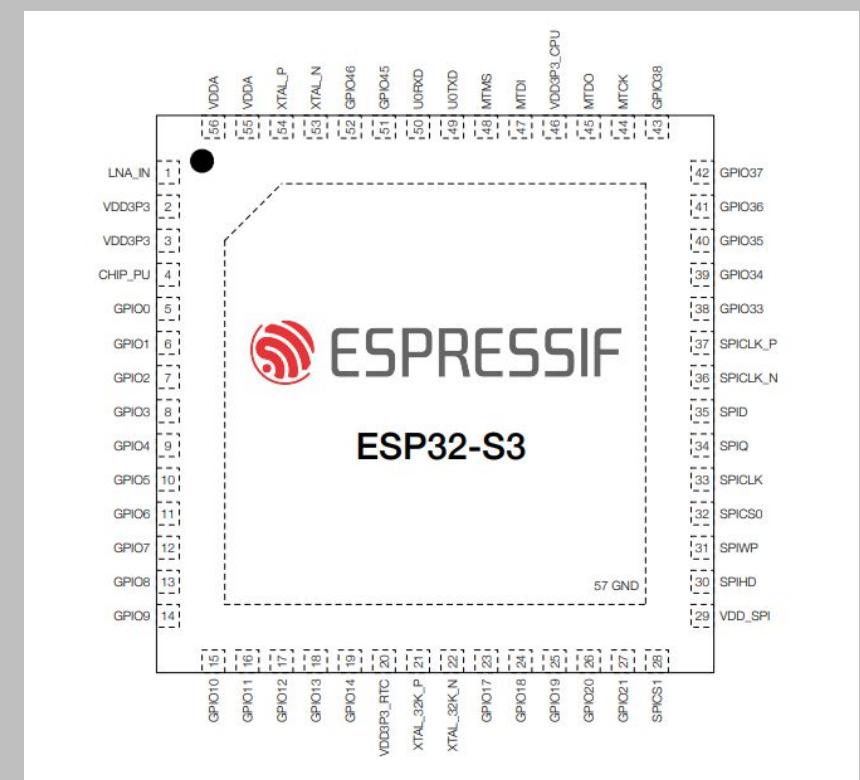
✓ 1) Pinos comprometidos normalmente

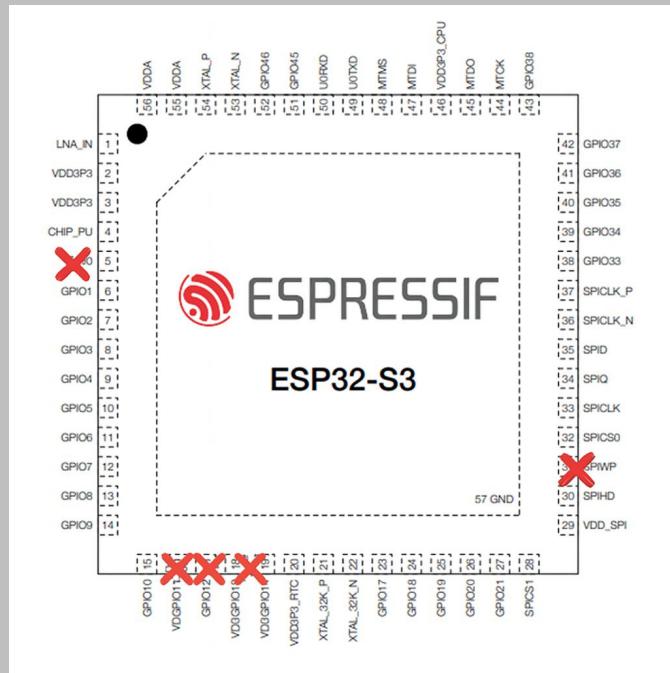
UART para USB

- **GPIO19 (U0RXD)** — Recebe dados do conversor USB/UART onboard (chip CP2102 ou equivalente).
- **GPIO20 (U0TXD)** — Transmite dados do conversor USB/UART onboard.
- **Impacto:** Usados para upload de firmware e logs via porta serial.

Resumo prático

Pino	Função onboard	Aviso
GPIO0	BOOT (flash mode)	Uso com cuidado
EN	Reset	Não é GPIO
GPIO19	UART RX	Evitar usar como GPIO
GPIO20	UART TX	Evitar usar como GPIO
GPIO38/48	LED RGB onboard	Evitar usar sem perder LED
GPIO46	Só entrada	Restrito (entrada apenas)



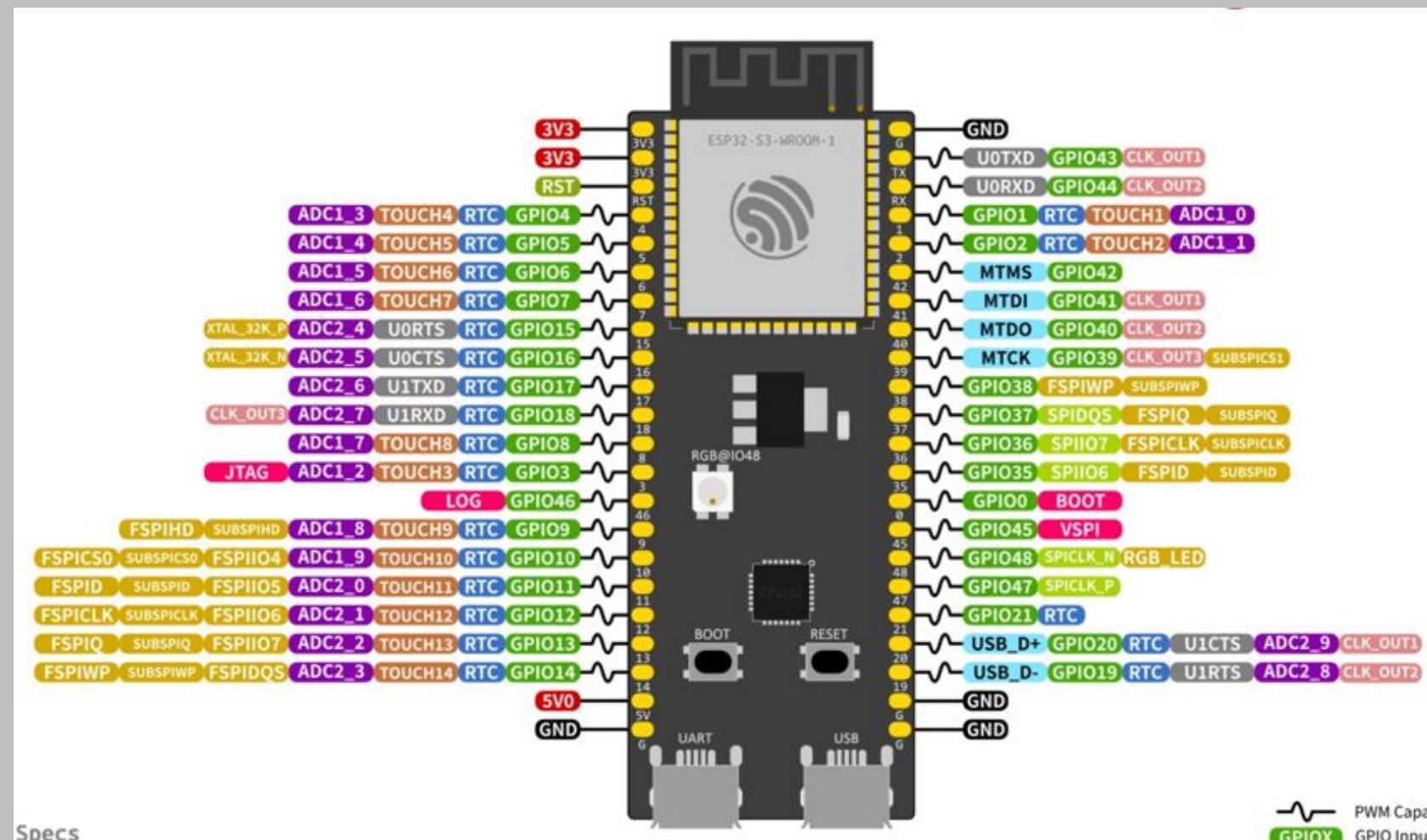


Resumo prático

Pino	Função principal	Motivo para não usar como GPIO normal
CHIP_PU	Reset / Enable	Controla energia, não é GPIO
GPIO0	BOOT mode	Interfere no boot
U0TXD/U0RXD	UART gravação/debug	Conflita com upload de firmware
VDD_*	Alimentação / Domínio RTC	Não são GPIOs
SPIWP/HD	Flash SPI	Interfere no boot do firmware

ADC2 não deve ser usado simultaneamente (Wi-Fi/BT)

No ESP32 (família original) existe uma restrição bem conhecida: o ADC2 é compartilhado com o rádio (Wi-Fi/BT) e, enquanto o driver de Wi-Fi estiver ativo, leituras confiáveis do ADC2 não são possíveis. Para o ESP32-S3 a restrição prática é a mesma: se for necessário manter Wi-Fi ativo e fazer leituras analógicas contínuas/confiáveis, use os canais do ADC1. ADC2 só deve ser usado quando você tiver controle sobre o estado do rádio (por exemplo Wi-Fi desligado/hibernado) ou quando aceitar leituras não garantidas.



UART TX/RX Serial

O chip **ESP32-S3** possui **3 controladores UART** independentes:

- **UART0**: Geralmente usada para comunicação com o computador (USB-Serial ou chip USB-UART na placa). É a que você usa para `Serial.print()`.
- **UART1**: Disponível para uso geral.
- **UART2**: Disponível para uso geral.

UARTs e Pinos Seguros no ESP32-S3 DevKitC-1

A grande vantagem do ESP32-S3 é que você pode **mapear os pinos TX e RX de qualquer UART para quase qualquer GPIO** disponível usando a matriz de IO MUX. Isso significa que não há pinos "fixos" para UART1 e UART2, você escolhe os que são seguros e convenientes.

- **Uso**: Esta é a UART que o Arduino IDE usa para upload de código e para a comunicação `Serial.print()`.
- **Segurança**: É seguro usar para debug e comunicação com o PC. Se você precisar de uma UART extra, **evite usar GPIO43 e GPIO44 para outras funções** se estiver usando `Serial.print()`.

2. UART1 (Uso geral)

- **Pinos seguros recomendados**: Você pode escolher **qualquer par de GPIOs seguros** que não estejam sendo usados por outras funções.

1. UART0 (Serial padrão)

- **Pinos padrão no DevKitC-1**:

- **TX0**: GPIO43
- **RX0**: GPIO44

- **Uso**: Esta é a UART que o Arduino IDE usa para upload de código e para a comunicação `Serial.print()`;

- **Segurança**: É seguro usar para debug e comunicação com o PC. Se você precisar de uma UART extra, **evite usar GPIO43 e GPIO44 para outras funções** se estiver usando `Serial.print()`;

- **Exemplos de pinos seguros para TX1/RX1**:

- **GPIO1, GPIO2, GPIO3, GPIO4, GPIO5, GPIO6, GPIO7, GPIO8, GPIO9, GPIO10, GPIO11, GPIO12, GPIO13, GPIO14, GPIO15, GPIO16, GPIO17, GPIO18**
- **GPIO38, GPIO39, GPIO40, GPIO41, GPIO42, GPIO45**

- **Como configurar**: Você usaria `Serial1.begin(baudRate, config, rxPin, txPin);`

3. UART2 (Uso geral)

- **Pinos seguros recomendados**: Assim como a UART1, você pode escolher **qualquer par de GPIOs seguros** que não estejam em uso.

- **Exemplos de pinos seguros para TX2/RX2**: Os mesmos listados para UART1.

- **Como configurar**: Você usaria `Serial2.begin(baudRate, config, rxPin, txPin);`

UART TX/RX Serial

Pinos a evitar para UARTs (e qualquer outra função)

Lembre-se dos pinos que **NÃO DEVEM SER USADOS** para evitar problemas de boot ou conflitos com a memória flash:

- **GPIO0, GPIO46**: Pinos de bootstrapping (podem causar problemas no boot).
- **GPIO19, GPIO20, GPIO21**: Usados para USB/JTAG/Serial (se você estiver usando o USB nativo do S3).
- **GPIO26-32**: Conectados à memória flash/PSRAM (NÃO USAR).
- **GPIO33-37**: Reservados para cristal e funções internas.

Resumo das UARTs no ESP32-S3 DevKitC-1

- **UART0**: 1 (usada para `Serial.print()`, pinos padrão `GPIO43/44`).
- **UART1**: 1 (totalmente configurável em pinos seguros).
- **UART2**: 1 (totalmente configurável em pinos seguros).

Total de UARTs disponíveis para seu projeto: 2 UARTs adicionais (UART1 e UART2), além da UART0 para debug.

I2C exemplo

Se você quiser ser **conservador** e NÃO causar conflito nenhum no DevKitC-1, use:

- GPIO8 até GPIO15
- GPIO33 até GPIO42

Esses são livres, versáteis e seguros para I2C.

```
#include <Wire.h>

#define I2C_SDA_PIN 8 // Exemplo: usando GPIO8 como SDA
#define I2C_SCL_PIN 9 // Exemplo: usando GPIO9 como SCL

void setup() {
    Wire.begin(I2C_SDA_PIN, I2C_SCL_PIN);
    Serial.begin(115200);
    Serial.println("I2C configurado!");
}

void loop() {
    // Seu código I2C aqui
}
```

SPI exemplo

Pinos seguros para SPI no DevKitC-1 (modo conservador):

Você pode montar um barramento SPI (MOSI, MISO, SCLK, CS) sem conflito usando:

GPIO8 ~ GPIO15

GPIO33 ~ GPIO42

SPI que NÃO podem ser usadas

SPI0 e SPI1 → NUNCA estão disponíveis para o usuário, porque o chip depende delas para dar boot e rodar os programas.

Esses blocos SPI já estão ligados fisicamente à memória **Flash** e **PSRAM** do módulo.

Se você tentar usar os pinos dessas interfaces (GPIO26–GPIO32), vai entrar em conflito total (o chip pode travar, não dar boot ou não rodar seu código).

Sugestão de mapeamento seguro de SPI:

- MOSI → GPIO11
- MISO → GPIO13
- SCLK → GPIO12
- CS → GPIO10 Chip Select (ou outro disponível, conforme o dispositivo)

SPI exemplo

Arduino ESP32 Core 3.x

A partir da versão 3.x do Arduino core para ESP32, houve uma padronização de nomenclatura:

- **SPIH** → é a instância da **SPI2 (HSPI no naming antigo)**
- **SPIv** → é a instância da **SPI3 (VSPI no naming antigo)**

No Arduino Core (ESP32 core ≤ 2.x), os nomes usados eram:

HSPI → referia-se ao **SPI2**

VSPI → referia-se ao **SPI3**

No framework oficial (ESP-IDF), elas são chamadas de:

SPI2_HOST

SPI3_HOST

- **Não existe diferença de hardware** — ambos são controladores SPI de uso geral, independentes e flexíveis.
- A diferença é **apenas de nomenclatura** (para ficar mais consistente com o ESP-IDF e evitar ambiguidades).

SPI exemplo

```
#include <SPI.h>          // Inclui a biblioteca de comunicação SPI

#define PIN_MOSI 11        // Define GPIO11 como pino MOSI (Master Out Slave In)
#define PIN_MISO 13        // Define GPIO13 como pino MISO (Master In Slave Out)
#define PIN_SCLK 12        // Define GPIO12 como pino SCLK (clock do barramento SPI)
#define PIN_CS  10          // Define GPIO10 como pino CS (Chip Select)

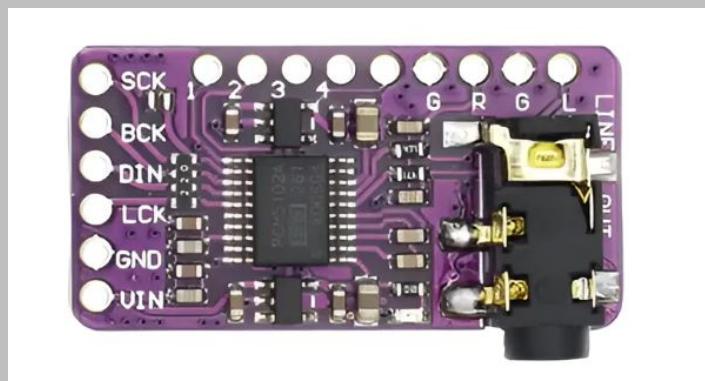
SPIClass SPIv(HSPI);      // Cria objeto SPI usando o barramento HSPI (SPI2)

void setup() {
  SPIv.begin(PIN_SCLK, PIN_MISO, PIN_MOSI, PIN_CS); // Inicializa SPI com pinos escolhidos
  pinMode(PIN_CS, OUTPUT);    // Configura o pino CS como saída digital
  digitalWrite(PIN_CS, HIGH); // Coloca CS em nível alto (dispositivo desativado)
}

void loop() {
  // Comunicação SPI aqui           // Local para código de envio/recebimento via SPI
}
```

I2S Interface digital de áudio

**Módulo Dac Pcm5102 I2s Estéreo
Conversor Áudio Digital com esp32-s3**



Características do Programa:

- Tom de teste:** Gera um Lá musical (440Hz) em estéreo
- Configuração segura:** Usa apenas pinos GPIO seguros (4, 5, 6)
- Comentários detalhados:** Cada linha explicada
- Tratamento de erros:** Verifica todas as operações I2S
- Funções extras:** Teste de múltiplas frequências

Pinos Recomendados (Seguros):

- GPIO4:** I2S_BCLK (Bit Clock)
- GPIO5:** I2S_DOUT (Data Output)
- GPIO6:** I2S_LRC (Left/Right Clock)

Configurações do PCM5102A:

- SCK → GND:** Usa BCK como clock do sistema
- FMT → GND:** Formato I2S
- FLT → 3V3:** Filtro normal
- XSMY → 3V3:** Sem mute

Para Testar:

- 1.Faça as conexões conforme a tabela
- 2.Carregue o código no ESP32-S3
- 3.Conecte fones ou alto-falantes na saída do PCM5102A
- 4.Você deve ouvir um tom contínuo de 440Hz

I2S Interface digital de áudio

Tabela de Ligações ESP32-S3 DevKitC-1 + PCM5102A

PCM5102A	ESP32-S3 DevKitC-1	Função
VCC	3V3	Alimentação 3.3V
GND	GND	Terra
BCK	GPIO4	Bit Clock (I2S_BCLK)
DIN	GPIO5	Data Input (I2S_DOUT)
LCK	GPIO6	Left/Right Clock (I2S_LRC)
SCK	GND	System Clock (conectar ao GND para usar BCK)
FLT	3V3	Filter Select (3V3 = filtro normal)
FMT	GND	Format Select (GND = I2S format)
XSMT	3V3	Soft Mute (3V3 = unmuted)

I2S Interface digital de áudio com core Arduino esp32 3.x

```
/* Teste do módulo PCM5102A com ESP32-S3 DevKitC-1
 * Gera um tom senoidal de 440Hz (Lá musical) para testar o áudio
 *
 * Conexões:
 * PCM5102A -> ESP32-S3
 * VCC -> 3V3
 * GND -> GND
 * BCK -> GPIO4
 * DIN -> GPIO5
 * LCK -> GPIO6
 * SCK -> GND
 * FLT -> 3V3
 * FMT -> GND
 * XSMT -> 3V3
 */
#include "driver/i2s.h"
#include <math.h>
// Definição dos pinos I2S (seguros para ESP32-S3)
#define I2S_BCLK_PIN    4    // Bit Clock
#define I2S_DOUT_PIN    5    // Data Output
#define I2S_LRC_PIN     6    // Left/Right Clock (Word Select)
```



Clique aqui e baixe o código fonte

Grupos de pinos com restrições importantes (explicação e como evitar)

CUIDADO !



• Boot/strapping pins

- O SoC lê alguns pinos no reset para decidir modo de boot. Se um circuito externo alterar o nível desses pinos no reset, a placa pode não inicializar.
- Evite conectar periféricos que prejudiquem o nível nesses pinos durante reset; se precisar usá-los, garanta buffering ou resistores que preservem o estado de boot.

• USB D+/D- e PHY

- Se você usar USB (CDC/console ou USB device/host), os pinos do PHY ficam ocupados. Não os reutilize para GPIOs quando o USB é necessário.

• JTAG / debug pins

- Enquanto o JTAG estiver habilitado/ em uso, os pinos de debug não podem ser usados para outras funções. Pode-se desabilitar JTAG para liberá-los, mas isso afeta debug.

• ADC / touch / RTC overlaps

- Alguns pinos servem para touch e ADC ou estão no domínio RTC (wake). Evite usar a mesma linha para touch e ADC simultaneamente; para wake do deep-sleep use apenas RTC_GPIOs.

• ADC2 vs rádio (observação prática)

- Na família ESP32 tradicional ADC2 tem restrição com Wi-Fi. No S3, recomendaciones prácticas são usar canais de ADC que não conflitam com rádio e, se necessário, preferir ADC1 ou ADC externo para leituras confiáveis com Wi-Fi ativo.

• Periféricos com canais limitados (LEDC, RMT, DMA, timers)

- Planeje se o projeto usa muitos PWMs/RMT/DMAs. Se necessário, utilize expanders (PCA9685 para PWM, MCP23017 para GPIOs) para aliviar o SoC.

SEGURO



Sugestão de mapeamento prático (exemplo de pinout de trabalho que evita colisões)

Estas atribuições são sugestões práticas para um projeto típico (Wi-Fi ativo, leituras analógicas confiáveis, vários sensores digitais):

- Console / Programação: use Serial (USB-Serial via UART0).
- I2C: escolha dois GPIOs livres que não sejam boot pins (ex.: pinos rotulados SDA/SCL no DevKit); use para ADS1115 (4× ADC 16-bit) e expanders MCP23017/PCA9555.
- SPI periféricos: use HSPI/VSPI (pinos mapeáveis, evitando GPIO6-11).
- ADC sensível: use somente os pinos ADC1 (para medições contínuas com Wi-Fi).
- Wake button: ligue ao RTC_GPIO indicado no datasheet (garanta pull resistors).
- Muitos PWMs/servos: use PCA9685 (I2C).

SEGURO



regras e escolhas seguras:

ADC (leituras confiáveis com Wi-Fi ON): use canais do ADC1 (evitar ADC2 quando Wi-Fi ativo).

Se precisar de mais canais/precisão, adicione ADS1115 (I2C, 4 ch, 16-bit) ou MCP3008/MCP3208 (SPI, 8 ch).

I2C (SDA/SCL): I2C é flexível (GPIO matrix) — escolha dois GPIOs que não sejam pinos de boot/flash. Caso a placa já indique SDA/SCL no serigrafado, use-os. Adicione pull-ups 4.7k–10k se necessário.

SPI periféricos: use HSPI/VSPI (evitar pinos 6–11). Mapeie MOSI/MISO/SCLK/CS para GPIOs livres que não conflitem com strapping.

UARTs: UART0 já é ligado ao USB-Serial (console). Para Serial1/Serial2, atribua GPIOs livres (evite pinos reservados/boot).

Wake from deep sleep: use RTC_GPIOs (somente eles despertam o chip em deep sleep). Verifique no datasheet quais são os RTC GPIOs e conecte botões/sinais de wake neles.

Muitos PWMs / servos: use PCA9685 (I2C) para não esgotar LEDC.

```
#include <Adafruit_NeoPixel.h>

// Define o pino do LED RGB (NeoPixel)
#define LED_PIN 48 // GPIO48 na versão 1.0 do DevKitC-1
#define NUMPIXELS 1 // Apenas 1 LED onboard

// Cria o objeto NeoPixel
Adafruit_NeoPixel pixels(NUMPIXELS, LED_PIN, NEO_GRB + NEO_KHZ800);

// Tempo de troca
unsigned long delay_ms = 500;

void setup() {
    pixels.begin(); // Inicializa o NeoPixel
}

void loop() {
    // Vermelho
    pixels.setPixelColor(0, pixels.Color(255, 0, 0));
    pixels.show();
    delay(delay_ms);

    // Verde
    pixels.setPixelColor(0, pixels.Color(0, 255, 0));
    pixels.show();
    delay(delay_ms);

    // Azul
    pixels.setPixelColor(0, pixels.Color(0, 0, 255));
    pixels.show();
    delay(delay_ms);
}
```

```
// FERNANDO KOYANAGI contato@fernandok.com
// Programa para ESP32-S3 DevKitC-1
// Acende GPIOs seguros um de cada vez por 100ms em sequência
// Feito para missão crítica - base em GPIOs confirmados pelas tabelas fornecidas
// 29 GPIOs diferentes estão sendo testados.
// Lista com os GPIOs seguros selecionados para teste sequencial
// -----
// *****SEGURIOS para uso geral (sem risco no boot)
// Esses podem ser usados livremente como entrada/saída digital, PWM, periféricos etc:
// GPIO3, 4, 5, 6, 7
// GPIO8, 9, 10, 11, 12, 13, 14
// GPIO17, 18, 21
// GPIO37, 38, 39, 40, 41, 42
// GPIO45, 47, 48
// -----
// Pinos com RESTRIÇÕES/ risco de uso
// GPIO2 → Bootstrap crítico
// Para boot normal pela Flash SPI, o GPIO2 deve estar em LOW (0) na inicialização.
// Se for usado com carga externa que mantenha HIGH, o chip pode falhar no boot.
// Pode ser usado, mas com muito cuidado (ex.: evitar LEDs com resistor para VCC, evitar circuitos que sustentem alto na energização).
// GPIO15 (U0RTS, XTAL_32K_P) → Bootstrap
// Está ligado à seleção de boot em alguns modos de inicialização.
// Recomendado não forçar nível elétrico durante reset.
// Também opcionalmente pode estar associado ao cristal de 32 kHz.
// GPIO16 (U0CTS, XTAL_32K_N) → Bootstrap
// Assim como o 15, usado em bootstrapping e opcionalmente para cristal de 32 kHz.
// Melhor evitar cargas pesadas que puxem o nível na energização.
// GPIO35, 36 → Atenção nos módulos
// No ESP32-S3-WROOM-1 (usado no DevKitC-1 básico), são livres.
// Mas no ESP32-S3-WROOM-2 ou WROVER, podem estar conectados internamente à Flash/PSRAM OCTAL.
// Se estiver usando o DevKitC-1 comum (WROOM-1) → são seguros, mas em designs futuros tenha cautela.
// -----
// RESTRIÇÕES FORTES (não usar para GPIO comum em muitos casos)
// GPIO19 / GPIO20 (USB D- e USB D+) → você não listou, mas é importante lembrar.
// Esses são usados pela interface USB nativa. Se tentar usar como GPIO em conjunto com USB, pode travar a comunicação.
```

```
const int gpio_list[] = {
    4, // GPIO4
    5, // GPIO5
    6, // GPIO6
    7, // GPIO7
    15, // GPIO15 (U0RTS, XTAL_32K_P) → Bootstrap
    16, // GPIO16 (U0CTS, XTAL_32K_N) → Bootstrap
    17, // GPIO17
    18, // GPIO18
    8, // GPIO8
    3, // GPIO3
    9, // GPIO9
    10, // GPIO10
    11, // GPIO11
    12, // GPIO12
    13, // GPIO13
    14, // GPIO14
    21, // GPIO21
    2, // GPIO2 // cuidado GPIO2 bootstrap deve estar em nível baixo (0) para permitir boot normal a partir da Flash SPI.
    42, // GPIO42
    41, // GPIO41
    40, // GPIO40
    39, // GPIO39
    38, // GPIO38
    37, // GPIO37
    36, // GPIO36 no ESP32-S3-WROOM-2 ou WROVER, podem estar conectados internamente à Flash/PSRAM OCTAL.
    35, // GPIO35 no ESP32-S3-WROOM-2 ou WROVER, podem estar conectados internamente à Flash/PSRAM OCTAL.
    45, // GPIO45
    48, // GPIO48 LED
    47, // GPIO47
    21 // GPIO21
};
```

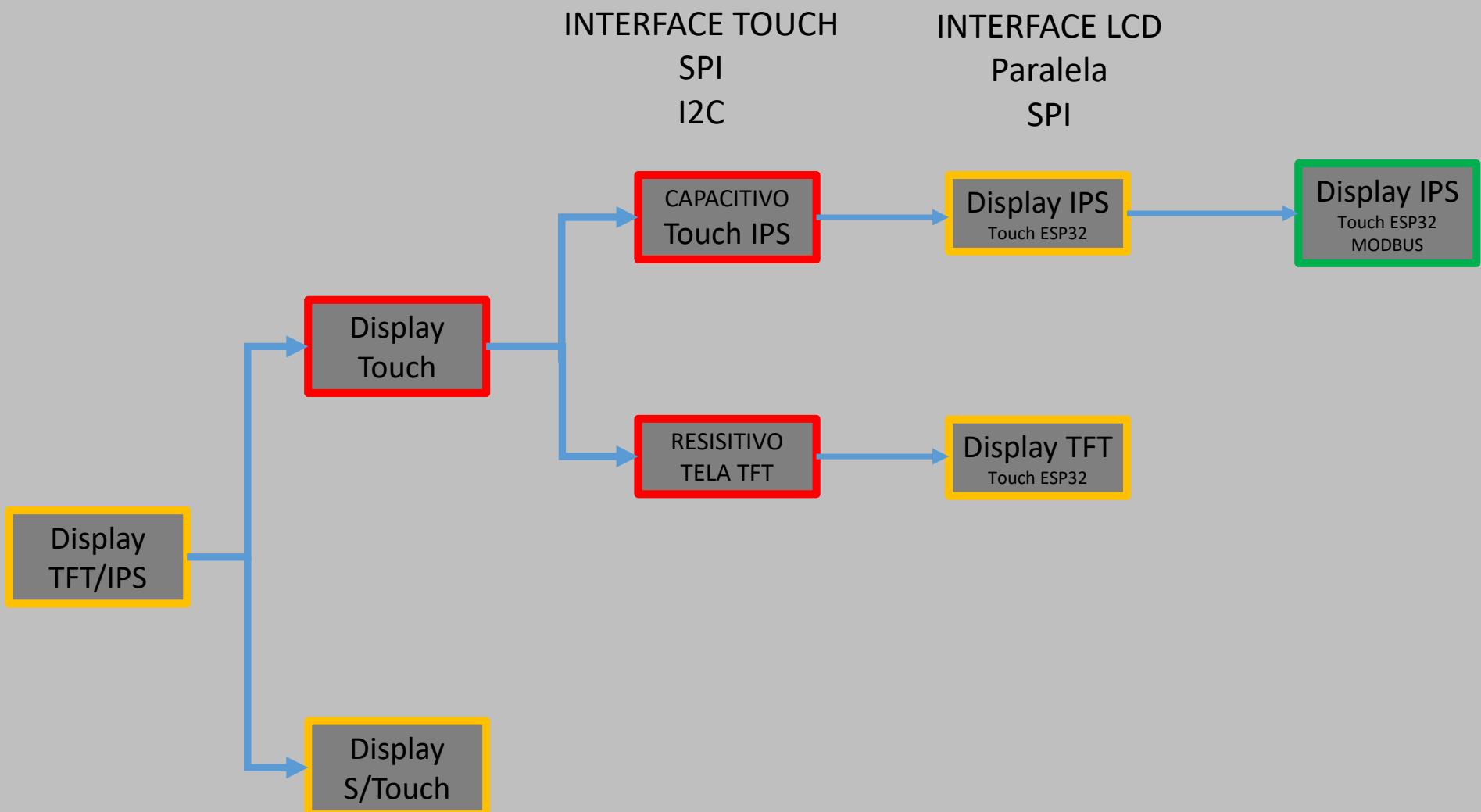
```
// Calcula quantos elementos existem no array (total de GPIOs a testar)
const int num_gpio = sizeof(gpio_list) / sizeof(gpio_list[0]);

void setup() {
    // Inicializa cada GPIO como saída digital e coloca em nível baixo (desligado)
    for (int i = 0; i < num_gpio; i++) {
        pinMode(gpio_list[i], OUTPUT);          // Configura o GPIO atual como saída
        digitalWrite(gpio_list[i], LOW);         // Garante que ele comece desligado
    }
}

void loop() {
    // Loop contínuo para testar GPIOs um por um
    for (int i = 0; i < num_gpio; i++) {
        digitalWrite(gpio_list[i], HIGH);       // Liga o GPIO atual
        delay(100);                          // Aguarda 100 milissegundos
        digitalWrite(gpio_list[i], LOW);        // Desliga o GPIO atual
    }
}
```

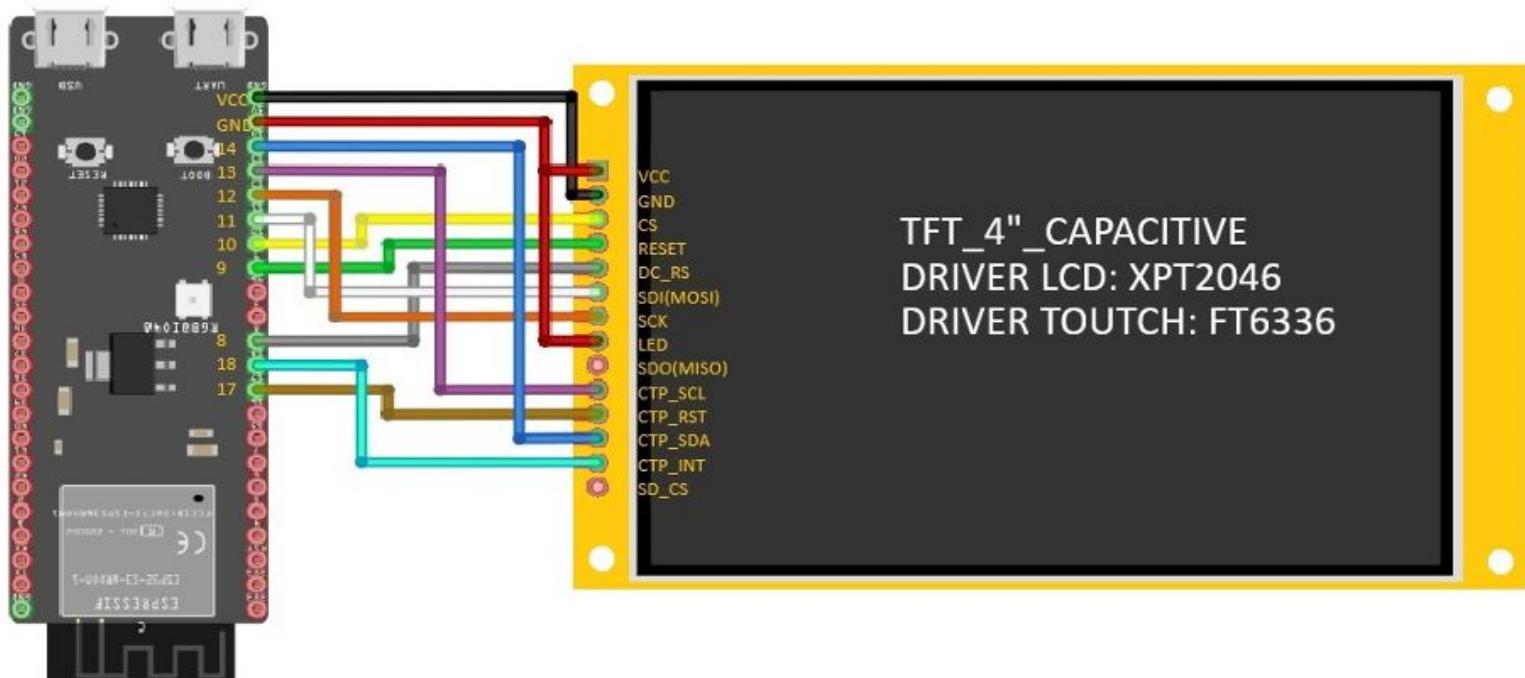
Displays





ESP32-S3 Capacitivo

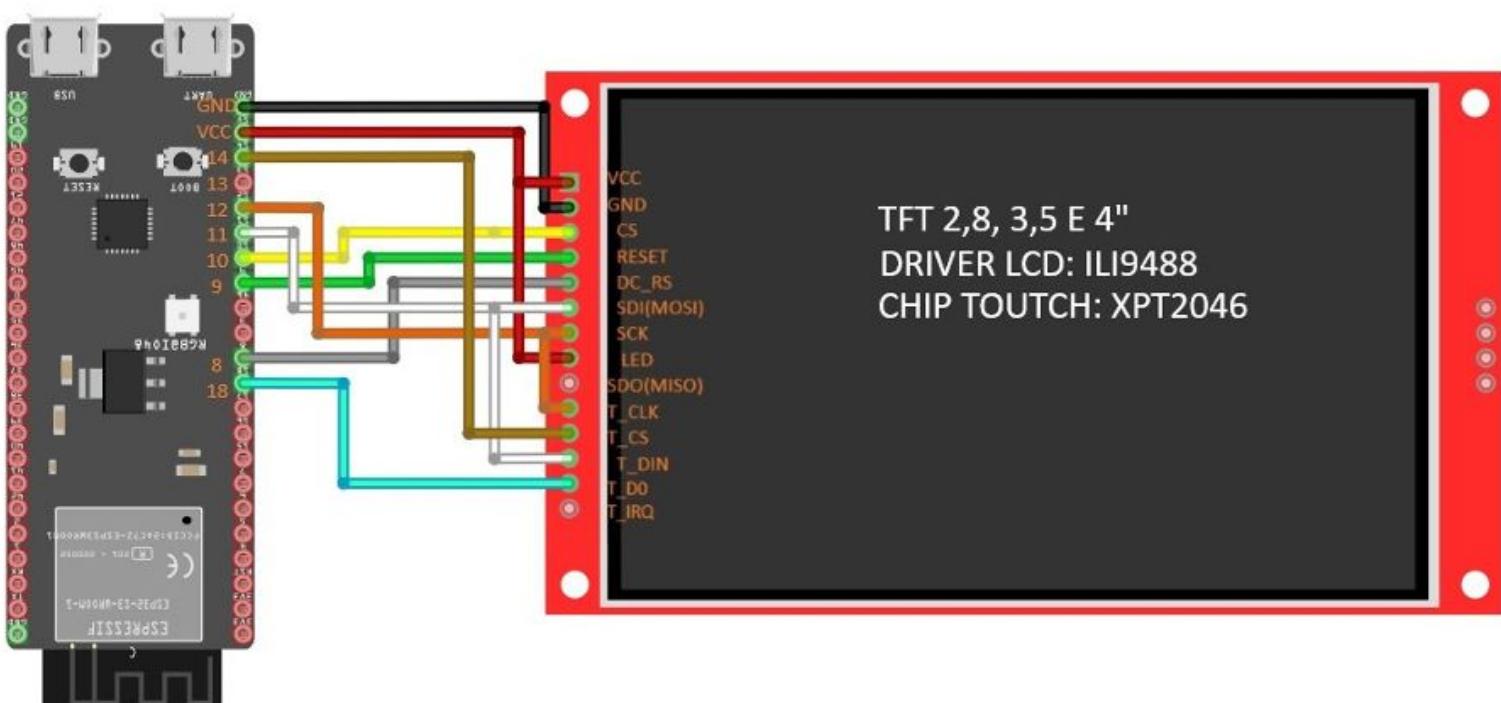
DISPLAY TFT, 4 capacitivo amarelo	
DISPLAY	esp32S3
CTP_CS	
CTP_INT	18
CTP_SDA	14
CTP_RST	17
CTP_SCL	13
SD0(MISO)	
LED	
SCK	12
SDI(MOSI)	11
DC_RS	8
RESET	9
CS	10
GND	
VCC	
SD_CS	
SD_MOSI	
SD_MISO	
SD_SCK	



TFT_4"_CAPACITIVE
DRIVER LCD: XPT2046
DRIVER TOUCH: FT6336

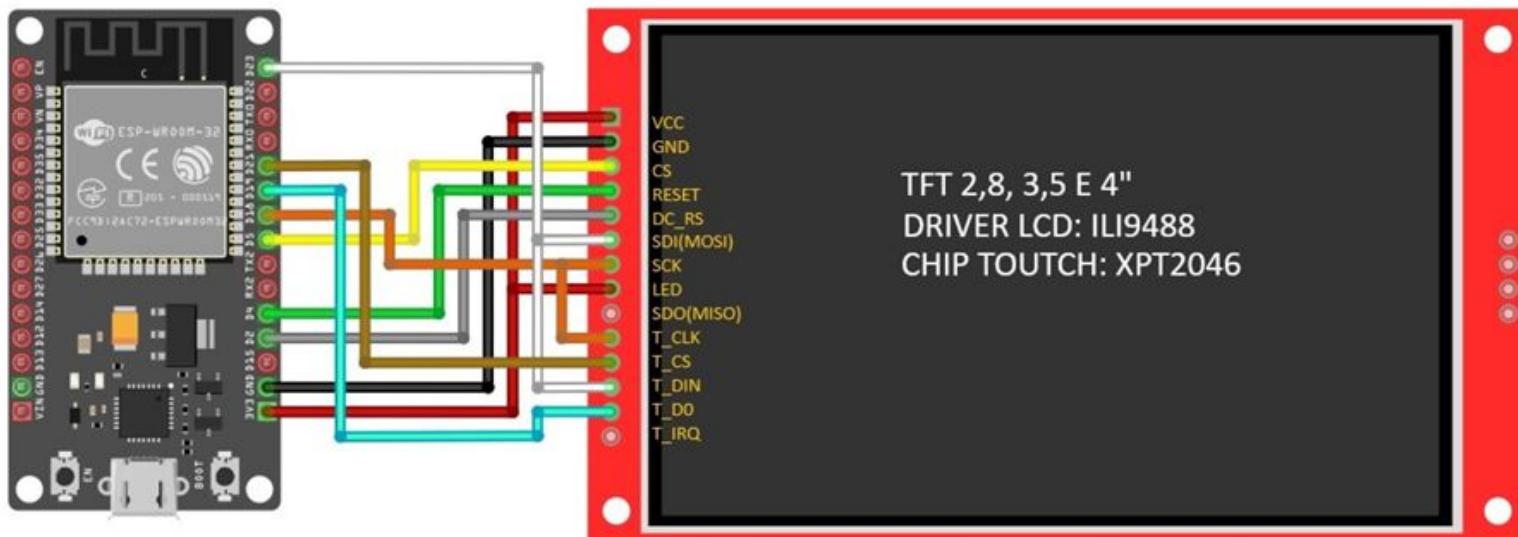
ESP32-S3 Resistivo

DISPLAY TFT 2,8, 3,5 E 4" Resistivo	
DISPLAY	esp32S3
T_IRQ	
T_D0	18
T_DIN	11
T_CS	14
T_CLK	12
SDO(MISO)	
LED	
SCK	12
SDI(MOSI)	11
DC_RS	8
RESET	9
CS	10
GND	
VCC	
SD_CS	
SD_MOSI	
SD_MISO	
SD_SCK	



ESP32-Devkit-V4

DISPLAY TFT 2,8, 3,5 E 4" Resistivo	
DISPLAY	ESP v4
T_IRQ	NOPE
T_D0	19
T_DIN	23
T_CS	21
T_CLK	18
SDO(MISO)	NOPE
LED	3V3
SCK	18
SDI(MOSI)	23
DC_RS	2
RESET	4
CS	5
GND	GND
VCC	3V3
SD_CS	14
SD_MOSI	12
SD_MISO	13
SD_SCK	27





ESP32-S3 4.3 Inch Touch LCD

LX7 Dual-core Processor Development Board

WiFi + BLE5
Onboard Antenna



RS485/CAN/I2C

Esp32 s3 4.3 Polegada lcd (b) 800x480 tela de toque captiva
lvgl placa desenvolvimento wifi ble5 com sensor pode i2c rs485

★★★★★ 5.0 1 avaliação | 29 vendidos

Promo Grandes Marcas
SuperOfertas

Termina : 27 ago, 23:59 (BRT)

R\$235,40 🔍 Poupe R\$245,02

R\$480,42

3% de Cashback

Compra internacional, +R\$105,04 em impostos estimados ⓘ

R\$21,38 x 12 meses >

R\$310,00 OFF em R\$1.900,00 >

cor: Without Case

https://s.click.aliexpress.com/e/_ooqnft

Preço do dia 20/08/2025



https://s.click.aliexpress.com/e/_ol8pNgz

<https://tinyurl.com/khcszw52>

AliExpress ⚙️ raspberry pi zero 2w kit



R\$88,20

6% de Cashback
Compra internacional, +R\$39,41 em impostos estimados ⓘ

R\$8,01 x 12 meses >

R\$360,00 de desconto em pedidos acima de R\$3.500,00 >

Novo módulo de placa amarela de toque capacitivo ips 4.0 Polegada spi ili9488 ft6236 esp32 14 pinos eletrônico 320*480

★★★★★ 4.7 27 avaliações | 119 vendidos

cor: IPS 4.0 CTP yellow





<https://tinyurl.com/khcszw52>

Preço do dia 20/08/2025



Preço do dia 20/08/2025

⚡ Em breve R\$60,08

Mega Saldão
SuperOfertas

R\$61,51

Começa : 13 : 06 : 58

R\$143,04 57% desc.

6% de Cashback

Compra internacional, +R\$27,49 em impostos estimados ⓘ

R\$5,58 x 12 meses >

R\$360,00 de desconto em pedidos acima de R\$3.500,00 >

ILI9488 FT6236 Módulo SPI vermelho de toque capacitivo de 3,5 polegadas 320*480 interface de painel de exibição LCD SPI TFT de 4 fios visão ampla

★★★★★ 5.0 1 avaliação | 11 vendidos

cor: TN 3.5 CTP FT6236

TN 3.5 CTP FT6236

Em www.fernandok.com

Download arquivo PDF e INO

