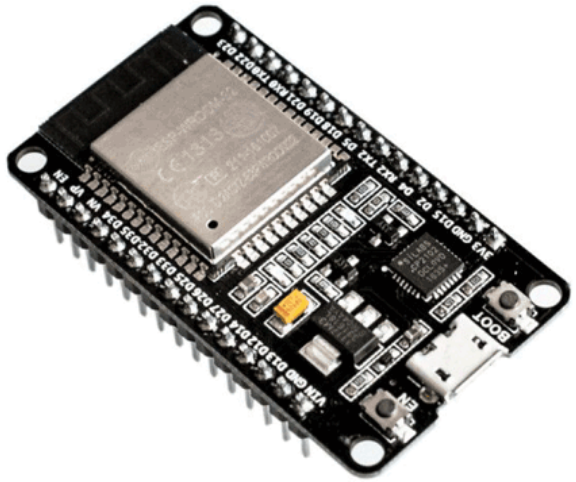


ESPRESSIF

Microcontroladores ESP32

Prof. Me. Hélio Esperidião

Entradas

As entradas, ou inputs, são sensores eletrônicos ou mecânicos que tomam os sinais (em forma de temperatura, pressão, umidade, contato, luz, movimento, ph, etc.) do mundo físico e converte em sinais de corrente ou voltagem.

Exemplos de entradas são sensores de gás, temperatura, pulsadores, fotocélulas, potenciômetros, sensores de movimento.

Saídas

As saídas, ou outputs, são atuadores, ou outros dispositivos que convertem os sinais de corrente ou voltagem em sinais fisicamente úteis como movimento, luz, som, força ou rotação, entre outros.

Exemplos de saídas são motores, LEDs ou sistemas de luzes que acendem automaticamente quando escurece ou um

Processamento

O processamento é realizado mediante circuitos conhecidos como microcontroladores.

São circuitos integrados construídos para manipular, interpretar e transformar os sinais de voltagem e corrente vindos dos sensores (entradas) e ativar determinadas ações nas saídas

Variável Digital

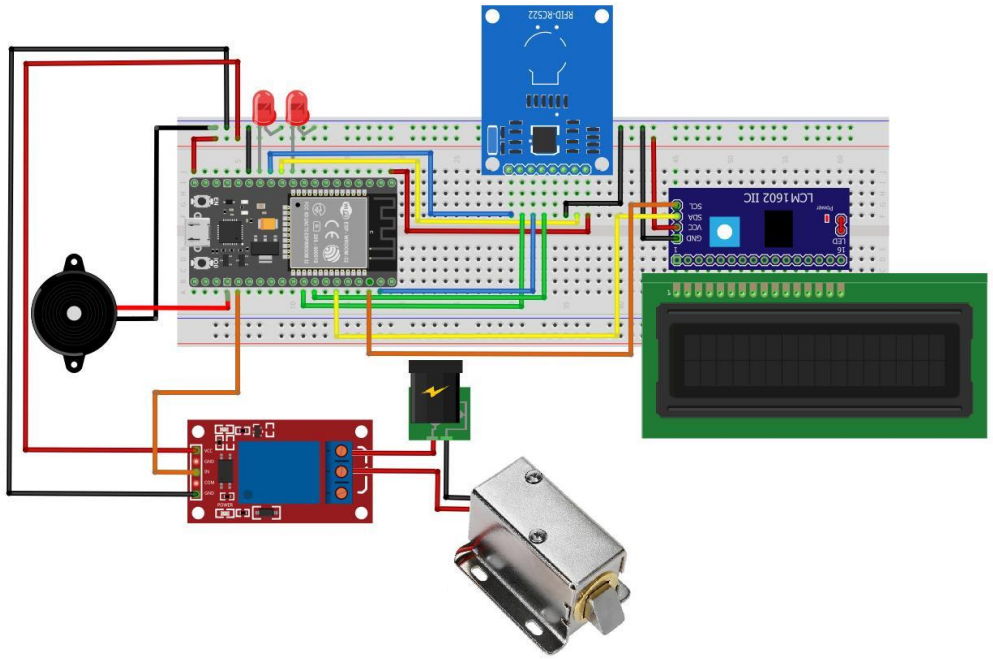
Também chamadas de variáveis discretas, se caracterizam por ter dois estados diferentes e portanto também podem ser chamadas de binárias (em lógica seria valores Verdadeiro (V) e Falso (F), ou poderiam ser 1 ou 0 respectivamente).

Um exemplo de um sinal digital é o interruptor da campainha da sua casa, porque ele tem somente dois estados, pulsado e sem pulsar.

Variável Analógica

- São aquelas que podem tomar um número infinito de valores compreendidos entre dois limites.
- A maioria dos fenômenos da vida real são sinais deste tipo (som, temperatura, luminosidade, etc.).

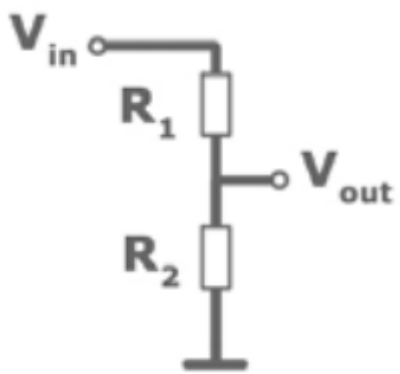
Entrada/Saída



SN	Product Name	Product Image	SN	Product Name	Product Image
1	DHT11 temperature and humidity sensor		12	DuPont Cable (Male to female)	
2	SR501 Infrared sensor		13	HC-SR04 Ultrasonic sensor	
3	GY-BMP280 Pressure Sensor		14	SW-420 Vibration switch sensor	
4	LM393 Photosensitive sensor		15	TCRT5000 Tracking module	
5	Soil Moisture sensor		16	KY-037 Microphone sensor	
6	KY-008 red laser sensor		17	KY-018 Photosensitive sensor	
7	TTP223B Touch sensor		18	Obstacle avoidance sensor	
8	Water level sensor		19	Raindrop sensor	
9	Voltage sensor		20	Angle Tilt Switch Sensor	
10	Speed sensor		21	KY-022 Infrared sensor	
11	KY-024 Linear magnetic Hall sensor		22	KY-003 Hall magnetic sensor	

Divisor de TENSÃO

- Em eletrônica, a regra do divisor de tensão é uma técnica de projeto utilizada para criar uma tensão elétrica (Vout) que seja proporcional à outra (Vin). Desta forma a voltagem de uma fonte é repartida entre uma ou mais resistências conectadas em série. Em um circuito deste tipo, dua o ligadas em série como no esquema a seguir:



$$V_{out} = \frac{R_2}{R_1 + R_2} \cdot V_{in}$$

Conversor Analógico-Digital

Um conversor analógicodigital (ou ADC em inglês - Analog-to-Digital Converter) é um dispositivo eletrônico capaz de gerar uma representação digital a partir de uma grandeza analógica, convertendo uma entrada analógica de voltagem em um valor binário.

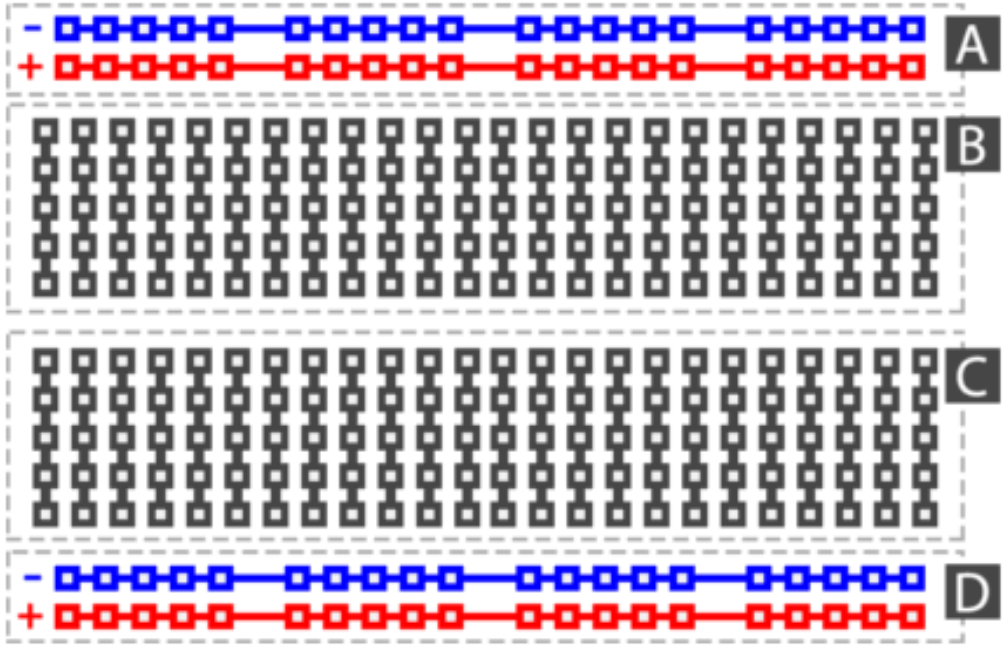
Se utiliza em equipamentos eletrônicos como computadores, gravadores de som e vídeo e equipamentos de telecomunicações.

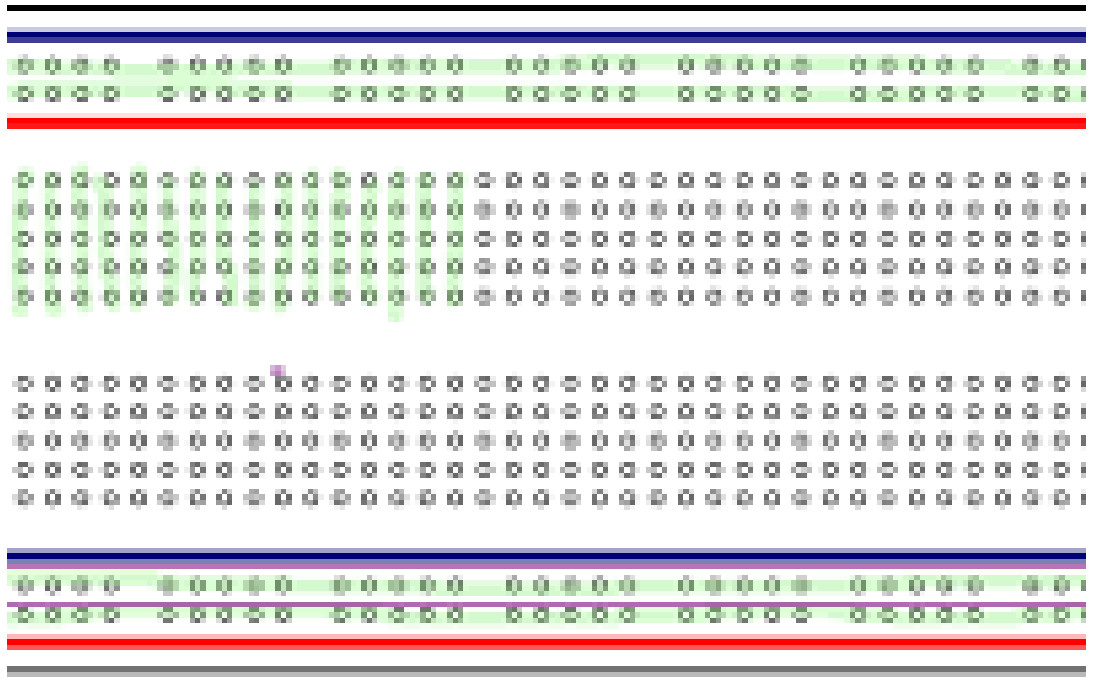
Comunicação Serial

É uma interface de comunicação de dados digitais em que a informação é enviada um bit de cada vez, sequencialmente.

É diferente da comunicação paralela, em que todos os bits de cada símbolo são enviados juntos.

A comunicação serial é usada em toda comunicação de longo alcance e na maioria das redes de computadores.



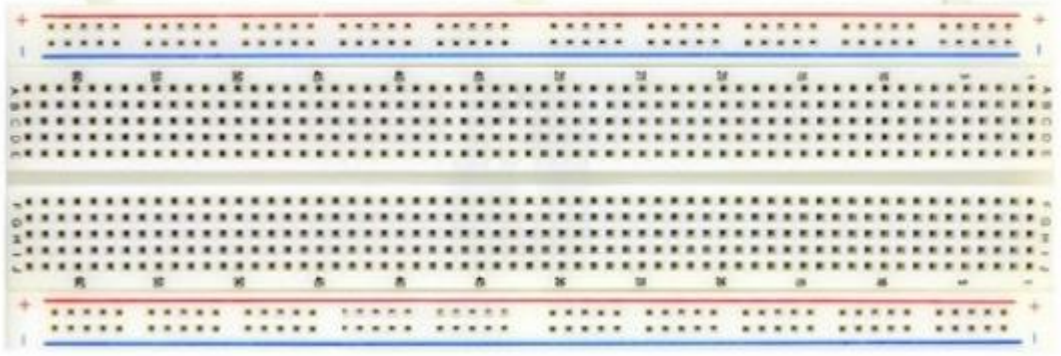


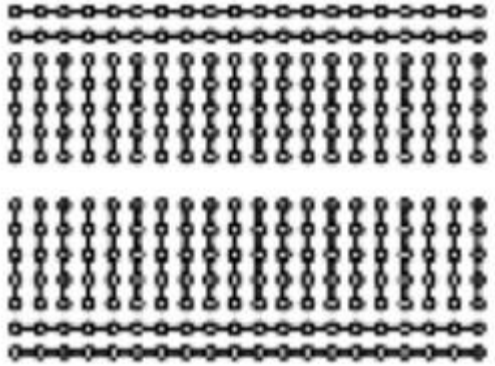
protoboard

- A – Azul em curto na horizontal
- A – Vermelho em curto na horizontal
- B – em curto na vertical
- C - Em curto na vertical
- D – Azul em curto na horizontal
- C – Vermelho em curto na horizontal

Protoboard

- É uma placa reutilizável usada para construir protótipos de circuitos eletrônicos sem solda. Uma protoboard é feita por blocos de plástico perfurados e várias lâminas finas de uma liga metálica de cobre

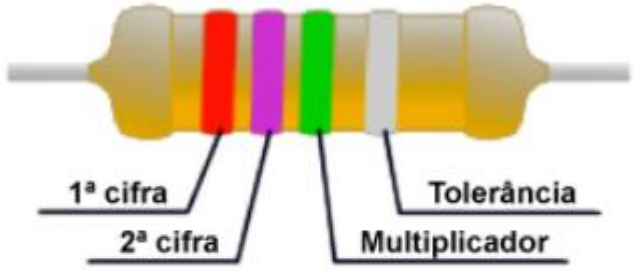




Resistor

- Formula =
- R = (1 Cifra “+” 2 Cifra) * Multiplicador

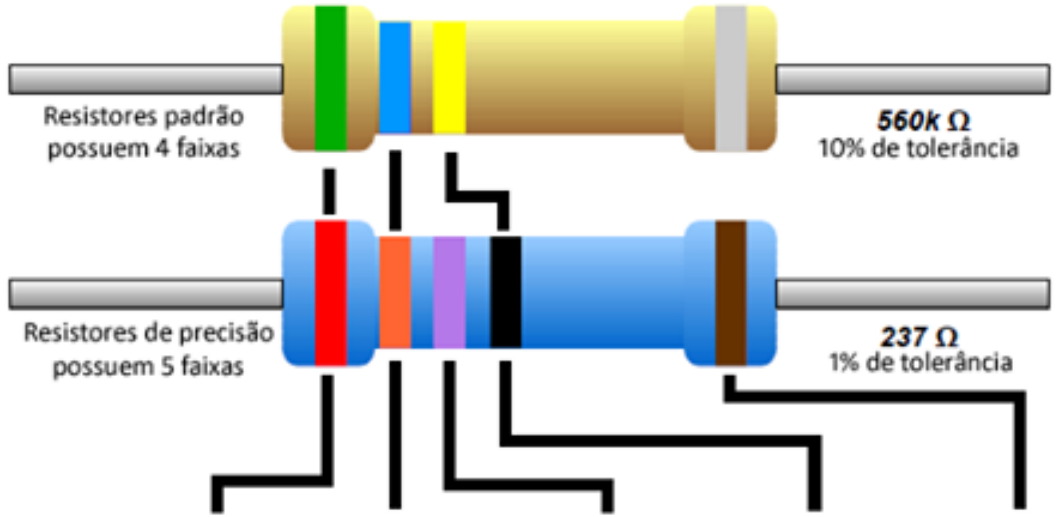
- Um resistor de 2.700.000Ω (2,7MΩ), com uma t seria representado pela figura.



1ª cifra: vermelho (2)
2ª cifra: violeta (7)
Multiplicador: verde (10^5)
Tolerância: prata (±10%)

Código de Cores

A extremidade com mais faixas deve apontar para a esquerda



Cor	1ª Faixa	2ª Faixa	3ª Faixa	Multiplicador	Tolerância
Preto	0	0	0	x 1 Ω	
Marrom	1	1	1	x 10 Ω	+/- 1%
Vermelho	2	2	2	x 100 Ω	+/- 2%
Laranja	3	3	3	x 1K Ω	
Amarelo	4	4	4	x 10K Ω	
Verde	5	5	5	x 100K Ω	+/- .5%
Azul	6	6	6	x 1M Ω	+/- .25%
Violeta	7	7	7	x 10M Ω	+/- .1%
Cinza	8	8	8		+/- .05%
Branco	9	9	9		
Dourado				x .1 Ω	+/- 5%
Prateado				x .01 Ω	+/- 10%

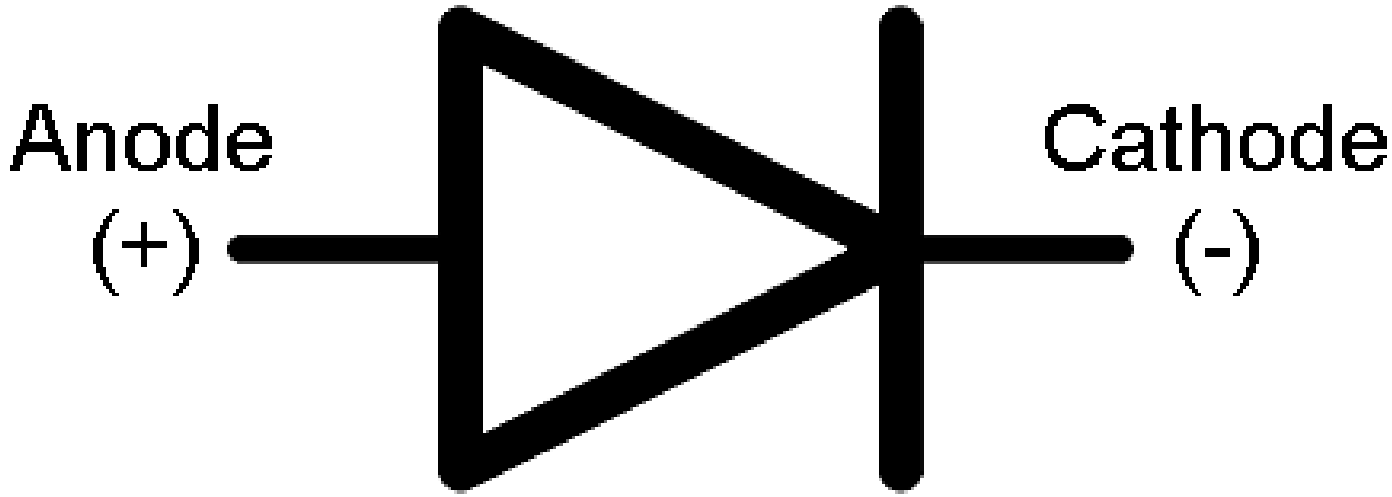
Termistor

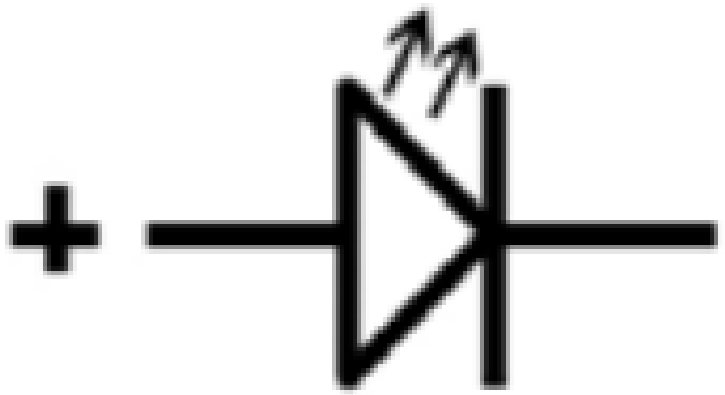
O termistor NTC (do inglês Negative Temperature Coefficient) é um componente eletrônico semicondutor sensível à temperatura, utilizado para controle, medição ou polarização de circuitos eletrônicos.

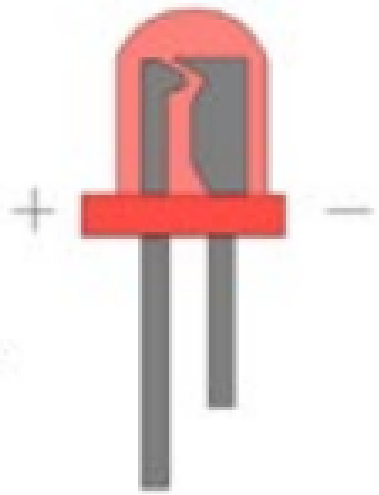
Possui um coeficiente de variação de resistência que varia negativamente conforme a temperatura aumenta, ou seja, a sua resistência elétrica diminui com o aumento da temperatura.

Diodo

- É o tipo mais simples de componente eletrônico semicondutor.
- É um componente que permite que a corrente atravesse somente em um sentido.





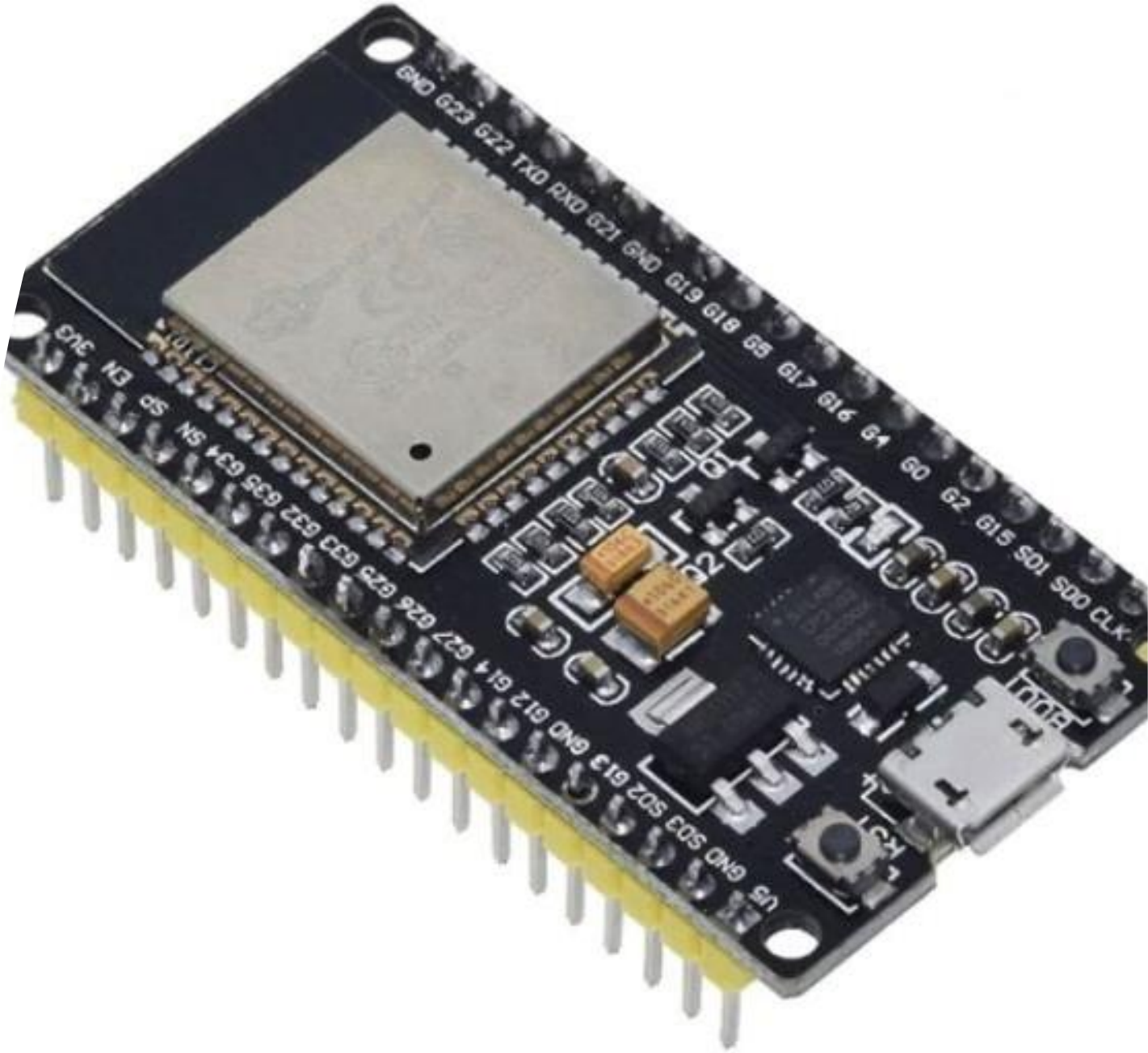


LED

- O LED (Light Emitting Diode) é um diodo que emite luz quando energizado. Os LED´s apresentam muitas vantagens sobre as fontes de luz incandescentes como um consumo menor de energia, maior tempo de vida, menor tamanho, grande durabilidade e confiabilidade.

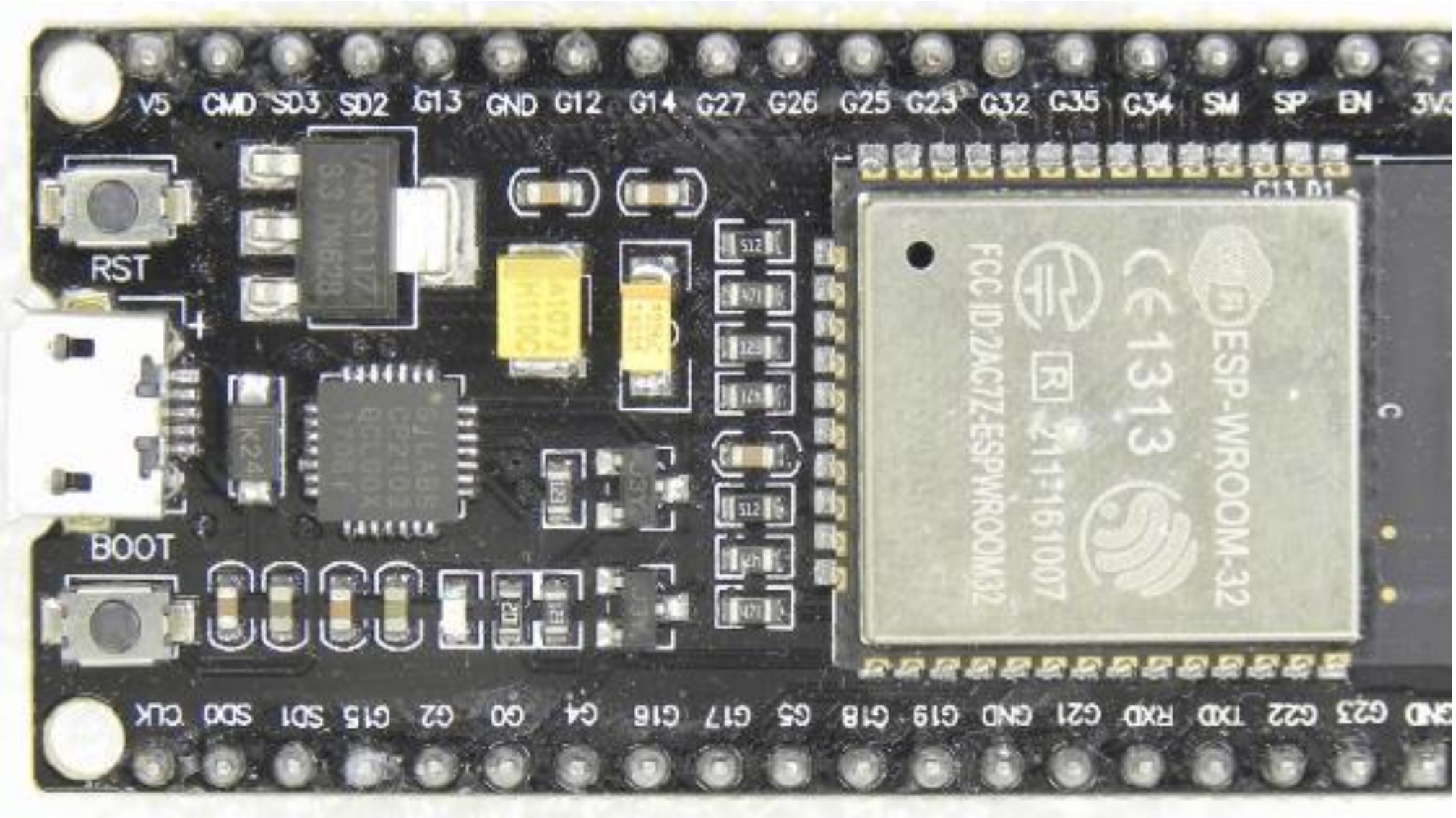
Qual comprar?

- Existem várias versões
- Recomendo a versão:
  - Placa Esp32 Wroom32 Cp2102 Wifi Bluetooth 38 Pinos
- O “Amarelinho”



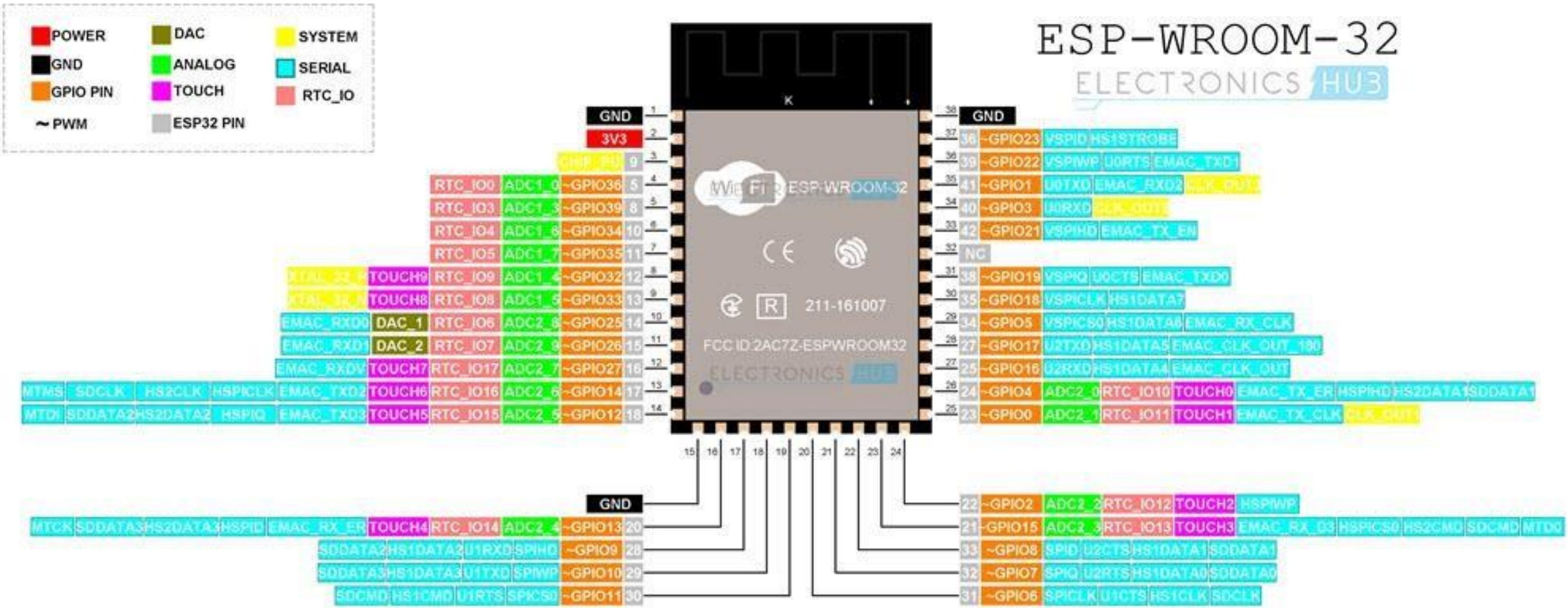
Existe uma versão muito semelhante:

Placa Esp32 Wroom32 Cp2102 Wifi Bluetooth 30 Pinos

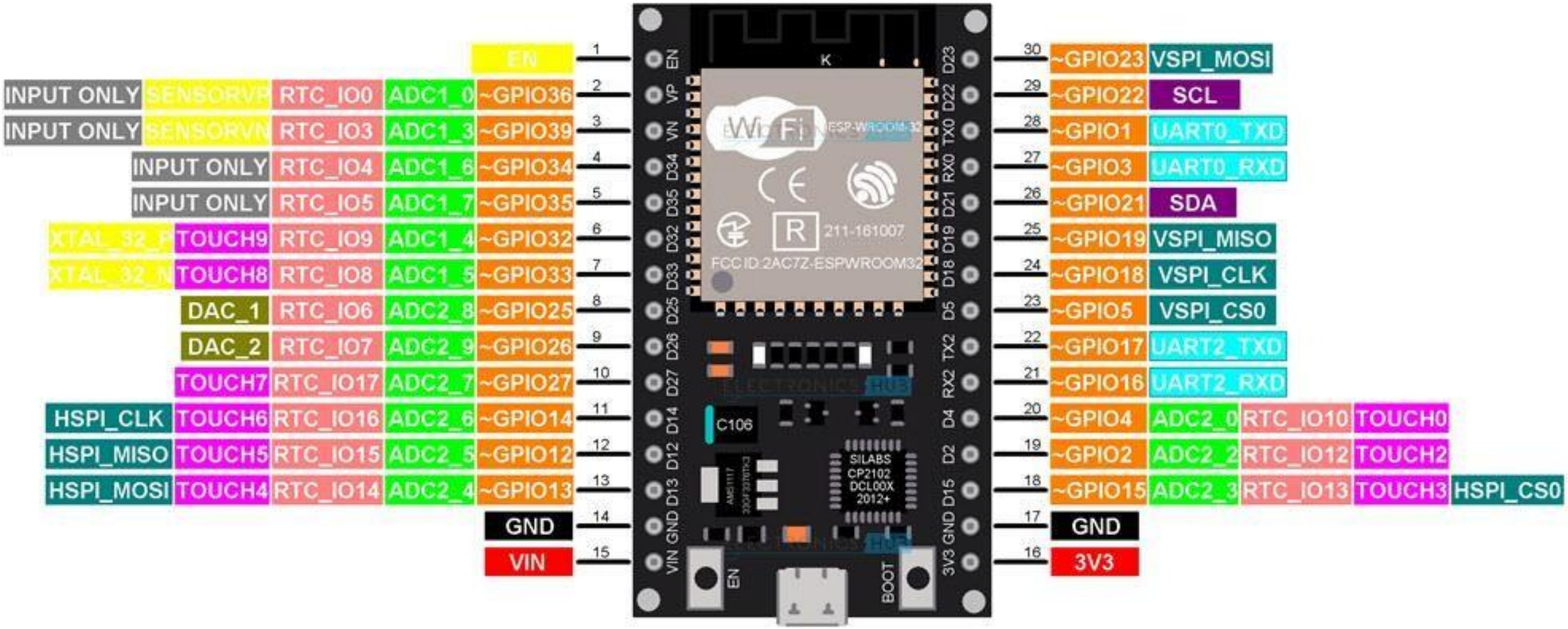


Microcontrolador encapsulado

- O microcontrolador de fato é encontrado dentro do encapsulamento mostrado em vermelho ao lado.
- O resto do circuito é apenas uma implementação de placa para fins educacionais.
- É possível construir uma placa totalmente personalizada para aplicações especificas.

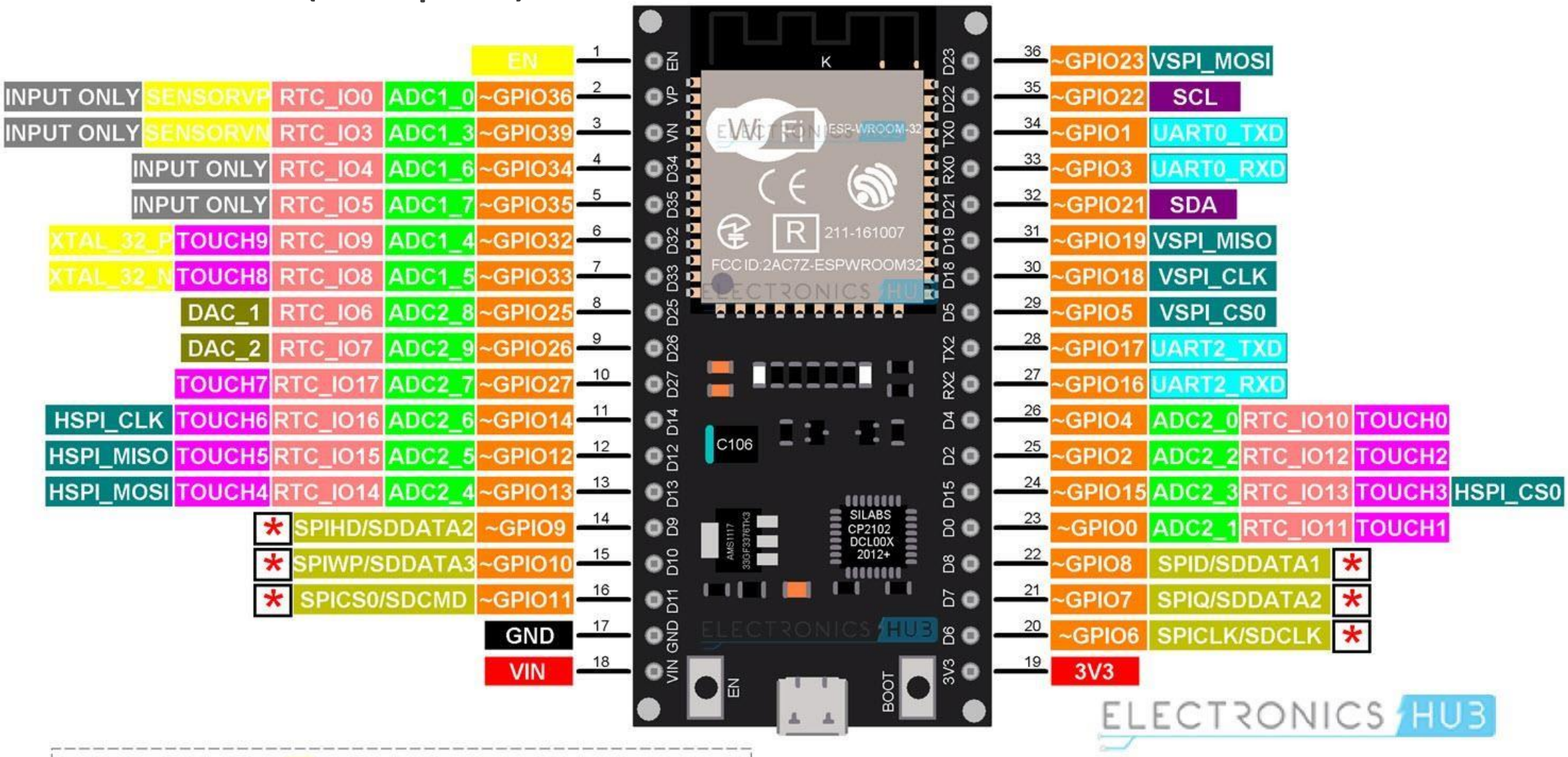

POWER
GND
GPIO PIN
~ PWM
DAC
ANALOG
TOUCH
ESP32 PIN
SYSTEM
SERIAL
RTC_IO
ESP-WROOM-32
ELECTRONICS HUB
ESP-WROOM-32
211-161007
FCC ID 2AC7Z-ESPWROOM32
GND 1
3V3 2
9 3
RTC_IO0 ADC1_0 ~GPIO36 5 4
RTC_IO3 ADC1_3 ~GPIO39 8 5
RTC_IO4 ADC1_6 ~GPIO34 10 6
RTC_IO5 ADC1_7 ~GPIO35 11 7
TOUCH9 RTC_IO9 ADC1_4 ~GPIO32 12 8
TOUCH8 RTC_IO8 ADC1_5 ~GPIO33 13 9
EMAC_RXD0 DAC_1 RTC_IO6 ADC2_8 ~GPIO25 14 10
EMAC_RXD1 DAC_2 RTC_IO7 ADC2_9 ~GPIO26 15 11
EMAC_RXDV TOUCH7 RTC_IO17 ADC2_7 ~GPIO27 16 12
MTMS SDCLK HS2CLK HSPICLK EMAC_TXD2 TOUCH6 RTC_IO16 ADC2_6 ~GPIO14 17 13
MTDI SDDATA2 HS2DATA2 HSPIQ EMAC_TXD3 TOUCH5 RTC_IO15 ADC2_5 ~GPIO12 18 14
15 16 17 18 19 20 21 22 23 24
GND
MTCK SDDATA3 HS2DATA3 HSPID EMAC_RX_ER TOUCH4 RTC_IO14 ADC2_4 ~GPIO13 20
SDDATA2 HS1DATA2 U1RXD SPIHD ~GPIO9 28
SDDATA3 HS1DATA3 U1TXD SPIWP ~GPIO10 29
SDCMD HS1CMD U1RTS SPICS0 ~GPIO11 30
38 GND
37 36 ~GPIO23 VSPID HS1STROBE
36 39 ~GPIO22 VSPIWP U0RTS EMAC_TXD1
35 41 ~GPIO1 U0TXD EMAC_RXD2
34 40 ~GPIO3 U0RXD
33 42 ~GPIO21 VSPIHD EMAC_TX_EN
32 NC
31 38 ~GPIO19 VSPIQ U0CTS EMAC_TXD0
30 35 ~GPIO18 VSPICLK HS1DATA7
29 34 ~GPIO5 VSPICS0 HS1DATA6 EMAC_RX_CLK
28 27 ~GPIO17 U2TXD HS1DATA5 EMAC_CLK_OUT_180
27 25 ~GPIO16 U2RXD HS1DATA4 EMAC_CLK_OUT
26 24 ~GPIO4 ADC2_0 RTC_IO10 TOUCH0 EMAC_TX_ER HSPIHD HS2DATA1 SDDATA1
25 23 ~GPIO0 ADC2_1 RTC_IO11 TOUCH1 EMAC_TX_CLK
22 ~GPIO2 ADC2_2 RTC_IO12 TOUCH2 HSPIWP
21 ~GPIO15 ADC2_3 RTC_IO13 TOUCH3 EMAC_RX_D3 HSPICS0 HS2CMD SDCMD MTDO
33 ~GPIO8 SPID U2CTS HS1DATA1 SDDATA1
32 ~GPIO7 SPIQ U2RTS HS1DATA0 SDDATA0
31 ~GPIO6 SPICLK U1CTS HS1CLK SDCLK

Pin out (30 pin)



ELECTRONICS HUB

Pin out (38 pin)



Pinos esp 32

GPIO	In	Out	Notes
0	pulled up	OK	Saída PWM signal no boot (HIGH no boot)
1	TX pin	OK	debug output at boot (HIGH no boot)
2	OK	OK	Pode estar conectado a um LED na placa
3	OK	RX pin	HIGH no boot
4	OK	OK	
5	OK	OK	Escreve sinal PWM signal no boot (HIGH no boot)
6	x	x	connected to the integrated SPI flash (HIGH no boot)
7	x	x	connected to the integrated SPI flash (HIGH no boot)
8	x	x	connected to the integrated SPI flash (HIGH no boot)
9	x	x	connected to the integrated SPI flash (HIGH no boot)
10	x	x	connected to the integrated SPI flash (HIGH no boot)

Pinos esp 32

GPIO	In	Out	Notes
10	x	x	connected to the integrated SPI flash (HIGH no boot)
11	x	x	connected to the integrated SPI flash (HIGH no boot)
12	OK	OK	boot falha se ligado no HIGH
13	OK	OK	
14	OK	OK	Escreve sinal PWM signal no boot (HIGH at boot)
15	OK	OK	Escreve sinal PWM signal no boot (HIGH at boot)
16	OK	OK	
17	OK	OK	
18	OK	OK	
19	OK	OK	
21	OK	OK	

Pinos esp 32

GPIO	Input	Output	Notes
22	OK	OK	
23	OK	OK	
25	OK	OK	
26	OK	OK	
27	OK	OK	
32	OK	OK	
33	OK	OK	
34	OK		Somente entrada de dados
35	OK		Somente entrada de dados
36	OK		Somente entrada de dados
39	OK		Somente entrada de dados

Pinos somente de entrada de dados

GPIO 34

GPIO 35

GPIO 36

GPIO 39

Pinos integrados a interface SPI da memória flash

GPIO 6 a GPIO 11 estão expostos em algumas placas de desenvolvimento ESP32.

No entanto, esses pinos estão conectados ao flash SPI integrado no chip ESP-WROOM-32 e não são recomendados para outros usos.

Então, não use esses pinos em seus projetos.

GPIO 6 (SCK/CLK)

GPIO 7 (SDO/SD0)

GPIO 8 (SDI/SD1)

GPIO 9 (SHD/SD2)

GPIO 10 (SWP/SD3)

GPIO 11 (CSC/CMD)

Capacitive touch GPIOs

O ESP32 possui 10 sensores de toque capacitivos internos.

Eles podem sentir variações em qualquer coisa que tenha uma carga elétrica, como a pele humana.

Assim, eles podem detectar variações induzidas ao tocar os GPIOs com o dedo.

Esses pinos podem ser facilmente integrados em pads capacitivos e substituir botões mecânicos.

Os pinos de toque capacitivos também podem ser usados para acordar o ESP32 do sono profundo.

Analog to Digital Converter (ADC)

O ESP32 possui canais de entrada ADC de 18 x 12 bits

Estes são os GPIOs que podem ser usados como ADC e respectivos canais:

ADC1_CH0 (GPIO 36)
ADC1_CH1 (GPIO 37)
ADC1_CH2 (GPIO 38)
ADC1_CH3 (GPIO 39)
ADC1_CH4 (GPIO 32)
ADC1_CH5 (GPIO 33)
ADC1_CH6 (GPIO 34)
ADC1_CH7 (GPIO 35)
ADC2_CH0 (GPIO 4)
ADC2_CH1 (GPIO 0)
ADC2_CH2 (GPIO 2)
ADC2_CH3 (GPIO 15)
ADC2_CH4 (GPIO 13)
ADC2_CH5 (GPIO 12)
ADC2_CH6 (GPIO 14)
ADC2_CH7 (GPIO 27)
ADC2_CH8 (GPIO 25)
ADC2_CH9 (GPIO 26)


GND G23 G22 TXD RXD G21 GND G19 G18 G5 G17 G16 G4 G0 G2 G15 SD1 SD0 CLK
3V3 EN SVP SVN G34 G35 G32 G33 G25 G26 G27 G14 G12 GND G13 SD2 SD3 GND 5V
EN
BOOT

C para ESP 32

Tipo de Dado

boolean	(true) ou falso (false)
char	um caractere
byte	8 bits
int	inteiro de 32 bits
long	inteiro de 64 bits
Float	32 bits
Double	64 bits
string	8 bits por letra

Expressões Aritméticas

Operador	Descrição	Hierarquia da Operação
+ -	Soma Subtração	3º 3º
* / (%)	Multiplicação Divisão Resto da divisão	2º 2º 2º
( )	Parênteses	1º 1º

Operadores relacionais

Operador	Função
>	Maior
<	Menor
>=	Maior ou igual
<=	Menor ou igual
!=	Diferente de
==	Igual a

Operadores lógicos

Operador	Função
&&	E
||	Ou
!	Inversor

Estrutura básica: main.cpp

```
void setup()
{
    // Configurações iniciais
}
void loop()
{
    // loop infinito
   // todo microcontrolador roda seu programa infinitamente
}
```

O que é um protocolo de comunicação?

- Para que dois dispositivos possam estabelecer comunicação, é essencial que ocorra a transferência de dados.
- Diante da diversidade de fabricantes, protótipos e aplicações, a criação de um padrão para essa transferência é fundamental para possibilitar a comunicação entre diferentes componentes.

- Um protocolo de comunicação consiste em um conjunto de regras que facilitam a interação entre placas, computadores e dispositivos eletrônicos de marcas distintas e concorrentes.
- Um exemplo amplamente utilizado nos dias atuais é o protocolo Wifi, presente em nossos dispositivos móveis, que permite a conexão à internet sem a necessidade de cabos.

Comunicação Serial

- Para estabelecer a comunicação entre placas e computadores, é imprescindível contar com um protocolo que facilite essa interação.

- O protocolo Serial é empregado por essas placas, viabilizando a transmissão sequencial de dados, um bit por vez, por meio de um barramento ou canal de comunicação.

- Suas características fundamentais incluem o método, o modo de transmissão e a taxa de comunicação.

comunicação Síncrona

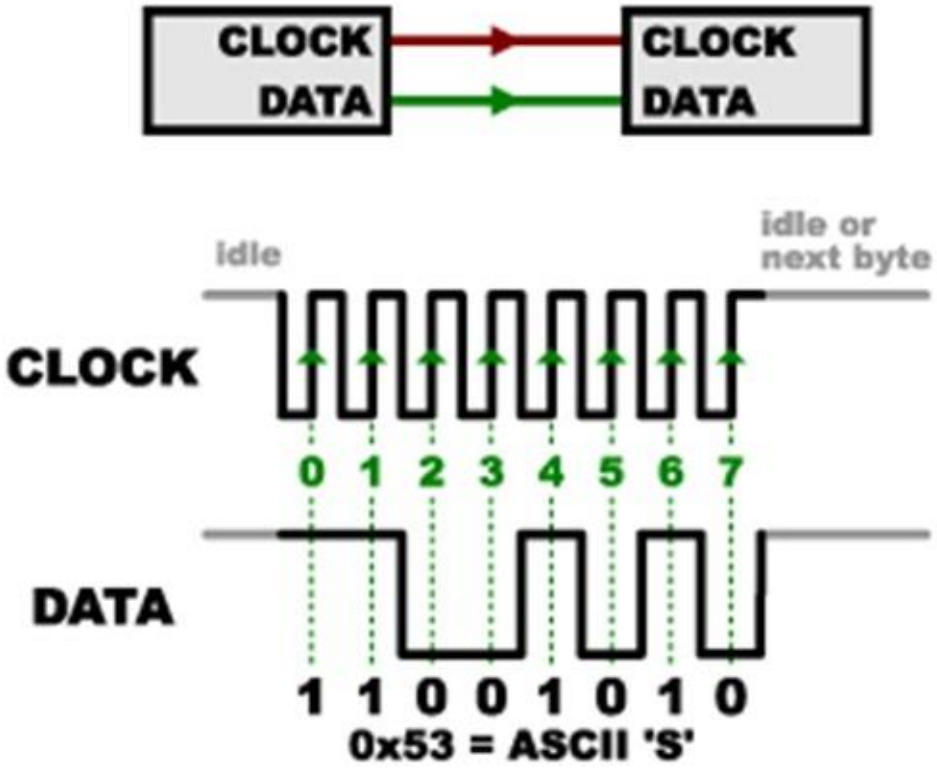
- O Transmissor e o Receptor devem utilizar a mesma frequência e fase, além do mesmo clock para a comunicação.

- Envolve a transferência de blocos, em que cada bloco inclui caracteres de sincronismo, dados e verificação de erro.
- Apresenta um baixo overhead, que se refere à quantidade de bits adicionados para garantir a chegada correta da informação ao destino.
- Em cada bloco, são acrescentados +3 bytes (2 para sincronismo e 1 para checksum).

- Exemplo: Para um bloco de 1024 bytes, o total seria 1027 bytes. Isso resulta em apenas um aumento de tempo de 0,3%. (1 byte = 8 bits)

Sincronismo com o clock

- Na subida do clock é enviado o dado

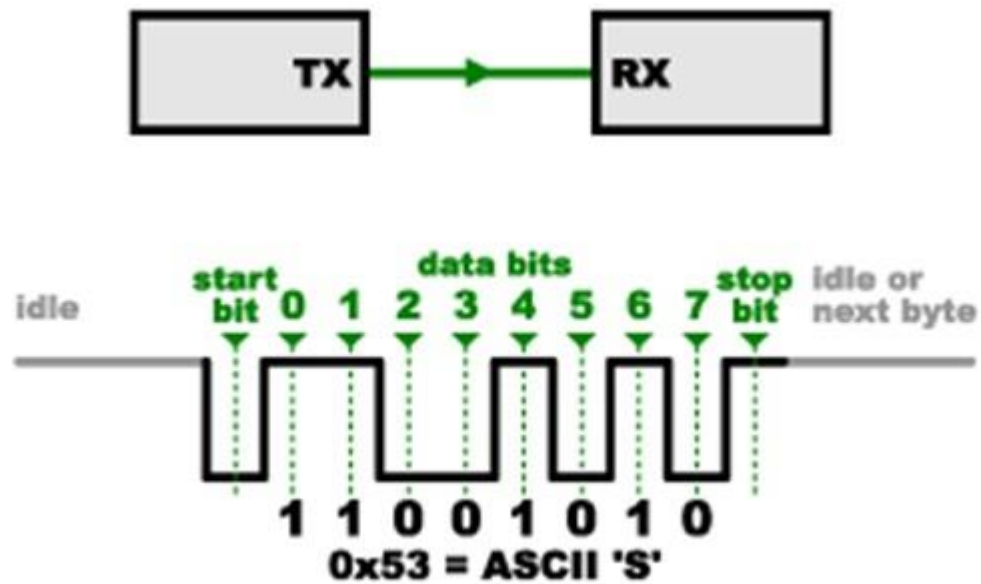


Comunicação Assíncrona

- O clock entre o transmissor e o receptor pode variar, uma vez que há ressincronização a cada byte.

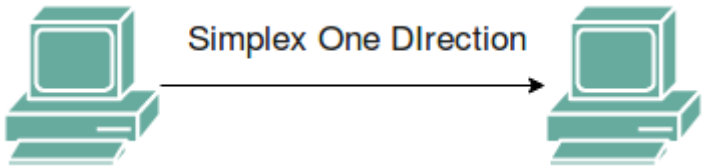
- Envolve a transferência de bytes.

- Para cada byte, são acrescentados 2 ou 3 bits, incluindo o bit de início, o bit de paridade e os bits de parada (1 a 2).
  - É feito para garantir a ressincronização
- Apresenta um alto overhead, pois para cada byte (8 bits), há a adição de 3 bits, totalizando 11 bits.
- Isso representa um aumento de 37,5% no tempo, resultando em uma velocidade menor.


TX
RX
idle
start bit
data bits
0
1
2
3
4
5
6
7
stop bit
idle or next byte
1 1 0 0 1 0 1 0
0x53 = ASCII 'S'

Formas de transmissão
Simplesx

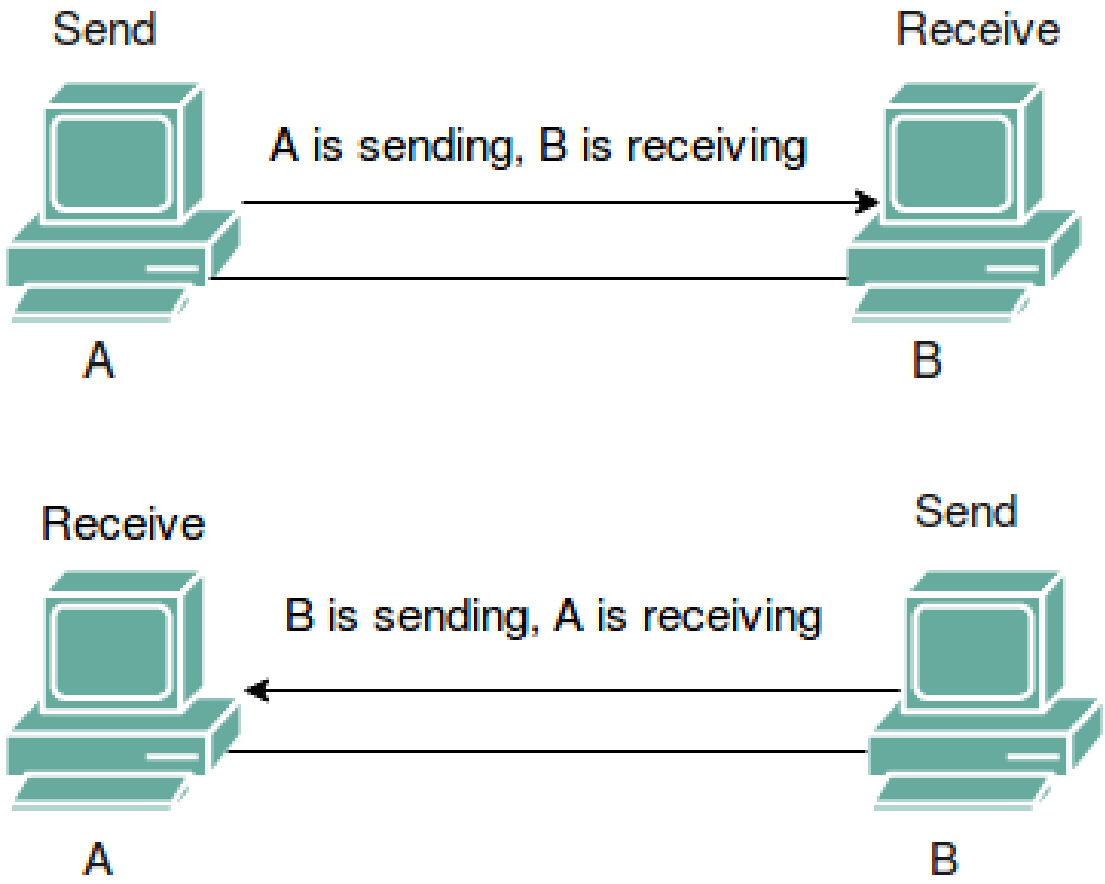
- O modo de comunicação chamado Simplesx é unidirecional, permitindo que apenas um dos dispositivos realize a transmissão, enquanto o outro apenas recebe informações.



Formas de transmissão
Half-Duplex

- No modo half-duplex, cada estação tem a capacidade de tanto transmitir quanto receber, porém, não simultaneamente.
- Quando um dispositivo está enviando, o outro pode apenas receber e vice-versa.
- Este modo é empregado em situações onde não é necessário realizar comunicação em ambas as direções simultaneamente.
- Dessa forma, toda a capacidade do canal pode ser dedicada a cada direção conforme necessário.

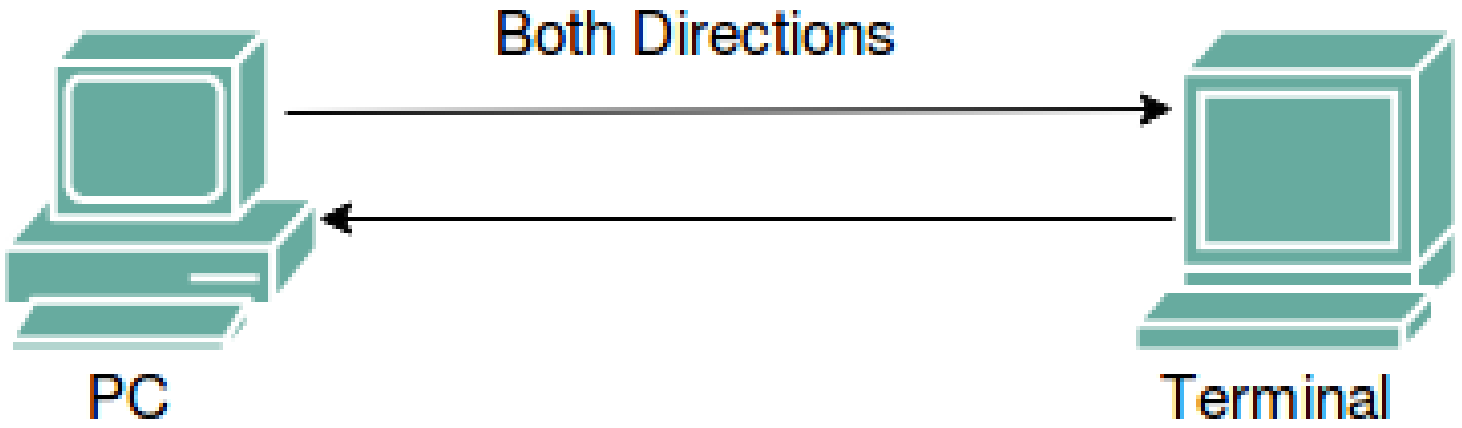
Formas de transmissão
Half-Duplex



Formas de transmissão full-duplex

- No modo full-duplex, ambas as estações têm a capacidade de transmitir e receber simultaneamente. Neste modo, os sinais que se dirigem em uma direção compartilham a capacidade do link com os sinais que seguem na direção oposta. Esse compartilhamento pode ser alcançado de duas maneiras:
- O link deve possuir dois caminhos de transmissão fisicamente separados, um dedicado ao envio e outro ao recebimento.
- A capacidade do link é dividida entre os sinais que viajam em ambas as direções.

Formas de transmissão
full-duplex



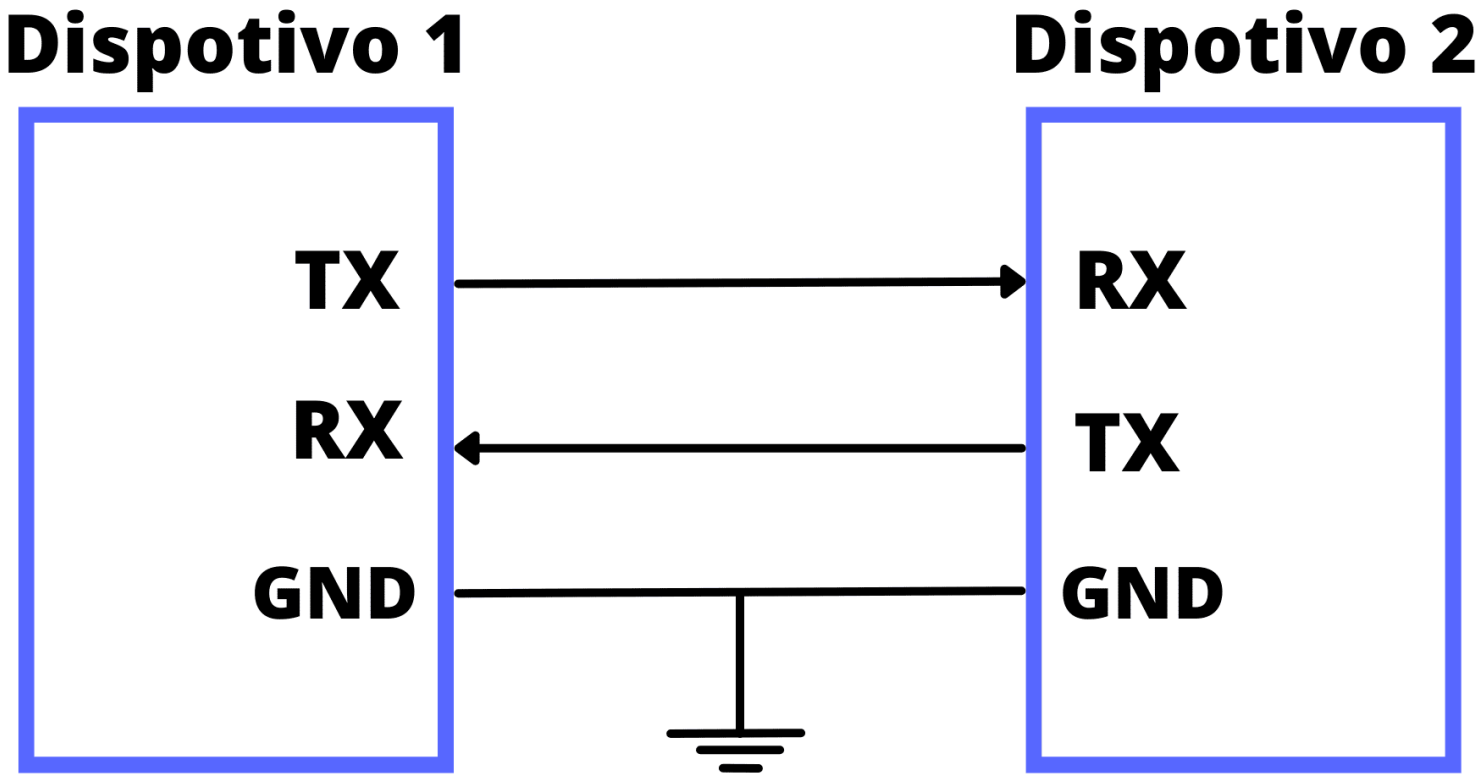
Principais protocolos e formas de comunicação.

Protocolo	Sentido	Método	Voltagem
UART	Full-duplex	Assíncrono	0 a 5V
I²C	Half-duplex	Síncrono	0 a 5V
SPI	Full-duplex	Síncrono	0 a 5V
RS232	Full-duplex	Síncrono ou Assíncrono	-15 a 25V
1Wire	Half-duplex	Assíncrono	2,8 a 6,0V
USB	Half-duplex ou Full duplex	Assíncrono	0 a 20V

Comunicação UART/ESP32

- UART - Universal Asynchronous Receiver/Transmitter
- Esse protocolo é comumente usado para transmitir dados de forma serial entre dispositivos, e no contexto do ESP32, é frequentemente empregado para a comunicação entre o microcontrolador e outros dispositivos, como um computador, sensores, ou módulos externos.
- Esse tipo de comunicação usa um fio para transmitir dados (TX - transmitir) e outro para receber dados (RX - receber).

Esquema básico UART



Taxa de transmissão

- A taxa de baud (baud rate) em uma comunicação UART (Universal Asynchronous Receiver/Transmitter) se refere à velocidade com que os bits de dados são transmitidos ou recebidos por segundo. Ela é medida em baud, que representa a quantidade de símbolos por segundo.

- Ao configurar a comunicação UART em um dispositivo como o ESP32, é necessário definir a taxa de baud apropriada, garantindo que ambos os dispositivos comunicantes estejam configurados com a mesma taxa. Se a taxa de baud não for a mesma em ambos os extremos da comunicação, a transmissão de dados pode ser interpretada incorretamente.

- Portanto, "UART baud rate" é um parâmetro importante a ser configurado ao usar comunicação UART para garantir uma transferência de dados confiável entre dispositivos.

Taxa de transmissão

- velocidade, ou taxa de baud, em uma comunicação UART é medida em baud.
- Os valores comuns para a taxa de baud incluem 9600, 19200, 38400, 57600, 115200, e assim por diante. Esses valores representam a quantidade de símbolos (ou bits de dados) transmitidos ou recebidos por segundo.

- A escolha da taxa de baud depende dos requisitos específicos do sistema e dos dispositivos envolvidos na comunicação. Dispositivos conectados devem ser configurados com a mesma taxa de baud para garantir uma comunicação adequada.

- Por exemplo, se você configurar a taxa de baud como 9600 em um dispositivo, o outro dispositivo também deve ser configurado para 9600 para que ambos possam entender corretamente os dados transmitidos e recebidos. A taxa de baud precisa ser a mesma em ambos os extremos da comunicação para garantir uma transferência de dados bem-sucedida.

Taxas mais baixas:

- **Distâncias mais longas**: Em comunicações seriais, taxas de baud mais baixas são mais tolerantes a interferências e perdas de sinal em distâncias maiores. Se você estiver trabalhando em um ambiente onde os dispositivos estão separados por longas distâncias, pode ser benéfico reduzir a taxa de baud para garantir uma comunicação estável.

- **Limitações de hardware**: Dispositivos mais antigos ou com hardware limitado podem ter restrições em termos de capacidade de processamento. Nesses casos, escolher uma taxa de baud mais baixa pode ser necessário para garantir uma comunicação confiável sem sobrecarregar o hardware.

- **Economia de energia**: Em sistemas alimentados por bateria ou onde a eficiência energética é crítica, usar taxas de baud mais baixas pode ajudar a reduzir o consumo de energia associado à comunicação serial.

Taxas mais altas:

- **Transferência de dados rápida**: Taxas de baud mais altas permitem uma transferência de dados mais rápida entre dispositivos. Isso é útil em situações em que a velocidade de comunicação é crítica, como em aplicações de transmissão de vídeo, transferência de arquivos grandes ou streaming de dados.

- **Comunicação em curtas distâncias**: Para comunicação em curtas distâncias, onde a atenuação do sinal não é um problema significativo, taxas de baud mais altas podem ser escolhidas para obter uma transferência rápida de dados.

- **Alta taxa de amostragem**: Em aplicações que requerem uma alta taxa de amostragem, como leitura de sensores em alta frequência, taxas de baud mais altas podem ser necessárias para garantir uma aquisição precisa e eficiente dos dados.

Comunicação serial UART: introdução

```
void setup()
{
    Serial.begin(9600);
}

void loop()
{
    Serial.printf("Ola mundo\n");
    delay(2000); // dorm 2000ms 2s
}
```

Declaração de variáveis.

```
void setup()
{
}
void loop()
{
    float x = 1.5f;
    double y = 3.2;
    char letra = 'a';
    String texto = "ola mundo";
    int inteiro = 1024;
    long grande = 1000L;
    delay(2000);
}
```

printf

```
void setup()
{
    Serial.begin(9600);
}
void loop()
{
    float x = 1.5f;
    double y = 3.2;
    char letra = 'a';
    String texto = "ola mundo";
    int inteiro = 1024;
    long grande = 1000L;
    Serial.printf("%.2f\n", x); // arredondamento de duas casas
    Serial.printf("%.2lf\n", y); // arredondamento de duas casas
    Serial.printf("%c\n", letra);
    Serial.printf("%s\n", texto);
    Serial.printf("%i\n", inteiro);
    Serial.printf("%ld\n", grande);
    delay(2000);
}
```

print/println – Escreve/ escreve quebra linha

```
void setup()
{
    Serial.begin(9600);
}
void loop()
{
    float x = 1.5f;
    double y = 3.2;
    char letra = 'a';
    String texto = "ola mundo";
    int inteiro = 1024;
    long grande = 1000L;
    Serial.println(x);
    Serial.println(y);
    Serial.println(letra);
    Serial.println(texto);
    Serial.println(inteiro);
    Serial.println(grande);
    Serial.print("\n");
    delay(2000);
}
```

While

```
void setup()
{
    Serial.begin(9600);
}
void loop()
{
    int i = 0;
    while (i < 10)
    {
        Serial.println("Ola mundo");
        i++;
    }
    delay(2000);
}
```

For

```
void setup() {
  Serial.begin(9600);
}
void loop() {

  for(int i=0;i<10;i++){
    Serial.println("Ola mundo :)");

  }
  delay(2000);
}
```

if

```
void setup() {
  Serial.begin(9600);

}
void loop() {
  int x = 1;
  if(x==1){
    Serial.println("entrou :D");
  }

}
```

```
#void setup() {
  Serial.begin(9600);

}
void loop() {
  int x = 1;
  if(x==1){
    Serial.println("entrou :D");
  }else{
     Serial.println("entrou :/");
  }

}
```

elseif

```
void setup() {
 Serial.begin(9600);
}
void loop() {
 int x = 3;
if(x==1){
Serial.println("entrou :D");
 }else if(x==2){
       Serial.println("entrou :/");
 }else{
  Serial.println(":)");
 }
```

Entrada dados serial

```
void setup() {
   Serial.begin(9600);
   }
void loop() {
  String dado="";
  if(Serial.available()){
    dado = Serial.readStringUntil('\n’);
  }
  int x = dado.toInt();
  // float x = dado.toFloat();
  if(dado!=""){
  Serial.println (x*2);
  }
}
```

Funções de tempo

delay(t) - O programa tem uma pausa de t milissegundos (1000 milissegundos = 1 segundo)

delayMicroseconds(t) - O programa tem uma pausa de t microssegundos

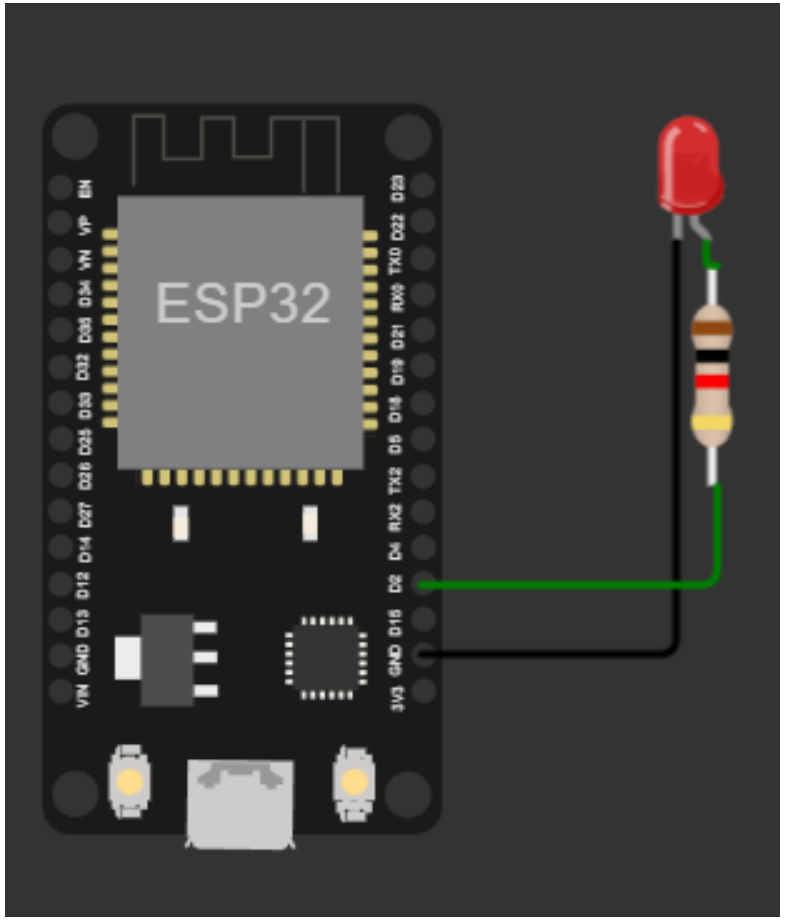
zillis() - Retorna o tempo, em milissegundos, desde que o programa começou a rodar

Saídas pelas portas digitais do esp32

- É preciso configurar a gpio para saída digital.
- Exemplo:
  - pinMode(13, OUTPUT);
- No exemplo o pino 13 é configurado como pino de saída.

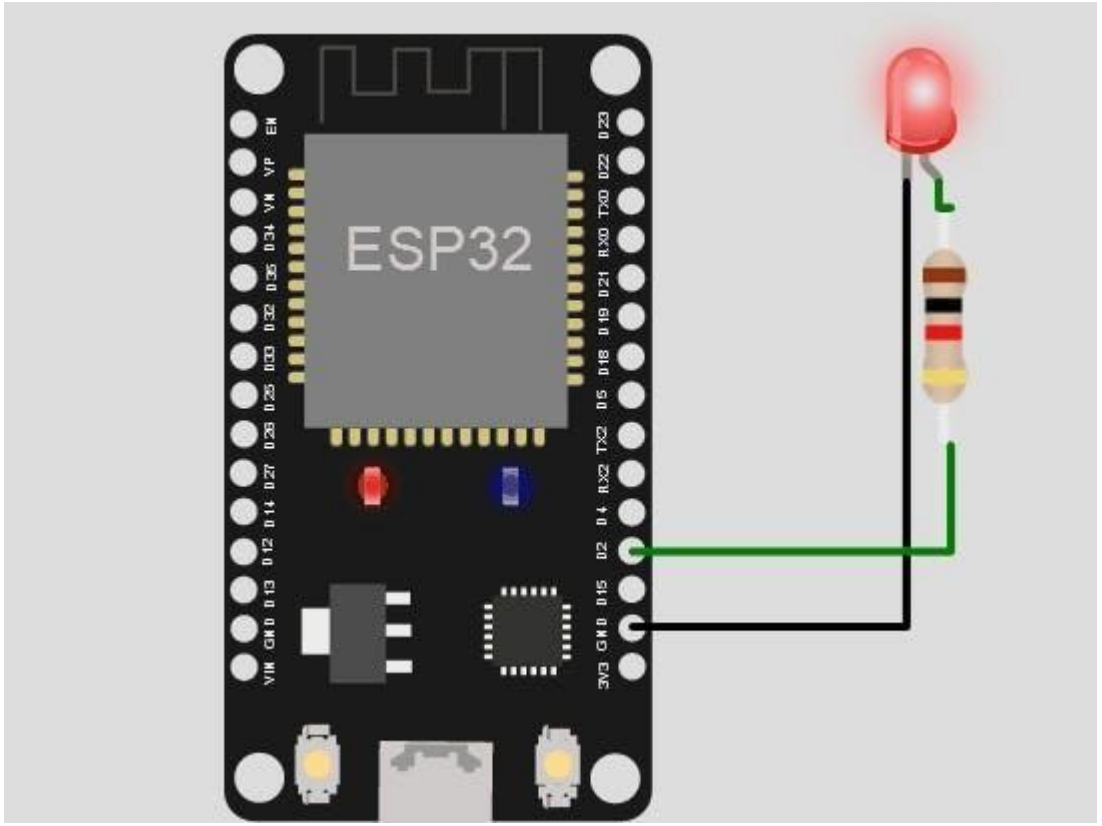
Led comum

```
#define LED 2 // declaração e constante

void setup() {
  pinMode(LED, OUTPUT);
}

void loop() {
  digitalWrite(LED, HIGH);
  delay(500);
  digitalWrite(LED, LOW);
  delay(500);
}
```



Digital write

```
#include <Arduino.h>
#define MEU_LED 2
void setup() {
  Serial.begin(9600);
  pinMode(MEU_LED, OUTPUT);
}
void loop() {
  digitalWrite(MEU_LED,HIGH);
  delay(2000);
  digitalWrite(MEU_LED,LOW);
  delay(2000);

}
```

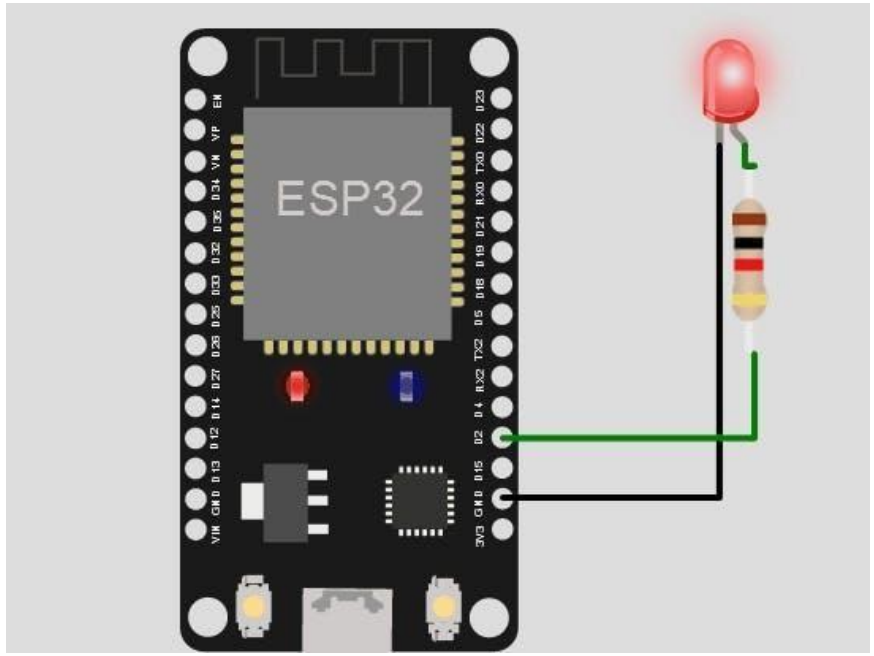


Ler dados pela porta serial UART

- A Função `Serial.available()` verifica se existe dado recebido no buffer.
  - A função retorna verdadeiro ou falso.
- Caso exista algum dado a leitura pode ser feita pela função :
  - Serial.readStringUntil('\n’)
  - Os dados são lidos até encontrar um \n
  - O \n é a quebra de linha(tecla enter no teclado).

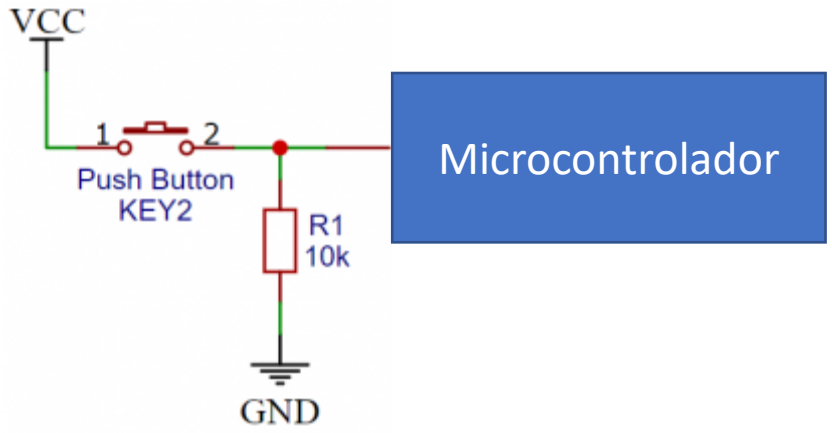
Digital write

```
#include <Arduino.h>
#define MEU_LED 2
void setup() {
  Serial.begin(9600);
  pinMode(MEU_LED, OUTPUT);
}
String dado="";
void loop() {
  if(Serial.available()){
      dado = Serial.readStringUntil('\n');
  }

  if(dado=="7"){
    digitalWrite(MEU_LED,HIGH);
  }
  if(dado=="9"){
    digitalWrite(MEU_LED,LOW);
  }
}
```



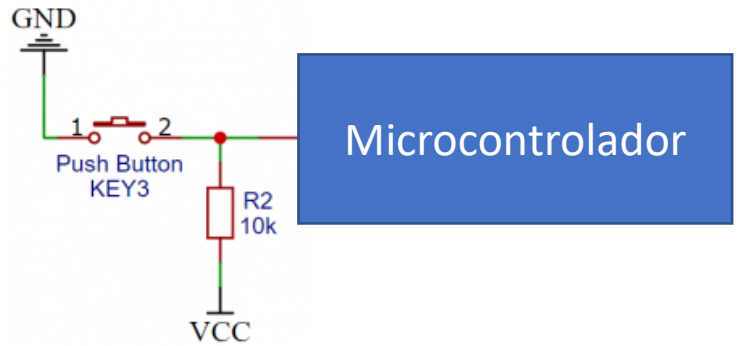
Pull-down

- Quando o botão não estiver pressionado o microcontrolador reconhecerá nível lógico 0 (Gnd)
  - por isso pull-down (puxar para baixo).
- Quando a chave for pressionada, o microcontrolador identificará nível lógico 1 .

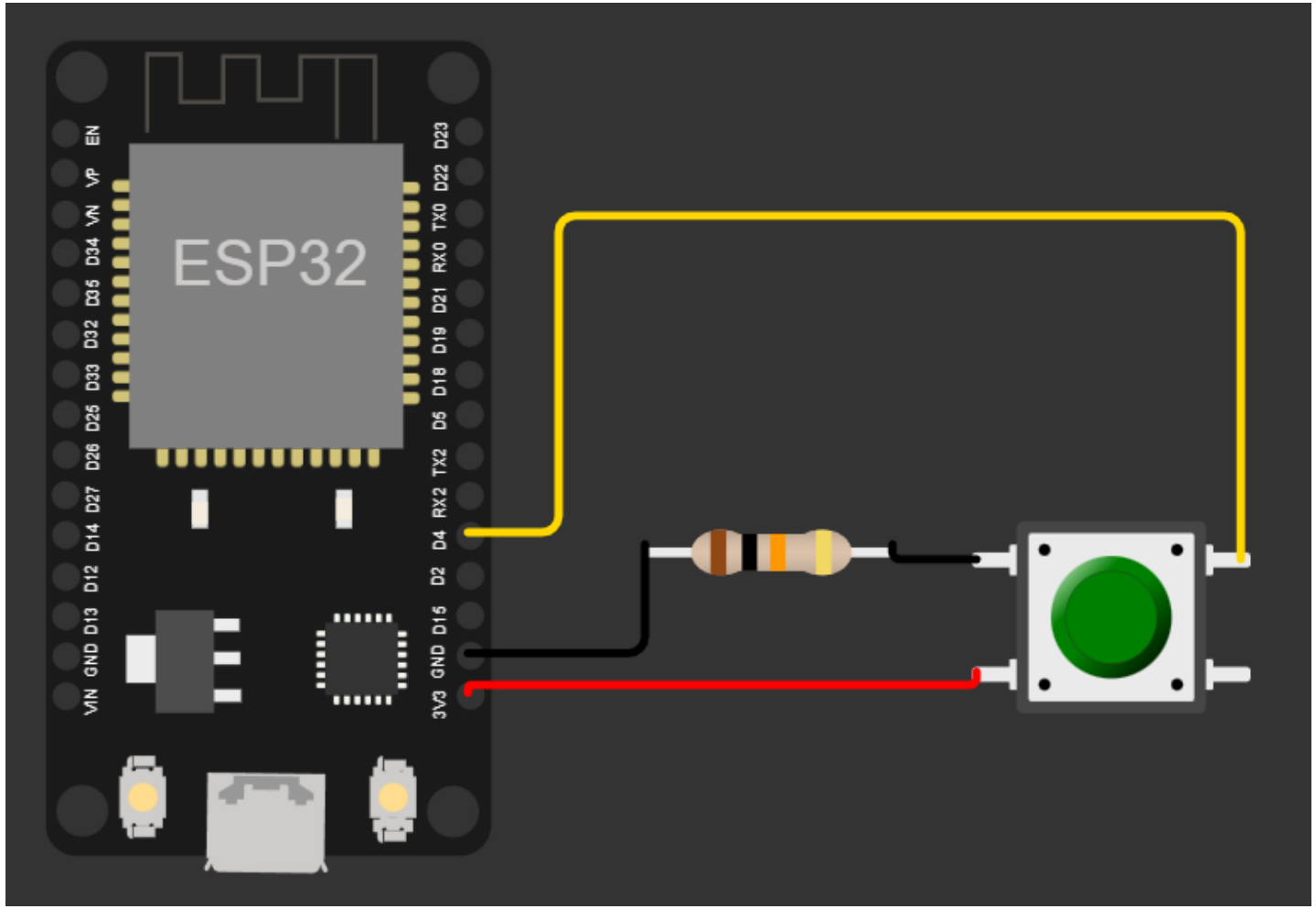


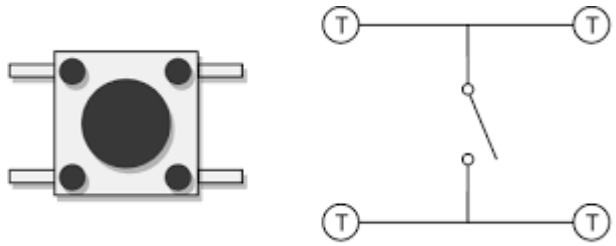
Pull-up

- Quando o botão não estiver pressionado o microcontrolador reconhecerá nível lógico 1
- Por isso pull-up (puxar para cima)
- Quando a chave for pressionada, o microcontrolador reconhecerá nível lógico 0.



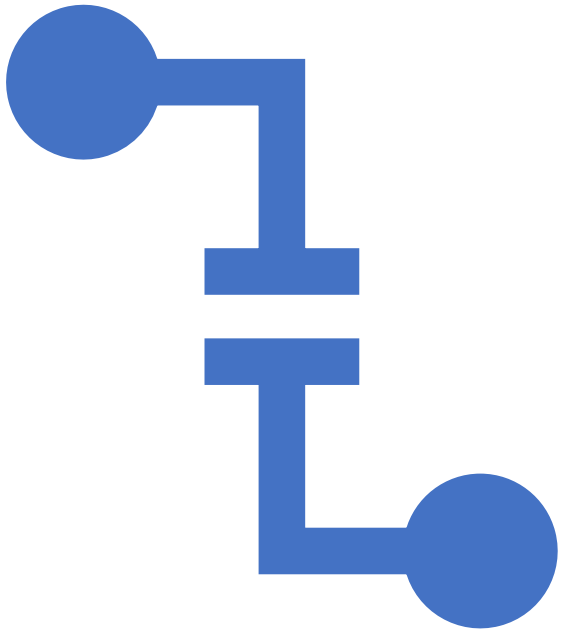
Resistor de 10k ohm pull up ou down?





Código

```
#include <Arduino.h>
#define LED_INTERNO 2
#define BOTAO_1 4
int     estadoBotao = 0;
void setup(){
  pinMode(BOTAO_1, INPUT);
  pinMode(LED_INTERNO, OUTPUT);
}
void loop(){
   estadoBotao = digitalRead(BOTAO_1);
  if (estadoBotao == HIGH) {
                digitalWrite(LED_INTERNO, HIGH);
  } else {
    digitalWrite(LED_INTERNO, LOW);
  }
  delay(10);
}
```



Simulador

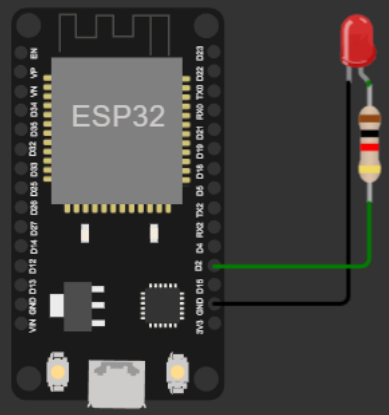
- https://wokwi.com/projects/305566932847821378

WOKWI SAVE SHARE esp32-arduino.ino by urish Docs SIGN UP

esp32-blink.ino | diagram.json | Library Manager

```ino
#define LED 2

void setup() {
  pinMode(LED, OUTPUT);
}

void loop() {
  digitalWrite(LED, HIGH);
  delay(500);
  digitalWrite(LED, LOW);
  delay(500);
}

```

Simulation




```
clk_drv:0x00,q_drv:0x00,d_drv:0x00,cs0_drv:0x00,hd_drv:0x00,wp_drv:0x00
mode:DIO, clock div:2
load:0x3fff0030,len:1156
load:0x40078000,len:11456
ho 0 tail 12 room 4
load:0x40080400,len:2972
entry 0x400805dc
```