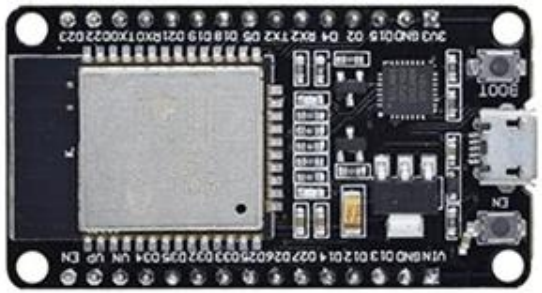
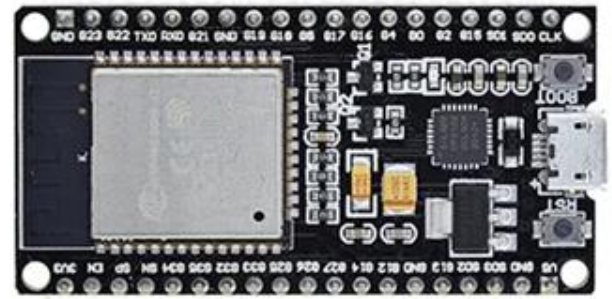
Introdução ao ESP32 DEVKIT

ESP32-WROOM-32E

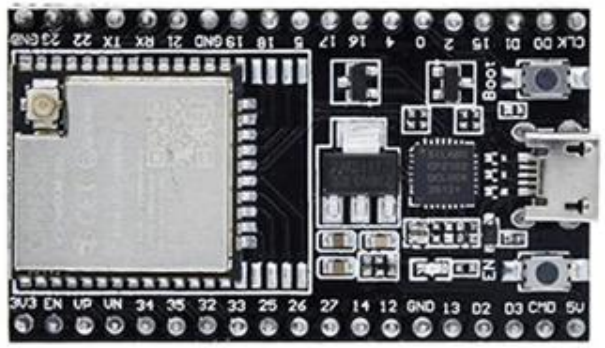
ESP32-WROOM-32UE



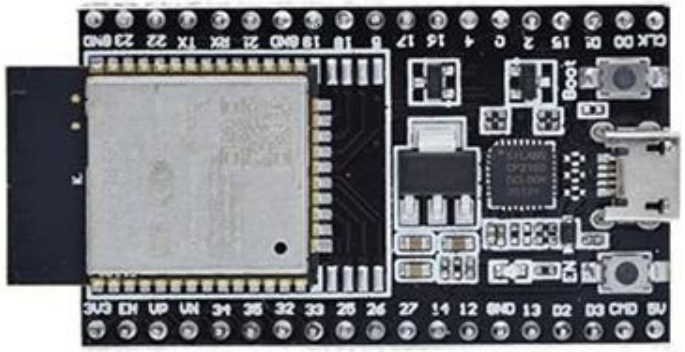
ESP-32 30PIN



ESP-32 38PIN



ESP32-WROOM-32U



ESP32-WROOM-32D

Comparação

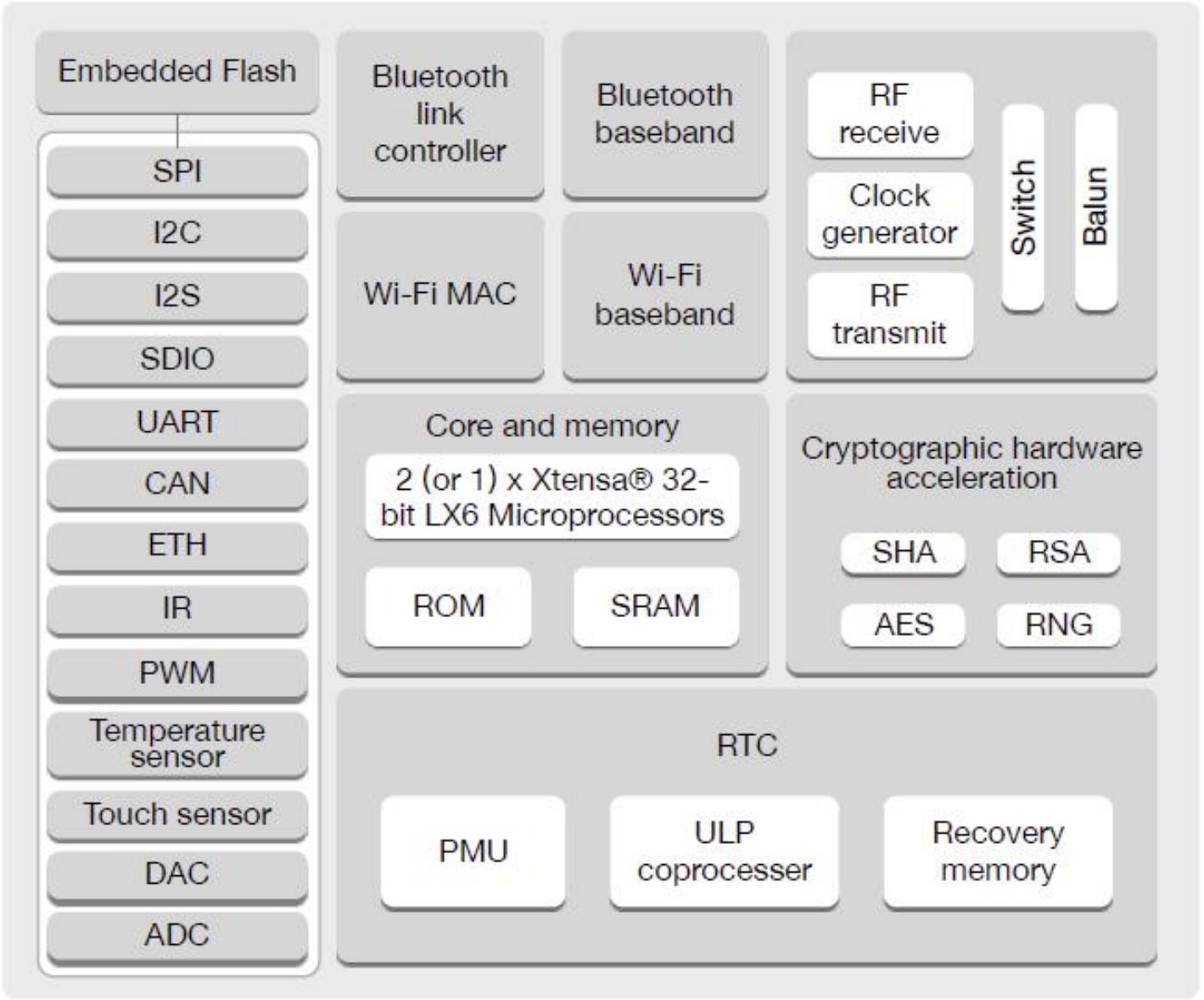
ESP8266 vs ESP32 vs ESP32-S2

FEATURES	ESP8266	ESP32
Release Year	2014	2016
Microcontroller	Xtensa single-core 32-bit L106	Xtensa single/dual-core 32-bit
Clock Frequency	80 MHz	160/240 MHz
Co-processor	✖	ULP
SRAM	160KB	520KB
RTC Memory	✖	16KB
External SPIRAM	Up to 16MB	Up to 16MB
External Flash	✖	✖
Wi-Fi (802.11 b/g/n)	HT20	HT20
ESP-MESH	✔	✔
Bluetooth	✖	BT 4.2, BR/EDR, BLE
Ethernet	✖	10/100 Mbps
CAN	✖	2
Time of Flight	✖	✖
GPIO (total)	16	34
Touch Sensors	✖	10
SPI	2	4
I2C	1 (soft)	2
I2S	2	2
UART	2 (1.5 actually)	3
ADC	1(10-bit)	18 (12-bit)
DAC	✖	2 (8-bit)
PWM (soft)	8	16
SDMMC	✖	✔
USB OTG	✖	✖
LCD Interface	✖	✖
Camera Interface	✖	✖
Temperature Sensor	✖	✔
Hall sensor	✖	✔
Security	✖	Secure boot Flash encryption 1024-bit OTP
Crypto	✖	AES, SHA-2, RSA, ECC, RNG
Low Power Consumption	20uA	10uA deep sleep

Links

- https://www.espressif.com/
- https://github.com/espressif
- https://en.wikipedia.org/wiki/ESP32
- https://www.espressif.com/en/products/socs/esp32
- https://docs.espressif.com/projects/esp-idf/en/latest/esp32/hw-reference/esp32/get-started-devkitc.html
- https://www.espressif.com/sites/default/files/documentation/esp32-wroom-32_datasheet_en.pdf
- https://www.espressif.com/en/products/devkits

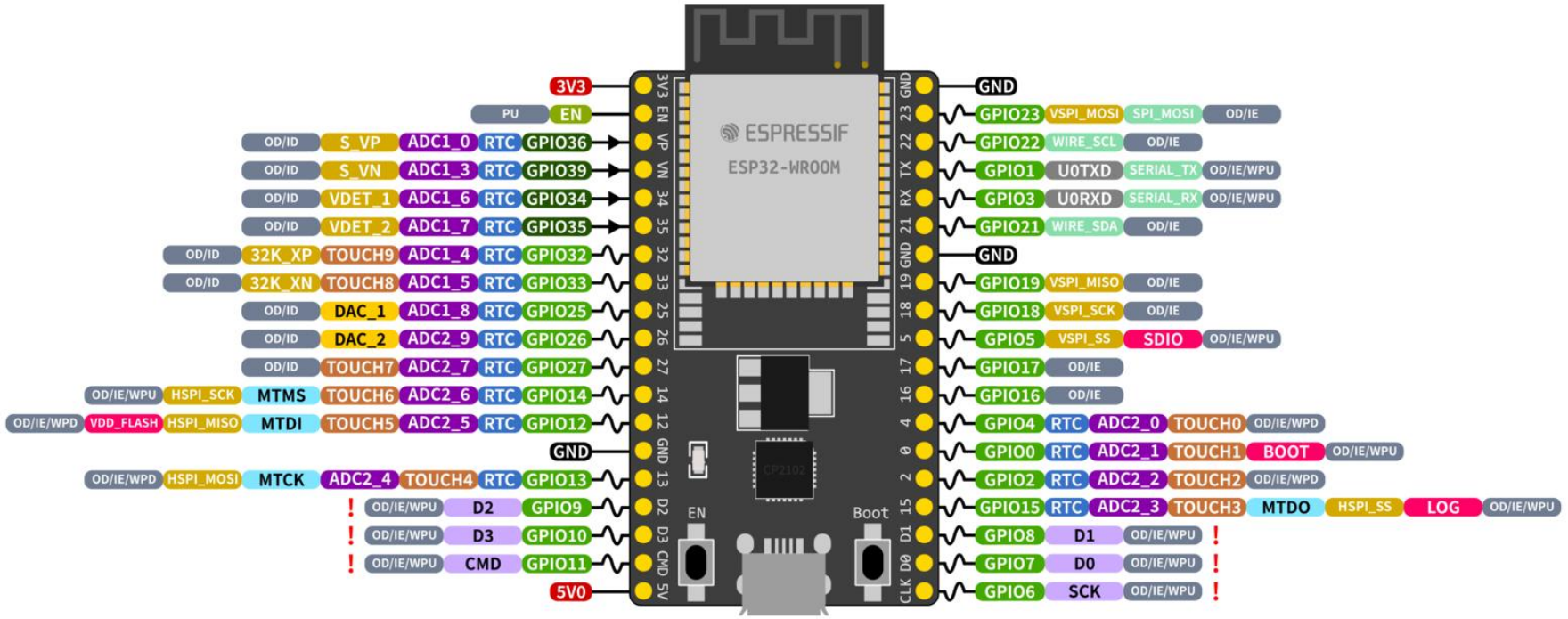
Diagrama de blocos



Pinagem (Pinout)

ESP32-DevKitC

ESPRESSIF



ESP32 Specs

32-bit Xtensa® dual-core @240MHz
Wi-Fi IEEE 802.11 b/g/n 2.4GHz
BLuetooth 4.2 BR/EDR and BLE
520 KB SRAM (16 KB for cache)
448 KB ROM
34 GPIOs, 4x SPI, 3x UART, 2x I2C,
2x I2S, RMT, LED PWM, 1 host SD/eMMC/SDIO,
1 slave SDIO/SPI, TWAI®, 12-bit ADC, Ethernet

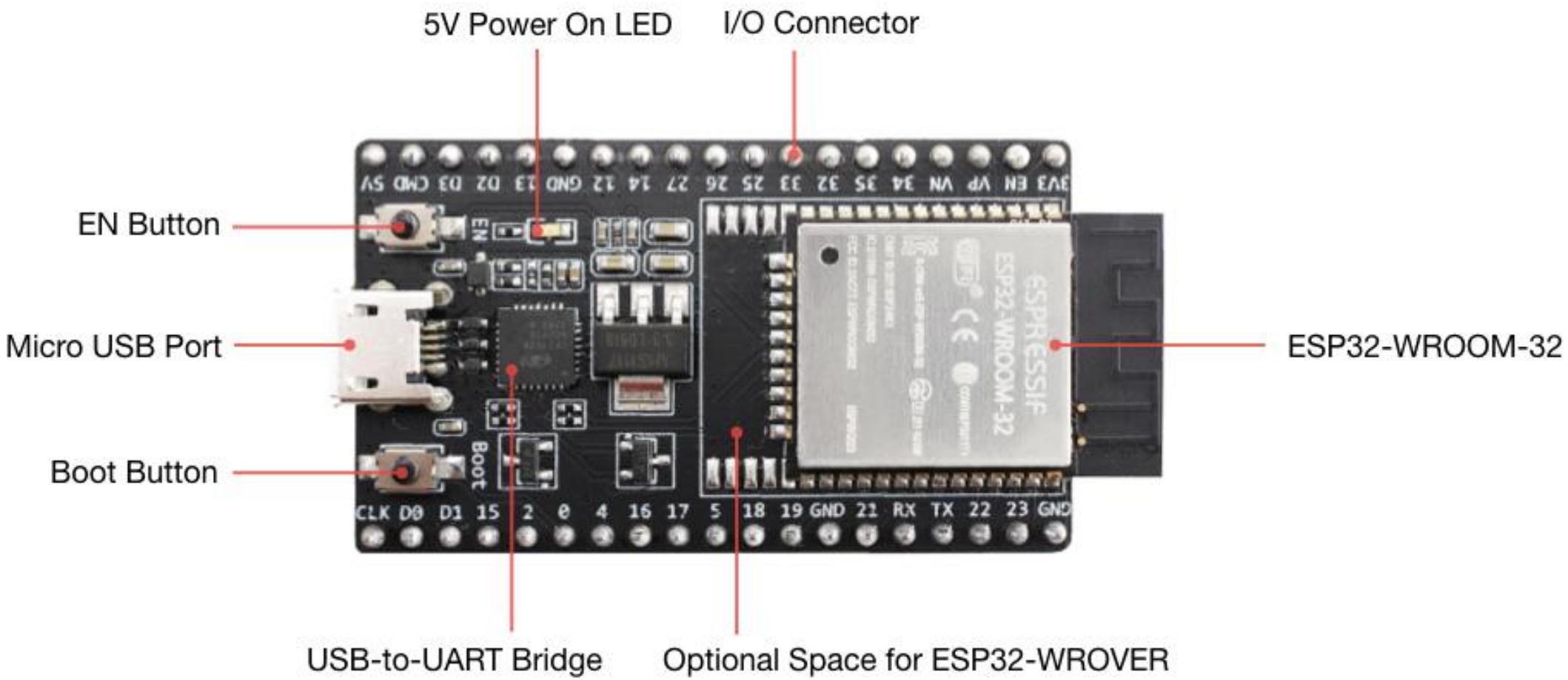
PWM Capable Pin
GPIOX GPIO Input Only
GPIOX GPIO Input and Output
DAC_X Digital-to-Analog Converter
DEBUG JTAG for Debugging
FLASH External Flash Memory (SPI)
ADCX_CH Analog-to-Digital Converter
TOUCHX Touch Sensor Input Channel
OTHER Other Related Functions
SERIAL Serial for Debug/Programming
ARDUINO Arduino Related Functions
STRAP Strapping Pin Functions

RTC RTC Power Domain (VDD3P3_RTC)
GND Ground
PWD Power Rails (3V3 and 5V)
! Pin Shared with the Flash Memory *Can't be used as regular GPIO*

GPIO STATE

WPU: *Weak Pull-up (Internal)*
WPD: *Weak Pull-down (Internal)*
PU: *Pull-up (External)*
IE: *Input Enable (After Reset)*
ID: *Input Disabled (After Reset)*
OE: *Output Enable (After Reset)*
OD: *Output Disabled (After Reset)*

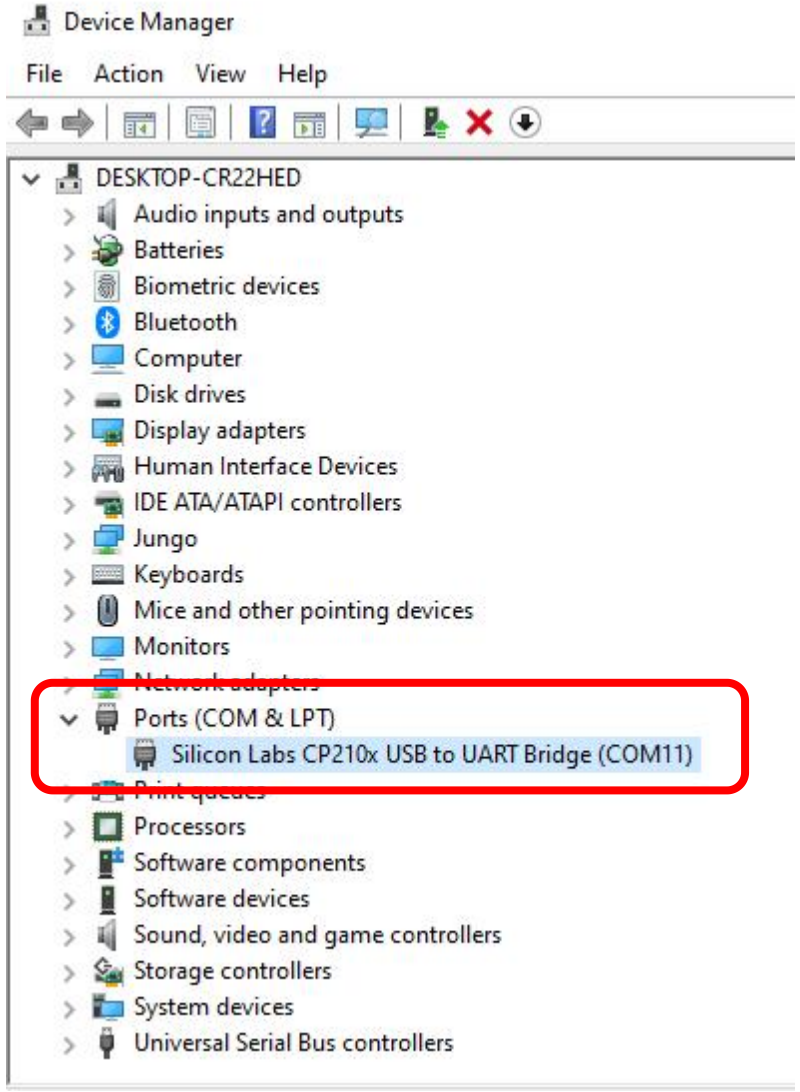
Placa


5V Power On LED
I/O Connector
EN Button
Micro USB Port
Boot Button
ESP32-WROOM-32
USB-to-UART Bridge
Optional Space for ESP32-WROVER

Instalando o driver CP210x

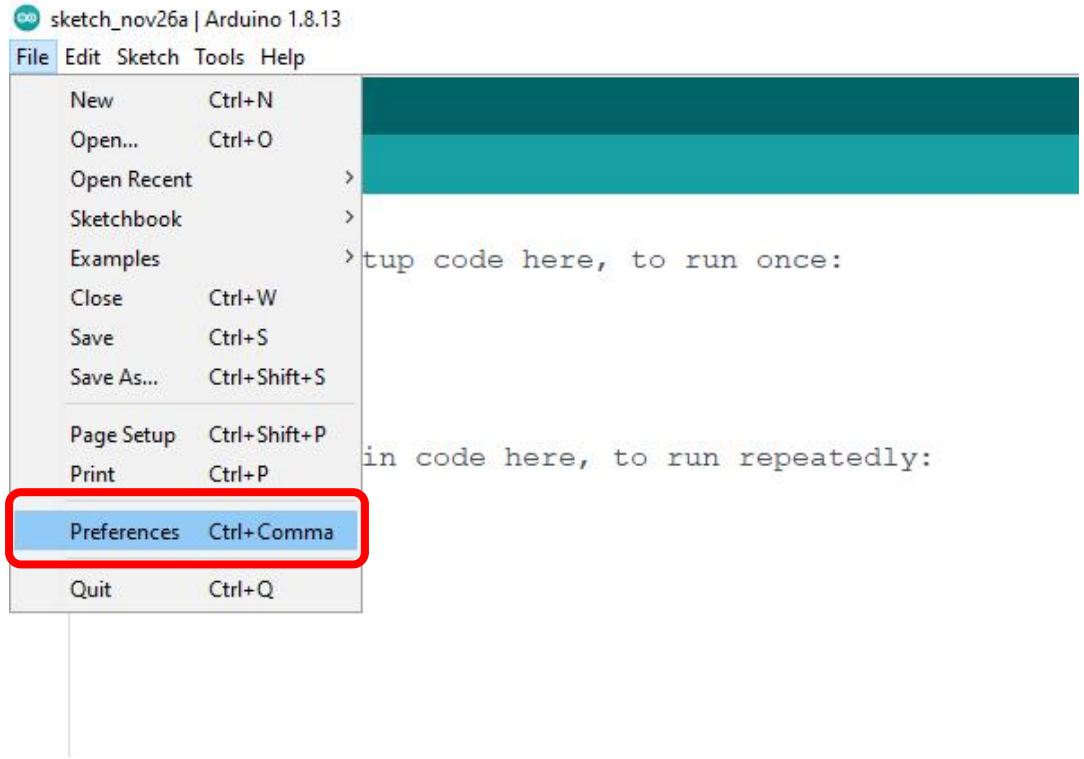
- Conecte o ESP32 na porta USB e veja se o pc reconheceu a porta USB.

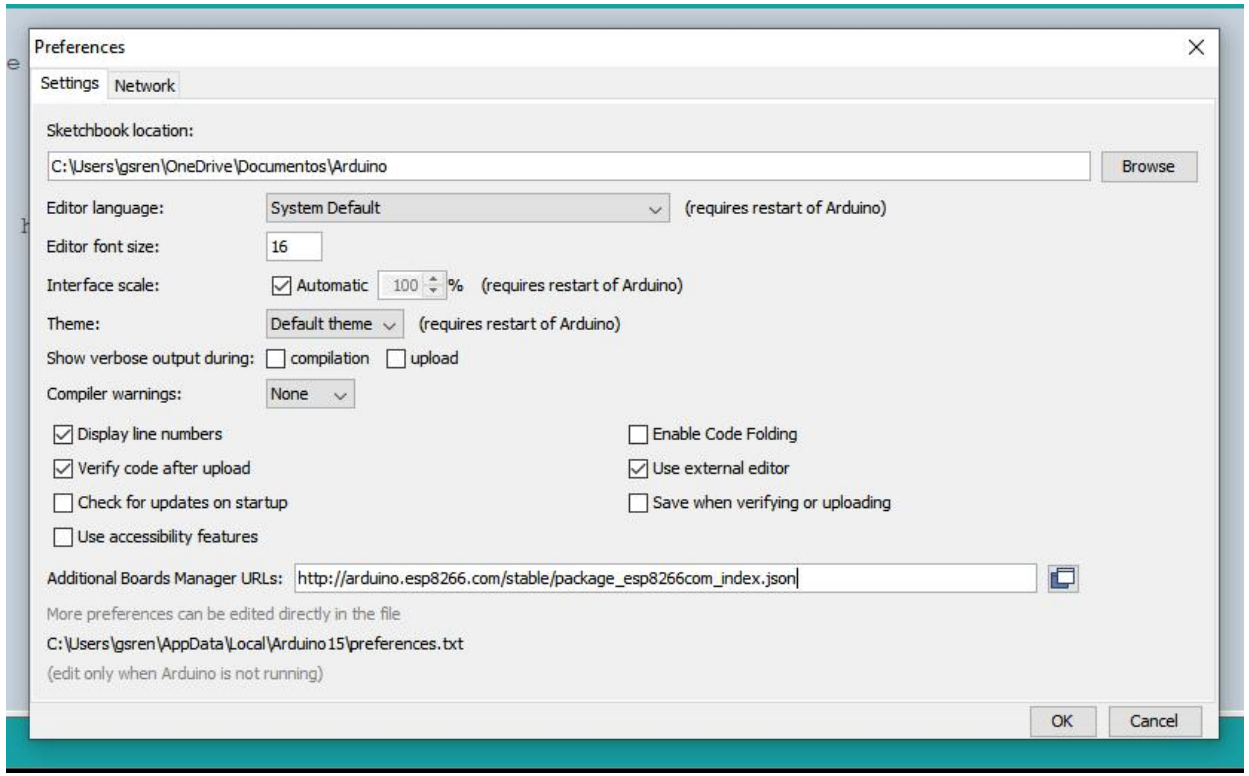
- Se não estiver mostrando a porta COM, instale o driver CP210x.



Instalando o ESP32 no Arduino IDE

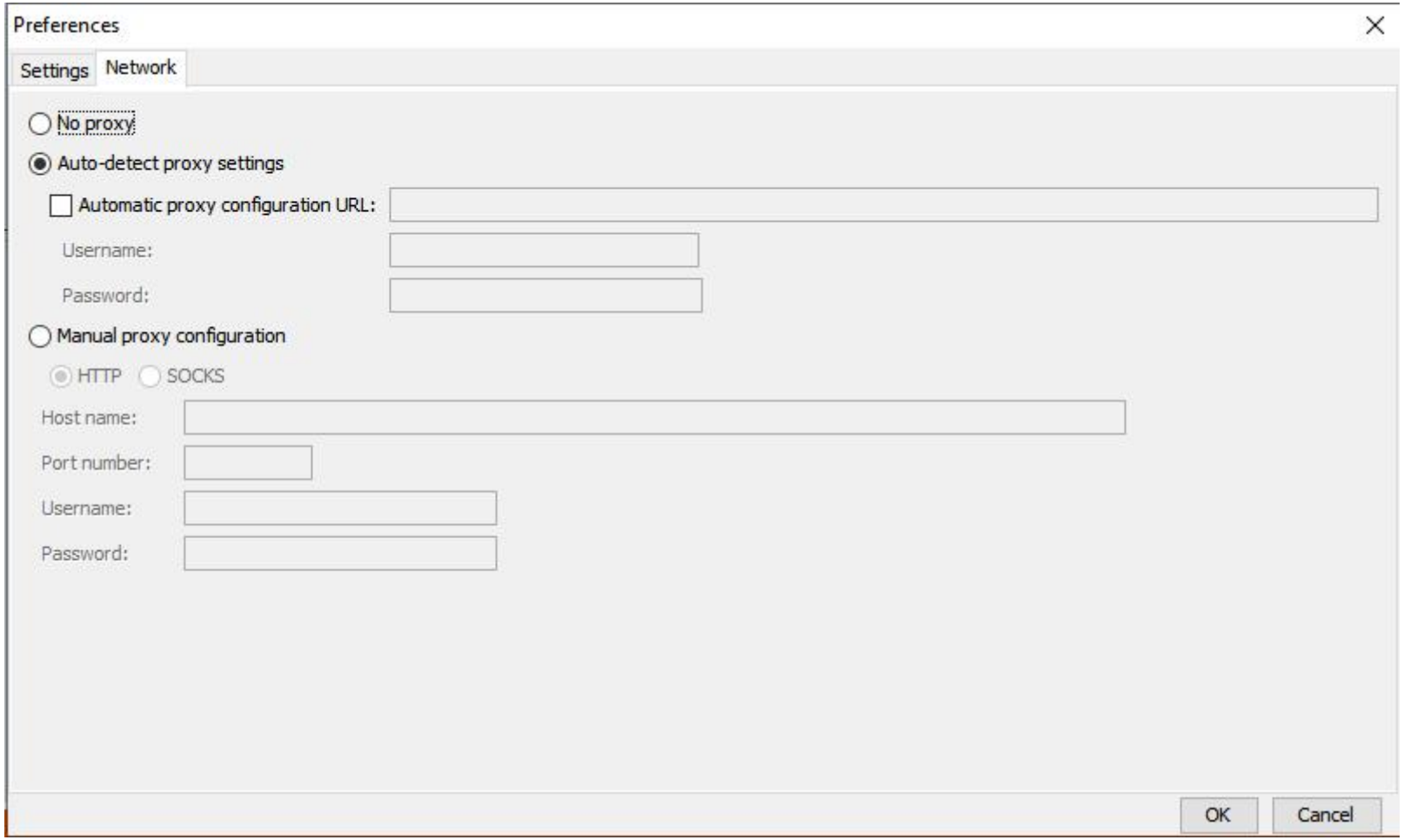
- Abra o Arduino IDE.
- Cole o link na area de preferências:
  - https://raw.githubusercontent.com/espressif/arduino-esp32/gh-pages/package_esp32_index.json
  - https://dl.espressif.com/dl/package_esp32_index.json





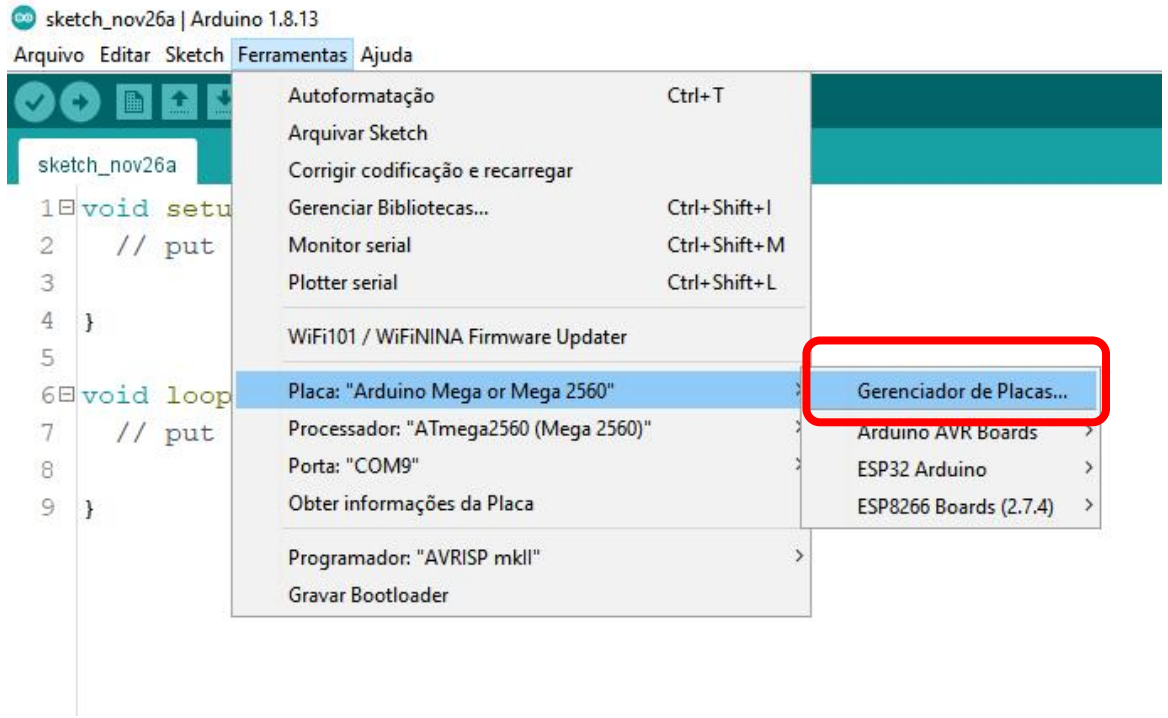
Instalando o ESP32 no Arduino IDE

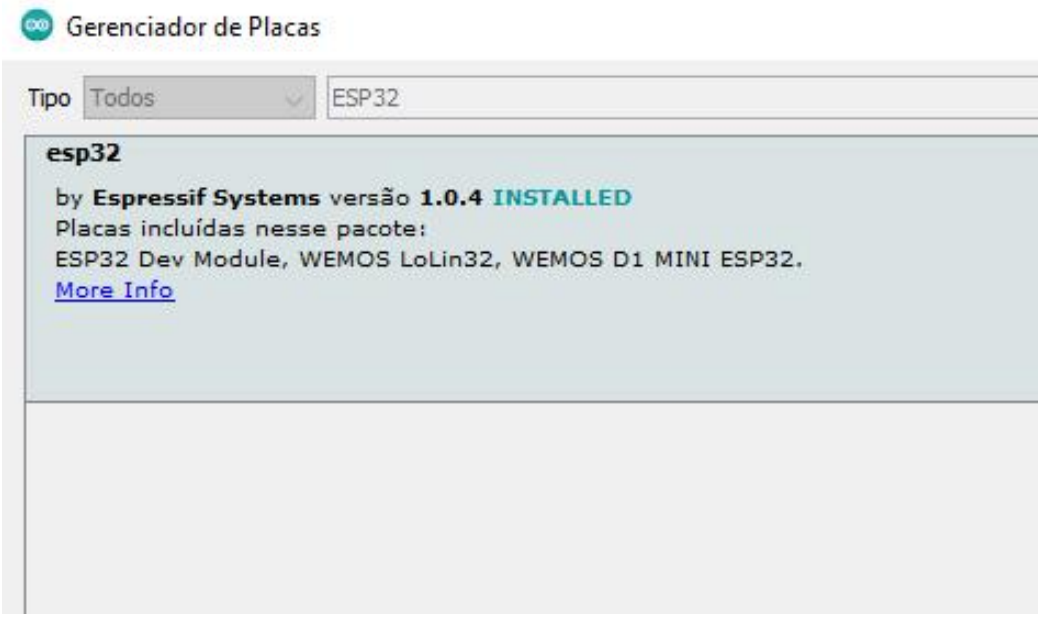
- Na aba Preferencias > Network configure o proxy como auto-detect.



Instalando o SDK (software development kit) no Arduino IDE

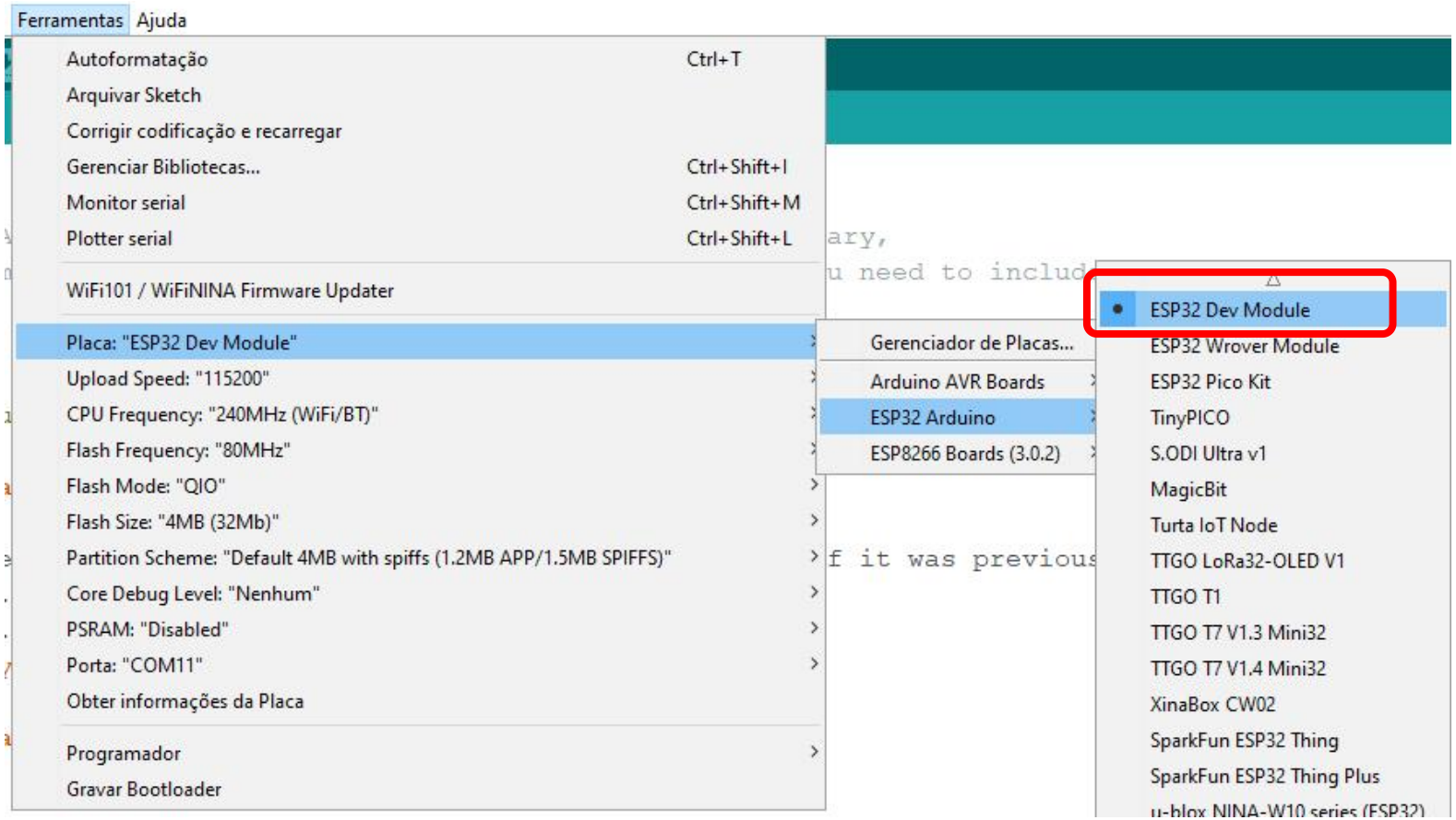
- Na aba Ferramentas > placa > Gerenciador de Placas.
- Digite ESP32 na busca e instale a biblioteca.



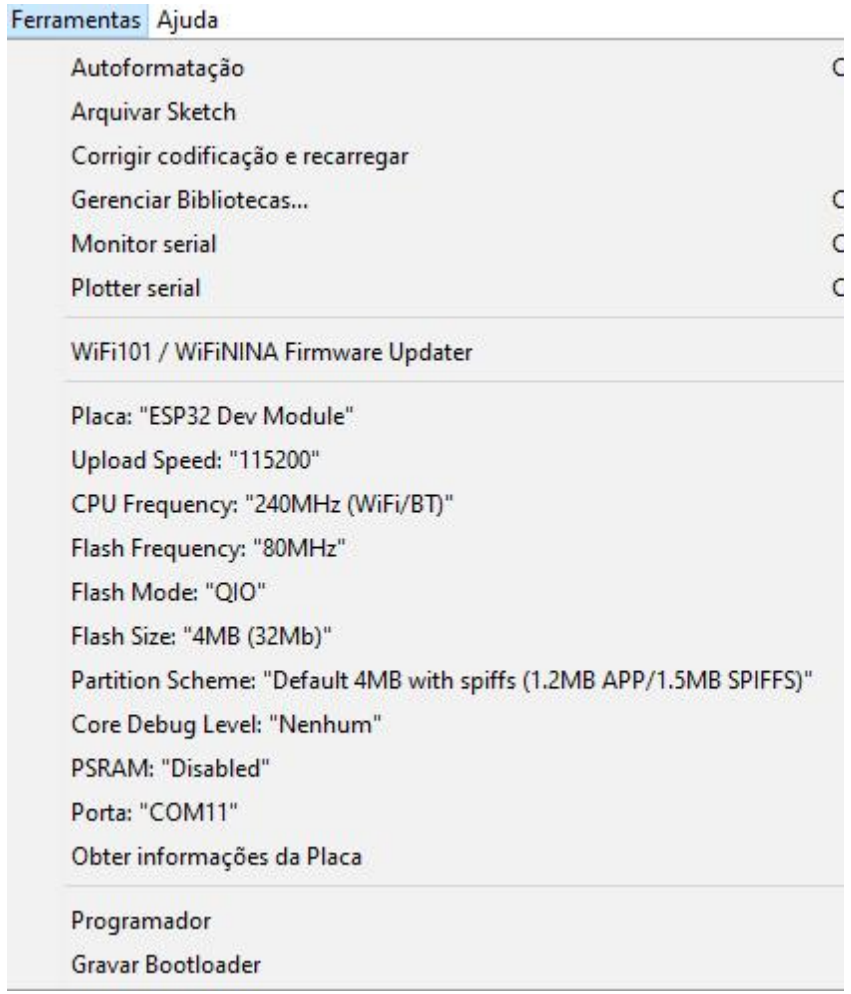


Instalando o SDK (software development kit) no Arduino IDE

- Selecione a placa ***ESP32 Dev Module***



- Configuração



Instalando esp32 no linux

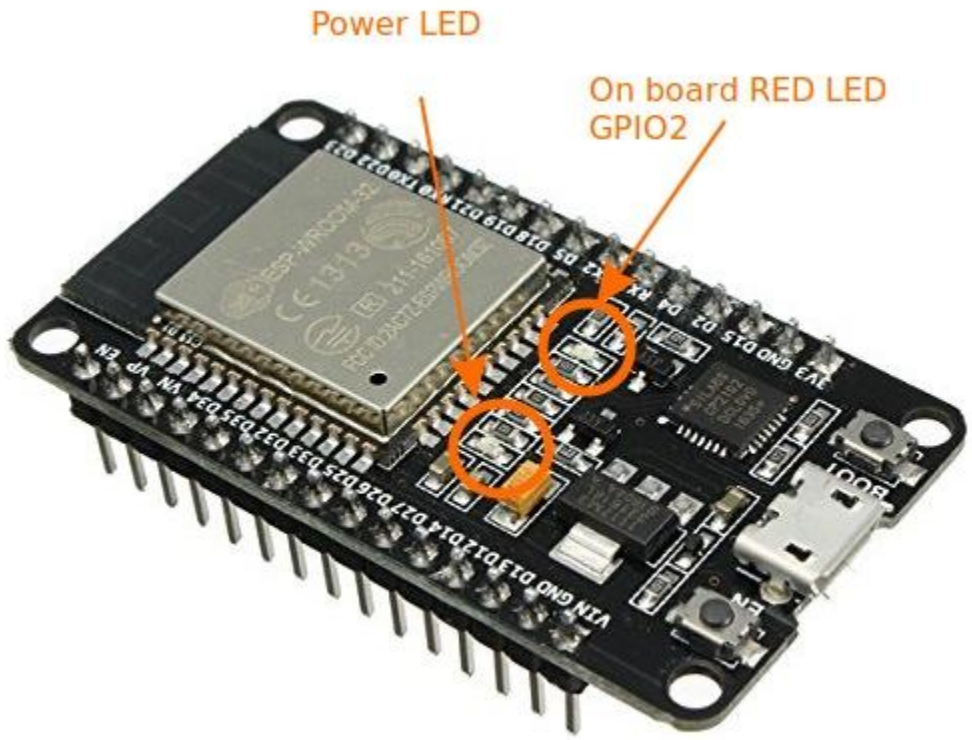
- se ocorre o erro de instalação da placa, copie o link que aparece na mensagem de erro.

- ***se ocorrer erro de compilação, instale a biblioteca pyserial:***
  - sudo apt install python3-pip
  - sudo pip3 install pyserial

Primeiro programa: Blink

- O led da placa pode estar conectado ao GPIO 2.

```
#define LED 2

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

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



Conectando ao Wi-Fi

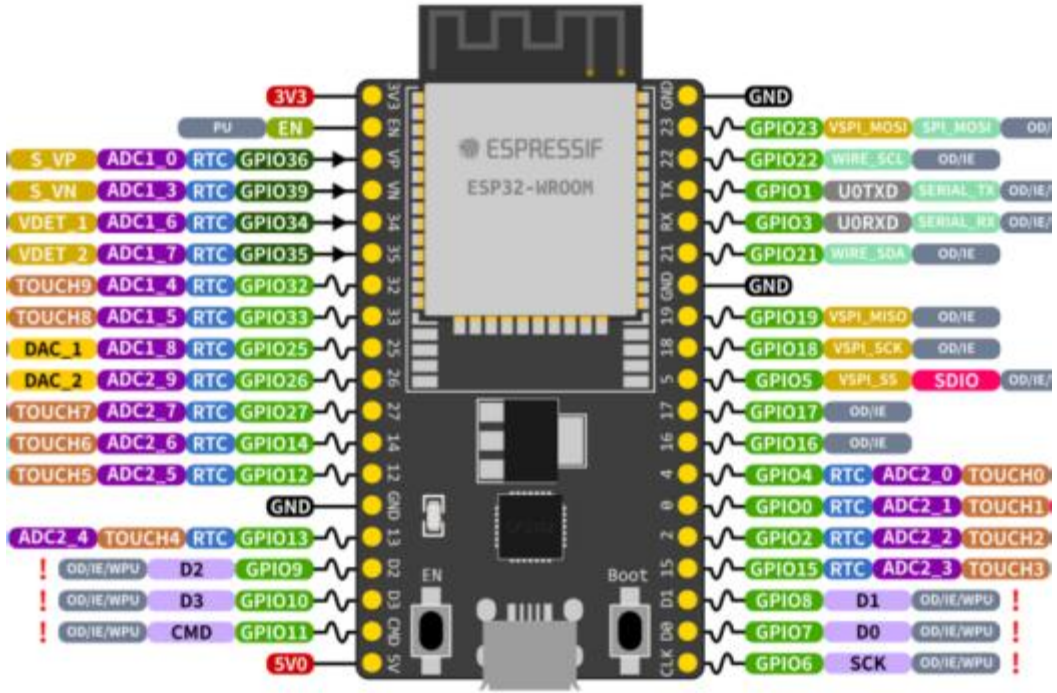
- Utilizando a biblioteca <***WiFi.h***>
- A biblioteca é dividida em classes. Cada classe possui uma função específica.

Sensor Touch

- função ***touchRead(GPIO)***

```
const int touchPin = 4;
const int led = 2;
const int ref = 30;

void setup() {
  Serial.begin(115200);
  pinMode (led, OUTPUT);
}

void loop() {
  int val = touchRead(touchPin);
  Serial.print(val);

  digitalWrite(led, LOW);
  Serial.println(" led desligado");

  if (val < ref) {
    digitalWrite(led, HIGH);
    Serial.println(" led ligado");
  }
  delay(100);
}
```

```
const int touchPin = 4;

void setup() {
  Serial.begin(115200);
 }
void loop() {
  int val = touchRead(touchPin);
  Serial.print(val);
}
```



Sensor de temperatura Built-in

- range: -40 a 125 °C.
- utilizado para medir a temperatura do núcleo.
- Problemas: sujeito às variações de temperatura da placa.
- Ideal para aplicações didáticas ou para detectar variações de temperatura.

```
#ifdef __cplusplus
extern "C" {
#endif
uint8_t temprature_sens_read();
#ifdef __cplusplus
}
#endif
uint8_t temprature_sens_read();

void setup() {
  Serial.begin(115200);
}

void loop() {
  Serial.print("Temperature: ");
  Serial.print((temprature_sens_read() - 32) / 1.8);//Fahrenheit para Celsius
  Serial.println(" C");
  delay(1000);
}
```

Sensor hall Built-in

- leituras de campos magnéticos positivos e negativos.
- Problemas: Este sensor deve ser utilizados para detectar grandes mudanças repentinas no campo magnético.

- função ***hallRead()***

```
void setup() {
  Serial.begin(115200);
}

void loop() {
  int val = hallRead();
  Serial.println(val);
  delay(500);
}
```

Deep Sleep

- O consumo padrão é de 240 mA. Dependendo do projeto pode chegar a 1 A. inviavel para bateria.

- Existem 4 modos de Deep Sleep:
  - **Modem Sleep:** desativa o rádio (WiFi e Bluetooth).
  - **Light Sleep**: pausa a CPU.
  - **Deep Sleep**: desativa periféricos, rádio, núcleos, exceto o co-processador e RTC. Toda informação na RAM é perdida.
  - **Hibernation**: apenas o timer do RTC e alguns GPIO do RTC permanecem ativos.

Power mode	Description	Power consumption
Active (RF working)	Wi-Fi Tx packet 13 dBm ~ 21 dBm	160 ~ 260 mA
	Wi-Fi / BT Tx packet 0 dBm	120 mA
	Wi-Fi / BT Rx and listening	80 ~ 90 mA
	Association sleep pattern (by Light-sleep)	0.9 mA@DTIM3, 1.2 mA@DTIM1
Modem-sleep	The CPU is powered on.	Max speed: 20 mA
		Normal speed: 5 ~ 10 mA
		Slow speed: 3 mA
Light-sleep	-	0.8 mA
Deep-sleep	The ULP co-processor is powered on.	0.15 mA
	ULP sensor-monitored pattern	25 μA @1% duty
	RTC timer + RTC memory	10 μA
Hibernation	RTC timer only	5 μA
Power off	CHIP_PU is set to low level, the chip is powered off	<0.1 μA

Deep Sleep

```
RTC_DATA_ATTR static int contador = 0;

void setup() {
  Serial.begin(115200);
  pinMode(2, OUTPUT);
  contador++;
  Serial.printf("\nNumero de boots: %d\n", contador); //"printf" somente no ESP
}
void loop() {
  //código
  digitalWrite(2, HIGH);
  delay(1000);// o delay deve ser suficiente para excutar todas as funcoes antes do Deep Sleep

  //deep sleep
  const int deep_sleep_sec = 5;
  ESP_LOGI(TAG, "Entering deep sleep for %d seconds", deep_sleep_sec);
  esp_deep_sleep(1000000LL * deep_sleep_sec);
}
```

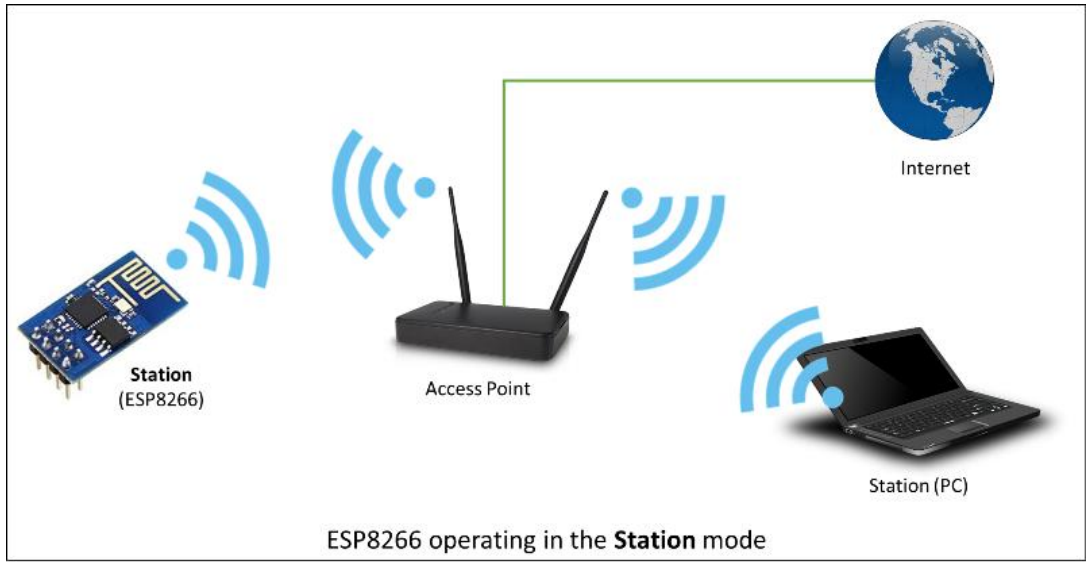
RTC Real Time Clock

- utiliza o padrão UNIX, que conta os segundos desde 01/01/1970 - https://www.unixtimestamp.com/
- Biblioteca **NTPClient.h**, Network Time Protocol, atualiza o tempo pela rede.

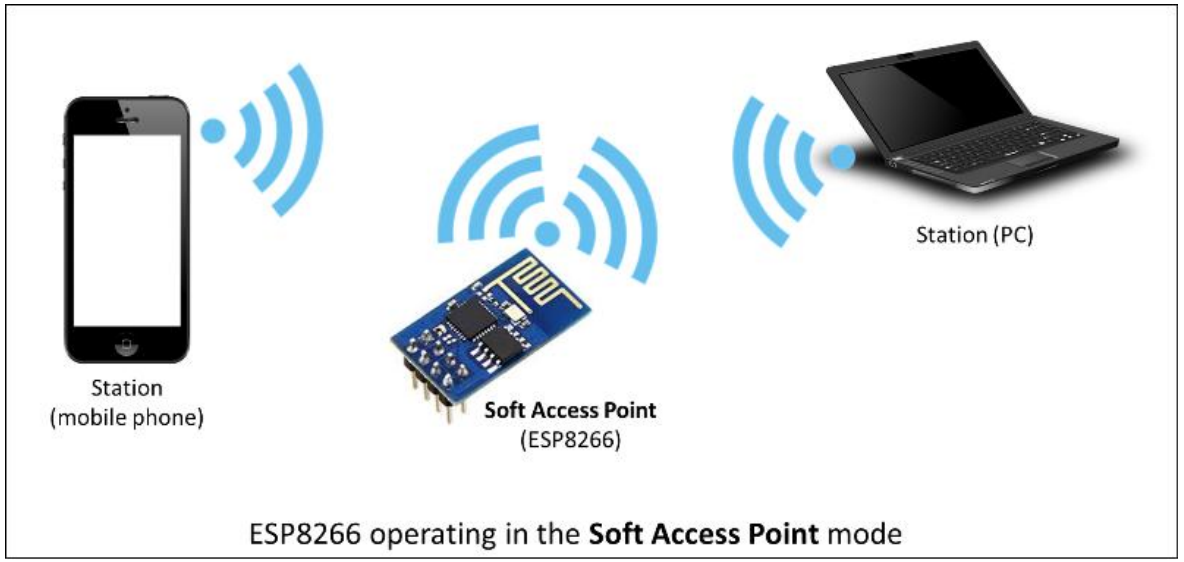
```
#include <WiFi.h>
#include <NTPClient.h>
#include <WiFiUdp.h>
const char *ssid     = "NET_2GF5B3F7";
const char *password = "51F5B3F7";
WiFiUDP ntpUDP;
NTPClient ntp(ntpUDP);
void setup(){
  Serial.begin(115200);
  WiFi.begin(ssid, password);
  while ( WiFi.status() != WL_CONNECTED ) {
    delay ( 500 );
    Serial.print ( "." );
  }
  ntp.begin();
   // -3 = -10800 (BRASIL)
  ntp.setTimeOffset(-10800);
}
void loop() {
  ntp.update();
  Serial.println(ntp.getFormattedTime());
  delay(1000);
}
```

Conectando ao Wi-Fi

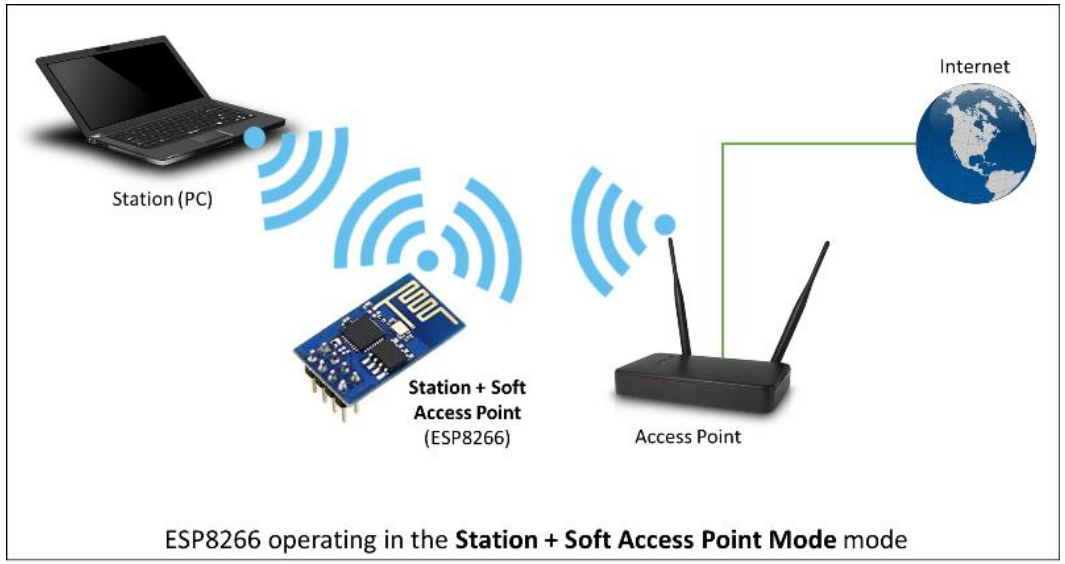
- O ESP32 pode concetar à internet como Estação, Ponto de Acesso ou ambos.



ESP8266 operating in the **Station** mode



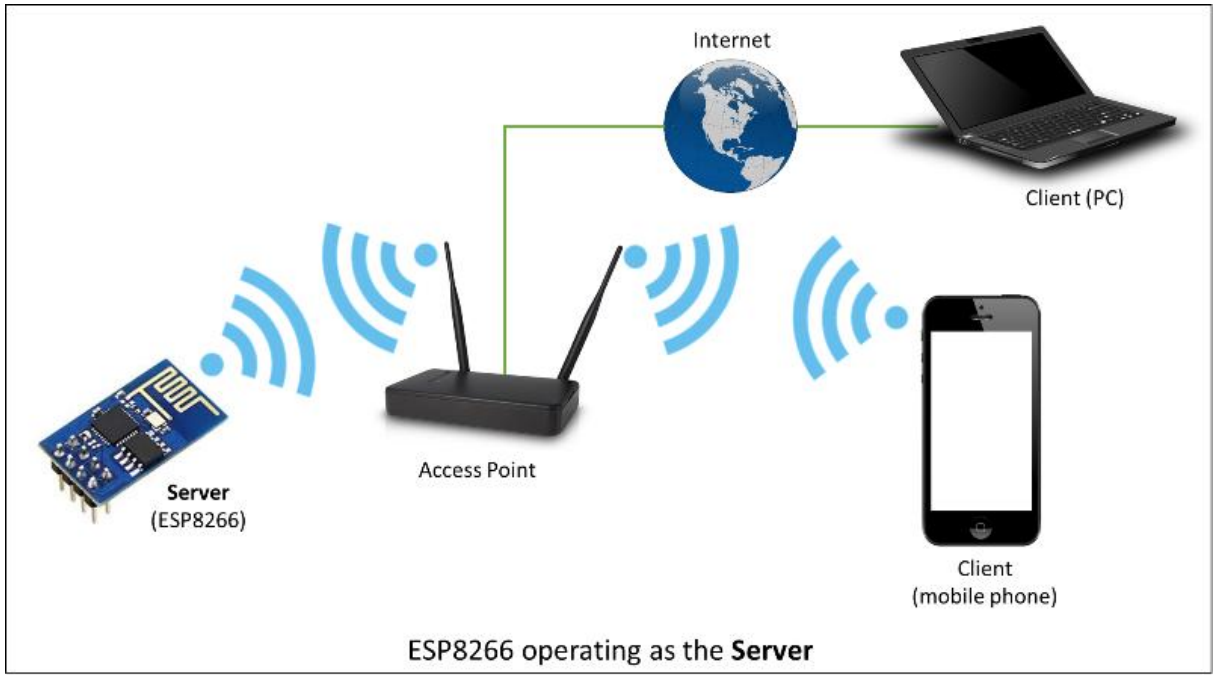
ESP8266 operating in the **Soft Access Point** mode



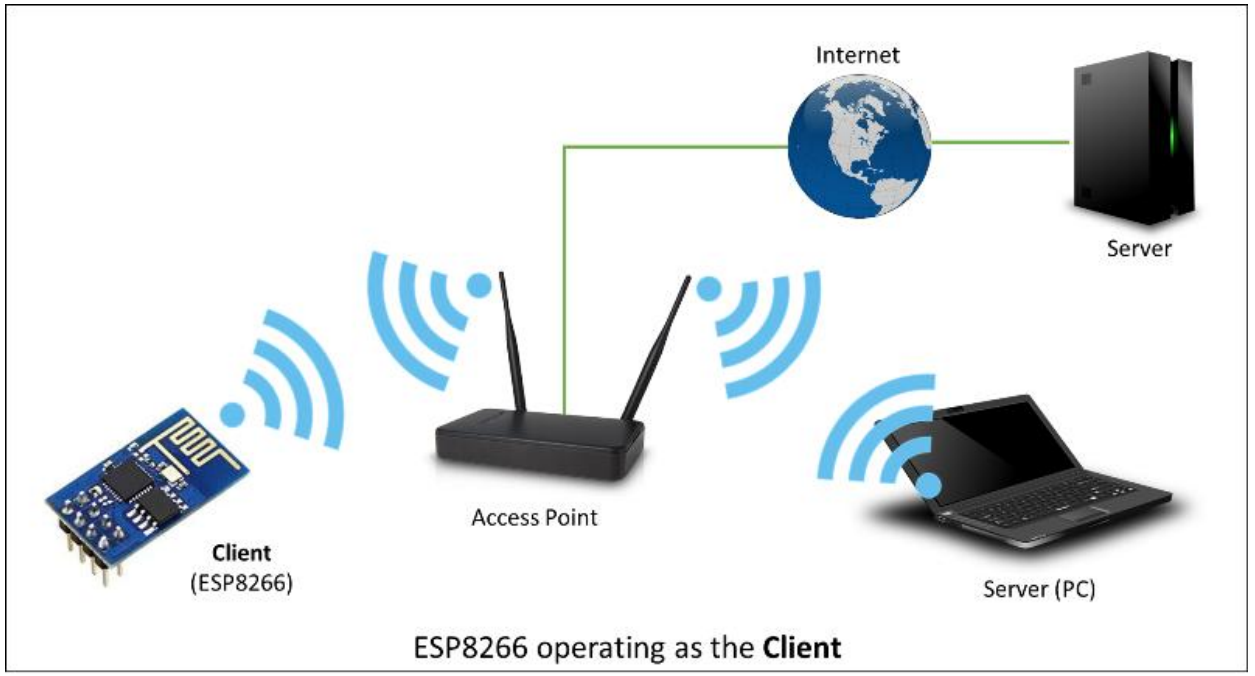
ESP8266 operating in the **Station + Soft Access Point Mode** mode

Conectando ao Wi-Fi

- O ESP32 também pode ser configurado como cliente ou servidor.



ESP8266 operating as the **Server**



ESP8266 operating as the **Client**

Atividades

1. Escaneando as redes disponíveis.
2. Conectando à rede.
3. Controle de led Web.
4. sensor touch
5. sensor hall
6. sensor de temperatura
7. Deep Sleep
8. RTC