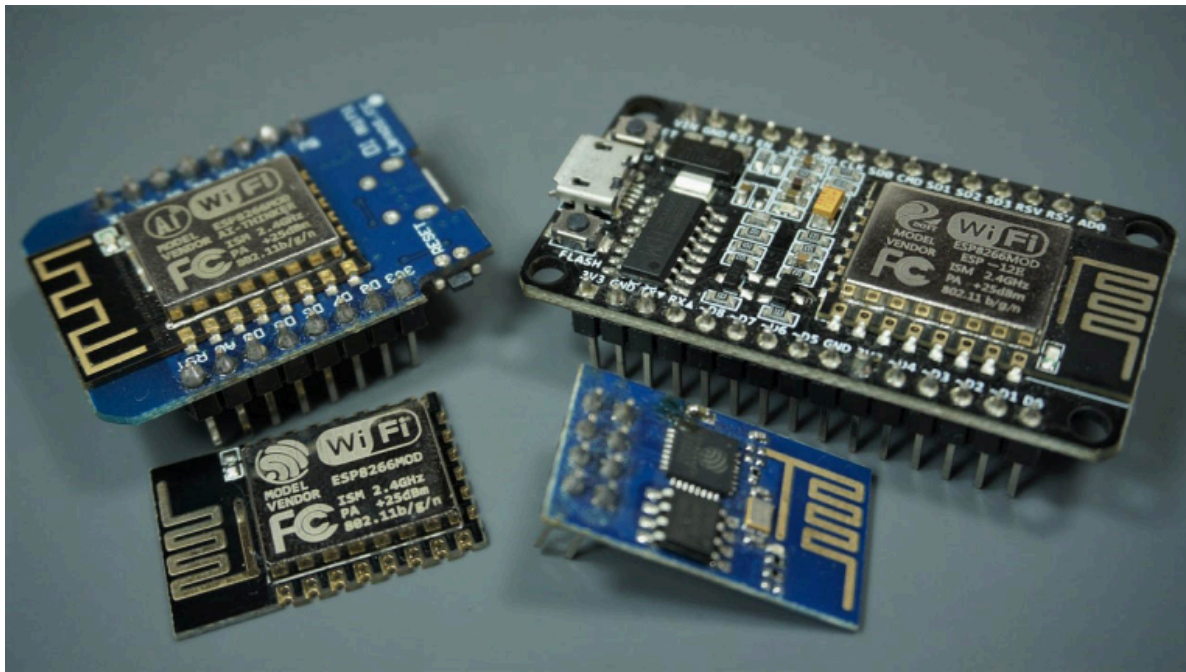


HOME ESP32 ESP8266 ESP32-CAM RASPBERRY PI MICROPYTHON RPi PICO ARDUINO  
REVIEWS

# ESP8266 Pinout Reference: Which GPIO pins should you use?

This article is a guide for the ESP8266 GPIOs: pinout diagrams, their functions and how to use them.



**Affiliate Disclosure:** *Random Nerd Tutorials* is a participant in affiliate advertising programs designed to provide a means for us to earn fees by linking to Amazon, eBay, AliExpress, and other sites. We might be compensated for referring traffic and business to these companies.



[Learn ESP32 with Arduino IDE \(2nd Edition\) Course »](#) Complete guide to program the ESP32 with Arduino IDE!

recommended to use, and others have very specific functions.

With this guide, you'll learn how to properly use the ESP8266 GPIOs and avoid hours of frustration by using the most suitable pins for your projects.

Note: not all GPIOs are accessible in all development boards, but each specific GPIO works in the same way regardless of the development board you're using. If you're just getting started with the ESP8266, we recommend checking out our ESP8266 Guides.

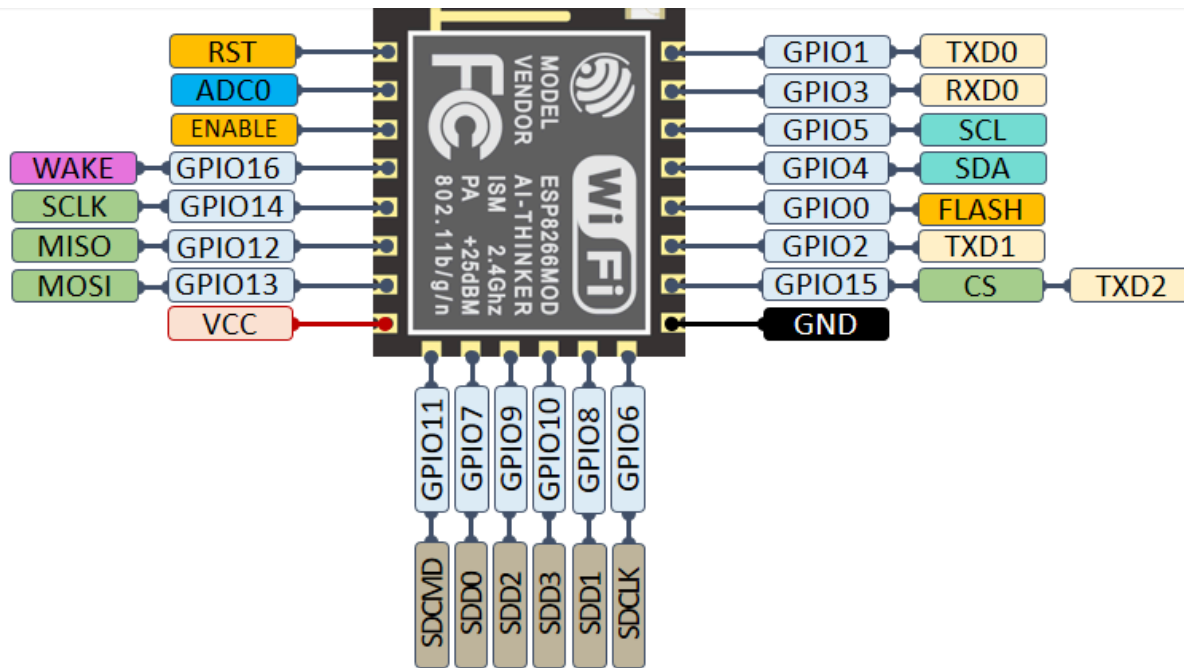
## ESP8266 12-E Chip Pinout

The following figure illustrates the ESP8266 12-E chip pinout. Use this diagram if you're using an ESP8266 bare chip in your projects.



**SMART HOME with Raspberry Pi, ESP32, and ESP8266 »** learn how to build a complete home automation system.






**Note:** not all GPIOs are accessible in all development boards, but each specific GPIO works in the same way regardless of the development board you're using. If you're just getting started with the ESP8266, we recommend reading our guide: [Getting Started with the ESP8266](#).

At the moment, there are a wide variety of development boards with the ESP8266 chip that differ in the number of accessible GPIOs, size, form factor, etc...

started guide to get the most out of the the Raspberry Pi Pico/Pico W (RP2040) microcontroller board using MicroPython programming language.

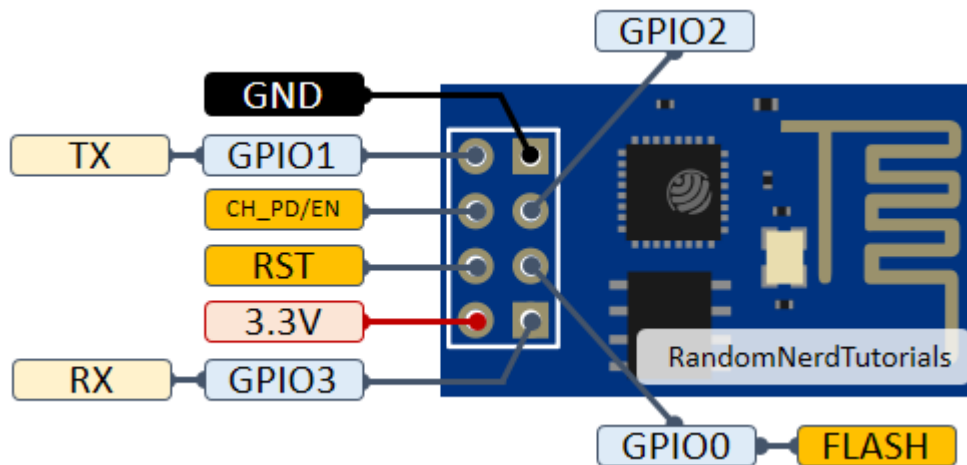


 **Learn LVGL: Build GUIs for ESP32 Projects »** Learn how to build Graphical User Interfaces (GUIs) for ESP32 Projects using LVGL (Light Versatile Graphics Library) with the Arduino IDE.

you can read this guide: [ESP8266 Wi-Fi Development Boards comparison](#).

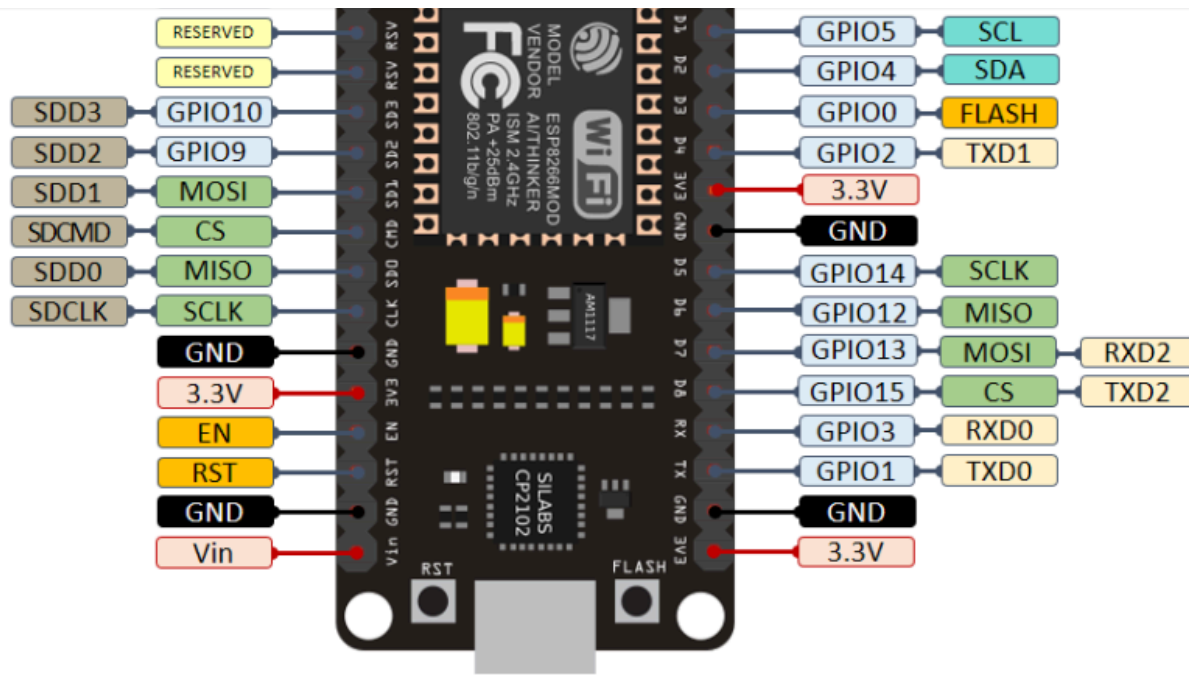
## ESP8266-01 Pinout

If you're using an ESP8266-01 board, you can use the following GPIO diagram as a reference.



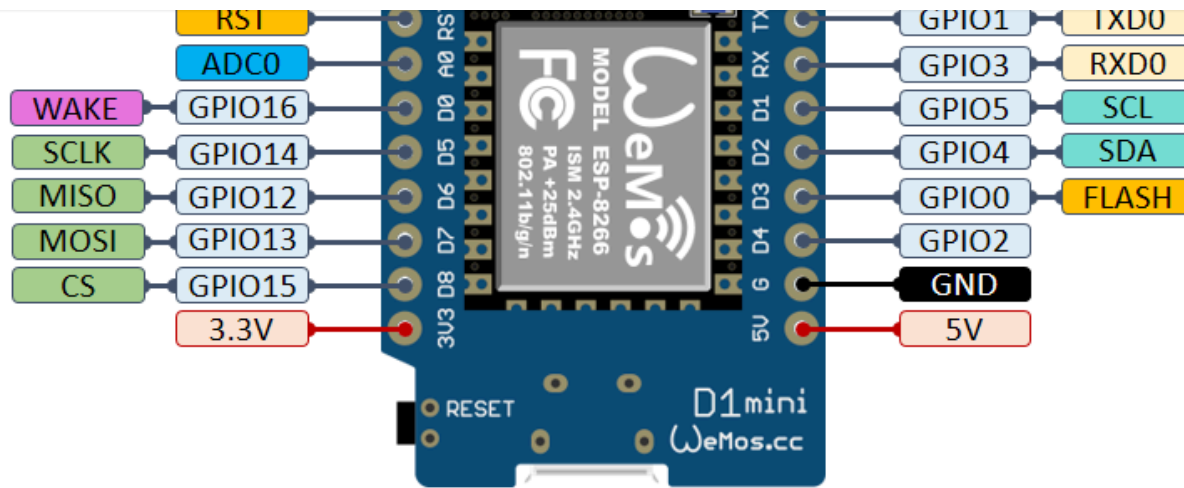
## ESP8266 12-E NodeMCU Kit

The ESP8266 12-E NodeMCU kit pinout diagram is shown below.



## Wemos D1 Mini Pinout

The following figure shows the WeMos D1 Mini pinout.



## Download PDF with ESP8266 Pinout Diagrams

We've put together a handy PDF that you can download and print, so you always have the ESP8266 diagrams next to you:

[Download PDF Pinout Diagrams »](#)

## ESP8266 Peripherals

The ESP8266 peripherals include:

- I2C (implemented on software)
- I2S interfaces with DMA
- UART
- 10-bit ADC

## Best Pins to Use – ESP8266

One important thing to notice about ESP8266 is that the GPIO number doesn't match the label on the board silkscreen. For example, D0 corresponds to GPIO16 and D1 corresponds to GPIO5.

The following table shows the correspondence between the labels on the silkscreen and the GPIO number as well as what pins are the best to use in your projects, and which ones you need to be cautious.

The pins highlighted in green are OK to use. The ones highlighted in yellow are OK to use, but you need to pay attention because they may have unexpected behavior mainly at boot. The pins highlighted in red are not recommended to use as inputs or outputs.

Label	GPIO	Input	Output	Notes
-------	------	-------	--------	-------

		interrupt	I2C support	deep sleep
D1	GPIO5	OK	OK	often used as SCL (I2C)
D2	GPIO4	OK	OK	often used as SDA (I2C)
D3	GPIO0	pulled up	OK	connected to FLASH button, boot fails if pulled LOW
D4	GPIO2	pulled up	OK	HIGH at boot connected to on-board LED, boot fails if pulled LOW
D5	GPIO14	OK	OK	SPI (SCLK)
D6	GPIO12	OK	OK	SPI (MISO)
D7	GPIO13	OK	OK	SPI (MOSI)
D8	GPIO15	pulled to GND	OK	SPI (CS) Boot fails if pulled HIGH



<b>TX</b>	<b>GPIO1</b>	TX pin	OK	HIGH at boot debug output at boot, boot fails if pulled LOW
<b>A0</b>	<b>ADC0</b>	Analog Input	X	

Continue reading for a more detailed and in-depth analysis of the ESP8266 GPIOs and its functions.

## GPIOs connected to the Flash Chip

GPIO6 to GPIO11 are usually connected to the flash chip in ESP8266 boards. So, these pins are not recommended to use.

## Pins used during Boot

The ESP8266 can be prevented from booting if some pins are pulled LOW or HIGH. The following list shows the state of the following pins on BOOT:

- **GPIO16:** pin is high at BOOT
- **GPIO0:** boot failure if pulled LOW
- **GPIO2:** pin is high on BOOT, boot failure if pulled LOW

- **GPIO1**: pin is high at BOOT, boot failure if pulled LOW
- **GPIO10**: pin is high at BOOT
- **GPIO9**: pin is high at BOOT

## Pins HIGH at Boot

There are certain pins that output a 3.3V signal when the ESP8266 boots. This may be problematic if you have relays or other peripherals connected to those GPIOs. The following GPIOs output a HIGH signal on boot:

- GPIO16
- GPIO3
- GPIO1
- GPIO10
- GPIO9

Additionally, the other GPIOs, except GPIO5 and GPIO4, can output a low-voltage signal at boot, which can be problematic if these are connected to transistors or relays. You can [read this article](#) that investigates the state and behavior of each GPIO on boot.

GPIO4 and GPIO5 are the most safe to use GPIOs if you want to operate relays.

---

The ESP8266 only supports analog reading in one GPIO. That GPIO is called **ADC0** and it is usually marked on the silkscreen as **A0**.

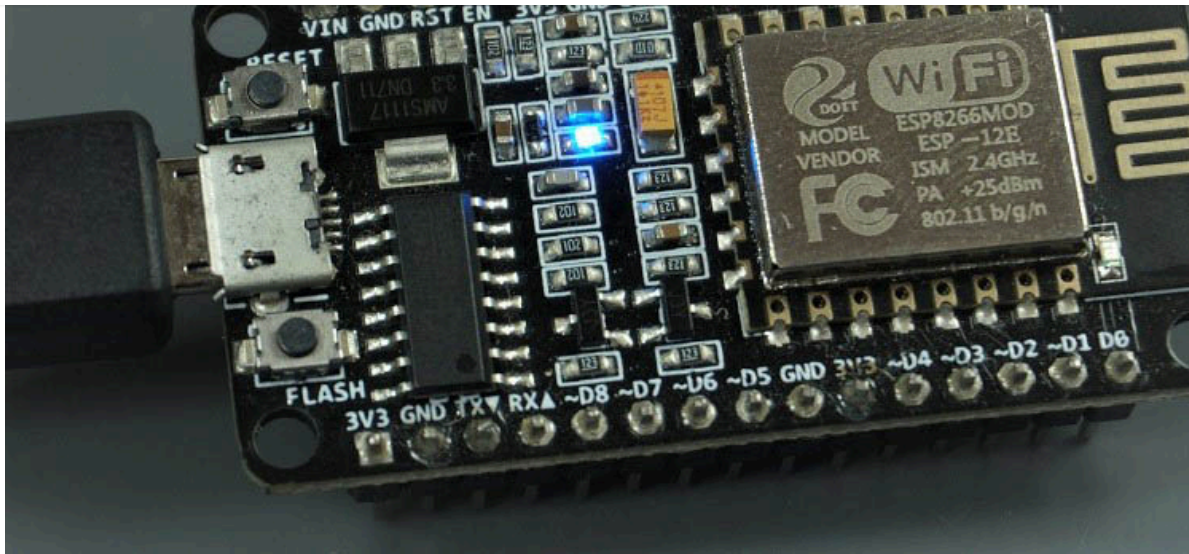
The maximum input voltage of the ADC0 pin is 0 to 1V if you're using the ESP8266 bare chip. If you're using a development board like the ESP8266 12-E NodeMCU kit, the voltage input range is 0 to 3.3V because these boards contain an internal voltage divider.

You can learn how to use analog reading with the ESP8266 with the following guide:

- [ESP8266 ADC – Read Analog Values with Arduino IDE, MicroPython and Lua](#)

## On-board LED

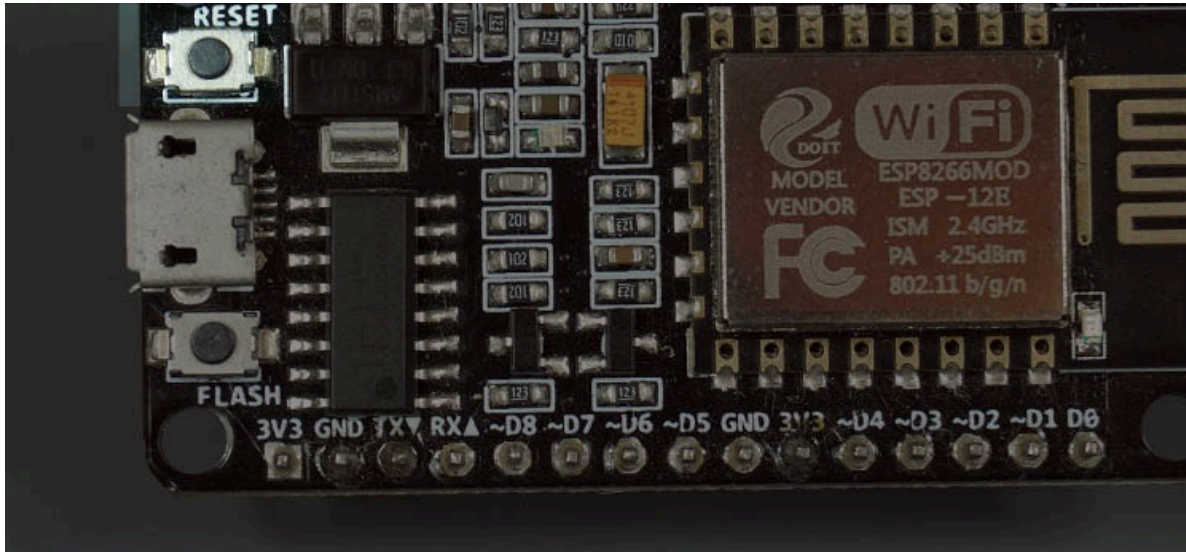
Most of the ESP8266 development boards have a built-in LED. This LED is usually connected to GPIO2.



The LED works with inverted logic. Send a HIGH signal to turn it off, and a LOW signal to turn it on.

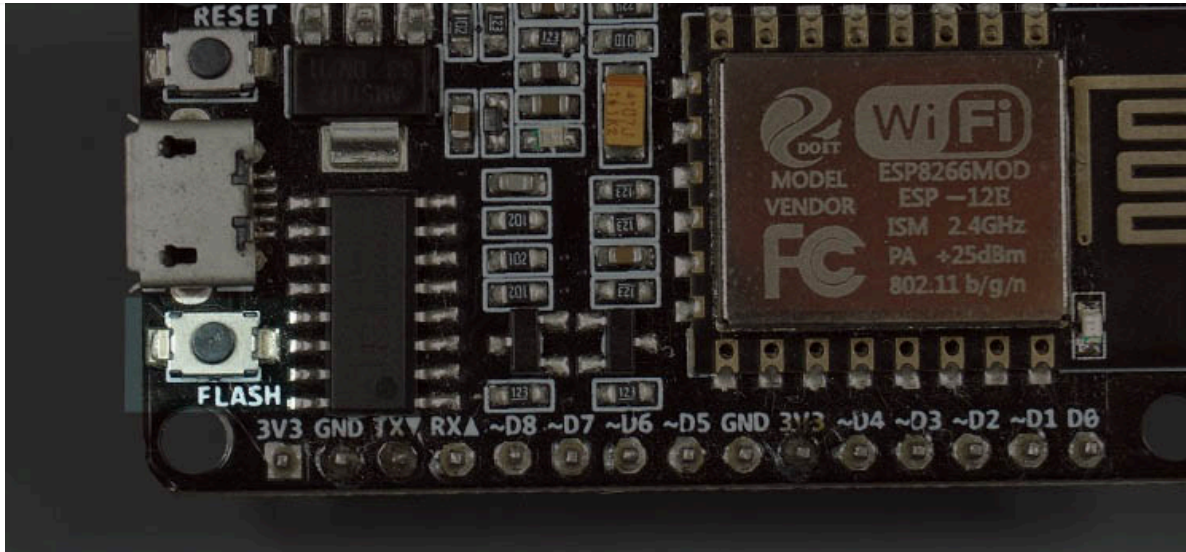
## RST Pin

When the RST pin is pulled LOW, the ESP8266 resets. This is the same as pressing the on-board RESET button.



## GPIO0

When GPIO0 is pulled LOW, it sets the ESP8266 into bootloader mode.  
This is the same as pressing the on-board FLASH/BOOT button.



## GPIO16

GPIO16 can be used to wake up the ESP8266 from deep sleep. To wake up the ESP8266 from deep sleep, GPIO16 should be connected to the RST pin. Learn how to put the ESP8266 into deep sleep mode:

- [ESP8266 Deep Sleep with Arduino IDE](#)
- [ESP8266 Deep Sleep with MicroPython](#)

## I2C

The ESP8266 doesn't have hardware I2C pins, but it can be implemented in software. So you can use any GPIOs as I2C. Usually, the following

- 
- **GPIO5:** SCL
  - **GPIO4:** SDA

## SPI

The pins used as SPI in the ESP8266 are:

- **GPIO12:** MISO
- **GPIO13:** MOSI
- **GPIO14:** SCLK
- **GPIO15:** CS

## PWM Pins

ESP8266 allows software PWM in all I/O pins: GPIO0 to GPIO15. PWM signals on ESP8266 have 10-bit resolution. Learn how to use ESP8266 PWM pins:

- [ESP8266 PWM with Arduino IDE](#)
- [ESP8266 PWM with MicroPython](#)

## Interrupt Pins

The ESP8266 supports interrupts in any GPIO, except GPIO16.

# Wrapping Up

We hope you've found this guide for the ESP8266 GPIOs useful. If you have some tips on how to use the ESP8266 GPIOs properly, you can write a comment below.

We also have a [similar guide for the ESP32 GPIOs that you can read](#).

If you're getting started with the ESP8266, we have some great content you might be interested in:

- [Home Automation using ESP8266 \(course\)](#)
- [Getting Started with the ESP8266](#)
- [30+ ESP8266 Projects and Tutorials](#)
- [ESP8266 Web Server Tutorial](#)
- [ESP32 vs ESP8266 – Pros and Cons](#)

Thanks for reading.