

ESP32 vs ESP8266 – Pros and Cons

May 1, 2019 By Sara Santos



Should you use ESP32 or ESP8266 in your projects? What's the difference between ESP32 and ESP8266? In this article we're going to answer these questions by comparing the ESP32 with the ESP8266 and cover the pros and cons of each board.

The ESP32 and ESP8266 are cheap Wi-Fi modules perfectly suited for DIY projects in the Internet of Things (IoT) field.

Both chips have a 32-bit processor. The ESP32 is dual core 160MHz to 240MHz CPU whereas the ESP8266 is a single core processor that runs at 80MHz.

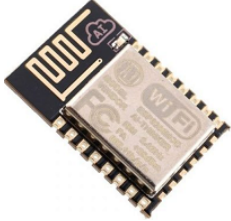

These modules come with GPIOs that support a wide variety of protocols like SPI, I2C, UART, and more. The best part is that they come with wireless networking included, which makes them apart from other microcontrollers like the [Arduino](#). This means that you can easily control and monitor devices remotely via Wi-Fi or Bluetooth (in case of ESP32) for a very low price.

Specifications: ESP32 vs ESP8266

The **ESP32** is the **ESP8266** successor. It adds an **extra CPU core**, faster Wi-Fi, more GPIOs, and supports Bluetooth 4.2 and **Bluetooth low energy**. Additionally, the ESP32 comes with touch sensitive pins that can be used to wake up the ESP32 from deep sleep, a **built-in hall effect sensor** and a built-in temperature sensor.

Both boards are very cheap, but the ESP32 costs slightly more. While the ESP32 can cost around \$6 to \$12, the ESP8266 can cost \$4 to \$6 (but it really depends on where you get them).

The following table shows the main differences between the ESP8266 and the ESP32 chips (table adapted from: [AMICA_IO](#)).

	ESP8266	ESP32
		
MCU	Xtensa Single-core 32-bit L106	Xtensa Dual-Core 32-bit LX6 with 600 DMIPS
802.11 b/g/n Wi-Fi	HT20	HT40
Bluetooth	X	Bluetooth 4.2 and BLE
Typical Frequency	80 MHz	160 MHz
SRAM	X	✓
Flash	X	✓
GPIO	17	36
Hardware /Software PWM	None / 8 channels	None / 16 channels
SPI/I2C/I2S/UART	2/1/2/2	4/2/2/2
ADC	10-bit	12-bit
CAN	X	✓
Ethernet MAC Interface	X	✓
Touch Sensor	X	✓
Temperature Sensor	X	✓

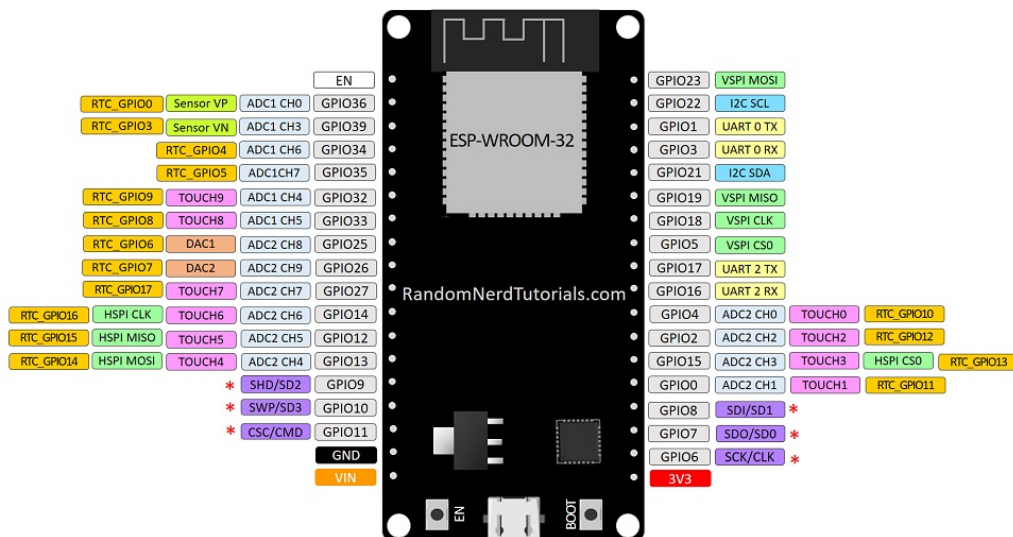
	ESP8266	ESP32
Hall effect sensor	X	✓
Working Temperature	-40°C to 125°C	-40°C to 125°C
Price	\$ (3\$ - \$6)	\$\$ (\$6 - \$12)
Where to buy	Best ESP8266 Wi-Fi Development Boards	ESP32 Development Boards Review and Comparison

More GPIOs on the ESP32

The **ESP32 has more GPIOs** than the **ESP8266**, and you can decide which pins are UART, I2C, SPI – you just need to set that on the code. This is possible due to the ESP32 chip's multiplexing feature that allows you to assign multiple functions to the same pin.

If you don't set them on the code, they will be on the pins defined by default as shown in the following figure (this is an example for the ESP32 DEVKIT V1 DOIT board – the pin location can change depending on the manufacturer).

ESP32 DEVKIT V1 – DOIT version with 36 GPIOs



* Pins SCK/CLK, SDO/SD0, SDI/SD1, SHD/SD2, SWP/SD3 and CSC/CMD, namely, GPIO6 to GPIO11 are connected to the integrated SPI flash integrated on ESP-WROOM-32 and are not recommended for other uses.

Recommended reading:

- [ESP32 Pinout Reference: Which GPIO pins should you use?](#)
- [ESP8266 Pinout Reference: Which GPIO pins should you use?](#)

PWM, ADC, and More

You can set **PWM signals in any GPIO** with configurable frequencies and duty cycle set on the code.

When it comes to the analog pins, these are static, but the ESP32 supports measurements on 18 channels (analog-enabled pins) versus just one 10-bit ADC pin on the ESP8266. The ESP32 also supports two 8-bit DAC channels.

Moreover, the ESP32 contains 10 capacitive sensing GPIOs, that detect touch and can be used to trigger events, or **wake-up the ESP32 from deep sleep**, for example.

Arduino IDE – ESP32 vs ESP8266

There are many ways to program the ESP32 and ESP8266 boards. Currently, both boards can be programmed using the Arduino IDE programming environment.



This is a good thing, specially for those who are used to program the Arduino and are familiar with the Arduino programming language.

Getting started with the ESP32 or ESP8266 using Arduino IDE and have your first project running is very simple. You can follow these guides:

- [Installing the ESP32 Board in Arduino IDE \(Windows instructions\)](#)
- [Installing the ESP32 Board in Arduino IDE \(Mac and Linux instructions\)](#)
- [How to Install the ESP8266 Board in Arduino IDE](#)

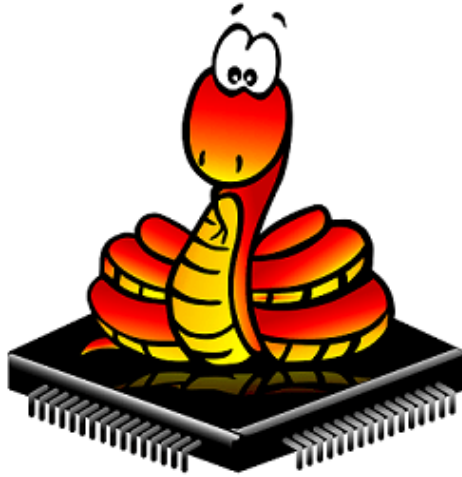
Although you can program both boards using Arduino IDE, they might not be compatible with the same libraries and commands. Some libraries are just compatible with one of the boards. This means that most of the times your ESP8266 code will not be compatible with the ESP32. However, usually you just need to make a few modifications.

We have a dedicated list of free tutorials and projects for the ESP32 and ESP8266 boards using the Arduino IDE that you might find useful:

- [ESP32 Projects and Tutorials](#)
- [ESP8266 Projects and Tutorials](#)

MicroPython Firmware – ESP32 vs ESP8266

Another popular way of programming the ESP32 and ESP8266 boards is using MicroPython firmware.



MicroPython is a re-implementation of Python 3 targeted for microcontrollers and embedded systems. MicroPython is very similar with regular Python. So, if you already know how to program in Python, you also know how to program in MicroPython.

In MicroPython, most Python scripts are compatible with both boards (unlike when using Arduino IDE). This means that most of the time you can use the same script for ESP32 and ESP8266.

You can get started with MicroPython firmware on the ESP32 and ESP8266 very quickly by following our free guides:

- [Getting Started with MicroPython on ESP32 and ESP8266](#)
- [Getting Started with Thonny MicroPython \(Python\) IDE for ESP32 and ESP8266](#)

Need Resources to Get Started?

If you want to get started with the ESP32 or ESP8266, you can take a look at our courses and projects:

- [Learn ESP32 with Arduino IDE](#)
- [Home Automation Using ESP8266](#)
- [MicroPython Programming with ESP32 and ESP8266](#)
- **Free projects for:** [ESP32](#) or [ESP8266](#)

ESP32 or ESP8266?

So, at this point you should be asking: **Should I get an ESP8266 or an ESP32?**



ESP32



ESP8266

It really depends on what you want to do. There is space for both boards, and both have pros and cons.

The ESP8266 is cheaper than the ESP32. Although it doesn't have as much functionalities, it works just fine for the majority of simple DIY IoT projects. Additionally, because it is "older" it is much more supported in terms of software, and you might find help easier. However, it has some limitations when it comes to the GPIO mapping, and it might not have enough pins for what you intend to do. If that's the case, you should get an ESP32.

The ESP32 is much more powerful than the ESP8266, contains more GPIOs with multiple functions, faster Wi-Fi, and also supports Bluetooth. Many people think that the ESP32 is more difficult to deal with than the ESP8266 because it is more complex. In our opinion, it is as easy to program the ESP32 as the ESP8266, specially if you intend to program it using the Arduino language or MicroPython.

On the other side, the ESP32 has some cons too. First, it is more expensive than the ESP8266. So, if you're making a simple IoT project, the ESP8266 might do the trick for a lower price. Additionally, because it is more recent than the ESP8266, not all software has been developed to take the most out of the ESP32 functionalities, there is less support, and more bugs. But on the long run these issues will be fixed, and there will be space for both boards.

My personal experience: in 2019, I use almost exclusively the ESP32 for IoT projects. It is more versatile, and it comes with much more functionalities like Bluetooth, different wake up sources, many peripherals, and much more. Additionally, the difference in price is not a big deal, in my opinion. I think that once you move to the ESP32, you will not want to go back to the ESP8266.

Wrapping U