Introduction

Pieter P

This is a tutorial on how to use the world's most popular $3 Wi-Fi board, the ESP8266.

This is going to be a very in-depth tutorial, covering some networking concepts as well. If you're a beginner, and just want to go straight to the more exciting Wi-Fi part and try some examples, feel free to do so. I included short *TL;DR's* in the longer, more theoretical parts.

A short overview of what I'll cover in this article:

1. What is an ESP8266?: A short overview of what an ESP8266 is, and what you can do with it
2. Deciding on what board to buy: There are a lot of different ESP8266 boards available these days, finding the one that's best for you isn't always easy
3. Installing the software: You need to install some software to program the ESP8266, and maybe a USB driver
4. Setting up the hardware: Some modules and boards need some external components before you can use them
5. The ESP8266 as a microcontroller: The ESP8266 can be used as a normal microcontroller, just like an Arduino
6. Network protocols: Before we start using the Wi-Fi capabilities of the ESP8266, I'll cover some of the network protocols involved
7. Setting up a Wi-Fi connection: That's probably why you're reading this, right?
8. Name resolution: Find the ESP8266 on your local network using mDNS
9. Setting up a simple web server: This enables you to add web pages to the ESP8266, and browse them from your computer or phone
10. Setting up a file server: A more advanced web server with a real file system that allows you to upload new files over Wi-Fi
11. OTA - uploading new programs over Wi-Fi: You don't have to upload programs over USB, you can use Wi-Fi instead
12. Wirelessly controlling your RGB lighting: Change the color of your LED strips using your phone or computer
13. Getting the time: Connect to a time server using NTP and sync the ESP's clock
14. Monitoring sensors: Log the temperature in your living room, save it in flash memory and show it in a fancy graph in your browser
15. Getting email notifications: Turn on a notification light when you've got unread emails
16. Advanced features: use DNS, captive portals, Wi-Fi connector libraries, OSC ...

This guide expects some basic knowledge of microcontrollers like the Arduino. If you're new to Arduino, I'd recommend to get familiar with it first, because I won't go into the basics in this article.
I really want to focus on the ESP8266-specific things, like Wi-Fi and other network protocols, the ESP's hardware, software, IoT, etc ...

What is an ESP8266?

The ESP8266 is a System on a Chip (SoC), manufactured by the Chinese company Espressif. It consists of a Tensilica L106 32-bit micro controller unit (MCU) and a Wi-Fi transceiver. It has 11 GPIO pins* (General Purpose Input/Output pins), and an analog input as well. This means that you can program it like any normal Arduino or other microcontroller. And on top of that, you get Wi-Fi communication, so you can use it to connect to your Wi-Fi network, connect to the Internet, host a web server with real web pages, let your smartphone connect to it, etc ... The possibilities are endless! It's no wonder that this chip has become the most popular IOT device available.

There are many different modules available, modules like the ESP-## series by AI Thinker, containing only the ESP8266 SoC and a flash memory chip, or complete development boards like the NodeMCU DevKit or the WeMos D1, that have a USB interface for programming the device, just like an Arduino board. Different boards may have different pins broken out, have different Wi-Fi antennas, or a different amount of flash memory on board.

(*) The ESP8266 chip itself has 17 GPIO pins, but 6 of these pins (GPIO 6-11) are used for communication with the on-board flash memory chip.

Programming

There are different ways to program the ESP8266, but I'll only cover the method using the Arduino IDE. This is really easy for beginners, and it's a very familiar environment if you've used Arduino boards before.

Just keep in mind that it's not limited to this option: there's also an official SDK available to program it in real C, this is very useful if you want to optimize your code or do some advanced tricks that aren't supported by the Arduino IDE. Keep in mind that the learning curve is quite steep, but if that's something you're interested in, I highly recommend Neil Kolban's book on the ESP8266.
Another possibility is to flash it with a LUA interpreter, so you can upload and run LUA scripts. Or maybe you're more familiar with Python? Then you could check out the MicroPython firmware to interpret MicroPython scripts. I'm sure there's other languages available as well.

Requirements

You'll need a couple of things in order to follow this guide:

- An ESP8266 board
- A computer that can run the Arduino IDE
- A USB-to-Serial converter, it is very important that you use a 3.3V model*
- A USB cable
- A 3.3V power supply or voltage regulator*
- A Wi-Fi network (with internet access) to connect to

(*) Your board may already include these. More information can be found in the next chapter.