NodeMCU ESP8266: Details and Pinout

by Fernando Koyanagi

Today, we will talk about ESP8266 pinning, or in other words, NodeMCU. Personally, I really like this component, as it already comes with USB input. But it is important to explain that the NodeMCU is formed by an ESP12E, which still has an ESP8266EX inside it. Thus, we'll learn the correct pin identification by doing the following: looking at the NodeMCU datasheet, knowing which of these pins work with digitalWrite, digitalRead, analogWrite, and analogRead, and understanding the boot more thoroughly.

As I program more with Arduino IDE, I practically see the NodeMCU as an Arduino. However, I must emphasize these devices have differences, especially concerning the pinning. If you watched the ESP32 video entitled "Internal Details and Pinout," you've learned there are pins that can't be used, or that are reserved for certain things. So I want to do something useful here related to this, but this time with ESP8266.

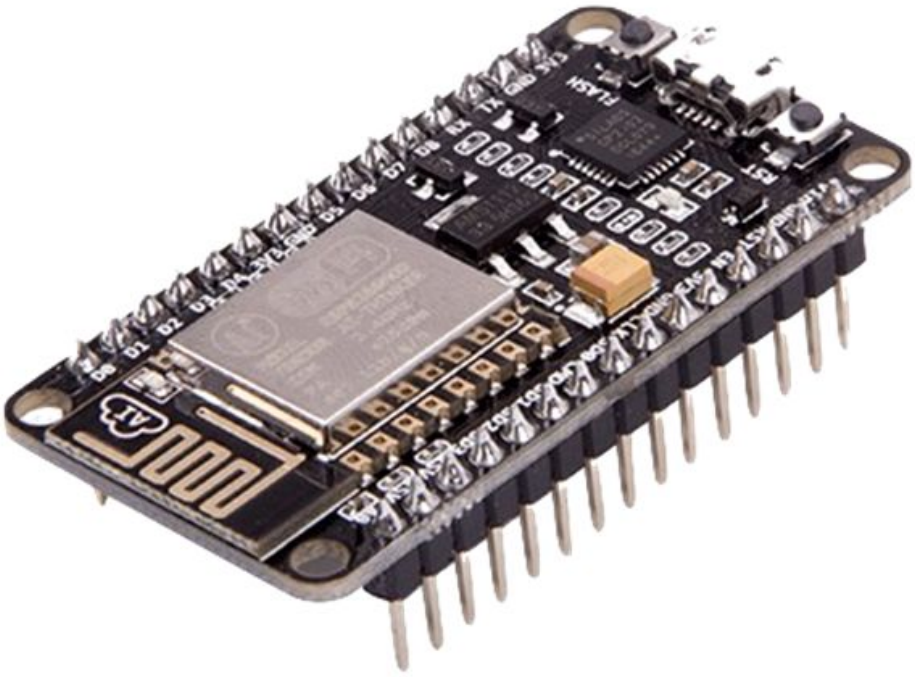
https://youtu.be/slKGGrPDNpk

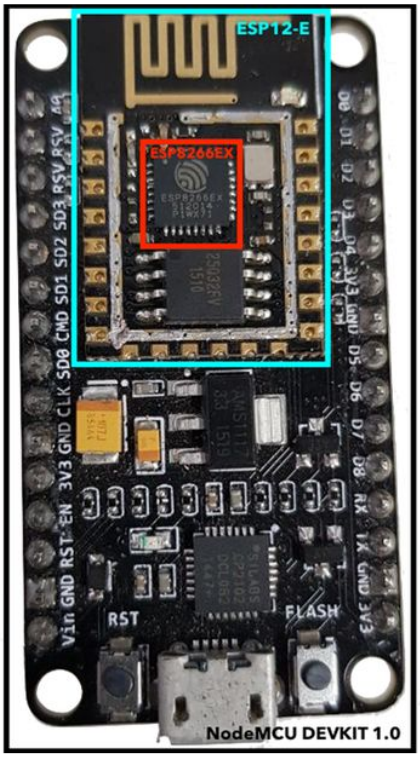
Step 1: NodeMCU Devkit 1.0

The term NodeMCU usually refers to the firmware, while the board is called Devkit.

NodeMCU Devkit 1.0 consists of an ESP-12E on a board, which facilitates its use.

It also has a voltage regulator, a USB interface.





Step 2: ESP-12E

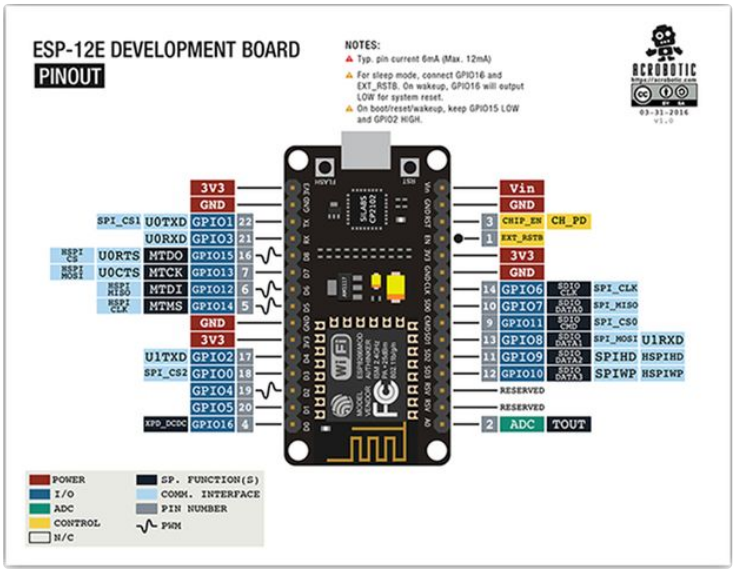
The ESP-12E is a board created by AI-THINKER, which consists of an ESP8266EX inside the metal cover.

Step 3: ESP8266EX

Made by Espressif, this microchip has integrated WiFi and low-power consumption.

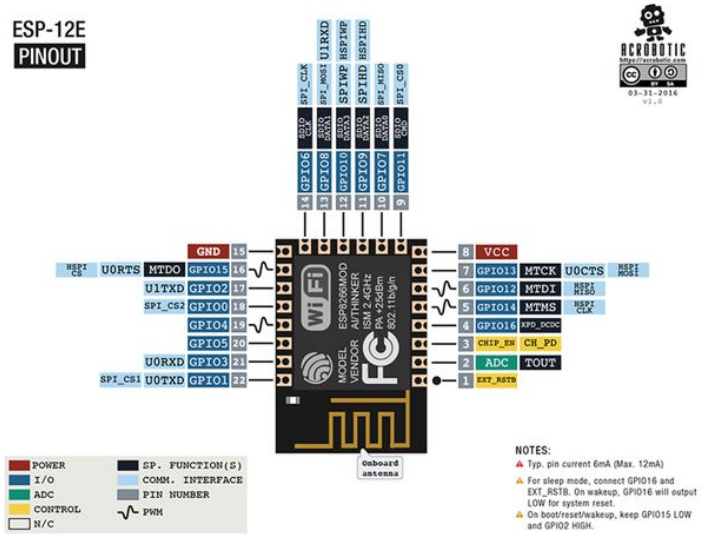
Processor RISC Tensilica L 106 32bit with a maximum clock of 160 MHz

Step 4: NodeMCU 1.0 ESP-12E Pinout



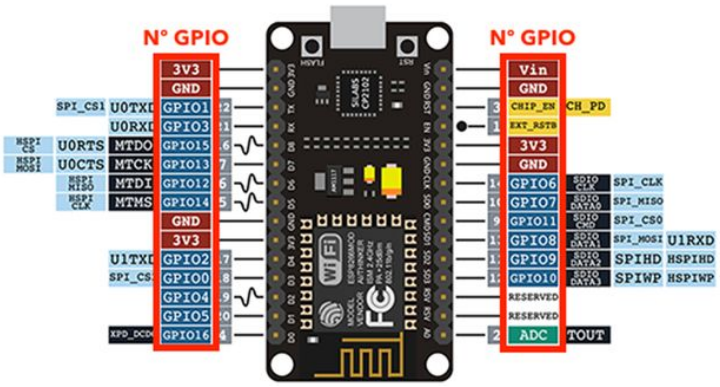
Step 5: ESP-12E Pinout

I want to emphasize that NodeMCU and ESP-12E are not the same things. In the case of the ESP-12E, the recording uses the serial, the UART. In NodeMCU, this is performed by the USB.



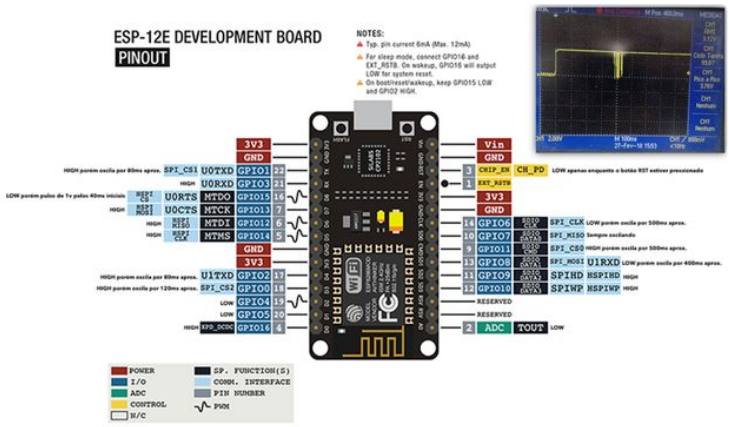
Step 6: And After All This, What's the Number to Put When Programming?

Use the number that is in front of the GPIO or the constants A0, D0, D1, D2, D3, D4, D5, D6, D7, and D8.



Step 7: Boot

We put the oscilloscope at the tip of each pin. This allows us to find, for example, that when we turn on the NodeMCU, its pins are not all the same. Some are up and others down, by default. See the comments on the behavior of each post after the boot in the image below.



Step 8: Constants That Are Already Predefined

Constante	Valor
D0	16
D1	5
D2	4
D3	0
D4	2
D5	14
D6	12
D7	13
D8	15
A0	17

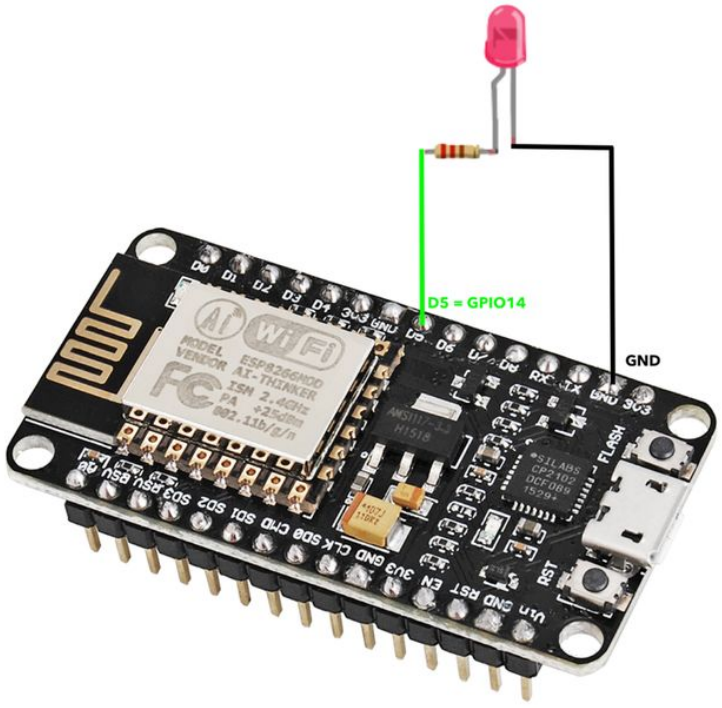
Step 9: Blink Example

In this example, we connected an LED on port D5, which is GPIO14. So the options are as follows:

```
 //O led está no GPIO14
 #define LED 6
//ou usar a constante D5 que já está definida
//#define LED D5

void setup() {
  pinMode(LED, FUNCTION_3);
}

void loop() {
  digitalWrite(LED, HIGH);
  delay(1000);
  digitalWrite(LED, LOW);
  delay(1000);
}
```



Step 10: INPUT / OUTPUT

When performing INPUT and OUTPUT tests on the pins, we obtained the following results:

- **digitalWrite** did NOT work with GPIOs 6, 7, 8, 11, and ADC (A0)
- **digitalRead** did NOT work with GPIOs 1, 3, 6, 7, 8, 11, and the ADC (A0)
- **analogWrite** did NOT work with GPIOs 6, 7, 8, 11, and ADC (A0) (GPIOs 4, 12, 14, 15 have hardware PWM, and the others are by software)
- **analogRead** worked only with the ADC (A0)

- **6**, **7**, **8**, **11** do NOT work for the above four commands

Step 11: PDF

Download the PDF.