

Taller ESP8266

El contenido es:

1. ¿Qué es el ESP8266?
2. ¿Cómo se programa? (Explicación de las diferentes formas para programarlo y configuración del IDE de Arduino para hacerlo)
3. Montaje de un programador para el ESP-12
4. Ejemplo LED-Blink (Y de ahí lo que se les ocurra)

Materiales

- cable usb-mini (AGREGAR IMAGEN)
- ftdi (AGREGAR IMAGEN)
- protoboard (AGREGAR IMAGEN)
- soldador (AGREGAR IMAGEN)
- un computador con arduino instalado (AGREGAR IMAGEN)
- ESP12 (AGREGAR IMAGEN)
- pcb para ESP12 (AGREGAR IMAGEN)
- Suiche on/off de 6 patas (AGREGAR IMAGEN)
- Pulsador (AGREGAR IMAGEN)

1. ¿Qué es el ESP8266?

<http://espressif.com/products/hardware/esp8266ex/overview>

High Integrated

ESP8266EX is among the most integrated WiFi chips in the industry with the size of 5mm x 5mm ; it integrates the antenna switches, RF balun, power amplifier, low noise receive amplifier, filters, power management modules while requires minimal external circuitry. The entire solution, including front-end module, is designed to occupy minimal PCB area.

32-bit MCU

ESP8266EX integrates Tensilica L106 32-bit micro controller (MCU) which features extra low power consumption and 16-bit RSIC. The CPU clock speed is 80 MHz. It can also reach a maximum value of 160 MHz. Real Time Operation System (RTOS) is enabled. Currently, only 20% of MIPS has been occupied by the Wi-Fi stack, the rest can all be used for user application programming and development.

Low Power

ESP8266EX has been designed for mobile, wearable electronics and Internet of Things applications with the aim of achieving the lowest power consumption with a combination of several proprietary technologies. The power saving architecture operates in 3 modes: active mode, sleep mode and deep

sleep mode.

Robustness

By integrating more components on-chip, we have made the solution to be the most robust and manufacturable. Our solutions also feature the widest operating temperature range, from -40°C to +125°C.

<https://github.com/esp8266/esp8266-wiki/wiki>

What is this ESP8266

It's a wireless SoC
It has GPIO, I2C, ADC, SPI, PWM and some more
It's running at 80MHz
64KBytes of instruction RAM
96KBytes of data RAM
64KBytes boot ROM
It has a Winbond W25Q40BVNIG SPI flash
It's a RISC architecture
The core is a 106micro Diamond Standard core (LX3) made by Tensilica
The ESP8266 chip is made by Espressif
Modules bearing this chip are made by various manufacturers

Features

802.11 b/g/n protocol
Wi-Fi 2.4 GHz, support WPA/WPA2
Super small module size (11.5mm x 11.5mm)
Integrated 10-bit ADC
Integrated TCP/IP protocol stack (ipv4 only at the moment)
Integrated TR switch, balun, LNA, power amplifier and matching network
Integrated PLL, regulators, and power management units
+20dBm output power in 802.11b mode
Supports antenna diversity
Deep sleep power <10uA, Power down leakage current < 5uA
Integrated low power 32-bit MCU
SDIO 2.0, SPI, UART, I2C
STBC, 1x1 MIMO, 2x1 MIMO
A-MPDU & A-MSDU aggregation & 0.4μs guard interval
Wake up and transmit packets in < 2ms
Standby power consumption of < 1.0mW (DTIM3)
Operating temperature range -40C ~ 125C

Status API Training Shop Blog About Pricing

2. ¿Cómo se programa?

Arduino

Installing with Boards Manager

Starting with 1.6.4, Arduino allows installation of third-party platform packages using Boards Manager. We have packages available for Windows, Mac OS, and Linux (32 and 64 bit).

1. Install Arduino 1.6.5 from the Arduino website.
2. Start Arduino and open Preferences window.
3. Enter http://arduino.esp8266.com/stable/package_esp8266com_index.json into Additional Board Manager URLs field. You can add multiple URLs, separating them with commas.
4. Open Boards Manager from Tools > Board menu and install esp8266 platform (and don't forget to select your ESP8266 board from Tools > Board menu after installation).

The best place to ask questions related to this core is ESP8266 community forum:

<http://www.esp8266.com/arduino>. If you find this forum or the ESP8266 Boards Manager package useful, please consider supporting it with a donation. Donate

NodeMCU - LUA

http://www.nodemcu.com/index_en.html

An open-source firmware and development kit that helps you to prototype your IOT product within a few Lua script lines

Features

Arduino-like hardware IO Advanced API for hardware IO, which can dramatically reduce the redundant work for configuring and manipulating hardware. Code like arduino, but interactively in Lua script.

Nodejs style network API Event-driven API for network applicaitons, which faciliates developers writing code running on a 5mm*5mm sized MCU in Nodejs style. Greatly speed up your IOT application developing process

Lowest cost WI-FI Less than \$2 WI-FI MCU ESP8266 integrated and esay to prototyping development kit. We provide the best platform for IOT application development at the lowest cost.

Development Kit The Development Kit based on ESP8266, integates GPIO, PWM, IIC, 1-Wire and ADC all in one board. Power your developement in the fastest way combining with NodeMCU Firmware!

Se programa con esptool.py <https://github.com/themadinventor/esptool> A cute Python utility to communicate with the ROM bootloader in Espressif ESP8266. It is intended to be a simple, platform independent, open source replacement for XTCOM.

Custom Builds de nodeMCU → <http://nodemcu-build.com/index.php> You customize your NodeMCU firmware and we build it. Just for you. On the spot.

MicroPython - Python

<http://www.micropython.org/>

MicroPython is a lean and fast implementation of the Python 3 programming language that is optimised to run on a microcontroller. The MicroPython board is a small electronic circuit board that runs MicroPython on the bare metal, and gives you a low-level Python operating system that can be used to control all kinds of electronic projects.

Smart.js - Javascript

<https://www.cesanta.com/developer/smartjs>

Smart.js is a generic, cross-platform, full-stack Internet of Things software platform. It makes it fast and easy to connect devices online. For the device part, Smart.js provides JavaScript-enabled firmware. Device part is integrated with the cloud part, which provides device management, OTA (Over The Air) reliable update functionality, and designed for easy integration with 3rd party database and analytics software.

Sming - C++

<https://github.com/SmingHub/Sming>

Sming - Open Source framework for high efficiency WiFi SoC ESP8266 native development with C++ language.

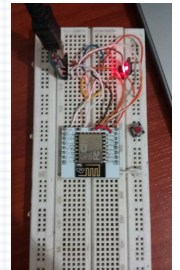
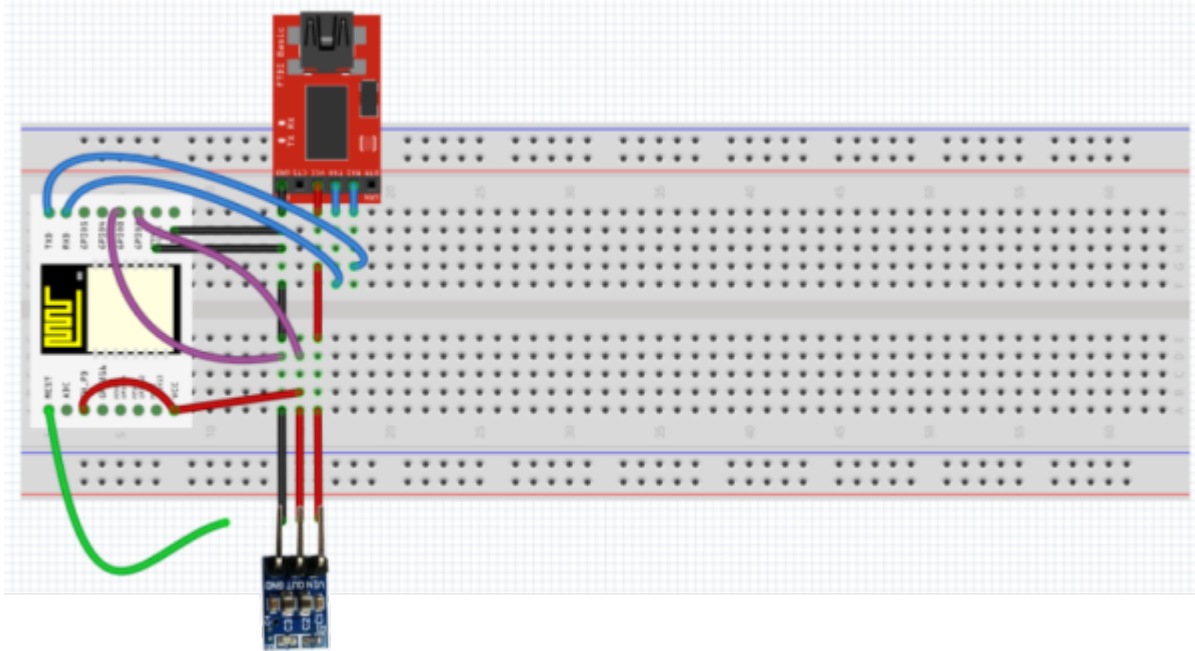
Summary

- Fast & user friendly development
- Work with GPIO in Arduino style
- High effective in performance and memory usage (this is native firmware!)
- Compatible with standard Arduino libraries - use any popular hardware in few lines of code
- rBoot OTA firmware updating
- Built-in file system: spiiffs
- Built-in powerfull network and wireless modules
- Built-in JSON library: ArduinoJson
- HTTP, AJAX, WebSockets support
- MQTT protocol based on libemqtt
- Networking based on LWIP stack
- Simple and powerfull hardware API wrappers
- Based on Espressif NONOS SDK 1.4.0 & 1.5.0

3. Montaje de un programador para el ESP-12

El esquema que se muestra a continuación es utilizado para programar el ESP12, se debe tener el

cuenta en cuenta que los pines **GPIO2** y **GPIO0** (cables morados en el esquema de la protoboard) se conectan de manera temporal mientras se programa. Para poner a funcionar el ESP-12 en modo de ejecución de programa estos dos pines deben ser desconectados. Adicionalmente es necesario agregar un cable al pin **RESET** (reset - cable verde en el esquema de la protoboard) que justo en el momento que se está subiendo el programa desde el IDE de Arduino, se debe hacer contacto con **GND** para resetear el integrado.



4. Ejemplo LED-Blink (Y de ahí lo que se les ocurra)

El builtin led para el ESP-12 no es el mismo que está definido dentro del código de arduino. Se debe renombrar esta variable y usar el **pin 2**

```
/*
  ESP8266 Blink by Simon Peter
  Blink the blue LED on the ESP-01 module
  This example code is in the public domain

  The blue LED on the ESP-01 module is connected to GPIO1
  (which is also the TXD pin; so we cannot use Serial.print() at the same
  time)

  Note that this sketch uses BUILTIN_LED to find the pin with the internal
  LED
  */

#define LED 2

void setup() {
  pinMode(LED, OUTPUT);    // Initialize the BUILTIN_LED pin as an output
}
```

```
// the loop function runs over and over again forever
void loop() {
  digitalWrite(LED, LOW); // Turn the LED on (Note that LOW is the voltage
level                                     // but actually the LED is on; this is
because                                // it is active low on the ESP-01)
                                     // Wait for a second
  delay(1000);
  digitalWrite(LED, HIGH); // Turn the LED off by making the voltage HIGH
  delay(2000);             // Wait for two seconds (to demonstrate
the active low LED)
}
```

From:
<https://wiki.unloquer.org/> -

Permanent link:
<https://wiki.unloquer.org/documentacion-proceso/tecnologicos/taller-esp8266?rev=1457747032>

Last update: **2016/03/12 01:43**

