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 > 1.6.4 (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
https://www.reddit.com/r/esp8266/wiki/index

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, 1×1 MIMO, 2×1 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

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

AT Firmware

Manipulación del ESP8266 mediante comandos AT → http://www.electrodragon.com/w/Category:ESP8266_Basic_Usage Para descargar el firmare binario → https://github.com/espressif/ESP8266_AT/tree/master/bin

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 combinating 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.

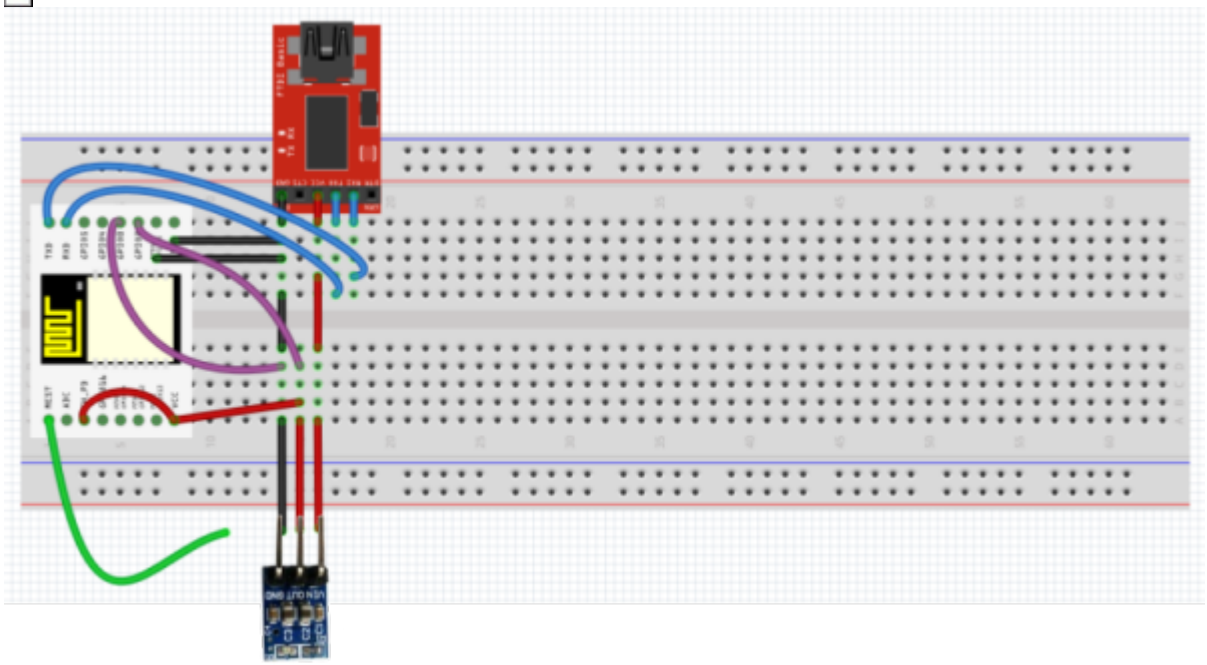
Summary

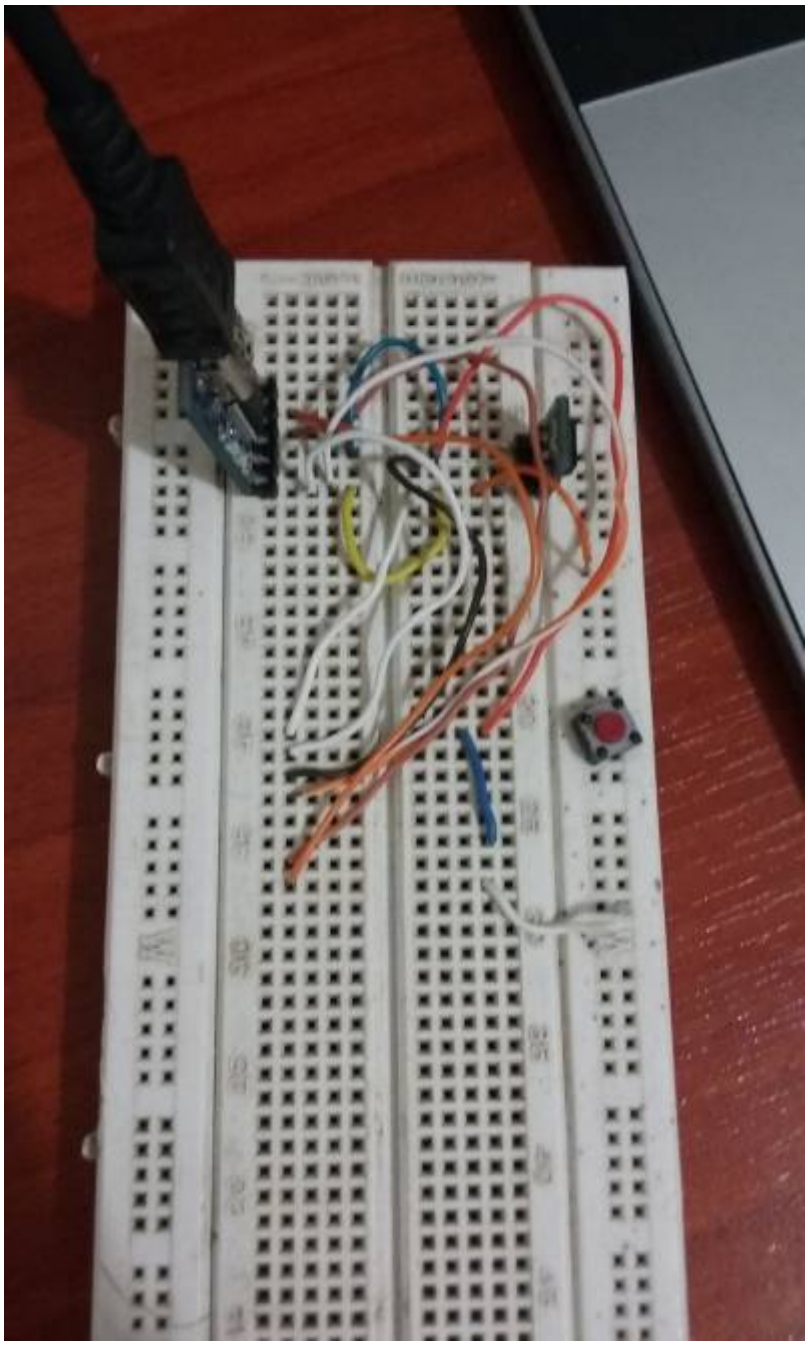
- Fast & user friendly development
- Work with GPIO in Arduino style
- High effective in perfomance and memory usage (this is native firmware!)
- Compatible with standard Arduino libraries - use any popular hardware in few lines of code
- rBoot OTA firmware updating
- Built-in file system: spiffs
- Built-in 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 **REST** (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. Ejemplos

LED-Blink

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**. A continuación se muestra el código probado con la modificación mencionada.

```
/*
 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 acive low on the ESP-01)
  delay(1000);              // Wait for a second
  digitalWrite(LED, HIGH);  // Turn the LED off by making the voltage HIGH
  delay(2000);              // Wait for two seconds (to demonstrate
the active low LED)
}
```

Relevo wifi

La idea es programar el ESP8266 para que de manera remota a través de una petición a un servidor web que se crea dentro del chip se encienda o se apage un pin. El código a continuación es tomado del ejemplo **WifiWebServer** de Arduino para ESP8266. Probamos usando el **pin GPIO2** para activar el relevo pero presentaba problemas con la conexión a la red. Lo cambiamos para el **pin GPIO4**

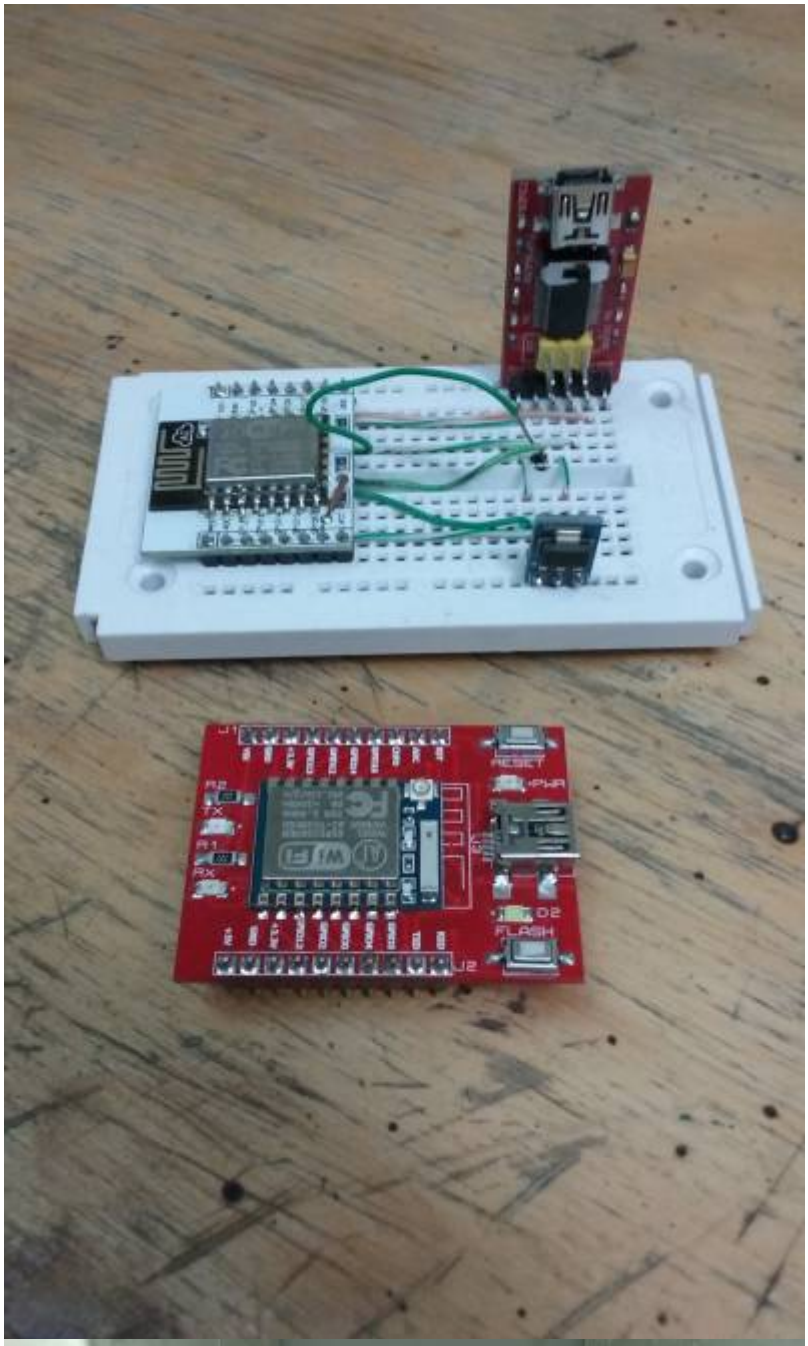
```
/*
 *  This sketch demonstrates how to set up a simple HTTP-like server.
 *  The server will set a GPIO pin depending on the request
 *    http://server_ip/gpio/0 will set the GPIO2 low,
 *    http://server_ip/gpio/1 will set the GPIO2 high
 *  server_ip is the IP address of the ESP8266 module, will be
 *  printed to Serial when the module is connected.
 */

#include <ESP8266WiFi.h>

const char* ssid = "ElNombreDeLaRed";
const char* password = "LaClaveDeLaRed";
```

```
// Create an instance of the server
// specify the port to listen on as an argument
WiFiServer server(80);

void setup() {
  Serial.begin(115200);
  delay(10);

  // prepare GPIO2
  pinMode(4, OUTPUT);
  digitalWrite(4, 0);

  // Connect to WiFi network
  Serial.println();
  Serial.println();
  Serial.print("Connecting to ");
  Serial.println(ssid);

  WiFi.begin(ssid, password);

  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }
  Serial.println("");
  Serial.println("WiFi connected");

  // Start the server
  server.begin();
  Serial.println("Server started");

  // Print the IP address
  Serial.println(WiFi.localIP());
}

void loop() {
  // Check if a client has connected
  WiFiClient client = server.available();
  if (!client) {
    return;
  }

  // Wait until the client sends some data
  Serial.println("new client");
  while(!client.available()){
    delay(1);
  }

  // Read the first line of the request
```

```
  String req = client.readStringUntil('\r');
  Serial.println(req);
  client.flush();

  // Match the request
  int val;
  if (req.indexOf("/gpio/0") != -1)
    val = 0;
  else if (req.indexOf("/gpio/1") != -1)
    val = 1;
  else {
    Serial.println("invalid request");
    client.stop();
    return;
  }

  // Set GPIO2 according to the request
  digitalWrite(4, val);

  client.flush();

  // Prepare the response
  String s = "HTTP/1.1 200 OK\r\nContent-Type: text/html\r\n\r\n<!DOCTYPE
HTML>\r\n<html>\r\nGPIO is now ";
  s += (val)?"high":"low";
  s += "</html>\n";

  // Send the response to the client
  client.print(s);
  delay(1);
  Serial.println("Client disonnected");

  // The client will actually be disconnected
  // when the function returns and 'client' object is detroyed
}
```

VID_20160312_240306565.ogv

Registro


RESET
PWR
FLASH
D2
TX
RX