ESP8266-EVB Programmer's Guide

The core of the microprocessor

Xtensa LX106 CPU It has a core ESP8089, which is the basis ESP8266
Here is the architecture of the microcontroller:

http://zeptobars.ru/ru/read/Espressif-ESP8266-wifi-serial-rs232-ESP8089-IoT

The virtual machine to compile (need to download approximatly 1 GB from internet).

If you use GNU/Linux Debian or Ubunta, you probably don't need to install VirtualBox and Vagrant to control the virtual machine from the command line.
On any operation system can be installed and run the virtual machine environment for Cross Compilation microcontroller ESP8266.

https://github.com/mziwisky/esp8266-dev

Before installation, you should already be installed the following programs:
VirtualBox (https://www.virtualbox.org/)
Vagrant (https://www.vagrantup.com/)

I tested it instalation under the MacBook Pro – Mac OS X v. 10.6.8. I had to also install the packages to access GitHub and GCC for assembly git from source.

The first firmware - flashing LED

After you configure a virtual machine Linux, try to flash Olimex ESP8266-EVB with MOD-WIFI-ESP8266-DEV.

http://homedevice.pro/product/esp8266-evb/

Chinese modules are similar, but you need to connect a source of DC power and a serial interface.

First LED blinked. To do this, follow the instructions:

https://olimex.wordpress.com/2015/01/29/esp8266-building-hello-world-blink-led-and-simple-web-server-to-drive-the-relay-and-check-button-status/

Notice in this example, SDK:

https://github.com/esp8266/esp8266-wiki/raw/master/sdk/esp_iot_sdk_v0.9.3_14_11_21.zip

and patches

https://github.com/esp8266/esp8266-wiki/raw/master/sdk/esp_iot_sdk_v0.9.3_14_11_21_patch1.zip

We use two cables that connect to the device:

Power cable:
http://homedevice.pro/product/sy0605e/

USB Cable Firmware:
http://homedevice.pro/product/usb-serial-cable-f/

The blue wire USB-Serial-Cable-F connects to Pin #2
Green wire USB-Serial-Cable-F to Pin #3
Red wire USB-Serial-Cable-F to Pin #4

By command lsusb - check the connection UART by USB cable. It should be something like:

```
Bus 001 Device 003: ID 067b:2303 Prolific Technology, Inc. PL2303 Serial Port
```

If the dial

```
ls /dev/tty*
```

we will see:

```
***
/dev/ttyUSB0
```

If everything is in order, go to

```
cd /opt/Espressif/ESP8266/ESP8266-EVB-blinkLED/
```

```
make
```

If no errors, it is possible to flash module. To do this, disconnect the power, Hold down the button and holding the button - connect the power. Further, from the working directory of the project - /opt/Espressif/ESP8266/ESP8266-EVB-blinkLED/ - type:

```
sudo make flash
```

Get the result (you have to wait):

```
/opt/Espressif/esptool-py/esptool.py --port /dev/ttyUSB0 write_flash 0x00000
firmware/0x00000.bin 0x40000 firmware/0x40000.bin

Connecting...
Erasing flash...
Writing at 0x00007000... (100 %)
Erasing flash...
Writing at 0x00063000... (100 %)

Leaving...
```

Go see - the LED starts blinking. The device is sewn correctly.

Now made of a web server from module ESP8266

Verify that the catalog /opt/Espressif/ have symlink esp8266_sdk to old SDK (on my system was a warning):

/opt/Espressif/ESP8266_SDK/esp_iot_sdk_v0.9.3/

To do this, move to the catalog

/opt/Espressif/ESP8266/esphttpd

In file /opt/Espressif/ESP8266/esphttpd/Makefile correct lines (it is necessary to add a slash between the path and file compiler):

```
# select which tools to use as compiler, librarian and linker
CC<---><----->:= $(XTENSA_TOOLS_ROOT)/xtensa-lx106-elf-gcc
AR<---><----->:= $(XTENSA_TOOLS_ROOT)/xtensa-lx106-elf-ar
LD<---><----->:= $(XTENSA_TOOLS_ROOT)/xtensa-lx106-elf-gcc
```

Before command make we must not forget to perform the command to copy the code (Without this code, the web server not assembly) to obtain a special code archiver, which is used by a Web server and consumes little memory, because it uses a stack to store variables.

```
git submodule init
git submodule update
```

```
make
```

Next, turn the power off, press, holding down the power on (switching module in flashing mode - We serve 5 volt flash) and enter the command:

```
sudo make flash
```

We get the results:

```
FW firmware/0x00000.bin
/opt/Espressif/esptool-py/esptool.py --port /dev/ttyUSB0 write_flash 0x00000
firmware/0x00000.bin 0x40000 firmware/0x40000.bin
Connecting...
Erasing flash...
Writing at 0x00008000... (100 %)
Erasing flash...
Writing at 0x00063000... (100 %)

Leaving...
```

Further, the power is turned back on, hold the button, turn on the power and enter the command for flashing Web pages:

```
sudo make htmlflash
```

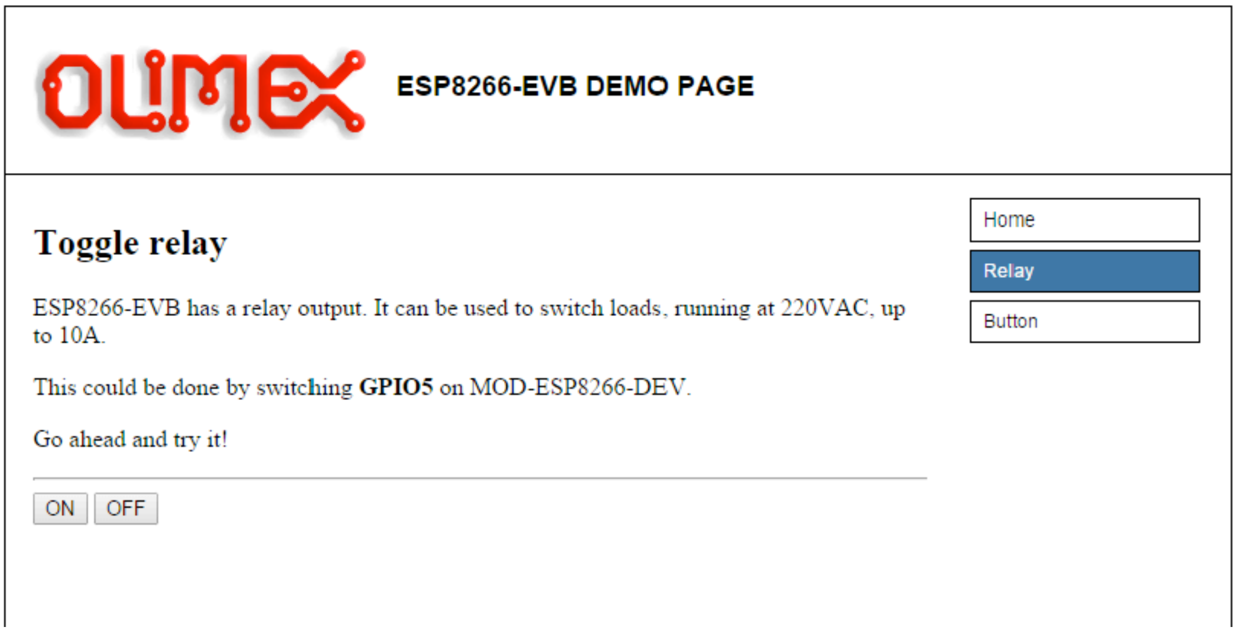
Result:

```
if [ $(stat -c '%s' webpages.espfs) -gt $(( 0x2E000 )) ]; then echo "webpages.espfs too big!";
false; fi
/opt/Espressif/esptool-py/esptool.py --port /dev/ttyUSB0 write_flash 0x12000 webpages.espfs
Connecting...
Erasing flash...
Writing at 0x0001b000... (100 %)

Leaving...
```

Now you can connect your computer to the ESP8266 unit by WiFi. Network name seems ESP_9C6165. Number depends from MAC address of your ESP8266 device.
We have to wait until our unit WiFi node start. And go to link http://192.168.4.1/index.html# , we see our website with 3 buttons: Home, Relay, Button.

By pressing the button Relay - we get on a page, where you can turn on and off Relay, By controlling the device remotely, which is connected to the relay. Unless something is not connected, nevertheless, we will see on the board idikatsiyu LED on the current status of the relay.


ESP8266-EVB DEMO PAGE

Toggle relay

ESP8266-EVB has a relay output. It can be used to switch loads, running at 220VAC, up to 10A.

This could be done by switching **GPIO5** on MOD-ESP8266-DEV.

Go ahead and try it!

Add relay control via HTTP GET request to a file relay.html :

```
28
29      function getQueryVariable(variable) {
30            var query = window.location.search.substring(1);
31            var vars = query.split("&");
32            for (var i = 0; i < vars.length; i++) {
33              var pair = vars[i].split("=");
34              if (pair[0] == variable) {
```

```
          return pair[1];
        }
      }
    return (false);
  }

  var press_btn = getQueryVariable('press_btn');
  if (press_btn == 'on') {
    setRelay(1);
  } else if (press_btn == 'off') {
    setRelay(0);
  }
```

The source code for the firmware ESP8266-EVB you can take in the branch "espain"

https://github.com/Ignat99/ESP8266/tree/espain

Update the following ways:

- update code in the directory: /opt/Espressif/ESP8266/esphttpd
- go to the directory /opt/Espressif/ESP8266/esphttpd and delete the file: webpages.espfs
- connect the cable USB, hold button to connect the power cord ESP8266-EVB
- enter the command:

```
sudo make htmlflash
```

Result:

```
cd html; find | sudo ../mkespfsimage/mkespfsimage  > ../webpages.espfs; cd ..
relay.html (65%)
ESP8266-EVB.jpg (100%)
index.html (65%)
140midley.min.js (45%)
home.html (88%)
logo.png (100%)
button.html (59%)
style.css (47%)
if [ $(stat -c '%s' webpages.espfs) -gt $(( 0x2E000 )) ]; then echo "webpages.espfs too big!";
false; fi
/opt/Espressif/esptool-py/esptool.py --port /dev/ttyUSB0 write_flash 0x12000 webpages.espfs
Connecting...
Erasing flash...
Writing at 0x0001b400... (100 %)

Leaving...
```

The relay can be the following query:

http://192.168.4.1/relay.html?press_btn=on
http://192.168.4.1/relay.html?press_btn=off

Android 2.3.7 application for relay control board ESP8266-EVB. When the application starts off the relay by pressing the button switches.

https://github.com/Ignat99/Android-Evaluation/tree/espain

It made under the mobile phone with API 10.

Web server ESP8266, which can measure the temperature, scan WiFi nets, share files on tftp, data through MQTT

http://harizanov.com/2015/02/wifi-thermostat-with-weekly-scheduler/

Highlights

- Option for on-board power supply
- Up to three high quality 10A relays
- Powered by the WiFi ESP8266 SoC module
- HTTP API to control the relays
- MQTT support
- NTP for network time
- HTTP daemon settings, including security/authentication setup
- HTTP UI for configuration and control
  - Thermostat function with weekly scheduling
  - Manual relay control
- Broadcast using HTTP GET to services like ThingSpeak and emonCMS
- Integration with ThingSpeak for charting/analytics visualization
- Temperature sensor support
  - DS18B20
  - DHT22

This server assembly with esp-Open-SDK. Therefore, we run a separate virtual machine in addition to a virtual machine on an official SDK.

https://github.com/mziwisky/esp8266-dev
Далее в запущенном Linux (Debian или Ubunta) надо установить новый автоконф:
http://ftp.gnu.org/gnu/autoconf/autoconf-latest.tar.gz

```
tar -xzf autoconf-latest.tar.gz
cd autoconf-2.69
./configure --prefix=/usr
make
make install
```

First, put the original esp-httpd (without the support of sensors and DDNS and no graphical interface and without thermostat MQTT)

Then do as instructed (the code for the Web server must be taken from the original repository).

http://git.spritesserver.nl/esphttpd.git/

Repository for advanced code (DDNS, GUI, MQTT, etc):

https://github.com/SCKStef/ESP8266_Relay_Board

For the advanced code necessary to make modifications. GPIO5 responsible for switching relays. The remaining 2 relay may be available if you buy additional device IO2:

http://homedevice.pro/product/mod-io2/

or plate with 4 relay:

http://homedevice.pro/product/mod-io/

Read more about the details of the installation is possible here:

http://www.esp8266.com/viewtopic.php?p=1629&sid=7382328539791b0c05f8b14a11a00a33#p1629

In Makefile we must add “/” after bin in the parameter XTENSA_TOOLS_ROOT,

```
cd crosstool-NG
ln -s .build builds
cd ..
```

In file:
esp-open-sdk/esp_iot_sdk_v1.0.0/include/c_types.h

need #if 0 (в веб-сервере Olimex не чего менять не надо)

```
#include <stdbool.h>
#if 0
typedef unsigned char       uint8_t;
```

change to #if 1

```
#include <stdbool.h>
#if 1
```

Makefile:

```
typedef unsigned char       uint8_t;export PATH=${PWD}/xtensa-lx106-elf/bin:$PATH
export XTENSA_TOOLS_ROOT=${PWD}/xtensa-lx106-elf/bin/
export SDK_BASE=${PWD}/esp_iot_sdk_v1.0.0/
export ESPTOOL=${PWD}/esptool/esptool.py
export ESPPORT=/dev/ttyUSB0
```

```
git clone http://git.spritesserver.nl/esphttpd.git/
cd esphttpd
git submodule init
git submodule update
```

Instructions for installing the native site:
https://nurdspace.nl/ESP8266/First_setup
The reference to the utility of the firmware code:
http://filez.zoobab.com/esp8266/esptool-0.0.2.zip

I have this utility doesn't work, so I just copied by directory /vagrant
firmvare first virtual machine to the web server of Olimex and updated the code from it:

```
export ESPTOOL=${PWD}/esptool/esptool
```

```
sudo make flash
sudo make htmlflash
```

POST API:
http://<Your_module_IP>/led.tpl
data - led=0 или led=1

This is an Android application, which can be controlled a relay by esphttpd POST API:
https://github.com/Ignat99/MjpegDemo

Assemble the source code can be under Java IDE Eclipse.

This application can also take video streaming with mjpeg-streamer. This software is installed on the video server, which will connect the camcorder. Either on IP CCTV cameras modules. The following repository there is a change in the code required for specific cameras (http://homedevice.pro/product/elp-usb100w04h-f170/):

https://github.com/Ignat99/mjpg-streamer

Or you can take the source code from the original site. To start the installation fulfill all the necessary packages:

```
sudo apt-get install imagemagick libav-tools libjpeg8-dev subversion
```

The downloaded and make mjpg-streamer:

```
sudo svn co https://svn.code.sf.net/p/mjpg-streamer/code/mjpg-streamer/ mjpg-streamer
 cd mjpg-streamer
make
```

Arduino IDE 1.6.1

https://olimex.wordpress.com/2015/03/31/programming-esp8266-evb-with-arduino-ide/

Additional information about Arduino sketches:

http://www.esp8266.com/viewforum.php?f=25