ESP Tool chain and SDK installation

Overview

This repository provides scripts to build a complete standalone SDK (with toolchain) for software development with the Espressif ESP8266 and ESP8266EX chips. And consists of:

1. Toolchain: Xtensa lx106 architecture toolchain (100% OpenSource), based on following projects:
    - https://github.com/jcmvbkbc/crosstool-NG
    - https://github.com/jcmvbkbc/gcc-xtensa
    - https://github.com/jcmvbkbc/newlib-xtensa
    - https://github.com/tommie/lx106-hal
2. SDK: ESP8266 IoT SDK from Espressif Systems. This component is only partially open source, (some libraries are provided as binary blobs). OpenSource components of the SDK are based on:
    - lwIP, http://savannah.nongnu.org/projects/lwip/
    - Contiki, http://www.contiki-os.org/
    - axTLS, http://axtls.sourceforge.net/
    - wpa_supplicant, http://w1.fi/wpa_supplicant/ (source withheld by Espressif)
    - net80211/ieee80211 (FreeBSD WiFi stack), http://www.unix.com/man-page/freebsd/9/NET80211 (source withheld by Espressif)

Requirements and Dependencies

To build the standalone SDK and toolchain, you need a GNU/POSIX system with the standard GNU development tools installed: bash, gcc, binutils, flex, bison, etc.

Debian/Ubuntu 14.04:

```
sudo apt-get install make unrar-free autoconf automake libtool gcc g++ gperf
sudo apt-get install flex bison texinfo gawk ncurses-dev libexpat-dev
sudo apt-get install python-dev python python-serial
sudo apt-get install sed git unzip bash help2man wget bzip2
```

Later Debian/Ubuntu versions may require:

```
sudo apt-get install libtool-bin
```

Building the Toolchain

The project can be built in two modes:

1. Where the toolchain and tools are kept separate from the vendor IoT SDK which contains binary

blobs. This makes licensing more clear, and helps facilitate upgrades to vendor SDK releases. To build the seperated toolchain and SDK: make STANDALONE=n
2. A completely standalone ESP8266 SDK with the vendor SDK files merged into the toolchain. This mode makes it easier to build software (no additinal -I and -L flags are needed), but redistributability of this build is unclear and upgrades to newer vendor IoT SDK releases are complicated. This mode is default for local builds. To build the self-contained, standalone toolchain+SDK: make STANDALONE=y

```
mkdir esp8266; cd esp8266
git clone --recursive https://github.com/pfalcon/esp-open-sdk.git
cd esp-open-sdk
make STANDALONE=y
```

Using the Toolchain

After make, the following sub directories are of interest:

- sdk/ (all sdk expressif libraries and include files)
- extensa-lx-106-elf/ (all the executables, compiler, linker, etc.)

Add xtensa-lx106-elf/bin/ subdirectory to your PATH environment variable to execute xtensa-lx106-elf-gcc and other tools. Edit .profile to add the path to the bin directories, by adding:

```
PATH=$PATH:/home/oscar/esp8266/esp-open-sdk/xtensa-lx106-elf/bin
ESP8266_SDK_ROOT=/home/oscar/esp8266/esp-open-sdk/sdk
export ESP8266_SDK_ROOT
#Logout/login for the .profile to take effect
```

If you chose the non-standalone SDK, run the compiler with the corresponding include and lib dir flags. The extra -I and -L flags are not needed when using the standalone SDK.

```
xtensa-lx106-elf-gcc -I$(THISDIR)/sdk/include -L$(THISDIR)/sdk/lib
```

Pulling updates

The project is updated from time to time, to get updates and prepare to build a new SDK, run:

```
$ make clean
$ git pull
$ git submodule sync
$ git submodule update --init
```

If you don't issue make clean (which causes toolchain and SDK to be rebuilt from scratch on next make), you risk getting broken/inconsistent results.