ESP8266

This guide explains how to set up the development environment and create applications for ESP8266 on Linux systems (tested on Ubuntu 14.04).

Connecting ESP8266 to PC (Linux)

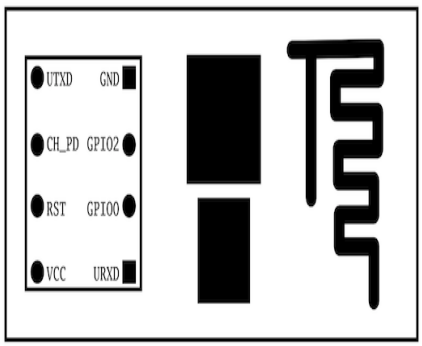
To connect the ESP8266 chip to PC, a 3.3V USB-to-TTL connector is required.

> This section guides you through the connection process for the ESP8266-01 module, please refer to the ESP8266 documentation for information about other modules.

> If you are going to connect ESP8266 in another way, make sure that the power source does not exceed ~3.6V

Flash mode

ESP8266	USB-to-TTL
URXD	TXD
UTXD	RXD
CH_PD	3.3V
GND	GND
VCC	3.3V
GPIO0	GND
GPIO2	3.3V



ESP8266 WiFi Pinout
Top View (Not to scale)

Run mode

ESP8266	USB-to-TTL
URXD	TXD
UTXD	RXD
CH_PD	3.3V
GND	GND
VCC	3.3V

Installing required software

Before building the SDK for the ESP8266 platform, you need to install ESP8266 RTOS SDK, Xtensa GCC toolchain, and esptool.py utitlity for uploading firmware.

1. Install prerequisites:

```
sudo apt-get install autoconf libtool libtool-bin bison build-essential gawk git
gperf flex texinfo libtool libncurses5-dev libc6-dev-amd64 python-serial
libexpat-dev python-setuptools
```

2. Install ESP RTOS SDK:

```
export ESPRESSIF_HOME=/opt/Espressif
sudo mkdir -p $ESPRESSIF_HOME
sudo chown `whoami`:root $ESPRESSIF_HOME
cd $ESPRESSIF_HOME
export ESP_SDK_HOME=$ESPRESSIF_HOME/esp-rtos-sdk
git clone https://github.com/espressif/esp_iot_rtos_sdk.git $ESP_SDK_HOME
cd $ESP_SDK_HOME
git reset --hard 169a436ce10155015d056eab80345447bfdfade5
wget -O lib/libhal.a
https://github.com/esp8266/esp8266-wiki/raw/master/libs/libhal.a
cd $ESP_SDK_HOME/include/lwip/arch
sed -i "s/#include \"c_types.h\"/\/\/#include \"c_types.h\"/" cc.h
```

3. Install Xtensa-lx106 GNU toolchain:

```
cd $ESPRESSIF_HOME
git clone -b lx106 git://github.com/jcmvbkbc/crosstool-NG.git
cd crosstool-NG
./bootstrap && ./configure --prefix=`pwd` && make && sudo make install
./ct-ng xtensa-lx106-elf
./ct-ng build
export PATH=$PATH:$ESPRESSIF_HOME/crosstool-NG/builds/xtensa-lx106-elf/bin
```

4. Install improved esptool.py:

```
cd $ESPRESSIF_HOME
git clone https://github.com/RostakaGmfun/esptool.git
cd esptool
sudo python setup.py install
```

Creating applications based on C SDK

Building C SDK

To build the C SDK for the ESP8266 platform, you will need to use GNU toolchain for Xtensa (which can be installed from sources as described above).

To create applications based on the C SDK, at first you should build a static library from the generated SDK. To do so, generate the C SDK in Admin UI, then extract the archive, `cd` to it and execute the following:

```
rm -rf build
mkdir -p build
cd build
cmake -DCMAKE_INSTALL_PREFIX=<YOUR_DESTINATION_PATH> -DKAA_PLATFORM=esp8266
-DCMAKE_TOOLCHAIN_FILE=../toolchains/esp8266.cmake ..
make install
```

The C SDK will be installed to the <*YOUR_DESTINATION_PATH*> directory. For more details on building the C SDK, please refer to this page.

Tips & tricks

- According to the ESP8266 SDK programming guide, the user application starts its execution in the `user_init()` function.
- To make the Kaa SDK work, you should establish network connection through WiFi. For more details on WiFi configuration, please refer to the ESP8266 official documentation.
- To see any output on the serial port, a UART driver is required.

You can use ConfigurationDemo (available in Kaa Sandbox) as an example of the UART and WiFi functionality coupled together with the Kaa endpoint SDK.

Linker Script

The ESP SDK provides a set of linker scripts for different Flash configurations. However, to run a Kaa client, some modifications to those scripts are required.

Basically, there are two sections where code can be placed. The first one is the `.text` section, which maps to the `iram` segment, and another one is the `.irom0.text` section, which maps to the `irom` segment.

The issue is that there is not enough space to place the Kaa SDK code in the `iram` segment (the `.text` section is mapped to the `iram` segment by default) . This is why the default SDK linker scripts are modified to force linking the Kaa SDK code to the `irom` segment.

You can find valid linker scripts in the `ld` directory of ConfigurationDemo for the ESP8266 platform.

Example

As an example, you can try out ConfigurationDemo. Put generated SDK from Kaa Sandbox into `libs/kaa` folder.

Connect ESP8266 to your PC through a USB-to-TTL connector as mentioned above, and execute the following in your terminal from demo root folder:

```
./build.sh deploy
```

The `deploy` command of `build.sh` script builds the C SDK for the ESP8266 platform, builds the ConfigurationDemo itself and a platform-dependent glue code for ESP8266 (WiFi connectivity, a UART driver), creates firmware images (`build/0x00000.bin` and `build/0x40000.bin`) and, finally, writes these images to Flash at the corresponding addresses. You will be prompted of your WiFi station SSID and password, so that ESP8266 can connect to it.
