AT指令版本编译指导

wireless-tag

版本1.0.0

启明云端

版权所有©2023

目录

wireless-tag

前言

本文档介绍了基于ESP-IDF开发环境配置、编译和烧录AT指令版本的操作使用方法。

wireless-tag

1、开始使用ESP-IDF

在编译 ESP-AT 项目之前，先了解 ESP-IDF，因为 ESP-AT 是基于 ESP-IDF 开发的。

1.1、环境安装

(1) 更新软件源：

```
sudo apt-get update -y
```

(2) 下载ESP-IDF相关依赖文件和库：

```
sudo apt-get install git wget flex bison gperf python3 python3-pip python3-setuptools cmake
ninja-build ccache libffi-dev libssl-dev dfu-util libusb-1.0-0
```

(3) 创建python3软链接

```
sudo ln -s /usr/bin/python3 /usr/bin/python
```

1.2、下载ESP-IDF

(1) 下载ESP-IDF

```
git clone https://gitee.com/EspressifSystems/esp-idf.git
```

(2) 下载ESP-IDF工具

```
git clone https://gitee.com/EspressifSystems/esp-gitee-tools.git
```

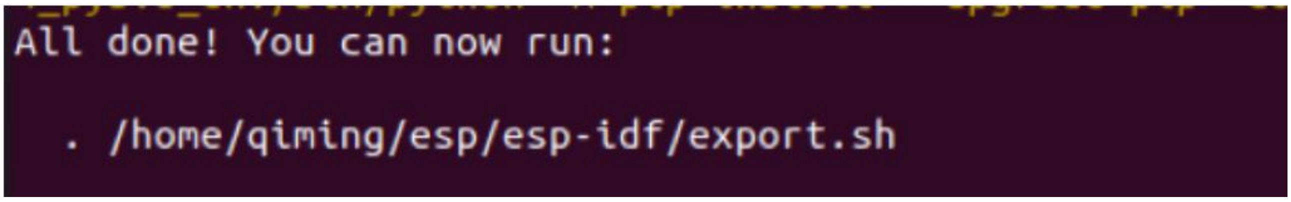
(3) 下载ESP-IDF子模块

```
cd  esp-gitee-tools
export EGT_PATH=$(pwd) 将工具路径添加到环境变量
cd  ../esp-idf/
$EGT_PATH/submodule-update.sh    在esp-idf文件夹里使用工具下载子模块
```

(4) 在esp-idf目录下载安装工具链

```
export IDF_GITHUB_ASSETS="dl.espressif.com/github_assets"
./install.sh
```

(5) 当出现类似如下图中的字样表明环境搭建成功了

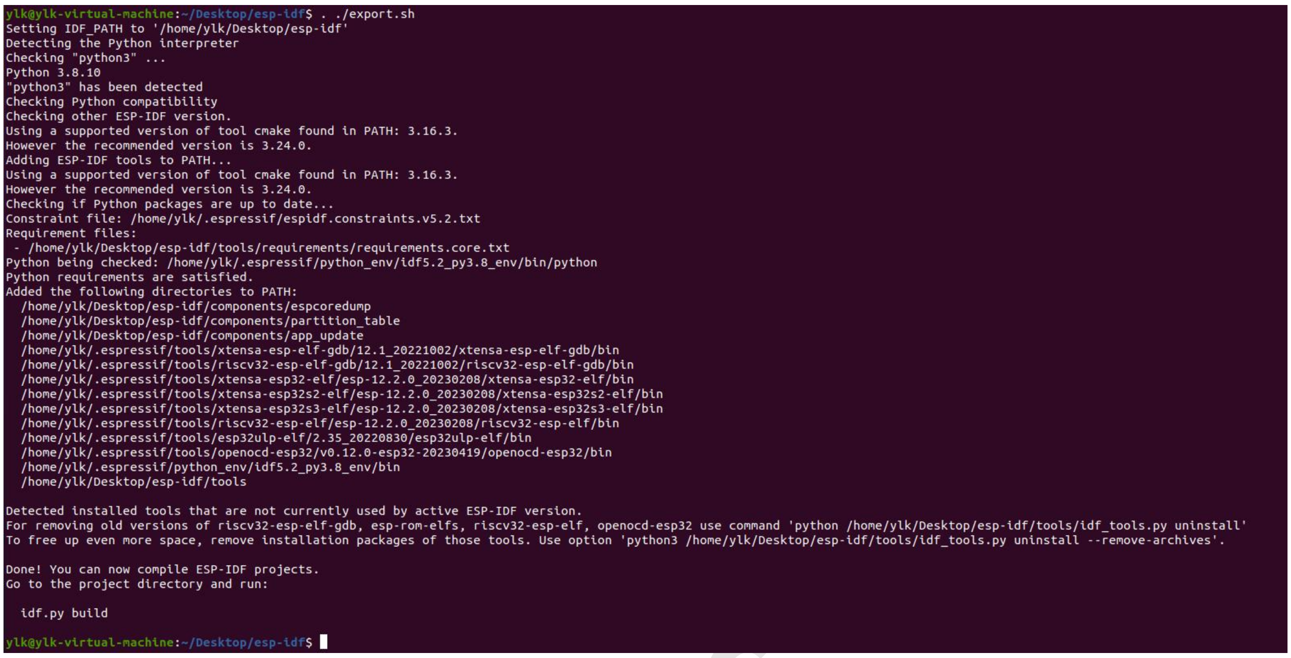


```
All done! You can now run:

  . /home/qiming/esp/esp-idf/export.sh
```

wireless-tag

(6) 接着在esp-idf目录下运行. ./export.sh进行配置工具链，如下图表明配置成功，至此就可以使用idf.py去编译ESP-IDF工程了



```
ylk@ylk-virtual-machine:~/Desktop/esp-idf$ . ./export.sh
Setting IDF_PATH to '/home/ylk/Desktop/esp-idf'
Detecting the Python interpreter
Checking "python3" ...
Python 3.8.10
"python3" has been detected
Checking Python compatibility
Checking other ESP-IDF version.
Using a supported version of tool cmake found in PATH: 3.16.3.
However the recommended version is 3.24.0.
Adding ESP-IDF tools to PATH...
Using a supported version of tool cmake found in PATH: 3.16.3.
However the recommended version is 3.24.0.
Checking if Python packages are up to date...
Constraint file: /home/ylk/.espressif/espidf.constraints.v5.2.txt
Requirement files:
 - /home/ylk/Desktop/esp-idf/tools/requirements/requirements.core.txt
Python being checked: /home/ylk/.espressif/python_env/idf5.2_py3.8_env/bin/python
Python requirements are satisfied.
Added the following directories to PATH:
  /home/ylk/Desktop/esp-idf/components/espcoredump
  /home/ylk/Desktop/esp-idf/components/partition_table
  /home/ylk/Desktop/esp-idf/components/app_update
  /home/ylk/.espressif/tools/xtensa-esp-elf-gdb/12.1_20221002/xtensa-esp-elf-gdb/bin
  /home/ylk/.espressif/tools/riscv32-esp-elf-gdb/12.1_20221002/riscv32-esp-elf-gdb/bin
  /home/ylk/.espressif/tools/xtensa-esp32-elf/esp-12.2.0_20230208/xtensa-esp32-elf/bin
  /home/ylk/.espressif/tools/xtensa-esp32s2-elf/esp-12.2.0_20230208/xtensa-esp32s2-elf/bin
  /home/ylk/.espressif/tools/xtensa-esp32s3-elf/esp-12.2.0_20230208/xtensa-esp32s3-elf/bin
  /home/ylk/.espressif/tools/riscv32-esp-elf/esp-12.2.0_20230208/riscv32-esp-elf/bin
  /home/ylk/.espressif/tools/esp32ulp-elf/2.35_20220830/esp32ulp-elf/bin
  /home/ylk/.espressif/tools/openocd-esp32/v0.12.0-esp32-20230419/openocd-esp32/bin
  /home/ylk/.espressif/python_env/idf5.2_py3.8_env/bin
  /home/ylk/Desktop/esp-idf/tools

Detected installed tools that are not currently used by active ESP-IDF version.
For removing old versions of riscv32-esp-elf-gdb, esp-rom-elfs, riscv32-esp-elf, openocd-esp32 use command 'python /home/ylk/Desktop/esp-idf/tools/idf_tools.py uninstall'
To free up even more space, remove installation packages of those tools. Use option 'python3 /home/ylk/Desktop/esp-idf/tools/idf_tools.py uninstall --remove-archives'.

Done! You can now compile ESP-IDF projects.
Go to the project directory and run:

  idf.py build

ylk@ylk-virtual-machine:~/Desktop/esp-idf$ █
```

1.3、测试hello_world例程以熟悉ESP-IDF的基本使用方法

(1) 首先 cd 到hello_world例程目录：

```
cd  esp-idf/examples/get-started/hello_world
```

(2) 设置要烧录的目标芯片:

```
idf.py set-target esp32
```

(3) 编译工程：

```
idf.py build
```

(4) 烧录固件到esp32设备：

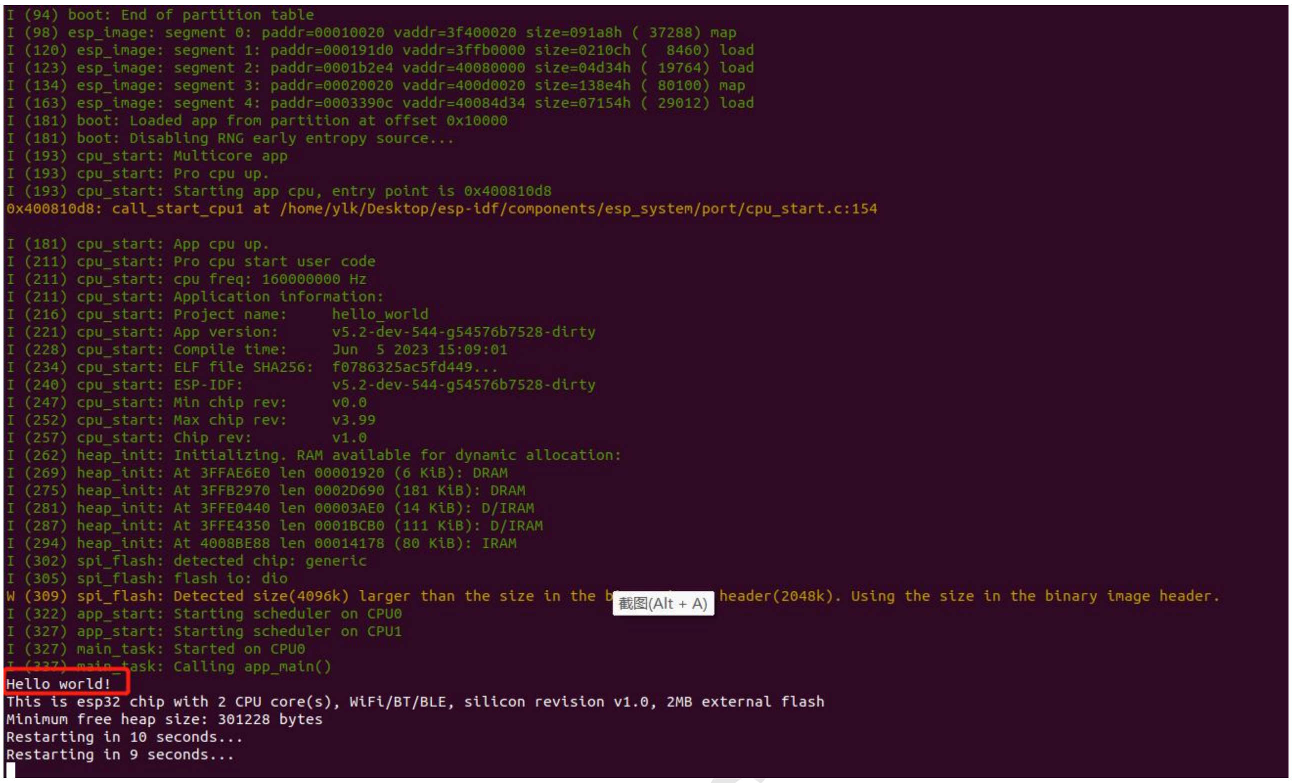
```
idf.py -p /dev/ttyUSB0 flash -b 115200
```

注意：/dev/ttyUSB0是我的esp32设备在Ubuntu上的串口号，需要根据实际情况修改

(5) 监视输出：

```
idf.py -p /dev/ttyUSB0 monitor
```

wireless



```
I (94) boot: End of partition table
I (98) esp_image: segment 0: paddr=00010020 vaddr=3f400020 size=091a8h ( 37288) map
I (120) esp_image: segment 1: paddr=000191d0 vaddr=3ffb0000 size=0210ch (  8460) load
I (123) esp_image: segment 2: paddr=0001b2e4 vaddr=40080000 size=04d34h ( 19764) load
I (134) esp_image: segment 3: paddr=00020020 vaddr=400d0020 size=138e4h ( 80100) map
I (163) esp_image: segment 4: paddr=0003390c vaddr=40084d34 size=07154h ( 29012) load
I (181) boot: Loaded app from partition at offset 0x10000
I (181) boot: Disabling RNG early entropy source...
I (193) cpu_start: Multicore app
I (193) cpu_start: Pro cpu up.
I (193) cpu_start: Starting app cpu, entry point is 0x400810d8
0x400810d8: call_start_cpu1 at /home/ylk/Desktop/esp-idf/components/esp_system/port/cpu_start.c:154

I (181) cpu_start: App cpu up.
I (211) cpu_start: Pro cpu start user code
I (211) cpu_start: cpu freq: 160000000 Hz
I (211) cpu_start: Application information:
I (216) cpu_start: Project name:     hello_world
I (221) cpu_start: App version:      v5.2-dev-544-g54576b7528-dirty
I (228) cpu_start: Compile time:     Jun  5 2023 15:09:01
I (234) cpu_start: ELF file SHA256:  f0786325ac5fd449...
I (240) cpu_start: ESP-IDF:          v5.2-dev-544-g54576b7528-dirty
I (247) cpu_start: Min chip rev:     v0.0
I (252) cpu_start: Max chip rev:     v3.99
I (257) cpu_start: Chip rev:         v1.0
I (262) heap_init: Initializing. RAM available for dynamic allocation:
I (269) heap_init: At 3FFAE6E0 len 00001920 (6 KiB): DRAM
I (275) heap_init: At 3FFB2970 len 0002D690 (181 KiB): DRAM
I (281) heap_init: At 3FFE0440 len 00003AE0 (14 KiB): D/IRAM
I (287) heap_init: At 3FFE4350 len 0001BCB0 (111 KiB): D/IRAM
I (294) heap_init: At 4008BE88 len 00014178 (80 KiB): IRAM
I (302) spi_flash: detected chip: generic
I (305) spi_flash: flash io: dio
W (309) spi_flash: Detected size(4096k) larger than the size in the b 截图(Alt + A) header(2048k). Using the size in the binary image header.
I (322) app_start: Starting scheduler on CPU0
I (327) app_start: Starting scheduler on CPU1
I (327) main_task: Started on CPU0
I (337) main_task: Calling app_main()
Hello world!
This is esp32 chip with 2 CPU core(s), WiFi/BT/BLE, silicon revision v1.0, 2MB external flash
Minimum free heap size: 301228 bytes
Restarting in 10 seconds...
Restarting in 9 seconds...
```

至此，您已经会ESP-IDF的基本使用了，接下来开始使用ESP-AT

2、获取ESP-AT

git clone https://gitee.com/EspressifSystems/esp-at.git

3、安装编译环境

1、cd esp-at/ 进入目录

2、执行./build.py install 以安装环境。此工具会自动安装 Python 软件包、ESP-IDF 仓库以及 ESP-IDF 使用的编译器和工具。

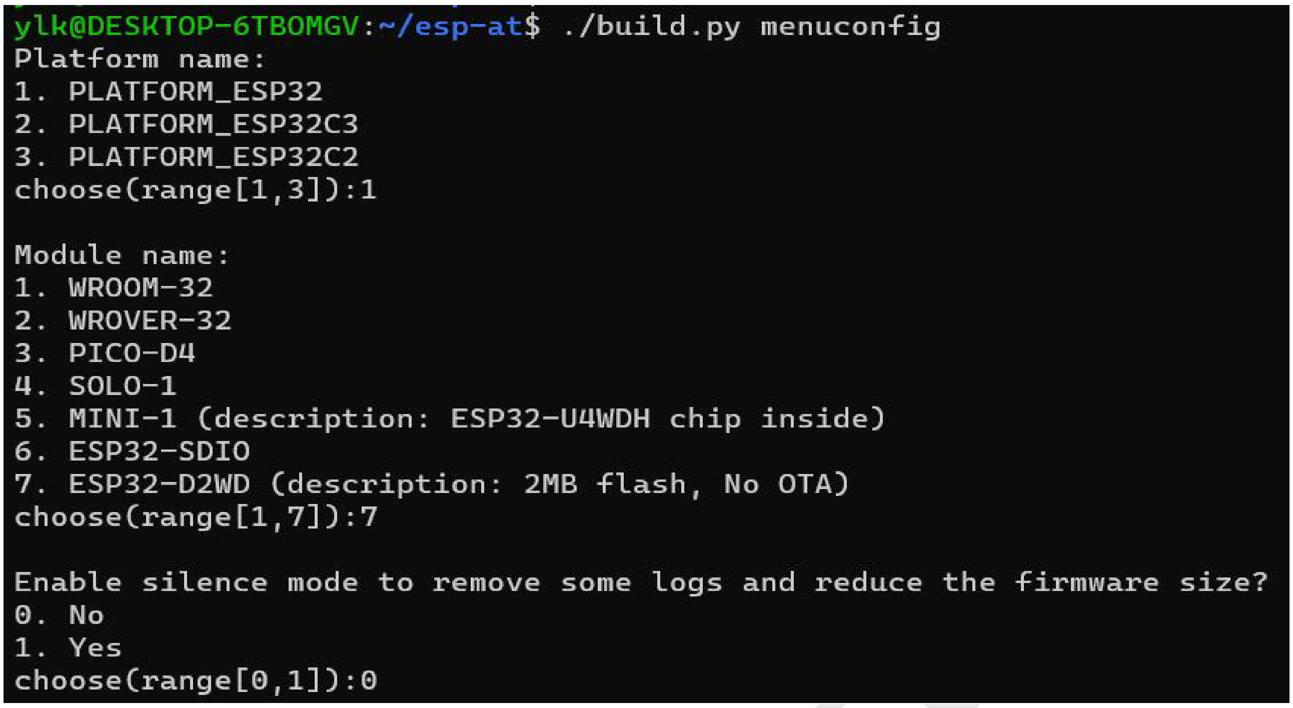
4、配置、编译、烧录

4.1 配置项目

执行./build.py menuconfig,此菜单设置特定于项目的配置，例如更改AT端口引脚，启用经典蓝牙功能等。如果未进行任何更改，它将使用默认配置运行。

如果您是第一次编译AT固件，您需要按照提示步骤选择ESP32系列芯片型号(silence mode通常设置为0)，如下图：

wireless

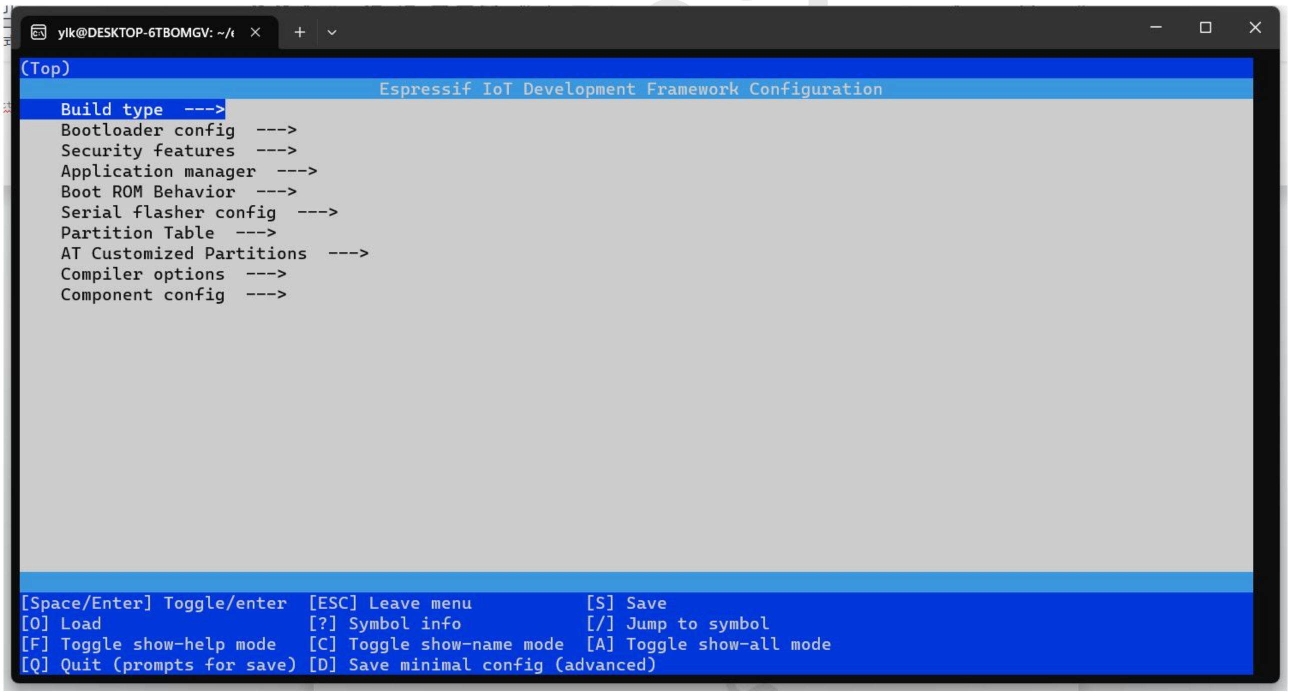


```
ylk@DESKTOP-6TBOMGV:~/esp-at$ ./build.py menuconfig
Platform name:
1. PLATFORM_ESP32
2. PLATFORM_ESP32C3
3. PLATFORM_ESP32C2
choose(range[1,3]):1

Module name:
1. WROOM-32
2. WROVER-32
3. PICO-D4
4. SOLO-1
5. MINI-1 (description: ESP32-U4WDH chip inside)
6. ESP32-SDIO
7. ESP32-D2WD (description: 2MB flash, No OTA)
choose(range[1,7]):7

Enable silence mode to remove some logs and reduce the firmware size?
0. No
1. Yes
choose(range[0,1]):0
```

正确安装和配置环境后将弹出以下固件配置界面：

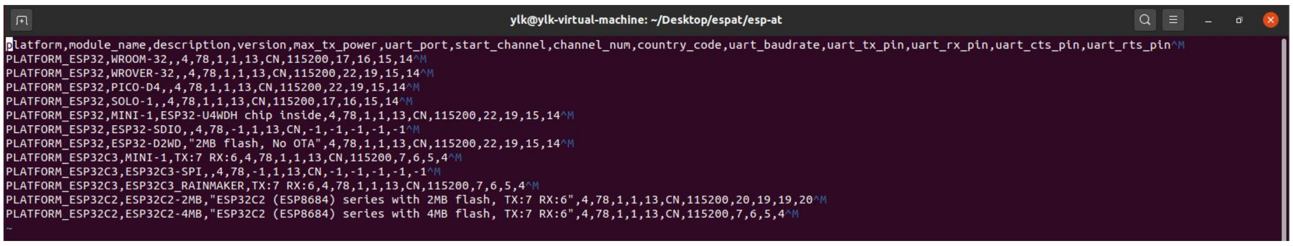


```
(Top)
                         Espressif IoT Development Framework Configuration
    Build type  --->
    Bootloader config  --->
    Security features  --->
    Application manager  --->
    Boot ROM Behavior  --->
    Serial flasher config  --->
    Partition Table  --->
    AT Customized Partitions  --->
    Compiler options  --->
    Component config  --->

[Space/Enter] Toggle/enter  [ESC] Leave menu           [S] Save
[O] Load                    [?] Symbol info            [/] Jump to symbol
[F] Toggle show-help mode   [C] Toggle show-name mode  [A] Toggle show-all mode
[Q] Quit (prompts for save) [D] Save minimal config (advanced)
```

修改发送AT指令的串口需要在下面这个文件中找到对应的引脚位置进行设置

```
vi  components/customized_partitions/raw_data/factory_param/factory_param_data.csv
```



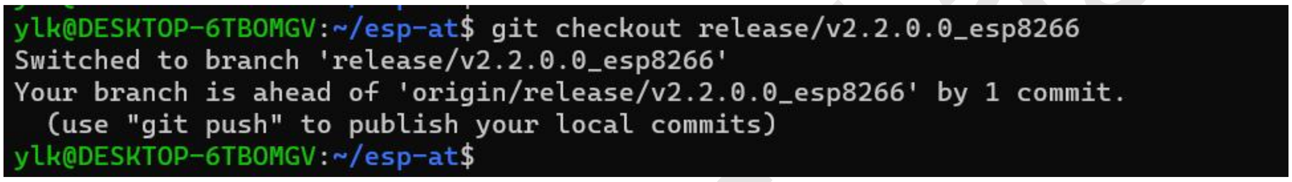
ylk@ylk-virtual-machine: ~/Desktop/espat/esp-at

```
platform,module_name,description,version,max_tx_power,uart_port,start_channel,channel_num,country_code,uart_baudrate,uart_tx_pin,uart_rx_pin,uart_cts_pin,uart_rts_pin^M
PLATFORM_ESP32,WROOM-32,,4,78,1,1,13,CN,115200,17,16,15,14^M
PLATFORM_ESP32,WROVER-32,,4,78,1,1,13,CN,115200,22,19,15,14^M
PLATFORM_ESP32,PICO-D4,,4,78,1,1,13,CN,115200,22,19,15,14^M
PLATFORM_ESP32,SOLO-1,,4,78,1,1,13,CN,115200,17,16,15,14^M
PLATFORM_ESP32,MINI-1,ESP32-U4WDH chip inside,4,78,1,1,13,CN,115200,22,19,15,14^M
PLATFORM_ESP32,ESP32-SDIO,,4,78,-1,1,13,CN,-1,-1,-1,-1,-1^M
PLATFORM_ESP32,ESP32-D2WD,"2MB flash, No OTA",4,78,1,1,13,CN,115200,22,19,15,14^M
PLATFORM_ESP32C3,MINI-1,TX:7 RX:6,4,78,1,1,13,CN,115200,7,6,5,4^M
PLATFORM_ESP32C3,ESP32C3-SPI,,4,78,-1,1,13,CN,-1,-1,-1,-1,-1^M
PLATFORM_ESP32C3,ESP32C3_RAINMAKER,TX:7 RX:6,4,78,1,1,13,CN,115200,7,6,5,4^M
PLATFORM_ESP32C2,ESP32C2-2MB,"ESP32C2 (ESP8684) series with 2MB flash, TX:7 RX:6",4,78,1,1,13,CN,115200,20,19,19,20^M
PLATFORM_ESP32C2,ESP32C2-4MB,"ESP32C2 (ESP8684) series with 4MB flash, TX:7 RX:6",4,78,1,1,13,CN,115200,7,6,5,4^M
~
```

注意：

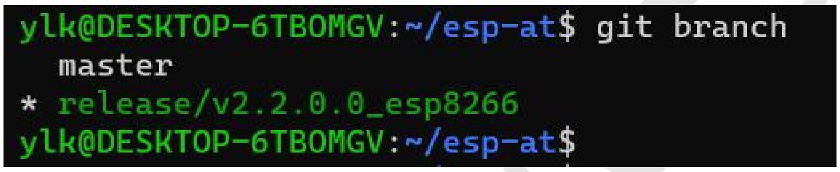
1、如果需要编译其他ESP32系列芯片的固件，需要将esp-at目录下的build目录删除，然后重新执行./build.py menuconfig

2、如果需要为ESP8266系列芯片编译AT固件，您需要切换至esp8266的SDK分支，在esp-at目录下执行：`git checkout release/v2.2.0.0_esp8266`



```
ylk@DESKTOP-6TBOMGV:~/esp-at$ git checkout release/v2.2.0.0_esp8266
Switched to branch 'release/v2.2.0.0_esp8266'
Your branch is ahead of 'origin/release/v2.2.0.0_esp8266' by 1 commit.
  (use "git push" to publish your local commits)
ylk@DESKTOP-6TBOMGV:~/esp-at$
```

查看当前分支：



```
ylk@DESKTOP-6TBOMGV:~/esp-at$ git branch
  master
* release/v2.2.0.0_esp8266
ylk@DESKTOP-6TBOMGV:~/esp-at$
```

4.2 编译项目

配置完成后保存退出，然后执行./build.py build进行编译

4.3 烧录程序

./build.py -p (PORT) flash

注：PORT是设备在系统上的串口号

固件在/esp-at/build/factory目录下

wireless-tag