ESP32 系列教程之二：Linux 搭建 esp-idf 环境

Date: 2021.7.09

Version: V1.1

关于文档

本文档为 ESP32 教程系列，旨在为客户进行 ESP32 系列芯片开发提供环境搭建、工程示例演示等方面的参考文档及视频演示，降低 ESP32 系列芯片、模组开发的入门难度。

ESP32 教程系列文档主要参考于乐鑫官网提供的 ESP32 入门教程：https://docs.espressif.com/projects/esp-idf/zh_CN/latest/esp32/get-started/index.html。由于个人经验有限，文档编写过程中难免存在失误、错漏之处，欢迎广大开发爱好者对本文档提出批评建议。

版本信息：

日期	版本	作者	说明
2021.6.22	V1.0	Amiliya	首次发布
2021.7.09	V1.1	Amiliya	修改部分描述 更改部分图片 增加常见问题处理（章节 4.3） 文档格式调整

文档变更通知：以启明云端官网版本为准，恕不另行通知。

意见提交邮箱：amiliya@wireless-tag.com

目　录

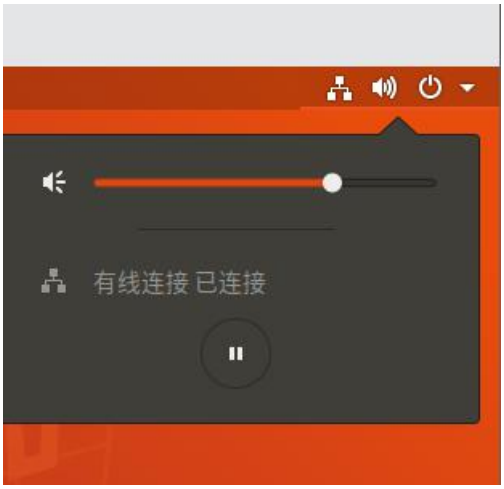
1 搭建 esp-idf 环境

1.1 安装虚拟机与 Ubuntu

本教程 esp-idf 环境搭建默认处于 Linux 环境下，非 Linux 环境请参考 ESP32 系列教程之一：安装虚拟机与 Ubuntu。

1.2 检查网络

打开终端，查看 Ubuntu 是否连接网络

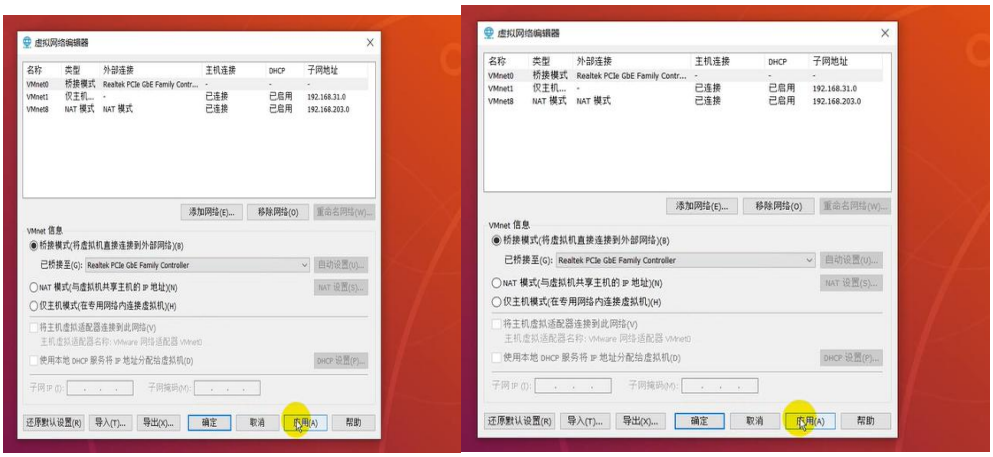


或使用命令检测：`ping www.baidu.com`

```
amiliya@ubuntu:~$ ping www.baidu.com
PING www.a.shifen.com (14.215.177.38) 56(84) bytes of data.
64 bytes from 14.215.177.38 (14.215.177.38): icmp_seq=1 ttl=55 time=6.46 ms
64 bytes from 14.215.177.38 (14.215.177.38): icmp_seq=2 ttl=55 time=6.65 ms
64 bytes from 14.215.177.38 (14.215.177.38): icmp_seq=3 ttl=55 time=7.03 ms
64 bytes from 14.215.177.38 (14.215.177.38): icmp_seq=4 ttl=55 time=6.83 ms
64 bytes from 14.215.177.38 (14.215.177.38): icmp_seq=5 ttl=55 time=7.51 ms
^C
--- www.a.shifen.com ping statistics ---
```

若未联网，按以下步骤设置：

编辑->虚拟网络编辑器->更改设置->桥接模式(桥接至电脑 ip 描述)->应用->确定



1.3 下载 Python 软件包

```
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
```

```
amiliya@ubuntu:~$ sudo apt-get install git wget flex bison gperf python3 python3-pip python3-setuptools cmake ninja-build ccache libffi-dev libssl-dev dfu-util libus
[sudo] amiliya 的密码：
正在读取软件包列表... 完成
正在分析软件包的依赖关系树
正在读取状态信息... 完成
libusb-1.0-0 已经是最新版 (2:1.0.21-2)。
libusb-1.0-0 已设置为手动安装。
python3 已经是最新版 (3.6.7-1~18.04)。
python3 已设置为手动安装。
wget 已经是最新版 (1.19.4-1ubuntu2.2)。
wget 已设置为手动安装。
将会同时安装下列软件：
  build-essential cmake-data cpp-7 dh-python dpkg-dev fakeroot g++ g++-7 gcc gcc-7 gcc-7-base gcc-8-base git-man libalgorithm-diff-perl libalgorithm-diff-xs-perl
  libalgorithm-merge-perl libasan4 libatomic1 libbison-dev libc-dev-bin libc6 libc6-dbg libc6-dev libcc1-0 libcilkrts5 libcurl4 libdpkg-perl liberror-perl libexpat1
  libfakeroot libfl-dev libfl2 libgcc-7-dev libgcc1 libgomp1 libitm1 libjsoncpp1 liblsan0 libmpx2 libpython3-dev libpython3.6 libpython3.6-dev libpython3.6-minimal
  libpython3.6-stdlib libquadmath0 librhash0 libsigsegv2 libssl1.1 libstdc++-7-dev libstdc++6 libtsan0 libubsan0 libuv1 linux-libc-dev m4 make manpages-dev python-pi
  python3-dev python3-distutils python3-lib2to3 python3-wheel python3.6 python3.6-dev python3.6-minimal
建议安装：
  bison-doc distcc cmake-doc gcc-7-locales debian-keyring flex-doc g++-multilib g++-7-multilib gcc-7-doc libstdc++6-7-dbg gcc-multilib autoconf automake libtool gcc-
  gcc-7-multilib libgcc1-dbg libgomp1-dbg libitm1-dbg libatomic1-dbg libasan4-dbg liblsan0-dbg libtsan0-dbg libubsan0-dbg libcilkrts5-dbg libmpx2-dbg libquadmath0-db
  git-daemon-run | git-daemon-sysvinit git-doc git-el git-email git-gui gitk gitweb git-svn git-mediawiki git-svn glibc-doc bzr libssl-doc libstdc++-7-doc m4-doc mak
  python-setuptools-doc python3.6-venv python3.6-doc binfmt-support
下列【新】软件包将被安装：
```

```
  python3.6-minimal
升级了 17 个软件包，新安装了 61 个软件包，要卸载 0 个软件包，有 449 个软件包未被升级。
需要下载 91.6 MB/116 MB 的归档。
解压缩后会消耗 279 MB 的额外空间。
您希望继续执行吗？ [Y/n] y
获取:1 http://us.archive.ubuntu.com/ubuntu bionic/main amd64 libsigsegv2 amd64 2.12-1 [14.7 kB]
获取:2 http://us.archive.ubuntu.com/ubuntu bionic/main amd64 m4 amd64 1.4.18-1 [197 kB]
获取:3 http://us.archive.ubuntu.com/ubuntu bionic/main amd64 flex amd64 2.6.4-6 [316 kB]
获取:4 http://us.archive.ubuntu.com/ubuntu bionic/main amd64 libbison-dev amd64 2:3.0.4.dfsg-1build1 [339 kB]
获取:5 http://us.archive.ubuntu.com/ubuntu bionic/main amd64 bison amd64 2:3.0.4.dfsg-1build1 [266 kB]
获取:6 http://us.archive.ubuntu.com/ubuntu bionic-updates/main amd64 libc-dev-bin amd64 2.27-3ubuntu1.4 [71.8 kB]
获取:7 http://us.archive.ubuntu.com/ubuntu bionic-updates/main amd64 linux-libc-dev amd64 4.15.0-144.148 [1,003 kB]
获取:8 http://us.archive.ubuntu.com/ubuntu bionic-updates/main amd64 libc6-dev amd64 2.27-3ubuntu1.4 [2,585 kB]
获取:9 http://us.archive.ubuntu.com/ubuntu bionic-updates/main amd64 libitm1 amd64 8.4.0-1ubuntu1~18.04 [27.9 kB]
获取:10 http://us.archive.ubuntu.com/ubuntu bionic-updates/main amd64 libatomic1 amd64 8.4.0-1ubuntu1~18.04 [9,192 B]
获取:11 http://us.archive.ubuntu.com/ubuntu bionic-updates/main amd64 libasan4 amd64 7.5.0-3ubuntu1~18.04 [358 kB]
获取:12 http://us.archive.ubuntu.com/ubuntu bionic-updates/main amd64 liblsan0 amd64 8.4.0-1ubuntu1~18.04 [133 kB]
获取:13 http://us.archive.ubuntu.com/ubuntu bionic-updates/main amd64 libtsan0 amd64 8.4.0-1ubuntu1~18.04 [288 kB]
```

1.4 安装 git 工具

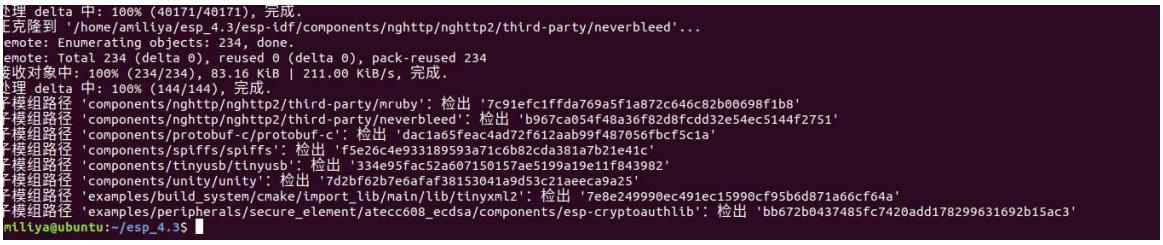
```
sudo apt install git
```

1.5 获取 ESP-IDF

使用终端命令创建工程文件夹，文件夹名称可以自己定义，例如我创建的文件夹名称为 esp_4.3

```
mkdir esp_4.3

cd esp_4.3

git clone -b release/v4.3 --recursive https://github.com.cnpmjs.org/espressif/esp-idf.git
```

（注：-b release/v4.3 表示 esp-idf 下载版本(最新版)，需要下载其它版本或版本介绍请移至乐鑫官网查看具体介绍）



若子模组未完全克隆，可参考章节 4.3.2

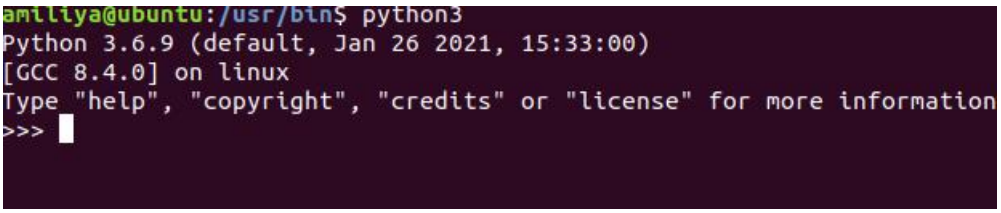
1.6 安装其它工具

除了 ESP-IDF 本身，您还需要安装 ESP-IDF 使用的各种工具，比如编译器、调试器、Python 包等!

1.6.1 查看当前 Python 版本

终端输入命令：`ls /usr/bin/python*`

这代表已安装 python3,输入 `python3` 查看 python 版本为 3.6.9

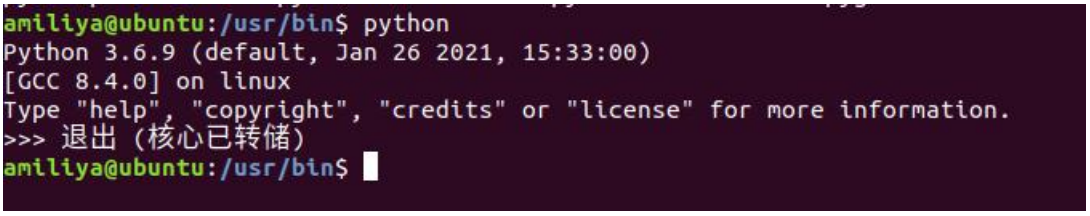


1.6.2 将 python3 设置为默认 python

进入用户 bin 目录下，将 python3.6 链接到 python 下。

```
cd /usr/bin
ln -s python3.6m python
```

执行命令：python，出现下图，说明链接成功



若系统 python 版本小于 3.6，可参考章节 4.3.3

安装 python3.6 的 pip：

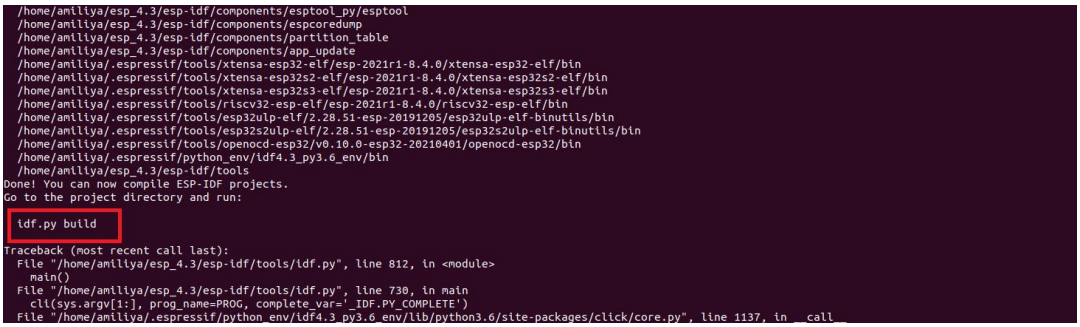
```
sudo apt-get install python3-pip
```

1.6.3 安装工具

运行 `./install.sh` 最终结果如下：

1.6.4 激活 esp-idf 环境

执行 `. ./export.sh`



到这一步表明 esp-idf 环境已经基本搭建完成，可以运行示例代码，若需要长期、多次运行 esp32 相关示例代码则建议将 esp-idf 环境加入环境变量。

1.7 将 esp-idf 环境加入环境变量

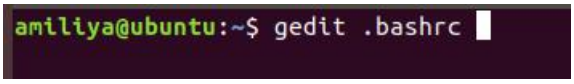
复制并粘贴以下命令到 shell 配置文件中（`.profile`，`.bashrc`，`.zprofile` 等）

```
alias get_idf='. $HOME/esp_4.3/esp-idf/export.sh'
```

(注：文件路径需要正确，如文件夹名称 esp_4.3)

以.bashrc 为例：

打开.bashrc 文件

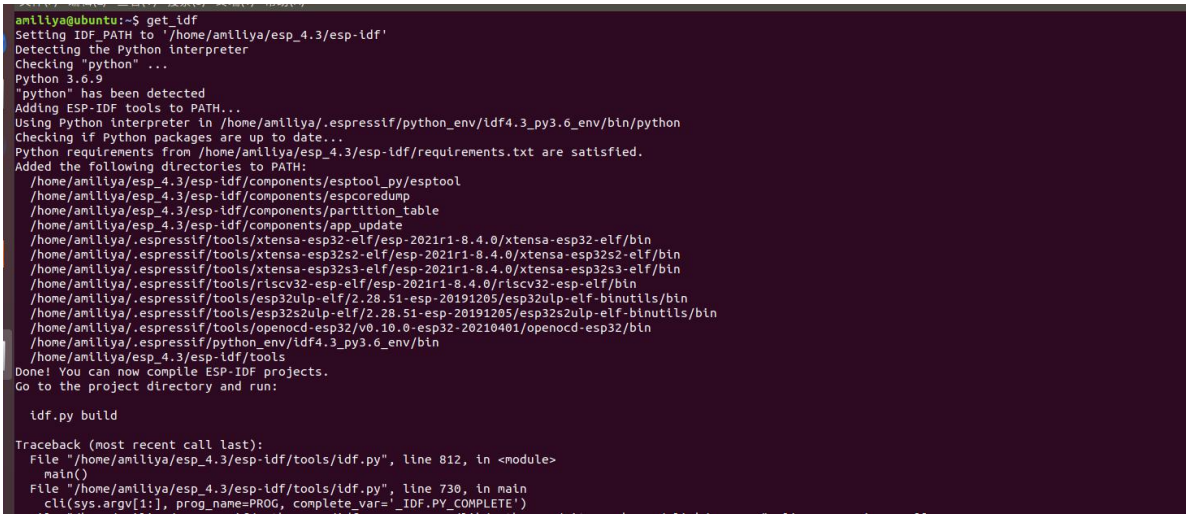


添加命令：（需要不同版本的 IDF 时都可以加入命令，只要变量名 get_idf 不同即可）

```
if ! shopt -oq posix; then
  if [ -f /usr/share/bash-completion/bash_completion ]; then
    . /usr/share/bash-completion/bash_completion
  elif [ -f /etc/bash_completion ]; then
    . /etc/bash_completion
  fi
fi

alias get_idf='. $HOME/esp_4.3/esp-idf/export.sh'
```

保存关闭，重启虚拟机或使用命令刷新配置文件：source ~/.bashrc

最终效果如下：执行 get_idf 便可以让端口进入 esp-idf 环境：



2 运行 hello_world 示例

2.1 拷贝工程

将 `esp-idf/examples/get-started/` 目录下的 `hello_world` 示例拷贝到 `esp_4.3` 下

```
cp esp-idf/examples/get-started/hello_world/ ./ -r
```

```
amiliya@ubuntu:~/esp_4.3$ cp esp-idf/examples/get-started/hello_world/ ./ -r
amiliya@ubuntu:~/esp_4.3$ ls
esp-idf  hello_world
amiliya@ubuntu:~/esp_4.3$
```

2.2 工程配置

进入 hello_world 示例，进行工程配置：

```
cd ~/esp_4.3/hello_world
```

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

(使用其它芯片请选择对应芯片类型，如：esp32c3)

(打开一个新项目后，应首先设置“目标”芯片 `idf.py set-target esp32`。注意，此操作将清除并初始化项目之前的编译和配置（如有）。 您也可以直接将“目标”配置为环境变量（此时可跳过该步骤）)

```
idf.py menuconfig
```

如果之前的步骤都正确，则会显示下面的菜单：

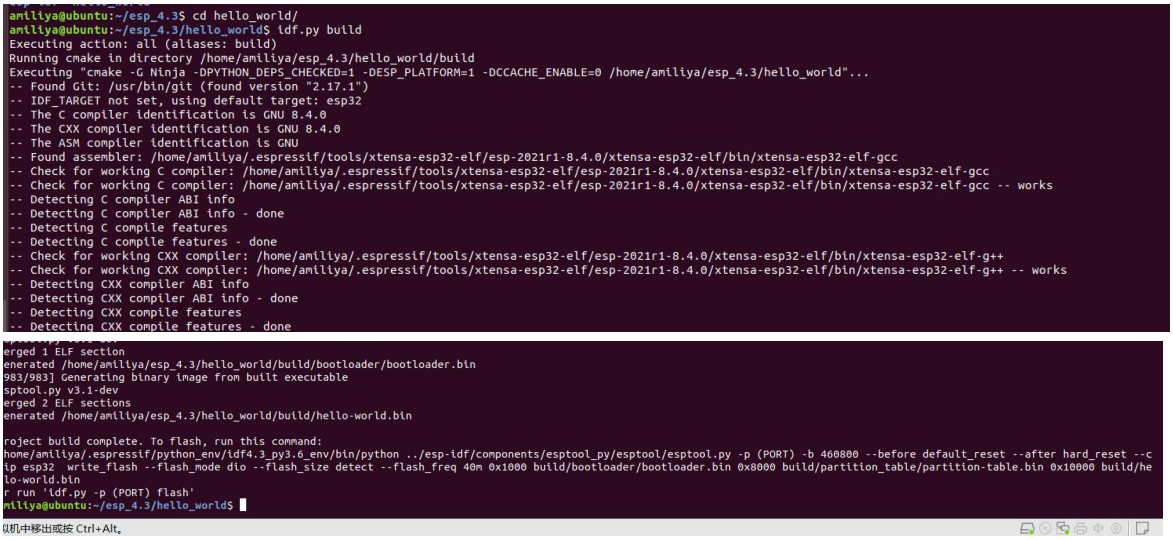
```
(Top)
                Espressif IoT Development Framework Configuration
    SDK tool configuration  --->
    Build type  --->
    Application manager  --->
    Bootloader config  --->
    Security features  --->
    Serial flasher config  --->
    Partition Table  --->
    Compiler options  --->
    Component config  --->
    Compatibility options  --->

[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)
```

工程配置 — 主窗口

您可以通过此菜单设置项目的具体变量，包括 Wi-Fi 网络名称、密码和处理器速度等. `hello_world` 示例项目会以默认配置运行，因此可以跳过使用 `menuconfig` 进行项目配置这一步骤。

2.3 编译工程

`idf.py build`



2.4 硬件连接

用 USB 线将 ESP32 模组与电脑连接：

选择连接到虚拟机->Ubuntu 64 位->确定

2.5 查看端口

查看下载端口：`ls /dev/tty*`

```
amiliya@ubuntu:~/esp_4.3/hello_world$ ls /dev/tty*
/dev/tty    /dev/tty15  /dev/tty22  /dev/tty3   /dev/tty37  /dev/tty44  /dev/tty51  /dev/tty59  /dev/tty9       /dev/ttyS14  /dev/ttyS21  /dev/ttyS29  /dev/ttyS8
/dev/tty0   /dev/tty16  /dev/tty23  /dev/tty30  /dev/tty38  /dev/tty45  /dev/tty52  /dev/tty6   /dev/ttyprintk  /dev/ttyS15  /dev/ttyS22  /dev/ttyS3   /dev/ttyS9
/dev/tty1   /dev/tty17  /dev/tty24  /dev/tty31  /dev/tty39  /dev/tty46  /dev/tty53  /dev/tty60  /dev/ttyS0      /dev/ttyS16  /dev/ttyS23  /dev/ttyS30  /dev/ttyUSB0
/dev/tty10  /dev/tty18  /dev/tty25  /dev/tty32  /dev/tty4   /dev/tty47  /dev/tty54  /dev/tty61  /dev/ttyS1      /dev/ttyS17  /dev/ttyS24  /dev/ttyS31
/dev/tty11  /dev/tty19  /dev/tty26  /dev/tty33  /dev/tty40  /dev/tty48  /dev/tty55  /dev/tty62  /dev/ttyS10     /dev/ttyS18  /dev/ttyS25  /dev/ttyS4
/dev/tty12  /dev/tty2   /dev/tty27  /dev/tty34  /dev/tty41  /dev/tty49  /dev/tty56  /dev/tty63  /dev/ttyS11     /dev/ttyS19  /dev/ttyS26  /dev/ttyS5
/dev/tty13  /dev/tty20  /dev/tty28  /dev/tty35  /dev/tty42  /dev/tty5   /dev/tty57  /dev/tty7   /dev/ttyS12     /dev/ttyS2   /dev/ttyS27  /dev/ttyS6
/dev/tty14  /dev/tty21  /dev/tty29  /dev/tty36  /dev/tty43  /dev/tty50  /dev/tty58  /dev/tty8   /dev/ttyS13     /dev/ttyS20  /dev/ttyS28  /dev/ttyS7
amiliya@ubuntu:~/esp_4.3/hello_world$
```

可以看见多出来的端口号为 `/dev/ttyUSB0,`即为下载端口号

2.6 工程烧录

进行烧录：`idf.py -p  /dev/ttyUSB0 flash`

出现如下错误说明我们对该端口没有读写权限：

```
  _main()
File "/home/amiliya/esp_4.1/esp-idf/components/esptool_py/esptool/esptool.py", line 3415, in _main
  main()
File "/home/amiliya/esp_4.1/esp-idf/components/esptool_py/esptool/esptool.py", line 3079, in main
  esp = chip_class(each_port, initial_baud, args.trace)
File "/home/amiliya/esp_4.1/esp-idf/components/esptool_py/esptool/esptool.py", line 261, in __init__
  self._port = serial.serial_for_url(port)
File "/home/amiliya/.espressif/python_env/idf4.1_py3.6_env/lib/python3.6/site-packages/serial/__init__.py", line 90, in serial_for_url
  instance.open()
File "/home/amiliya/.espressif/python_env/idf4.1_py3.6_env/lib/python3.6/site-packages/serial/serialposix.py", line 325, in open
  raise SerialException(msg.errno, "could not open port {}: {}".format(self._port, msg))
erial.serialutil.SerialException: [Errno 13] could not open port /dev/ttyUSB0: [Errno 13] Permission denied: '/dev/ttyUSB0'
sptool.py failed with exit code 1
```

通过以下命令，将用户添加到 `dialout` 组，从而获许读写权限:

```
sudo usermod -a -G dialout $USER
```

`reboot` (重启虚拟机)

重新烧录：

```
Flash will be erased from 0x00010000 to 0x00037fff...
Compressed 3072 bytes to 103...
Writing at 0x00008000... (100 %)
Wrote 3072 bytes (103 compressed) at 0x00008000 in 0.1 seconds (effective 429.2 kbit/s)...
Hash of data verified.
Compressed 25152 bytes to 15450...
Writing at 0x00001000... (100 %)
Wrote 25152 bytes (15450 compressed) at 0x00001000 in 0.7 seconds (effective 278.0 kbit/s)...
Hash of data verified.
Compressed 162976 bytes to 85495...
Writing at 0x00010000... (16 %)
Writing at 0x0001acae... (33 %)
Writing at 0x0002048a... (50 %)
Writing at 0x00025c9f... (66 %)
Writing at 0x0002f4e3... (83 %)
Writing at 0x000366b0... (100 %)
Wrote 162976 bytes (85495 compressed) at 0x00010000 in 2.4 seconds (effective 538.9 kbit/s)...
Hash of data verified.

Leaving...
Hard resetting via RTS pin...
Done
amiliya@ubuntu:~/esp_4.3/hello_world$
```

烧录成功！

2.7 监视工程

查看监视器：`idf.py -p  /dev/ttyUSB0 monitor`

```
W (296) spi_flash: Detected size(4096k) larger than the size in the binary image header(2048k). Using the s
I (307) cpu_start: Starting scheduler on PRO CPU.
I (0) cpu_start: Starting scheduler on APP CPU.
Hello world!
This is ESP32 chip with 2 CPU cores, WiFi/BT/BLE, silicon revision 1, 2MB external flash
Restarting in 10 seconds...
Restarting in 9 seconds...
Restarting in 8 seconds...
Restarting in 7 seconds...
Restarting in 6 seconds...
Restarting in 5 seconds...
Restarting in 4 seconds...
```

成功打印 hello_world.

3 参考视频

（注：为保证视频质量，演示视频并未录制声音，同时对于一些长时间下载过程进行了缩略录制，视频仅作为本文档参考文件。）


Video_1.wmv


Video_2.wmv

4 后　记

4.1 注意事项

（1）Ubuntu 版本建议在 16.0 以上，搭建 esp-idf 环境需要保证网络通畅
（2）python 版本必须保证 3.6 以上且为系统默认 python
（3）环境变量可添加多个，可通过调用不同变量名选择不同版本的 IDF
（4）工程配置中选择目标芯片的过程对某些芯片是必要的，如 ESP32-C3
（5）端口权限也可以通过命令 `chmod` 修改，但那只是暂时的
（6）进行 ESP32-C3 的开发必需使用最新版（V4.3）IDF 的支持

4.2 相关建议

使用 Linux 系统进行开发是众多开发爱好者较为普遍的选择，Linux 较强的兼容性是其最大的优势，众多命令的使用让它在开发层面自由度更高。但是由于缺少图形化的文本编辑器，Ubuntu 在代码的查看与编辑方面较为弱势，因此常与其它文本编辑器联合使用，如 VS Code。因此，ESP32 系列教程之三将继续介绍 VS Code 远程连接 Linux 的方法，将代码的编写与烧录变得更加方便与简洁。

除了 Linux 外，ESP32 系列教程后续将介绍在 Windows 搭建 esp-idf 环境的方法，习惯使用 Windows 系统进行开发的开发爱好者可查看 ESP32 系列教程之四(Windows)和 ESP32 系列教程之五（Eclipse）了解相关操作过程。

4.3 常见问题

由于 esp-idf 环境搭建过程中由于各种原因容易遇到各种问题，此章节整理了一些常见的问题以供大家参考，其它相关问题可提交至意见提交邮箱。

4.3.1 ESP-IDF 的获取

部分开发者由于网络等原因无法从 github 获取 esp-idf，可参考下方步骤从国内的 gitee 获取 esp-idf。

（1）获取 esp-idf

```
git clone -b release/v4.3 https://gitee.com/EspressifSystems/esp-idf.git
```

```
amiliya@DESKTOP-LDC9RD8:~/esp$ git clone -b release/v4.3 https://gitee.com/EspressifSystems/esp-idf.git
正克隆到 'esp-idf'...
remote: Enumerating objects: 30044, done.
remote: Counting objects: 100% (30044/30044), done.
remote: Compressing objects: 100% (11812/11812), done.
remote: Total 250881 (delta 20441), reused 24664 (delta 17190), pack-reused 220837
接收对象中: 100% (250881/250881), 139.96 MiB | 5.57 MiB/s, 完成.
处理 delta 中: 100% (185486/185486), 完成.
正在更新文件: 100% (8306/8306), 完成.
amiliya@DESKTOP-LDC9RD8:~/esp$ cd esp-idf/
```

（2）下载工具

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

```
amiliya@DESKTOP-LDC9RD8:~/esp$ git clone https://gitee.com/EspressifSystems/esp-gitee-tools.git
正克隆到 'esp-gitee-tools'...
remote: Enumerating objects: 48, done.
remote: Counting objects: 100% (48/48), done.
remote: Compressing objects: 100% (47/47), done.
remote: Total 48 (delta 19), reused 0 (delta 0), pack-reused 0
展开对象中: 100% (48/48), 15.36 KiB | 249.00 KiB/s, 完成.
amiliya@DESKTOP-LDC9RD8:~/esp$ ls
esp-gitee-tools  esp-idf
amiliya@DESKTOP-LDC9RD8:~/esp$
```

（3）下载子模块和工具链

进入工具文件夹，保存路径

```
cd esp-gitee-tools
```

```
export EGT_PATH=$(pwd)
```

进入 esp-idf 文件夹，下载子模块和工具链

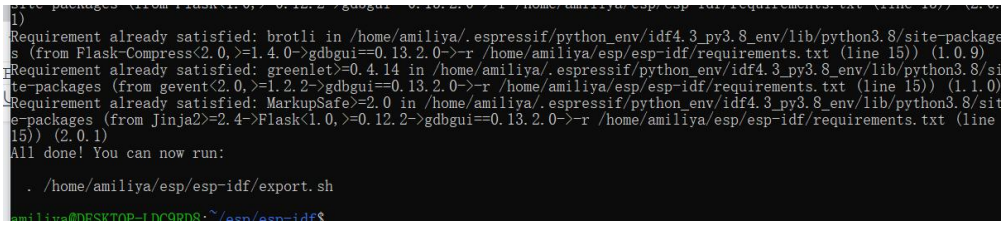
```
cd ~/esp-idf/
```

```
$EGT_PATH/submodule-update.sh
```

```
amiliya@DESKTOP-LDC9RD8:~$ cd esp/esp-gitee-tools/
amiliya@DESKTOP-LDC9RD8:~/esp/esp-gitee-tools$ export EGT_PATH=$(pwd)
amiliya@DESKTOP-LDC9RD8:~/esp/esp-gitee-tools$ cd ../esp-idf/
amiliya@DESKTOP-LDC9RD8:~/esp/esp-idf$ $EGT_PATH/submodule-update.sh
子模组 'components/asio/asio' (https://gitee.com/espressif/asio.git) 已对路径 'components/asio/asio' 注册
子模组 'components/bootloader/subproject/components/micro-ecc/micro-ecc' (https://gitee.com/kmackay/micro-ecc.git) 已对
路径 'components/bootloader/subproject/components/micro-ecc/micro-ecc' 注册
子模组 'components/bt/controller/lib_esp32' (https://gitee.com/espressif/esp32-bt-lib.git) 已对路径 'components/bt/con
oller/lib_esp32' 注册
子模组 'components/bt/controller/lib_esp32c3_family' (https://gitee.com/espressif/esp32c3-bt-lib.git) 已对路径 'compone
ts/bt/controller/lib_esp32c3_family' 注册
子模组 'components/bt/host/nimble/nimble' (https://gitee.com/espressif/esp-nimble.git) 已对路径 'components/bt/host/ni
le/nimble' 注册
子模组 'components/cbor/tinycbor' (https://gitee.com/intel/tinycbor.git) 已对路径 'components/cbor/tinycbor' 注册
子模组 'components/cmock/CMock' (https://gitee.com/ThrowTheSwitch/CMock.git) 已对路径 'components/cmock/CMock' 注册
子模组 'components/coap/libcoap' (https://gitee.com/obgm/libcoap.git) 已对路径 'components/coap/libcoap' 注册
子模组 'components/esp_wifi/lib' (https://gitee.com/espressif/esp32-wifi-lib.git) 已对路径 'components/esp_wifi/lib' 注
册
子模组 'components/esptool_py/esptool' (https://gitee.com/espressif/esptool.git) 已对路径 'components/esptool_py/esptoo
' 注册
```

```
$EGT_PATH/install.sh
```

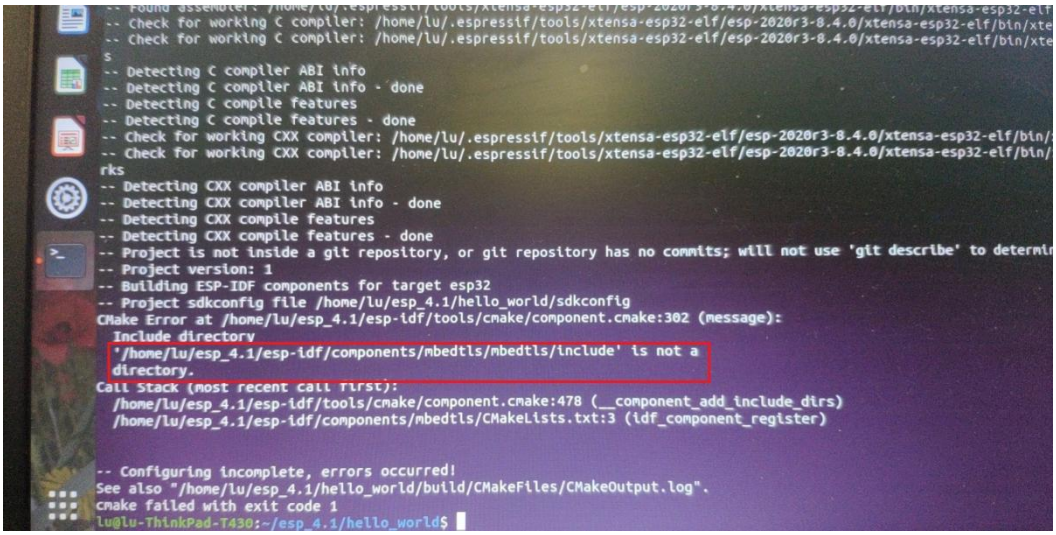
```
amiliya@DESKTOP-LDC9RD8:~/esp/esp-idf$ $EGT_PATH/install.sh
Installing ESP-IDF tools
Installing tools: xtensa-esp32-elf, xtensa-esp32s2-elf, xtensa-esp32s3-elf, riscv32-esp-elf, esp32ulp-elf, esp32s2ulp-el
f, openocd-esp32
Skipping xtensa-esp32-elf@esp-2021r1-8.4.0 (already installed)
Skipping xtensa-esp32s2-elf@esp-2021r1-8.4.0 (already installed)
Skipping xtensa-esp32s3-elf@esp-2021r1-8.4.0 (already installed)
Skipping riscv32-esp-elf@esp-2021r1-8.4.0 (already installed)
Skipping esp32ulp-elf@2.28.51-esp-20191205 (already installed)
Skipping esp32s2ulp-elf@2.28.51-esp-20191205 (already installed)
Skipping openocd-esp32@v0.10.0-esp32-20210401 (already installed)
Installing Python environment and packages
Python 3.8.10
pip 21.1.3 from /home/amiliya/.espressif/python_env/idf4.3_py3.8_env/lib/python3.8/site-packages/pip (python 3.8)
```



（4）其余步骤参照章节 1.6-1.7

4.3.2 子模组缺失

绝大多数开发者所遇到的无法启动菜单、工程编译失败等问题都源于 ESP-IDF 子模组未下载完全，因此保证子模组的完全下载对完整搭建 esp-idf 环境至关重要。



缺少相应子模块时的错误 log

在 esp-idf 文件夹下使用以下命令下载缺失子模组

```
git submodule update --init --recursive
```

若命令长时间无响应可取消后重试，此命令也可用于检测子模组是否下载完全（下载完全则直接返回命令行），也可进入缺失子模组的目录下单独克隆该子模组。

4.3.3 python 版本

步骤 1.6.3（./install.sh）执行过程中需要的 python 版本至少大于 3.6，如果执行 python 后发现 python 版本小于 3.6，则执行以下步骤安装 python3.6 并将其设置为默认 python

```
hu@hu:~/esp/esp-idf$ python
Python 3.5.2 (default, Oct  7 2020, 17:19:02)
[GCC 5.4.0 20160609] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> exit()
hu@hu:~/esp/esp-idf$
```

1、安装依赖包

```
$ sudo apt-get update
$ sudo apt-get install software-properties-common
```

2、添加 deadsnakes PPA 源

```
$ sudo add-apt-repository ppa:deadsnakes/ppa
$ sudo apt-get update
```

> 在进行 sudo apt-get update时：
> 如果报错：
> 正在读取软件包列表... 完成 W: GPG
> 错误：http://ppa.launchpad.net/fossfreedom/indicator-sysmonitor/ubuntu
> xenial InRelease: 由于没有公钥，无法验证下列签名： NO_PUBKEY 82EB5823F4FE239D W: 仓库
> “http://ppa.launchpad.net/fossfreedom/indicator-sysmonitor/ubuntu
> xenial InRelease” 没有数字签名。 N: 无法认证来自该源的数据，所以使用它会带来潜在风险。 N: 参见
> apt-secure(8) 手册以了解仓库创建和用户配置方面的细节。

执行如下命令

```
sudo apt-key adv --keyserver keyserver.ubuntu.com --recv-keys 82EB5823F4FE239D  # 这个是填写自己报错后的那一串字符！
```

```
sudo apt-get install python3.6
```

```
cd /usr/bin
```

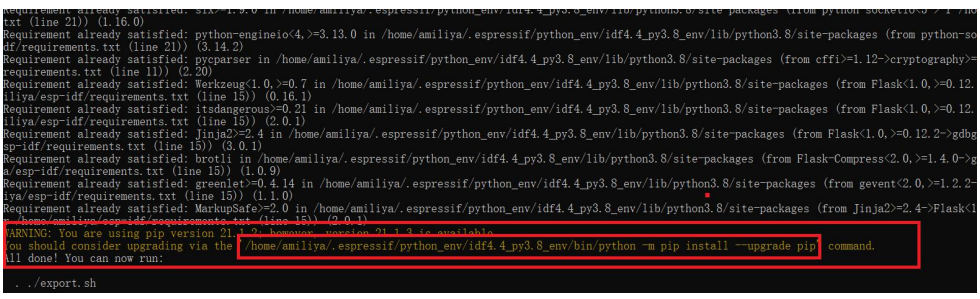
```
 rm python
```

```
ln -s python3.6m python
```

（参考链接：https://blog.csdn.net/weixin_49938318/article/details/113129408?utm_source=app&app_version=4.9.0&code=app_1562916241&uLinkId=usr1mkqgl919blen）

4.3.4 工具链设置

少部分开发者在搭建环境时会遇到此类问题，而这一类问题的形成原因较为复杂，推测可能与所用系统版本有关，因此仅能依据错误提示 log 做出相应处理，暂无完整解决方式，以下仅供参考。



按照提示执行相应命令可消除警告（pip 版本问题）