

## ESP32-C3 环境准备

### 1. Windows 环境搭建

#### a) 获取 ESP-IDF

Windows 去官方网站下载一个 ESP-IDF 工具安装器

<https://dl.espressif.com/dl/esp-idf/?idf=4.4>

我这里使用的是离线安装器

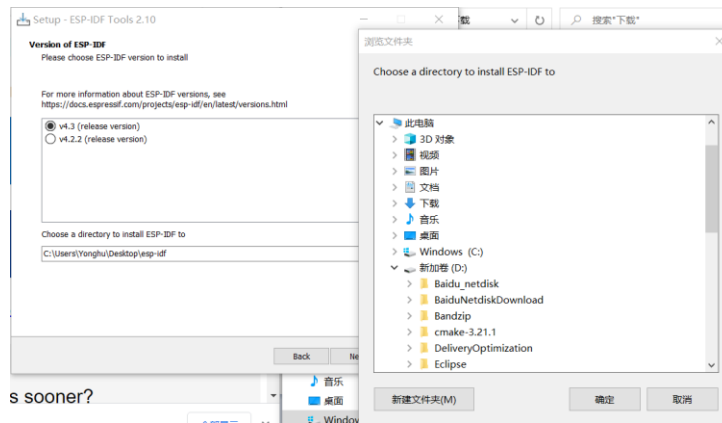
#### ESP-IDF Windows Installer Download

Open Source IoT Development Framework for ESP32



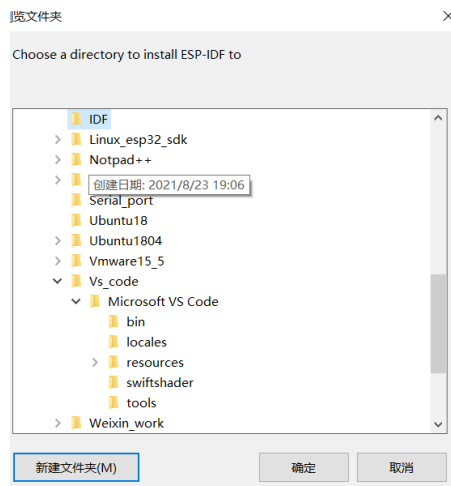
Installation instructions: [ESP-IDF documentation](#) and [Espressif Systems Youtube channel](#).

#### b) 安装

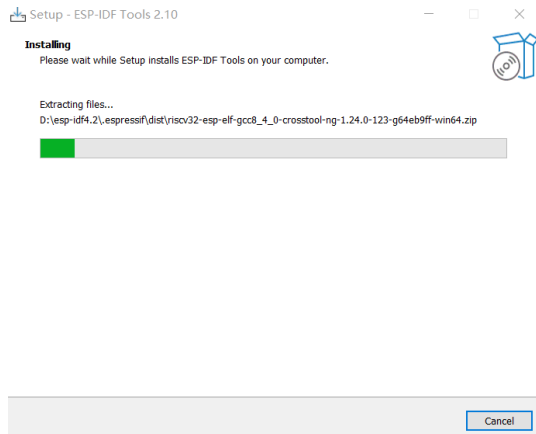


建议在 D 盘创建一个文件

夹去保存该 ESP-IDF，因为后面需要把工程放在同一个目录下



建立一个 IDF 文件夹去保存该 esp-idf 直接一直下一步到安装



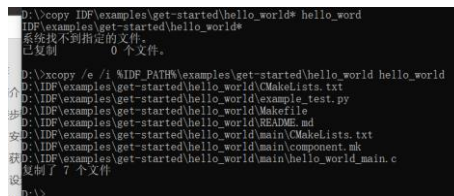
安装完成后会在桌面生成



可以以管理员身份运行（但是不推荐），如图已经可以使用 `idf.py build` 这样环境搭建完成，下面开始编译工程

## 2. 工程

### a) 拷贝工程



将 `hello_world` 复制到 d 盘下

```
xcopy /e /i %IDF_PATH%\examples\get-started\hello_world hello_world
```

这里安装没安装到 IDF 子目录下，所以选择直接在 d 盘选择一个文件夹作为工程目录  
使用 `dir` 指令 查看当前所有文件

```

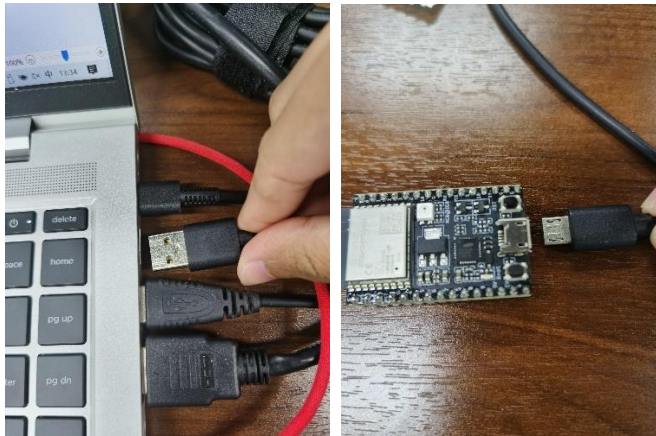
D:\>dir
驱动器 D 中的卷是 新加卷
卷的序列号是 72D3-B965

D:\ 的目录
2021/08/26 17:52 <DIR> BaiduNetdiskDownload
2021/08/25 10:46 <DIR> Baidu_netdisk
2021/08/25 11:17 <DIR> Bandzip
2021/08/24 14:41 <DIR> cmake-3.21.1
2021/08/23 11:28 <DIR> DeliveryOptimization
2021/08/24 13:50 <DIR> Eclipse
2021/08/24 14:44 <DIR> esp-idf4.2
2021/09/01 14:09 <DIR> Esp32_idf
2021/08/23 14:16 <DIR> ESP8266
2021/08/24 13:56 <DIR> Fotiaoqiang
2021/08/23 18:24 <DIR> git
2021/09/02 11:53 <DIR> hello_world
2021/09/02 11:32 <DIR> IDF

```

这个便是已经复制好的工程文件了

#### b) 设备连接



#### c) 设置目标芯片：

idf.py set-target esp32c3 设置目标芯片

```

ansport D:/IDF/components/tcpip_adapter D:/IDF/components/tinyusb
/components/wear_levelling D:/IDF/components/wifi_provisioning D
-- Configuring done
-- Generating done
-- Build files have been written to: D:/hello_world/build
D:\hello_world>

```

idf.py menuconfig 打开工程配置主窗口命令

```

(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
[D] 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)

```

#### d) 编译工程

进入到已经克隆的文件 hello\_world 编译命令：idf.py build

```

elf/bin/riscv32-esp-elf-g++-esp
-- Check for working CXX compiler: D:/esp-idf4.2/.espressif/tools/riscv32-esp-elf/1.24.0.123_64eb0ff-8
elf/bin/riscv32-esp-elf-g++-esp -- works
-- Detecting CXX compiler ABI info
-- Detecting CXX compiler ABI info - done
-- Detecting CXX compile features
-- Detecting CXX compile features - done
-- Building ESP-IDF components for target esp32c3
-- Project sdkconfig file D:/hello_world/sdkconfig
-- Adding linker script D:/IDF/components/esp32c3/ld/esp32c3_peripherals.ld
-- Adding linker script D:/IDF/components/esp_rom/esp32c3/ld/esp32c3_rom.ld
-- Adding linker script D:/IDF/components/esp_rom/esp32c3/ld/esp32c3_rom_api.ld
-- Adding linker script D:/IDF/components/esp_rom/esp32c3/ld/esp32c3_rom_libgcc.ld
-- Adding linker script D:/IDF/components/esp_rom/esp32c3/ld/esp32c3_rom_newlib.ld
-- Adding linker script D:/IDF/components/bootloader/subproject/main/ld/esp32c3_bootloader.ld
-- Components: bootloader bootloader_support efuse esp32c3 esp_common esp_hw_support esp_rom esp_system
log main micro-ecce newlib partition_table riscv soc spi_flash
-- Component paths: D:/IDF/components/bootloader D:/IDF/components/bootloader_support D:/IDF/component
components/esp32c3 D:/IDF/components/esp_common D:/IDF/components/esp_hw_support D:/IDF/components/esp
ments/esp_system D:/IDF/components/esp_tool_py D:/IDF/components/hal D:/IDF/components/log D:/IDF/comp
subproject/main D:/IDF/components/bootloader/subproject/components/micro-ecce D:/IDF/components/newlib
ts/partition_table D:/IDF/components/riscv D:/IDF/components/soc D:/IDF/components/api_flash
-- Configuring done
-- Generating done
-- Build files have been written to: D:/hello_world/build/bootloader
C:\W\361\Building C object esp-idf-esp_event\Makefiles... idf_esp_event_dir/default_event_loop.c.obj

```

e) 烧入到设备 `idf.py -p port [-b BAUD] flash`

`-p port` 表示需要指定的端口号 `-b BAUD` 可以不用写指的是波特率不写默认烧入波特率为：  
460800

f) 环境 打开设备管理器 找到

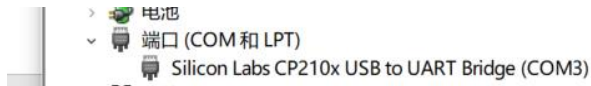
按下 windows 按键/鼠标单击左小角



键盘输入设备管理器



找到 COM 和 LPT 选项



当 COM 口未连接设备不会有该选项

`idf.py -p (PORT) monitor`

这个 PORT 即端口号，如上则是 COM3 命令：`idf.py -p COM3 monitor`

也可以使用软件：串口调试工具

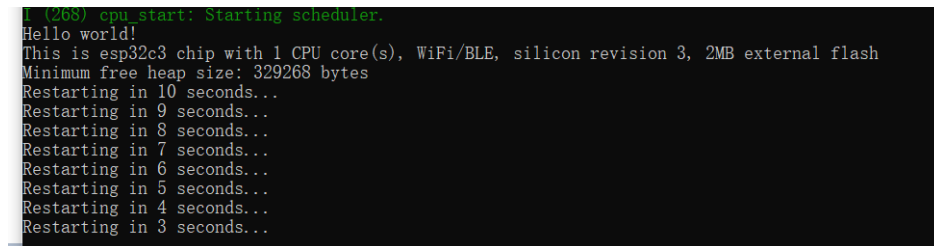


g) 烧入

h) 监视器

windows 也可以运行以下命令，一次性执行构建、烧录和监视过程：

idf.py -p PORT flash monitor Linux 用 monitor



i) 点击软件打开口



可以看到打印所有信息

## Ubuntu 下的环境安装

### 1. 下载 cmake 及 python3 环境 和 ninja-build git

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  
编译的软件包

```
yonghu@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 libusb-1.0-0
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 dh-python dpkg-dev fakeroot g++ g++-7 gcc gcc-7
  git-man libalgorithm-diff-perl libalgorithm-diff-xs-perl
  libalgorithm-merge-perl libasan4 libatomic1 libbison-dev libc-dev-bin
升级了 0 个软件包，新安装了 61 个软件包，要卸载 0 个软件包，有 5 个软件包未被升级。
需要下载 91.6 MB 的归档。
解压缩后会消耗 278 MB 的额外空间。
您希望继续执行吗？ [Y/n]
```

键入 Y

我们装好了工具，现在安装 esp-idf 先 cd 到你想要安装到的路径我这里选择桌面

```
yonghu@ubuntu:~$ cd /home/yonghu/Desktop/
```

### 2. 在当前路径下创建文件夹 esp 这样可以看见桌面生成了一个文件夹

cd ./esp 转到刚刚创建的文件夹

git clone --recursive https://github.com/espressif/esp-idf.git

克隆文件到当前路径下，也就是刚刚创建的文件夹 esp

```
.cache/ .dbus/ Documents/ .gnupg/ Music/ Public/ Templates/
.config/ Desktop/ Downloads/ .local/ Pictures/ .ssh/ Videos/
yonghu@ubuntu:~$ cd /home/yonghu/Desktop/
yonghu@ubuntu:~/Desktop$ cd /home/yonghu/Desktop/
yonghu@ubuntu:~/Desktop$ mkdir ./esp
yonghu@ubuntu:~/Desktop$ cd esp/
yonghu@ubuntu:~/Desktop/esp$ git clone --recursive https://github.com/espressif/esp-idf.git
```

### 3. Linux 则需要先 cd ~/esp/esp-idf 即刚刚下载的路径

这里的 ~ 表示 家目录 /home/yonghu

然后：./install.sh esp32c3

```
Install.sh: 未找到命令
yonghu@ubuntu:~/Desktop/esp/esp-idf$ ./install.sh
Detecting the Python Interpreter
Checking "python" ...
/home/yonghu/Desktop/esp/esp-idf/tools/detect_python.sh: 行 16: python: 未找到命令
Checking "python3" ...
Python 3.6.9
"python3" has been detected
Installing ESP-IDF tools
WARNING: File /home/yonghu/.espressif/idf-env.json was not found.
Creating /home/yonghu/.espressif/idf-env.json
WARNING: File /home/yonghu/.espressif/idf-env.json can not be created.
Selected targets are:
Installing tools: xtensa-esp32-elf, xtensa-esp32s2-elf, xtensa-esp32s3-elf, riscv32-esp-elf, esp32ulp-elf, esp32s2ulp-elf, openocd-esp32
```

```
File "/home/yonghu/Desktop/esp/esp-idf/tools/idf_tools.py", line 10, in
    main(sys.argv[1:])
File "/home/yonghu/Desktop/esp/esp-idf/tools/idf_tools.py", line 10, in
    action_func(args)
File "/home/yonghu/Desktop/esp/esp-idf/tools/idf_tools.py", line 10, in
    on_install
File "/home/yonghu/Desktop/esp/esp-idf/tools/idf_tools.py", line 10, in
    tool_obj.download(tool_version)
File "/home/yonghu/Desktop/esp/esp-idf/tools/idf_tools.py", line 10, in
    raise DownloadError()
main .DownloadError
```

报错解决办法

输入 `git submodule update --init --recursive` 更新 需要多次尝试直到完全克隆

```
yonghu@ubuntu:~/esp$ cd esp-idf/
yonghu@ubuntu:~/esp/esp-idf$ git clone --recursive https://github.com/espressif/esp-idf.git
正克隆到 'esp-idf'...
remote: Enumerating objects: 267064, done.
remote: Counting objects: 100% (494/494), done.
remote: Compressing objects: 100% (283/283), done.
remote: Total 267064 (delta 187), reused 471 (delta 186), pack-reused 266570
接收对象中: 100% (267064/267064), 148.63 MiB | 5.42 MiB/s, 完成。
处理 delta 中: 90% (177837/197578)
```

```
Python 3.6.0 pip 9.0.2 pyserial 2.3.1 pyserial 3.5 python-engineio 3.14.2 python-socketio 4
.6.1 reedsolo 1.5.4 six 1.16.0 typing-extensions 3.10.0.2 zipp 3.5.0
WARNING: You are using pip version 21.2.3; however, version 21.2.4 is available.
You should consider upgrading via the '/home/yonghu/.espressif/python_env/idf4.4
_py3.6_env/bin/python -m pip install --upgrade pip' command.
All done! You can now run:
. ./export.sh
yonghu@ubuntu:~/Desktop/esp/esp-idf$
```

```
-- Configuring incomplete, errors occurred!
See also "/home/yonghu/Desktop/esp/hello_world/build/CMakeFiles/CMakeOutput.log"
cmake failed with exit code 1
yonghu@ubuntu:~/Desktop/esp/hello_world$
```

可能遇见找不到文件

解决办法：`find /usr/include -name "io.h"`

```
yonghu@ubuntu:~/Desktop/esp/hello_world$ find /usr/include -name "io.h",
yonghu@ubuntu:~/Desktop/esp/hello_world$ find /usr/include -name "io.h"
/usr/include/x86_64-linux-gnu/sys/io.h
yonghu@ubuntu:~/Desktop/esp/hello_world$
```

可以看到 `sys` 目录下有

`/usr/include` 下没有，但是在 `/usr/include/sys` 下有，我把 `io.h` 复制到了 `/usr/include` 下，就行了

由于上面出错可能导致该命令未生效

接着：`./export.sh esp32c3` （`esp32c3` 打不打无所谓）

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

#### 4. 写入环境

`.$HOME/esp/esp-idf/export.sh`

此外，如果您希望在当下命令提示符窗口使用 ESP-IDF，请使用下方代码：

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/esp-idf/export.sh'`

通过重启终端窗口或运行 `source [path to profile]`，如 `source ~/.bashrc` 来刷新配置文件。

#### 5. 拷贝目录

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

```

examples/   export.bat  export.fish  export.ps1  export.sh
yonghu@ubuntu:~/Desktop/esp$ cp ./esp-idf/ex
examples/   export.bat  export.fish  export.ps1  export.sh
yonghu@ubuntu:~/Desktop/esp$ cp ./esp-idf/ex
examples/   export.bat  export.fish  export.ps1  export.sh
yonghu@ubuntu:~/Desktop/esp$ cp ./esp-idf/examples/get-started/hello_world/ ./ -
r
yonghu@ubuntu:~/Desktop/esp$

```

cd hello\_world 跑到 hello\_world 目录下

```

yonghu@ubuntu:~/Desktop/esp$ ls
esp-idf  hello_world
yonghu@ubuntu:~/Desktop/esp$ cd hello_world/

```

idf.py build 编译该工程

ls /dev/ttyUSB\* 显示当前串口连接

```

gec@ubuntu: ~/Desktop/esp/hello_world$ ls /dev/tty*
/dev/tty0      /dev/tty23    /dev/tty39    /dev/tty54    /dev/ttyS10   /dev/ttyS26
/dev/tty1      /dev/tty24    /dev/tty4     /dev/tty55    /dev/ttyS11   /dev/ttyS27
/dev/tty10     /dev/tty25    /dev/tty40    /dev/tty56    /dev/ttyS12   /dev/ttyS28
/dev/tty11     /dev/tty26    /dev/tty41    /dev/tty57    /dev/ttyS13   /dev/ttyS29
/dev/tty12     /dev/tty27    /dev/tty42    /dev/tty58    /dev/ttyS14   /dev/ttyS3
/dev/tty13     /dev/tty28    /dev/tty43    /dev/tty59    /dev/ttyS15   /dev/ttyS30
/dev/tty14     /dev/tty29    /dev/tty44    /dev/tty6     /dev/ttyS16   /dev/ttyS31
/dev/tty15     /dev/tty30    /dev/tty45    /dev/tty60    /dev/ttyS17   /dev/ttyS4
/dev/tty16     /dev/tty31    /dev/tty46    /dev/tty61    /dev/ttyS18   /dev/ttyS5
/dev/tty17     /dev/tty32    /dev/tty47    /dev/tty62    /dev/ttyS19   /dev/ttyS6
/dev/tty18     /dev/tty33    /dev/tty48    /dev/tty63    /dev/ttyS2    /dev/ttyS7
/dev/tty19     /dev/tty34    /dev/tty49    /dev/tty7     /dev/ttyS20   /dev/ttyS8
/dev/tty20     /dev/tty35    /dev/tty50    /dev/tty8     /dev/ttyS21   /dev/ttyS9
/dev/tty21     /dev/tty36    /dev/tty51    /dev/tty9     /dev/ttyS22   /dev/ttyUSB0
/dev/tty22     /dev/tty37    /dev/tty52    /dev/ttyprintk /dev/ttyS23
/dev/tty23     /dev/tty38    /dev/tty53    /dev/ttyS1    /dev/ttyS24
/dev/tty24     /dev/tty39    /dev/tty54    /dev/ttyS25
gec@ubuntu:~/Desktop/esp/hello_world$ sudo chmod 777 /dev/ttyUSB0
build/          example_test.py  Makefile        sdkconfig
CMakeLists.txt  main/           README.md
gec@ubuntu:~/Desktop/esp/hello_world$ sudo chmod 777 /dev/ttyUSB0
[sudo] gec 的密码:

```

得到 端口号为 /dev/ttyUSB0

每次烧入前必须输入 sudo chmod 777 改变该 USB 的权限

idf.py -p PORT [-b BAUD] flash 烧入即可看见

```

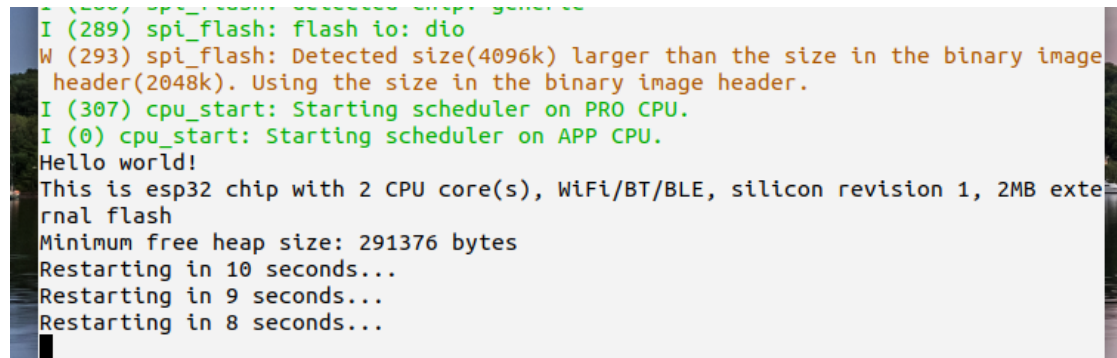
gec@ubuntu:~/Desktop/esp/hello_world$ idf.py -p /dev/ttyUSB0 flash
Executing action: flash
Running ninja in directory /home/gec/Desktop/esp/hello_world/build
Executing "ninja flash"...
[1/4] Performing build step for 'bootloader'
ninja: no work to do.
[1/2] cd /home/gec/Desktop/esp/esp-idf...nents/esptool_py/run_serial_tool.cmake
esptool.py esp32 -p /dev/ttyUSB0 -b 460800 --before=default_reset --after=hard_r
eset write_flash --flash_mode dio --flash_freq 40m --flash_size 2MB 0x8000 parti
tion_table/partition-table.bin 0x1000 bootloader/bootloader.bin 0x10000 hello-wo
rld.bin
esptool.py v3.1-dev
Serial port /dev/ttyUSB0
Connecting.....
Chip is ESP32-D0WD (revision 1)
Features: WiFi, BT, Dual Core, 240MHz, VRef calibration in efuse, Coding Scheme
None
Crystal is 40MHz
MAC: 10:52:1c:a4:c7:a8

```

## 6. 观察串口打印信息

`idf.py -p (PORT) monitor`

这个 PORT 即端口号，如上则是 `/dev/ttyUSB0` 命令：`idf.py -p /dev/ttyUSB0 monitor`



```
I (289) spi_flash: flash io: dio
W (293) spi_flash: Detected size(4096k) larger than the size in the binary image
header(2048k). Using the size in the binary image header.
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 core(s), WiFi/BT/BLE, silicon revision 1, 2MB exte
rnal flash
Minimum free heap size: 291376 bytes
Restarting in 10 seconds...
Restarting in 9 seconds...
Restarting in 8 seconds...
```

就可以看见所有打印信息