

# Get Quick Started

---

快速入门 (传统 GNU Make)

---

:link\_to\_translation: en:[English]

.. include:: ../../gnu-make-legacy.rst

本文档旨在指导用户搭建 **ESP32** 硬件开发的软件环境，

通过一个简单的示例展示如何使用 **ESP-IDF** (Espressif IoT Development Framework) 配置菜单，并编译、下载固件至 **ESP32** 开发板等步骤。

.. include:: ./\_build/inc/version-note.inc

## 概述

**ESP32 SoC** 芯片支持以下功能：

- 2.4 GHz Wi-Fi
- 蓝牙 4.2 标准
- 高性能双核
- 超低功耗协处理器
- 多种外设

**ESP32** 采用 40 nm 工艺制成，具有最佳的功耗性能、射频性能、稳定性、通用性和可靠性，适用于各种应用场景和不同功耗需求。

乐鑫为用户提供完整的软、硬件资源，进行 **ESP32** 硬件设备的开发。其中，乐鑫的软件开发环境 **ESP-IDF** 旨在协助用户快速开发物联网 (IoT) 应用，可满足用户对 Wi-Fi、蓝牙、低功耗等方面的要求。

## 准备工作

硬件：

- 一款 **ESP32** 开发板
- **USB** 数据线（USB A/Micro USB B）
- PC (Windows、Linux 或 Mac OS)

软件：

- 设置 工具链，用于编译 **ESP32** 应用程序；
- 获取 **ESP-IDF** 软件开发框架。该框架已经基本包含 **ESP32** 使用的 API（软件库和源代码）和运行 工具链 的脚本；
- 安装 C 语言编程（工程）的 文本编辑器，例如 [Eclipse](https://www.eclipse.org/) [\\_](https://www.eclipse.org/)。

.. figure:: ../../\_static/what-you-need.png

:align: center

:alt: **ESP32** 应用程序开发

:figclass: align-center

**ESP32** 应用程序开发

# 开发板简介

请点击下方连接，了解有关具体开发板的详细信息。

```
.. toctree::  
:maxdepth: 1
```

```
ESP32-DevKitC <../hw-reference/get-started-devkit>  
ESP-WROVER-KIT <../hw-reference/get-started-wrover-kit>  
ESP32-PICO-KIT <../hw-reference/get-started-pico-kit>  
ESP32-Ethernet-Kit <../hw-reference/get-started-ethernet-kit>
```

```
.. _get-started-step-by-step-legacy:
```

## 详细安装步骤

请根据下方详细步骤，完成安装过程。

设置开发环境

---

- :ref: [get-started-setup-toolchain-legacy](#)
- :ref: [get-started-get-esp-idf-legacy](#)
- :ref: [get-started-setup-path-legacy](#)
- :ref: [get-started-get-packages-legacy](#)

创建您的第一个工程

---

- :ref: [get-started-start-project-legacy](#)
- :ref: [get-started-connect-legacy](#)
- :ref: [get-started-configure-legacy](#)
- :ref: [get-started-build-and-flash-legacy](#)
- :ref: [get-started-monitor-legacy](#)

```
.. _get-started-setup-toolchain-legacy:
```

## 第一步：设置工具链

工具链指一套用于编译代码和应用程序的程序。

为了加快开发进度，您可以直接使用乐鑫提供的预制工具链。请根据您的操作系统，点击下方对应的链接，并按照链接中的指导进行安装。

```
.. toctree::  
:hidden:
```

```
Windows <windows-setup>  
Linux <linux-setup>  
MacOS <macos-setup>
```

```
+-----+-----+-----+  
| windows-logo | linux-logo | macos-logo |  
+-----+-----+-----+  
| Windows <windows-legacy> | Linux <linux-legacy> | Mac OS <macos-legacy> |  
+-----+-----+-----+
```

```
.. |windows-logo| image:: ../../_static/windows-logo.png  
:target: ../get-started-legacy/windows-setup.html
```

```
.. |linux-logo| image:: ../../_static/linux-logo.png
```

```
:target: ../get-started-legacy/linux-setup.html  
.. |macos-logo| image:: ../../_static/macos-logo.png  
:target: ../get-started-legacy/macos-setup.html  
.. _Windows-legacy: ../get-started-legacy/windows-setup.html  
.. _Linux-legacy: ../get-started-legacy/linux-setup.html  
.. _Mac OS-legacy: ../get-started-legacy/macos-setup.html  
.. note::
```

在本文档中，**Linux** 和 **MacOS** 操作系统中 **ESP-IDF** 的默认安装路径为 ``~/**esp**``；**Windows** 操作系统的默认路径为 ``%**userprofile%**\esp``。您也可以将 **ESP-IDF** 安装在任何其他路径下，但请注意在使用命令行时进行相应替换。注意，**ESP-IDF** 不支持带有空格的路径。

此外，您也可以根据自身经验和实际需求，对环境进行个性化设置，而非使用预制工具链。此时，请前往

:ref: [工具链的个性化设置<get-started-customized-setup-legacy>](#) 章节获取更多信息。

```
.. _get-started-get-esp-idf-legacy:
```

## 第二步：获取 **ESP-IDF**

除了工具链，您还需要供 **ESP32** 使用的 **API**（软件库和源代码），具体请见

**ESP-IDF 仓库** <https://github.com/espressif/esp-idf>。

获取本地副本：打开终端，切换到你要存放 **ESP-IDF** 的工作目录，使用 **git clone** 命令克隆远程仓库。

打开终端，后运行以下命令：

```
.. include:: /_build/inc/git-clone-bash.inc
```

**ESP-IDF** 将下载至 **~/esp/esp-idf**。

请前往 :doc: [/versions](#)，查看 **ESP-IDF** 不同版本的具体适用场景。

```
.. include:: /_build/inc/git-clone-notes.inc
```

```
.. note::
```

在克隆远程仓库时，不要忘记加上 ``--recursive`` 选项。否则，请接着运行以下命令，获取所有子模块： ::

```
cd esp-idf
git submodule update --init
```

```
.. _get-started-setup-path-legacy:
```

## 第三步：设置环境变量

工具链通过环境变量 **IDF\_PATH** 获得 **ESP-IDF** 的目录。因此，您需要在 **PC** 中设置该环境变量，否则无法编译工程。

您可以在每次重启会话时手动设置，也可以在用户配置中进行永久设置，具体请前往 :doc: [add-idf\\_path-to-profile](#) 章节，查看 :ref: [Windows <add-idf\\_path-to-profile-windows-legacy>](#)

、:ref: [Linux 及 MacOS <add-idf\\_path-to-profile-linux-macos-legacy>](#) 操作系统的具体设置方式。

```
.. _get-started-get-packages-legacy:
```

## 第四步：安装 **Python** 软件包

ESP-IDF 所需 Python 软件包位于 `IDF_PATH/requirements.txt` 中。您可以运行以下命令进行安装：::

```
python -m pip install --user -r $IDF_PATH/requirements.txt
```

.. note::

请注意查询您所使用的 Python 解释器的版本（运行命令 `python --version`），并根据查询结果将上方命令中的 `python` 替换为 `python2`, `python2.7`，例如：::

```
python2.7 -m pip install --user -r $IDF_PATH/requirements.txt
```

.. \_get-started-start-project-legacy:

## 第五步：开始创建工程

现在，您可以开始准备开发 ESP32 应用程序了。您可以从 ESP-IDF 中 :idf: `examples` 目录下的 :example: `get-started/hello_world` 工程开始。

将 :example: `get-started/hello_world` 复制至您本地的 `~/esp` 目录下：

Linux 和 MacOS 操作系统

---

.. code-block:: bash

```
cd ~/esp
cp -r $IDF_PATH/examples/get-started/hello_world .
```

Windows 操作系统

---

.. code-block:: batch

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

ESP-IDF 的 :idf: `examples` 目录下有一系列示例工程，都可以按照上面的方法进行创建。您可以按照上述方法复制并运行其中的任何示例，也可以直接编译示例，无需进行复制。

.. important::

ESP-IDF 编译系统不支持带有空格的路径。

.. \_get-started-connect-legacy:

## 第六步：连接设备

现在，请将您的 ESP32 开发板连接到 PC，并查看开发板使用的串口。

通常，串口在不同操作系统下显示的名称有所不同：

- Windows 操作系统： `COM1` 等
- Linux 操作系统：以 `/dev/tty` 开始

- **MacOS** 操作系统：以 `/dev/cu.` 开始

有关如何查看串口名称的详细信息，请见 :doc: [establish-serial-connection](#)。

.. note::

请记住串口名，您会在下面的步骤中用到。

.. \_get-started-configure-legacy:

## 第七步：配置

请进入 :ref: [get-started-start-project-legacy](#) 中提到的 `hello_world` 目录，并运行工程配置工具 `menuconfig`。

**Linux 和 MacOS** 操作系统

.. code-block:: bash

```
cd ~/esp/hello_world  
make menuconfig
```

**Windows** 操作系统

.. code-block:: batch

```
cd %userprofile%\esp\hello_world  
make menuconfig
```

如果之前的步骤都正确，则会显示下面的菜单：

.. figure:: ../../\_static/project-configuration.png  
:align: center  
:alt: 工程配置 — 主窗口  
:figclass: align-center

工程配置 – 主窗口

进入菜单后，选择 `Serial flasher config > Default serial port` 配置串口（设备将通过该串口加载工程）。按回车键确认选择，点击 `< Save >` 保存配置，然后点击 `< Exit >` 退出 `menuconfig`。

`menuconfig` 工具的常见操作见下。

- `上下箭头`：移动
- `回车`：进入子菜单
- `ESC 键`：返回上级菜单或退出
- `英文问号`：调出帮助菜单（退出帮助菜单，请按回车键）。
- `空格、Y 键 或 N 键`：使能/禁用 `[*]` 配置选项
- `英文问号`：调出有关高亮选项的帮助菜单
- `/ 键`：寻找配置项目

.. note::

如果您是 \*\*Arch Linux\*\* 用户，请前往 ``SDK tool configuration``，并将 ``Python 2 interpreter`` 的名称从 ``python`` 替换为 ``python2``。

.. attention::

如果您使用的是 ESP32-DevKitC（板载 ESP32-SOLO-1 模组），请在烧写示例程序前，前往 ``menuconfig`` 中使能单核模式（:ref:`CONFIG\_FREERTOS\_UNICORE`）。

.. \_get-started-build-and-flash-legacy:

## 第八步：编译和烧录

请使用以下命令，编译烧录工程：

```
make flash
```

运行以上命令可以编译应用程序和所有 **ESP-IDF** 组件，接着生成 **bootloader**、分区表和应用程序二进制文件。接着，这些二进制文件将被烧录至 **ESP32** 开发板。

如果一切顺利，您可在烧录完成后看到类似下方的打印信息（代表加载进程）。接着，开发板将会复位，应用程序“hello\_world”开始启动。

.. highlight:: none

::

```
esptool.py v2.0-beta2
Flashing binaries to serial port /dev/ttyUSB0 (app at offset 0x10000)...
esptool.py v2.0-beta2
Connecting.....—
Uploading stub...
Running stub...
Stub running...
Changing baud rate to 921600
Changed.
Attaching SPI flash...
Configuring flash size...
Auto-detected Flash size:4MB
Flash params set to 0x0220
Compressed 11616 bytes to 6695...
Wrote 11616 bytes (6695 compressed) at 0x00001000 in 0.1 seconds (effective 920.5 kbit/s)...
Hash of data verified.
Compressed 408096 bytes to 171625...
Wrote 408096 bytes (171625 compressed) at 0x00010000 in 3.9 seconds (effective 847.3 kbit/s)...
Hash of data verified.
Compressed 3072 bytes to 82...
Wrote 3072 bytes (82 compressed) at 0x00008000 in 0.0 seconds (effective 8297.4 kbit/s)...
Hash of data verified.

Leaving...
Hard resetting...
```

如果您希望使用 **Eclipse IDE**，而非 `make` 编译系统，请参考 :doc:`Eclipse 指南 <eclipse-setup>`。

.. \_get-started-monitor-legacy:

## 第九步：监视器

您可以使用 `make monitor` 命令，监视“hello\_world”的运行情况。

运行该命令后，:doc: IDF 监视器 <../api-guides/tools/idf-monitor> 应用程序将启动：::

```
$ make monitor
MONITOR
--- idf_monitor on /dev/ttyUSB0 115200 ---
--- Quit:Ctrl+] | Menu:Ctrl+T | Help:Ctrl+T followed by Ctrl+H ---
ets Jun  8 2016 00:22:57

rst:0x1 (POWERON_RESET),boot:0x13 (SPI_FAST_FLASH_BOOT)
ets Jun  8 2016 00:22:57
...
...
```

此时，您就可以在启动日志和诊断日志之后，看到打印的“Hello world!”了。

.. code-block:: none

```
...
Hello world!
Restarting in 10 seconds...
I (211) cpu_start:Starting scheduler on APP CPU.
Restarting in 9 seconds...
Restarting in 8 seconds...
Restarting in 7 seconds...
```

您可使用快捷键 **Ctrl+]**，退出 IDF 监视器。

如果 IDF 监视器在烧录后很快发生错误，或打印信息全是乱码（见下），很有可能是因为您的开发板选用了 **26 MHz** 晶振，而 **ESP-IDF** 默认支持大多数开发板使用的 **40 MHz** 晶振。

.. figure:: ../../\_static/get-started-garbled-output.png  
:align: center  
:alt: 乱码输出  
:figclass: align-center

此时，请您：

1. 退出监视器。
2. 打开 :ref: `menuconfig <get-started-configure>`，
3. 进入 `Component config` -> `ESP32-specific` -> `Main XTAL frequency` 进行配置，将 :ref: `CONFIG_ESP32_XTAL_FREQ_SEL` 设为 **26 MHz**。
4. 然后，请重新 :ref: `编译和烧录 <get-started-build-and-flash-legacy>` 应用程序。

.. note::

您也可以运行以下命令，一次性执行构建、烧录和监视过程：::

```
make flash monitor
```

此外，请前往 :doc: IDF 监视器 <../api-guides/tools/idf-monitor>，了解更多使用 IDF 监视器的快捷键和其他详情。

恭喜，您已完成 **ESP32** 的入门学习！

现在，您可以尝试一些其他 :ref: `examples`，或者直接开发自己的应用程序。

## 环境变量

用户可以在使用 `make` 命令时直接设置部分环境变量，而无需进入 `make menuconfig` 进行重新配置。这些变量包括：

+	+
变量   描述与使用方式	

```
+-----+
| ESPPORT | 覆盖 flash 和 monitor 命令使用的串口。 |
+-----+
|| 例： make flash ESPPORT=/dev/ttyUSB1 , make monitor ESPPORT=COM1 |
+-----+
| ESPBAUD | 覆盖烧录 ESP32 时使用的串口速率。 |
+-----+
|| 例： make flash ESPBAUD=9600 |
+-----+
| MONITORBAUD | 覆盖监控时使用的串口速率。 |
+-----+
|| 例： make monitor MONITORBAUD=9600 |
+-----+
```

.. note::

您可导出环境变量（例：```export ESPPORT=/dev/ttyUSB1```）。  
在同一会话窗口中，如果未被同步覆盖，所有 ```make``` 命令均会使用导出的环境变量值。

## 更新 ESP-IDF

乐鑫会不时推出更新版本的 **ESP-IDF**，修复 **bug** 或提出新的特性。因此，在使用时，您也应注意更新您本地的版本。最简单的方法是：直接删除您本地的 `esp-idf` 文件夹，然后按照 :ref: `get-started-get-esp-idf-legacy` 中的指示，重新完成克隆。

如果您希望将 **ESP-IDF** 克隆到新的路径下，请务必 :doc: `重新设置 IDF_PATH <add-idf_path-to-profile>`。否则，工具链将无法找到 **ESP-IDF**。

此外，您可以仅更新变更部分。具体方式，请前往 :ref: `更新 <updating>` 章节查看。

## 相关文档

.. toctree::  
:maxdepth: 1

```
add-idf_path-to-profile
establish-serial-connection
make-project
eclipse-setup
../api-guides/tools/idf-monitor
toolchain-setup-scratch
```

.. \_Stable version: [https://docs.espressif.com/projects/esp-idf/zh\\_CN/stable/](https://docs.espressif.com/projects/esp-idf/zh_CN/stable/)  
.. \_Releases page: <https://github.com/espressif/esp-idf/releases>