



ESP-IDF 开发入门

2022 全国大学生物联网设计竞赛

在线培训课程

ESP-College



目录

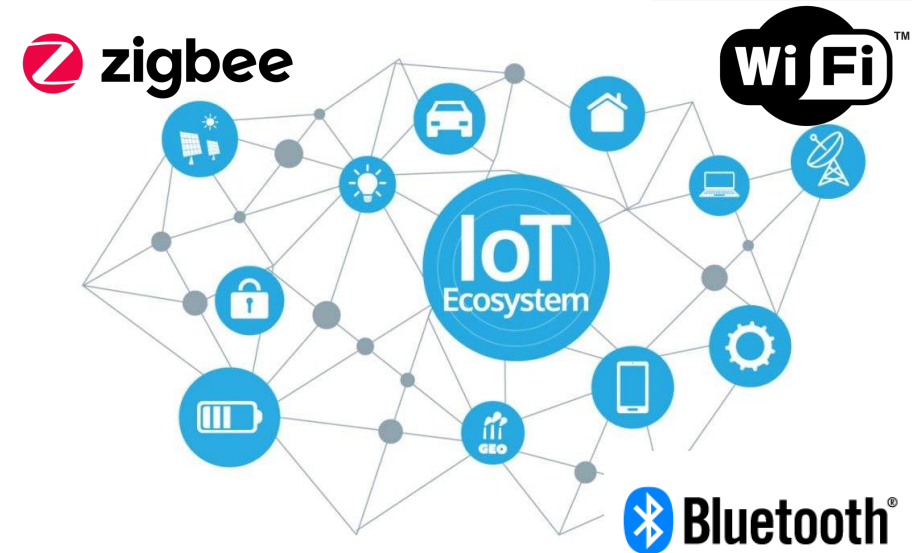
-  ESP32-S3 芯片资源介绍
-  ESP-IDF 物联网开发框架详解
-  ESP-IDF 开发环境搭建教程
-  ESP-IDF 编程实践

/01

ESP32-S3 芯片资源介绍

乐鑫 AIoT 系列芯片

特性	ESP32 系列	ESP32-S2 系列	ESP32-C3 系列	ESP32-S3 系列
发布时间	2016	2020	2020	2020
产品型号	请参考 ESP32 技术规格书 (PDF)	请参考 ESP32-S2 技术规格书 (PDF)	请参考 ESP32-C3 技术规格书 (PDF)	请参考 ESP32-S3 技术规格书 (PDF)
内核	搭载低功耗 Xtensa® LX6 32 位双核/单核处理器	搭载低功耗 Xtensa® LX7 32 位单核处理器	搭载 RISC-V 32 位单核处理器	搭载低功耗 Xtensa® LX7 32 位双核处理器
Wi-Fi 协议	802.11 b/g/n、2.4 GHz	802.11 b/g/n、2.4 GHz	802.11 b/g/n、2.4 GHz	802.11 b/g/n、2.4 GHz
Bluetooth®	Bluetooth v4.2 BR/EDR 和 Bluetooth Low Energy	×	Bluetooth 5.0	Bluetooth 5.0
主频	240 MHz (ESP32-S0WD 为 160 MHz)	240 MHz	160 MHz	240 MHz
SRAM	520 KB	320 KB	400 KB	512 KB



ESP32-S3 芯片、模组、开发板

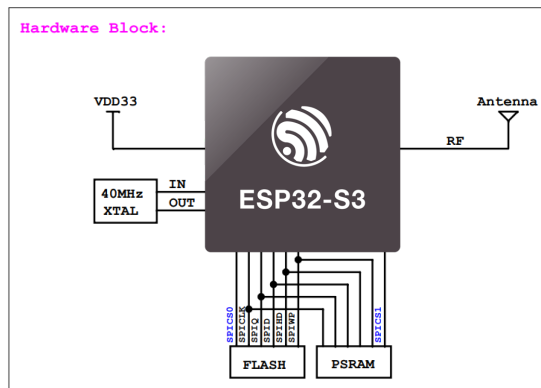
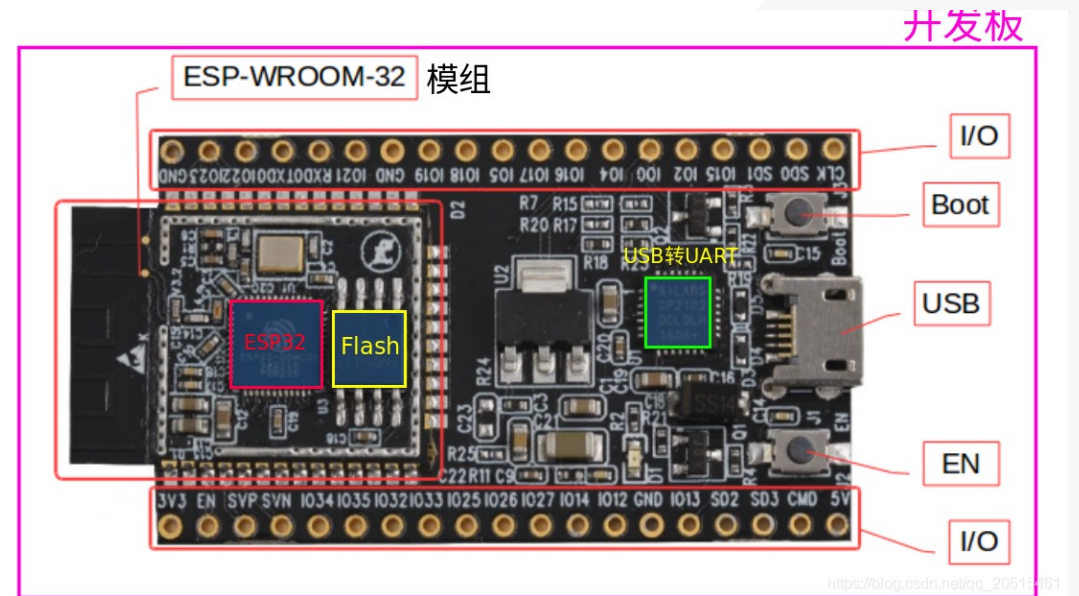


图 1: 模组功能块图

芯片



模组



开发板

ESP32-S3 硬件资源



无线功能:

- ✓ Wi-Fi: IEEE 802.11 b/g/n-compliant
- ✓ Bluetooth LE: Bluetooth 5, mesh

模拟外设:

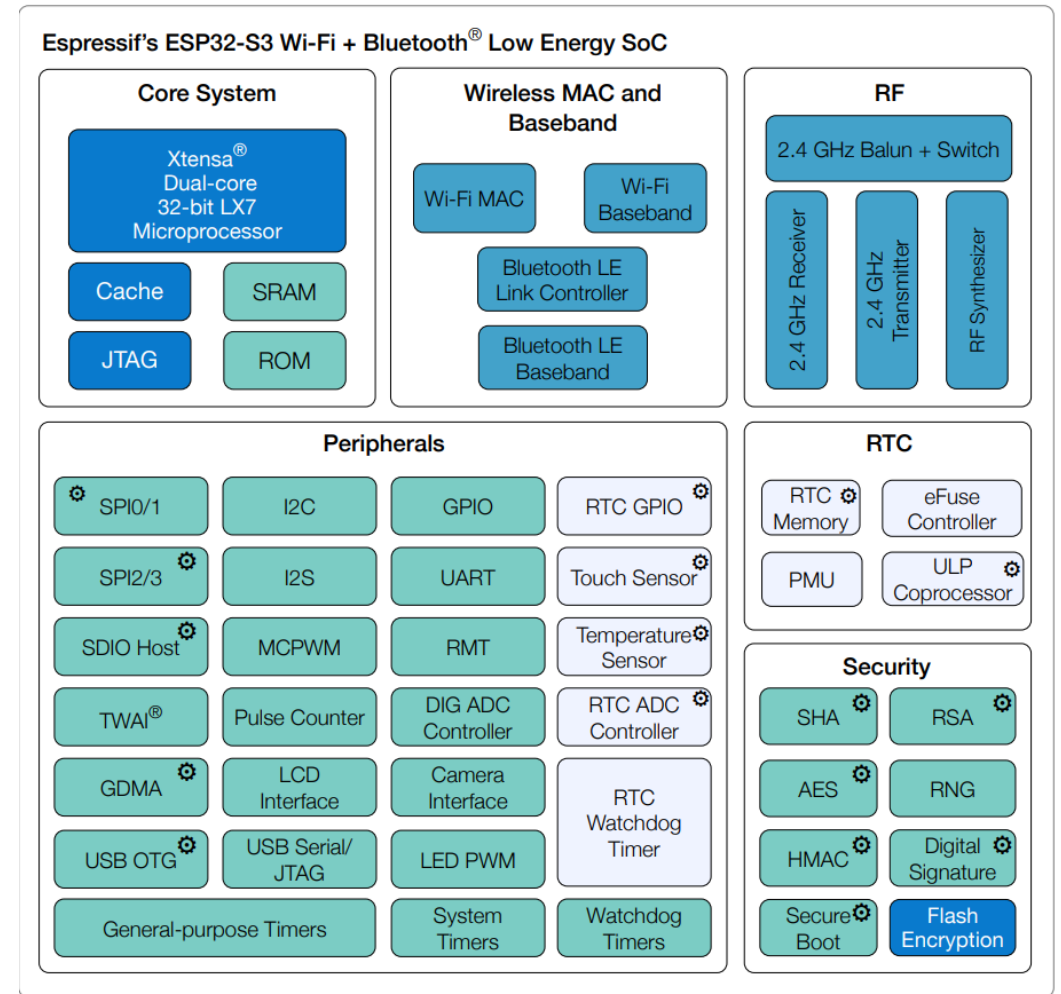
- ✓ 12bit ADC * 20 channels
- ✓ Temperature sensor * 1
- ✓ Touch sensor (触摸) * 14

硬件加速:

- ✓ Hash, RSA, AES ...

数字外设:

- ✓ GPIO * 45
- ✓ SPI * 4
- ✓ UART * 3
- ✓ I2C * 2
- ✓ I2S * 2
- ✓ LEDC/PWM * 8
- ✓ RMT * 1 (红外 发送/接收)
- ✓ MCPWM * 2 (电机控制)
- ✓ USB_OTG * 1 (USB 主机/设备)
- ✓ SDIO * 1 (SD 卡)
- ✓ TWAI * 1 (兼容 CAN 总线)



Modules having power in specific power modes:

- Active
- Active and Modem-sleep
- Active, Modem-sleep, and Light-sleep; optional in Light-sleep
- all modes; optional in Deep-sleep

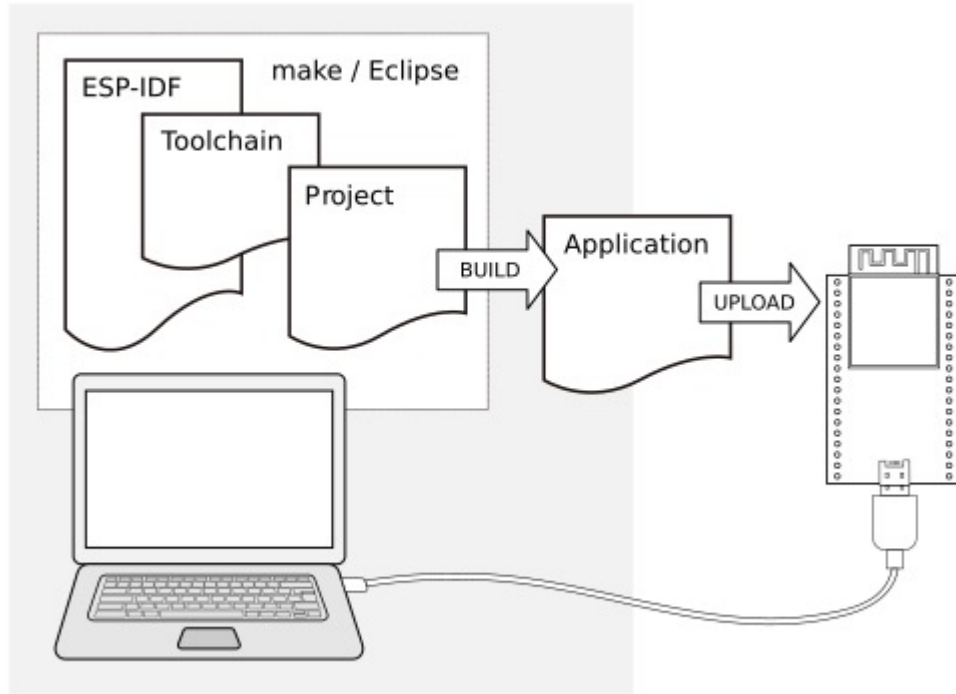
ESP32-S3 硬件 Tips

1. 大多数外设信号可以任意配置引脚
2. 部分引脚具有特殊功能或固定功能 (GPIO 0,3,45,46, 19,20, 26-32..)
3. 外设包含 LL、HAL、以及系统层 Driver, 开发者不必了解寄存器细节
4. ESP32S3 为同构双核, 共同地址空间, 任务可在两个内核上自由调度
5. NOR Flash 存储程序和数据, 通过 4 线或 8 线 SPI 连接, 通常为 4MB, 8MB, 16MB
6. 内置 512KB RAM, 可增加 PSRAM 扩展内存, 与 Flash 挂在到同一 SPI 总线, 通常为 2MB, 4MB, 8MB
7. CPU 频率可配置, 默认 160MHz, 可提升到 240MHz (ESP32S3_DEFAULT_CPU_FREQ_MHZ)
8. ESP32S3 支持多种低功耗模式, 可同时配置多种唤醒源 (timer、IO、触摸、uart 等)

/02

ESP-IDF 物联网开发框架详解

ESP-IDF 概述

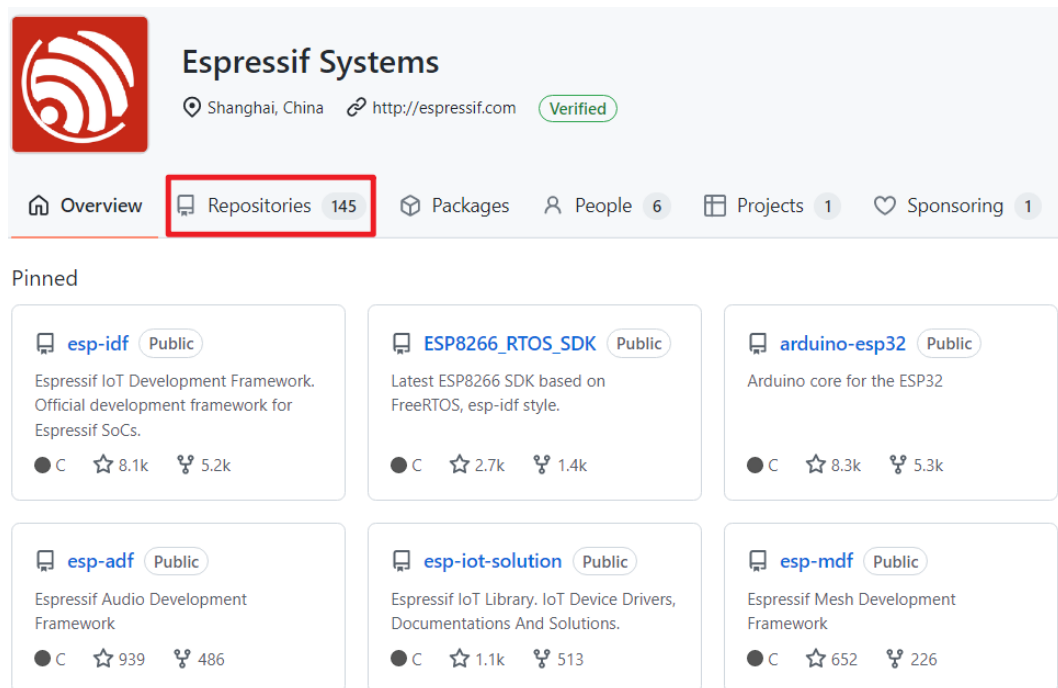


Development of applications for ESP32

ESP-IDF (Espressif IoT Development Framework) 是使用乐鑫芯片进行物联网开发时，必要的**基础代码库**、**示例程序**、**文档**和**工具**的集合。

- Toolchain to compile code for ESP32
- Build tools - CMake and Ninja to build a full Application for ESP32
- ESP-IDF that essentially contains API (software libraries and source code) for ESP32 and scripts to operate the Toolchain
- Text editor to write programs (Projects) in C

ESP-IDF 开源仓库



Espressif Systems
Shanghai, China <http://espressif.com> Verified

Overview **Repositories 145** Packages People 6 Projects 1 Sponsoring 1

Pinned

- esp-idf** Public
Espressif IoT Development Framework. Official development framework for Espressif SoCs.
● C ☆ 8.1k 🍴 5.2k
- ESP8266_RTOS_SDK** Public
Latest ESP8266 SDK based on FreeRTOS, esp-idf style.
● C ☆ 2.7k 🍴 1.4k
- arduino-esp32** Public
Arduino core for the ESP32
● C ☆ 8.3k 🍴 5.3k
- esp-adf** Public
Espressif Audio Development Framework
● C ☆ 939 🍴 486
- esp-iot-solution** Public
Espressif IoT Library. IoT Device Drivers, Documentations And Solutions.
● C ☆ 1.1k 🍴 513
- esp-mdf** Public
Espressif Mesh Development Framework
● C ☆ 652 🍴 226

ESP-IDF 使用 **Apache License 2.0** 开源协议：
通过开源软件开发协作，提供可靠且长久不衰的软件产品。

1. 修改后的代码可以闭源，但要提及使用了开源代码，并标明原出处
2. 修改后的代码可以开源，需要保留开源声明并为改动的代码文件添加修改说明
3. 授予版权和专利许可，可以商业使用



[Github: Espressif Systems \(https://github.com/espressif\)](https://github.com/espressif)

[Gitee: 乐鑫开源 \(https://gitee.com/EspressifSystems\)](https://gitee.com/EspressifSystems)

ESP-IDF 系统组件



RTOS 内核



标准编程接口



外设驱动程序



Wi-Fi



经典蓝牙和低功耗蓝牙



网络协议



电源管理



存储



安全性



网络配置



构建系统

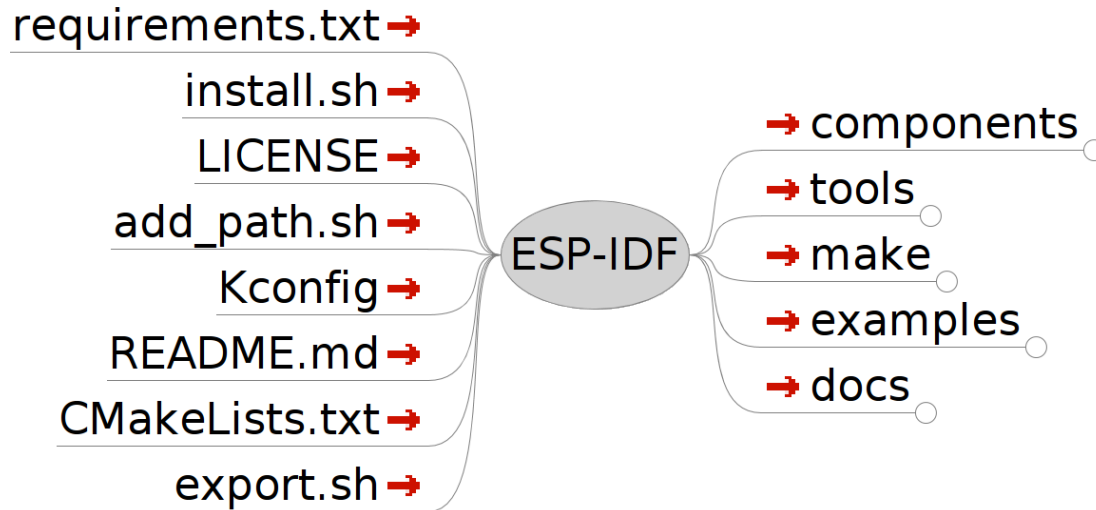


开发工具



IDE 支持

ESP-IDF 目录结构



components: 以组件的形式提供的基础代码库

examples: 示例程序，包括外设示例等

docs: ESP-IDF 文档

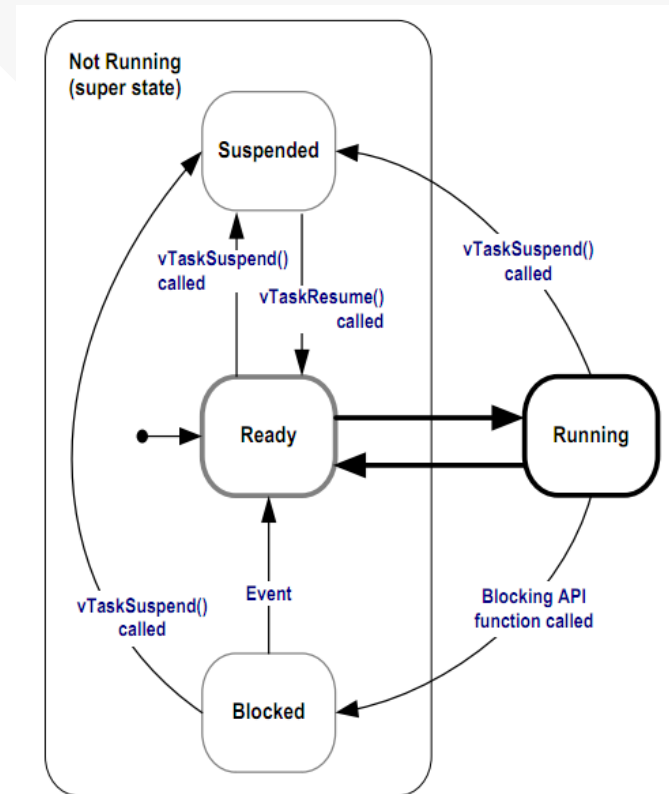
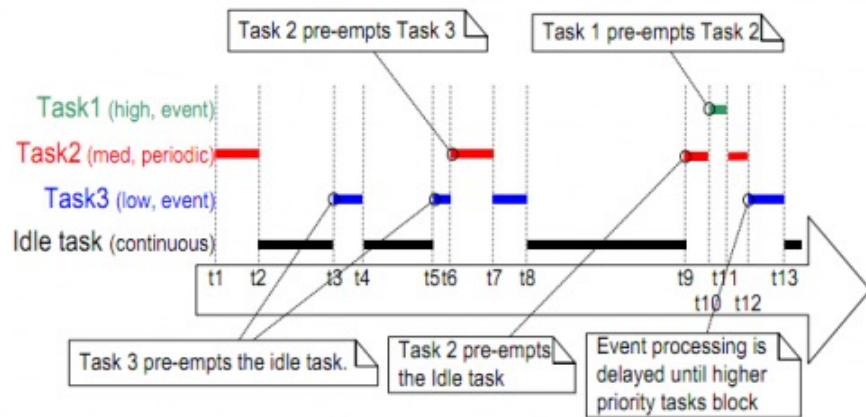
在线版本 <https://docs.espressif.com/projects/esp-idf>

tools: 脚本工具，包括 idf.py, kconfig 等

make: 默认构建规则，包括 project.mk 等

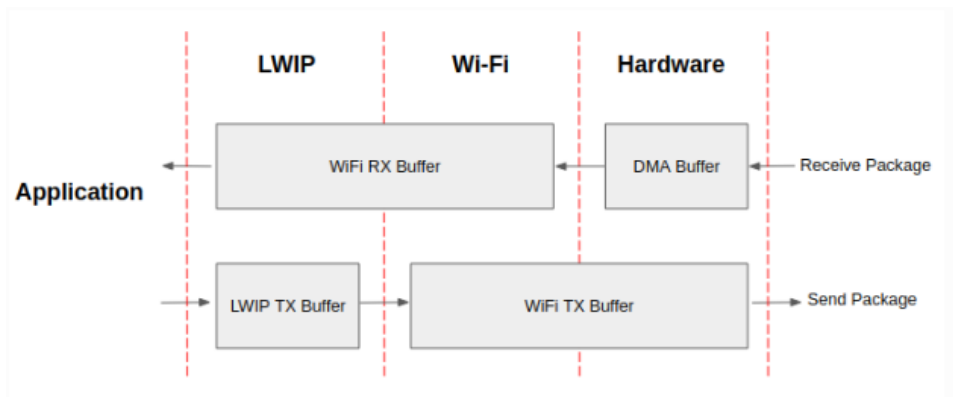
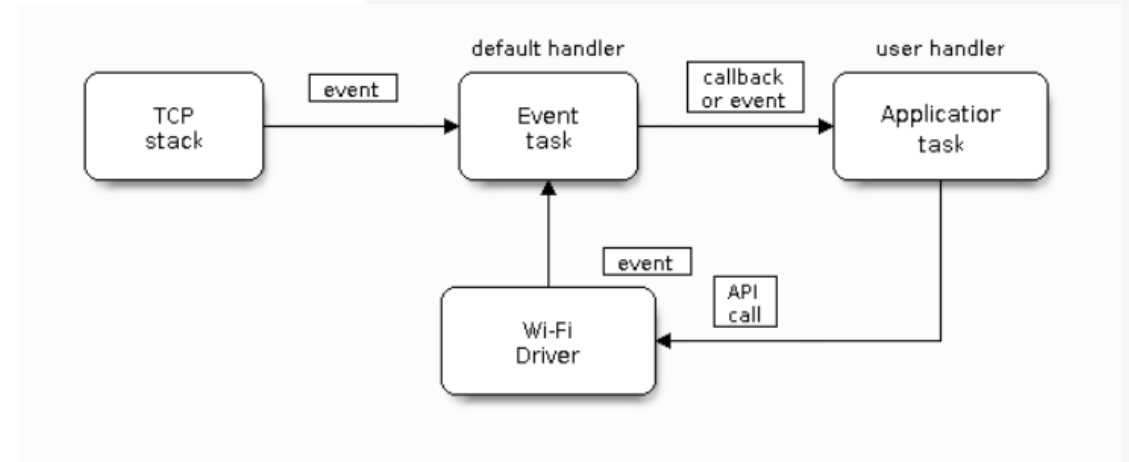
RTOS 内核

- 乐鑫为 FreeRTOS 开发了多核支持，可在 ESP32-S3 双核芯片上运行
- 多任务、抢占式调度、时间片轮转调度
- 队列、信号量、事件组等操作系统通信机制



Wi-Fi / 蓝牙驱动

- 乐鑫自研 Wi-Fi 协议栈，支持 STA、SoftAP
- 乐鑫自研 Mesh 组网、ESP-NOW 协议
- 开放底层接口，支持用户收发 802.11 底层包，支持收发 CSI 信息
- 支持多种 Wi-Fi 配网方式（BLE, SoftAP, DPP, Console）
- 支持多种 Wi-Fi 低功耗模式



通信协议相关组件

- ESP-IDF 官方支持数十种[网络协议栈](#)
 - [TCP/IP](#): 基于 LWIP, 支持 Socket 接口, 支持 ICMP、DHCP、mDNS、NAPT
 - [MQTT](#): 支持 MQTT over TCP/ WebSocket/ SSL with mbedtls
 - HTTP(S): 支持 [Server](#) 模式、[Client](#) 模式
 - [WebSocket Client](#): 支持 WebSocket over TCP, TLS with mbedtls
 - [Modbus](#): 支持 Modbus RTU, Modbus ASCII, Modbus TCP/IP
 - [ESP-NOW](#): 乐鑫 2.4G 自定义通信协议
 - [ESP Local Control](#): 基于 Wi-Fi + HTTPS or BLE 乐鑫自主本地控制协议
 - [ESP-Modem](#): 支持 PPP 协议拨号上网, 可对接 2G/3G/4G/5G 模组
 - [BLE-MESH](#): 标准 BLE MESH 组网协议
 - [ESP Wi-Fi MESH](#): 乐鑫 Wi-Fi MESH 组网协议



外设驱动相关组件

- ESP-IDF 官方支持[多种外设接口](#)
 - [ADC](#): 支持单次读取或 DMA 模式, 12bit, 最大测量范围 0~3100mV
 - [LEDC \(PWM\)](#): 可配置 8 路输出, 最大 40MHz 输出频率, 支持自动占空比, 用于驱动 LED 或电机
 - [I2C](#): 支持 Master 模式或 Slave 模式, 支持 8bit 或 10 bit 地址模式, 已适配[多种传感器](#)
 - [SPI](#): 支持 Master 模式或 Slave 模式, 支持挂载 Flash 等存储设备, SPI 接口的显示设备
 - [SDIO](#): 支持挂载高速 SD 卡, 已适配文件系统
 - [I2S](#): 支持 Philips 模式、PCM 模式等多种格式, 已适配[多种 codec 芯片](#)
 - [LCD](#): 支持 SPI、8080、RGB、I2C 等多种接口的屏幕
 - [Touch](#): 内置触摸传感器, 14 通道, 支持触摸按键、滑条、接近感应等
 - [RMT](#): 支持多种收发模式, 已适配 [NEC 协议](#)、[DShot 协议](#)、[LED 灯条](#) 等
 - [USB](#): 支持主机模式或设备模式, 支持 [U 盘](#)、[USB 摄像头](#) 等
 - ...

外设相关示例程序：<https://github.com/espressif/esp-idf/tree/master/examples/peripherals>

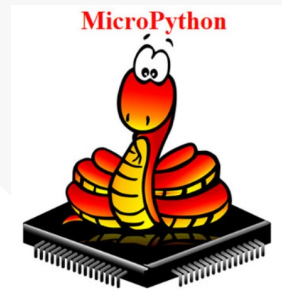
其它外接模块驱动程序：<https://github.com/espressif/esp-iot-solution>

ESP32-S3 软件 Tips

1. ESP-IDF 可**跨芯片平台**，同一份代码可以编译在 ESP32、S2、S3、C3.. 不同芯片
2. ESP-IDF 中的**外设驱动可能不支持线程安全**，尽量将对同一外设的操作放在一个任务中
3. RTOS 为 ESP-IDF 核心组件，**默认不支持裸机编程**
4. 尽量避免使用全局变量，**任务间通信可使用队列、信号量、事件循环等操作系统接口**
5. **默认双核同时开启**，也可以只开一个内核（CONFIG_FREERTOS_UNICORE）
6. 任务创建**默认开启负载均衡**，也可以指定工作内核（xTaskCreatePinnedToCore）

ESP32(-S2/S3/C3) 第三方开发环境

- * **Arduino:** 是一个开源硬件和开源软件平台，支持包括 ESP32 在内的大量微控制器，它定义了基于 C++ 语言的 Arduino API，由于接口简单且标准，被广泛应用在原型开发和教学领域。
- * **MicroPython:** 是一个可在 ESP32 上运行的 Python 语言解析器，支持通过简单的脚本语言调用 ESP32 的外设资源和通信功能，借助脚本语言的特性，开发者不再需要重复代码的编译和烧录过程。
- * **NodeMCU:** 是一个针对 ESP 系列芯片开发的 LUA 语言解析器，几乎支持 ESP 芯片所有外设功能，相比 MicroPython 也更加轻量。同样它拥有脚本语言无需重复编译的优点。



/03

ESP-IDF 开发环境搭建教程

ESP-IDF 版本介绍

Chip	v3.3	v4.1	v4.2	v4.3	v4.4	v5.0
ESP32	supported	supported	supported	supported	supported	supported
ESP32-S2			supported	supported	supported	supported
ESP32-C3				supported	supported	supported
ESP32-S3				preview	supported	supported
ESP32-H2					preview	preview
ESP32-C2						preview



本次大赛推荐选择稳定版本 v4.4.1

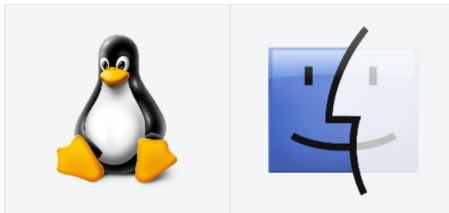
1. 对于入门开发者，推荐选择稳定版本 v4.4 及其修订版本
2. 如果有量产需求，推荐使用最新稳定版本，以便获得最及时的技术支持
3. 如需尝试新芯片或者预研产品新功能，请使用 master 分支，最新版本包含所有最新特性，但存在已知或未知的 Bug
4. 如需使用稳定版本没有的新特性，又想降低使用 master 分支的风险，请使用对应的发布分支，例如 release/v4.4 分支（ESP-IDF GitHub 会先创建 release/v4.4 分支，等完成全部功能开发和测试，再基于该分支的某一历史节点发布稳定版本 v4.4）

Linux / MacOS 环境搭建

在 Linux 系统上开发 ESP32 将获得最佳的编译速度和开发体验。我们这里推荐使用 Ubuntu 20.04 及以上系统

Installation Step by Step:

<https://docs.espressif.com/projects/esp-idf/en/release-v4.4/esp32s3/get-started/index.html#installation-step-by-step>



问题 1: Linux 发行版?

- 建议使用 Ubuntu20.04 及以上版本

问题 2: 下载速度慢?

- Ubuntu 的源切换为中国的服务器 (Server for China)
- 使用 Gitee 加速 GitHub 代码下载, 参考 <https://gitee.com/EspressifSystems/esp-gitee-tools.git>
- 使用乐鑫服务器, 加速工具链下载: [Espressif download server 设置](#)

问题 3: Python 版本不受支持?

- ESP-IDF v4.3 以上版本要求 Python 版本不低于 v3.6, 切换默认 Python 版本请参考: [使用 update-alternatives 切换 Python 版本](#)

Windows 环境搭建

推荐使用 Windows 10、11 系统

方法 1. 使用安装器安装:

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

方法 2. WSL2 安装, 同 Linux, 需要注意 USB 驱动问题 (建议使用 USBIPD)



↓
Universal Online Installer 2.15
Windows 10, 11
Size: 4 MB

在线安装工具, 自行勾选版本或组件, 安装过程中下载

↓
Espressif-IDE 2.4.2 with ESP-IDF
v4.4
Windows 10, 11
Size: 1 GB

v4.4 最新稳定版本 + Espressif IDE 离线安装包

↓
ESP-IDF v4.4.1 - Offline Installer
Windows 10, 11
Size: 600 MB

v4.4 最新稳定版本离线安装包

↓
ESP-IDF v4.3.2 - Offline Installer
Windows 10, 11
Size: 570 MB

v4.3 更新版本离线安装包

↓
ESP-IDF v4.2.3 - Offline Installer
Windows 10, 11
Size: 376 MB

v4.2 更新版本离线安装包

↓
ESP-IDF v4.1.3 - Offline Installer
Windows 10, 11
Size: 353 MB

v4.1 更新版本离线安装包

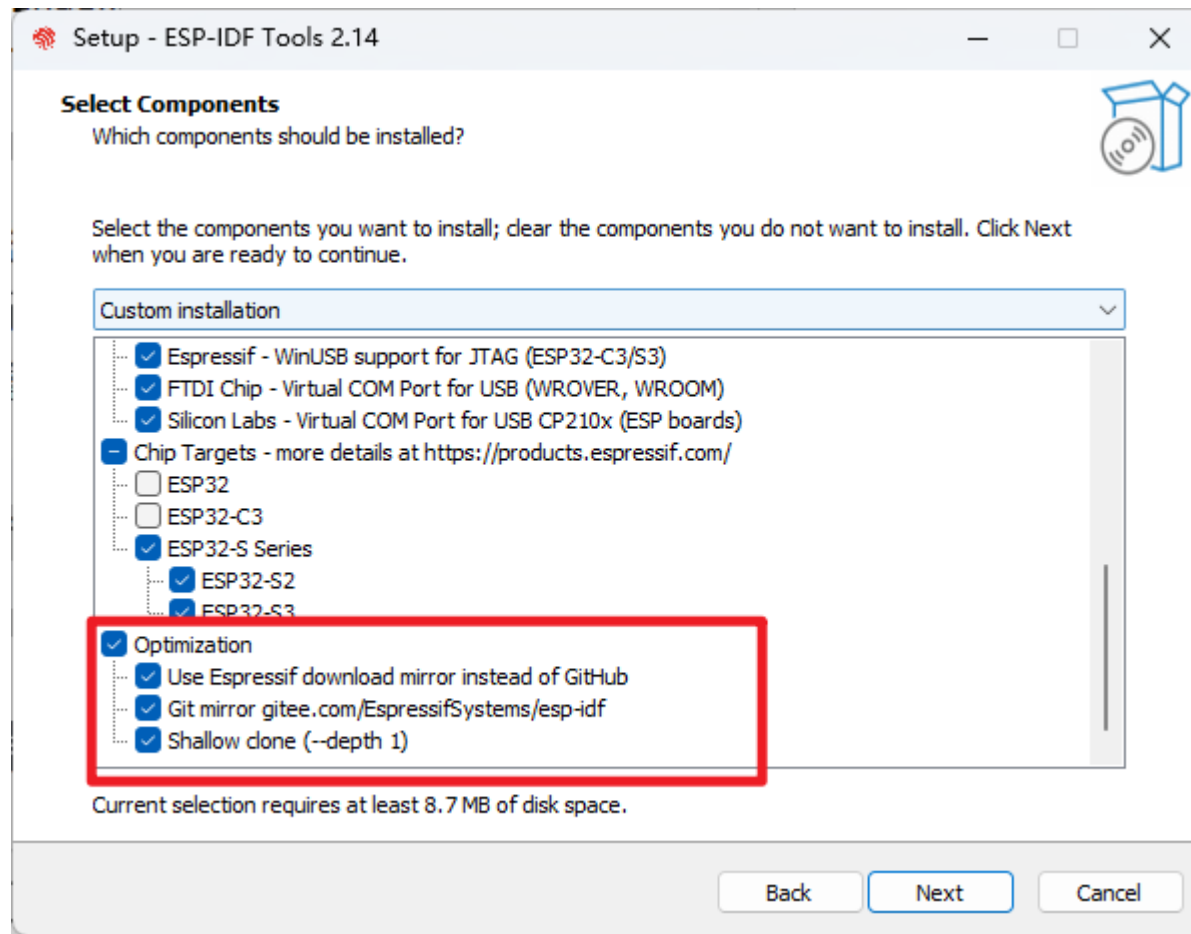
Windows 环境搭建

推荐使用 Windows 10、11 系统

方法 1. 使用安装器安装:

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

方法 2. WSL2 安装, 同 Linux, 需要注意 USB 驱动问题 (建议使用 USBIPD)



/04 ESP-IDF 编程实战

ESP-IDF 示例工程结构

```
- myProject/  
  - CMakeLists.txt  
  - sdkconfig  
  - components/  
    - component1/  
      - CMakeLists.txt  
      - Kconfig  
      - src1.c  
    - component2/  
      - CMakeLists.txt  
      - Kconfig  
      - src1.c  
      - include/ - component2.h  
  - main/  
    - CMakeLists.txt  
    - src1.c  
    - src2.c  
  - build/
```

- **CMakeList.txt**: 根目录的 CMakeList 描述了项目的构建行为，每个 component 中的 CMakeList 描述了组件的构建行为，例如指定编译哪些 .c 文件，包含哪些 .h 文件
- **sdkconfig**: 保存了项目宏定义，在使用 menuconfig 时自动生成
- **components**: 用户组件目录，可自定义组件，也可对 IDF_PATH 下的同名组件进行覆盖
- **main**: “伪组件”，与 components 中的组件遵循相同的编译规则，其中的 app_main 函数为默认的执行入口
- **build**: 编译时自动保存的文件，无需修改

idf.py 常用指令

- **idf.py help**: 将输出指令列表和使用说明
- **idf.py set-target <target>**: 设置编译目标, <target> 请替换为 esp32s3 等
- **idf.py menuconfig**: 运行 menuconfig 终端图像化配置工具, 可以选择或修改配置选项, 配置结果将保存在 sdkconfig 文件
- **idf.py build**: 开始编译代码, 编译产生的中间文件和最终的可执行程序, 将默认保存在项目 build 目录, 编译过程是增量式的, 如果仅对一个源文件进行修改, 下次编译讲只重新编译已修改的文件
- **idf.py fullclean**: 删除整个 build 目录下的内容, 包括所有 CMake 的配置输出文件。下次构建项目时, CMake 会从头开始配置项目。请注意, 该命令会递归删除构建目录下的所有文件, 请谨慎使用。但项目配置文件同样不会被删除。
- **idf.py flash**: 将 build 生成的可执行程序二进制文件烧录进目标 ESP32-C3 设备中。-p <port_name> 和 -b <baud_rate> 选项可分别设置串口的设备名和烧录时的波特率, 如果不指定将自动搜索串口, 并使用默认波特率
- **idf.py monitor** 用于显示目标 ESP32-C3 设备的串口输出。同样 -p 选项可用于设置主机端串口的设备名, 串口打印期间, 可按下组合键 Ctrl-] 退出监视器。

代码编译和调试过程

```
ESP-IDF 4.4 PowerShell
IDF_PYTHON_ENV_PATH      C:\Users\LEEB0-AMD\.espressif\python_env\idf4.4_py3.8_env
Added to PATH
-----
D:\Espressif\frameworks\esp-idf-v4.4.1\components\esptool_py\esptool
D:\Espressif\frameworks\esp-idf-v4.4.1\components\app_update
D:\Espressif\frameworks\esp-idf-v4.4.1\components\espcoredump
D:\Espressif\frameworks\esp-idf-v4.4.1\components\partition_table
C:\Users\LEEB0-AMD\.espressif\tools\xtensa-esp32s2-elf\esp-2021r2-patch3-8.4.0\xtensa-esp32s2-elf\bin
C:\Users\LEEB0-AMD\.espressif\tools\xtensa-esp32s3-elf\esp-2021r2-patch3-8.4.0\xtensa-esp32s3-elf\bin
C:\Users\LEEB0-AMD\.espressif\tools\riscv32-esp-elf\esp-2021r2-patch3-8.4.0\riscv32-esp-elf\bin
C:\Users\LEEB0-AMD\.espressif\tools\esp32s2ulp-elf\2.28.51-esp-20191205\esp32s2ulp-elf-binutils\bin
C:\Users\LEEB0-AMD\.espressif\tools\cmake\3.20.3\bin
C:\Users\LEEB0-AMD\.espressif\tools\openocd-esp32\v0.11.0-esp32-20211220\openocd-esp32\bin
C:\Users\LEEB0-AMD\.espressif\tools\ninja\1.10.2\
C:\Users\LEEB0-AMD\.espressif\tools\idf-exe\1.0.3\
C:\Users\LEEB0-AMD\.espressif\tools\ccache\4.3\ccache-4.3-windows-64
C:\Users\LEEB0-AMD\.espressif\tools\dfu-util\0.9\dfu-util-0.9-win64
D:\Espressif\frameworks\esp-idf-v4.4.1\tools
Checking if Python packages are up to date...
Python requirements from D:\Espressif\frameworks\esp-idf-v4.4.1\requirements.txt are satisfied.

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

PS D:\Espressif\frameworks\esp-idf-v4.4.1>
```



乐鑫学习资源-Part1

1. 乐鑫 ESP-IDF 官方文档: <https://docs.espressif.com/projects/esp-idf/en/v4.4.1/esp32s3/index.html>
2. 乐鑫 Github 开源仓库: <https://github.com/espressif>
 1. ESP-IDF 开发框架: <https://github.com/espressif/esp-idf>
 2. ESP-DL 深度学习引擎: <https://github.com/espressif/esp-dl>
 3. ESP-WHO AI 图像: <https://github.com/espressif/esp-who>
 4. ESP-SR AI 语音: <https://github.com/espressif/esp-sr>
 5. ESP-BOX AIoT 套件: <https://github.com/espressif/esp-box>
 6. ESP-CSI 无线感知: <https://github.com/espressif/esp-csi>
 7.
3. ESP32-S3 技术参考手册 https://www.espressif.com/sites/default/files/documentation/esp32-s3_technical_reference_manual_en.pdf



Thank you !

