INTRODUCTION TO RISC-V

RISC-V入门教程

ISA | 汇编指令 | 系统编程 | 组成原理 | 嵌入式应用

基于RISC-V的嵌入式系统开发

主讲 邢建国

中国开放指令生态（RISC-V）联盟 | 浙江中心
China RICS-V Alliance | Zhejiang Center

浙江图灵算力研究院
ZHEJIANG TURING INSTITUTE

嵌入式系统开发

- 嵌入式系统
- RISC-V在嵌入式系统中的应用
- FPGA与嵌入式系统开发
- 嵌入式系统开发模式

嵌入式系统

嵌入式系统是一种为特定功能设计的专用计算机系统，通常嵌入在更大的设备或系统中，结合硬件和软件以实现高效、可靠的实时控制或数据处理。

- 专用性：针对特定任务优化，如智能手表的时间管理或汽车ABS的刹车控制。
- 集成性：硬件（如微控制器）与定制化软件紧密结合，体积小、成本低。
- 实时性：许多系统需即时响应，如医疗设备的心跳监测。

■ 嵌入式系统

主要特征

- 资源受限：有限的内存（如KB级）、处理能力（低功耗CPU）和存储空间。
- 低功耗设计：依赖电池的设备（如传感器）需优化能耗，采用休眠模式。
- 高可靠性：工业控制系统需在恶劣环境（高温、震动）中稳定运行。
- 长生命周期：工业设备可能使用10年以上，软件需长期维护。

嵌入式系统

典型应用场景

- 消费电子：扫地机器人（路径规划）、智能电视（视频解码）。
- 汽车电子：自动驾驶模块（传感器融合）、车载娱乐系统。
- 工业自动化：PLC控制机械臂、温度监控系统。
- 医疗设备：胰岛素泵（精准给药）、MRI机器控制。
- 物联网：智能电表（远程抄表）、农业传感器（土壤湿度监测）。

嵌入式系统

系统组成

- 硬件部分：
  - 处理器：ARM Cortex-M系列（低功耗）、Raspberry Pi（高性能应用）。
  - 传感器/执行器：陀螺仪（无人机平衡）、电机（机器人关节控制）。
  - 通信模块：Wi-Fi（智能家居）、蓝牙（无线耳机）。
- 软件部分：
  - 实时操作系统（RTOS）：FreeRTOS（无人机飞控）、Zephyr（IoT设备）。
  - 中间件：协议栈（如TCP/IP）、数据库（轻量级SQLite）。
  - 应用逻辑：基于C/Python的算法，如人脸识别（边缘AI摄像头）

嵌入式系统

开发技术与挑战

- 硬件设计：PCB布局优化（减少电磁干扰）、选择低功耗芯片（如ESP32）。
- 软件优化：减少内存占用（静态分配代替动态）、算法加速（FFT运算）。
- 调试工具：JTAG调试器、逻辑分析仪（信号抓取）。
- 安全挑战：防止固件篡改（加密签名）、数据加密（TLS协议）。

嵌入式系统

未来趋势

- AIoT融合：端侧AI（如TensorFlow Lite在微控制器运行）。
- 边缘计算：本地数据处理（减少云依赖），如智能摄像头的实时分析。
- RISC-V架构：开源硬件（降低成本，定制指令集）。
- 功能安全：ISO 26262（汽车，《道路车辆功能安全》）、IEC 62304（医疗，《医疗器械软件 – 软件生命周期流程》）标准合规。

RISC-V在嵌入式系统中的应用

- RISC-V作为一种开源的指令集架构（ISA），在嵌入式系统领域展现出独特的优势，并正在快速改变行业格局。
- RISC-V凭借开源自由、灵活定制、成本优势，正在嵌入式系统领域快速渗透，尤其在IoT、工业控制、边缘AI等场景表现突出。
- 未来随着向量计算、功能安全、生态标准化的推进，其应用范围将进一步扩大，成为挑战ARM主导地位的重要力量。
- 短期内可能形成“ARM主导高端，RISC-V覆盖中低端+定制领域”的格局，长期看有望重塑嵌入式处理器市场。

RISC-V在嵌入式系统中的核心优势

- 开源免授权费，降低开发成本
  - 零许可费：RISC-V指令集开源，无需向ARM或MIPS支付授权费用，尤其适合中小企业和初创公司。
  - 自主可控：企业可基于开源代码定制芯片（如阿里平头哥玄铁系列），避免受制于商业架构的供应链限制。
- 高度可定制化
  - 模块化指令集：
    - 基础指令集（RV32I/RV64I）精简（仅40条指令），适合资源受限的嵌入式场景（如传感器）。
    - 支持扩展指令（如浮点运算F/D、向量计算V、自定义指令），灵活适配不同需求：
      - 工业控制：添加实时中断处理指令。
      - AI边缘设备：集成AI加速扩展（如Tensor操作指令）。
  - 微架构自由设计：
    - 可自主优化流水线、缓存结构（如SiFive E7系列针对低功耗优化），平衡性能与功耗。

RISC-V在嵌入式系统中的核心优势

- 低功耗与高性能平衡
  - 精简设计：RISC-V架构无历史包袱，指令解码逻辑简单（如单周期指令执行），适合低功耗MCU（如GD32V系列功耗比同类ARM Cortex-M低20%）。
  - 并行化潜力：支持多核异构设计（如蜂鸟E203支持RISC-V+NPU），满足实时性与算力需求。
- 生态系统快速成熟
  - 工具链支持：
    - 编译器：GCC/LLVM已全面支持RISC-V。
    - 调试工具：OpenOCD、J-Link适配RISC-V调试接口。
  - 操作系统适配：
    - FreeRTOS、Zephyr、RT-Thread等嵌入式RTOS原生支持。
    - Linux已支持64位RISC-V（如StarFive VisionFive开发板）。

RISC-V在嵌入式系统的典型应用

- 物联网（IoT）设备
  - 低功耗传感器节点：
    - 芯片示例：Espressif ESP32-C5（Wi-Fi 6 + RISC-V核）。
    - 优势：通过定制指令优化无线协议栈（如LoRaWAN），续航提升30%。
  - 边缘AI终端：
    - 案例：嘉楠科技K510芯片集成RISC-V双核+NPU，支持人脸识别和语音唤醒。
- 工业控制
  - 实时控制器：
    - 芯片示例：SiFive E76 MCU（400MHz，支持CAN总线）。
    - 应用：PLC逻辑控制，中断响应延迟<100ns。
  - 功能安全：
    - 方案：芯来科技N100系列通过ISO 26262 ASIL-D认证，用于汽车ECU。

RISC-V在嵌入式系统的典型应用

- 消费电子
  - 智能穿戴设备：
    - 案例：华米Amazfit手表采用RISC-V协处理器，负责传感器数据预处理，延长主控续航。
    - 家电控制：
    - 方案：兆易创新GD32VF103用于空调变频控制，支持PWM波形动态调整。
- AI与边缘计算
  - 端侧推理加速：
    - 芯片：平头哥曳影1520，集成4核RISC-V + 4TOPS NPU，运行YOLOv5s仅需5ms。
  - 语音交互：
    - 方案：Syntiant NDP200芯片（RISC-V+神经网络加速器），实现离线语音唤醒。

RISC-V在嵌入式领域的未来趋势

- . 架构创新加速
  - 向量计算扩展（RVV）普及：
    - RVV 1.0标准支持SIMD操作，将推动边缘AI性能提升（如图像处理速度翻倍）。
  - 安全性增强：
    - 专用安全扩展（如Pointer Masking、内存隔离），满足医疗/汽车等高安全场景需求。
- 垂直领域深度定制
  - 领域专用架构（DSA）：
    - 汽车：针对AUTOSAR标准优化实时性指令。
    - 医疗：集成生物信号处理加速单元（如ECG滤波算法硬件化）。

RISC-V在嵌入式领域的未来趋势

- 生态融合与标准化
  - 跨平台兼容：
    - RISC-V International推动统一ABI标准，简化跨厂商代码移植（如与ARM二进制兼容性探索）。
  - 云-端协同：
    - 阿里云“无剑”平台提供RISC-V芯片设计+云端训练一体化方案，加速AIoT落地。
- 挑战与竞争
  - 生态成熟度：
    - 仍需完善中间件（如AI框架适配）、商业EDA工具支持（Synopsys/Cadence全面兼容）。
  - 与ARM的竞争：
    - 在高端市场（如Cortex-A系列替代）仍需突破，但在嵌入式MCU领域已形成替代趋势（2023年RISC-V MCU市占率超15%）。

FPGA与嵌入式系统开发

- FPGA在嵌入式系统中是解决高性能、实时性、灵活性需求的利器，尤其适合以下场景：
  - 需要硬件加速的复杂算法。
  - 定制化接口或非标准协议实现。
  - 高可靠性、低延迟的工业控制。
  - 异构系统中的任务卸载与协同处理。
- 对于资源受限或需求明确的项目，可优先考虑MCU/SoC；若需突破性能瓶颈或实现硬件级优化，FPGA是理想选择。

FPGA与嵌入式系统开发

- 控制密集型
  - RISC-V软核
- 计算密集型、实时性
  - FPGA

FPGA内嵌MCU软核

FPGA与嵌入式系统开发

- 算法加速
  - FPGA通过并行硬件逻辑实现算法加速，适用于需要高速计算的场景，如：
  - 数字信号处理（DSP）：FFT、滤波器、图像处理（如边缘检测）。
  - 加密/解密：AES、RSA等算法的硬件级加速。
  - 机器学习推理：在边缘设备中加速神经网络（如CNN）运算。
- 实时性要求
  - FPGA的硬件逻辑直接执行任务，无需操作系统调度，可达到纳秒级延迟，适用于：
  - 工业控制（如电机控制、机器人运动规划）。
  - 高速数据采集（如ADC/DAC信号处理）。

FPGA与嵌入式系统开发

- 定制化接口与协议实现
  - 多协议支持
    - FPGA可灵活实现非标准或自定义的通信协议，例如：
      - 工业总线协议（CAN、Modbus、EtherCAT）。
      - 高速接口（PCIe、USB 3.0、MIPI）。
      - 传感器专用接口（如摄像头CMOS信号解析）。
  - 接口扩展
    - 当嵌入式主控（如MCU）接口不足时，FPGA可作为“胶合逻辑”扩展外设：
      - 多路SPI/I2C/UART复用。
      - 高速ADC/DAC数据缓存与预处理。

FPGA与嵌入式系统开发

- 可重构性与动态适配
  - 动态硬件重构
    - FPGA支持运行时重新配置部分逻辑（部分重配置），适用于：
    - 多任务系统：不同时段切换硬件功能（如白天处理图像，夜晚处理传感器数据）。
    - 远程固件更新：通过无线更新硬件逻辑（OTA）。
  - 原型验证与快速迭代
    - FPGA常用于ASIC（专用芯片）的验证阶段，通过硬件仿真降低开发风险。

FPGA与嵌入式系统开发

- 低功耗与高可靠性
  - 能效优化
    - FPGA可通过定制化逻辑减少冗余操作，在特定任务中比通用处理器更高效。
    - 示例：低功耗传感器节点中的信号预处理，仅唤醒主控MCU进行关键决策。
  - 容错设计
  - FPGA支持冗余逻辑设计，适用于高可靠性场景（如航空航天、医疗设备）。

FPGA与嵌入式系统开发

FPGA vs. 传统MCU/SoC

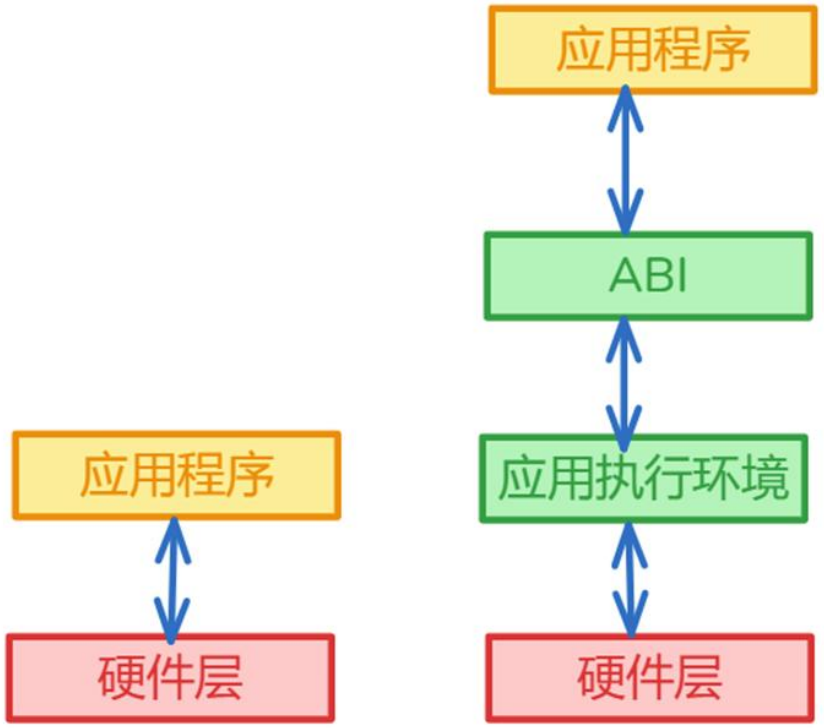
特性	FPGA	MCU/SoC
灵活性	硬件逻辑可完全自定义	固定外设与处理架构
并行性	支持大规模并行处理	依赖多核/多线程
开发周期	较长（需硬件设计）	较短（基于软件编程）
功耗	静态功耗较高，动态功耗可优化	整体功耗较低
成本	高（适合复杂任务）	低（适合通用任务）

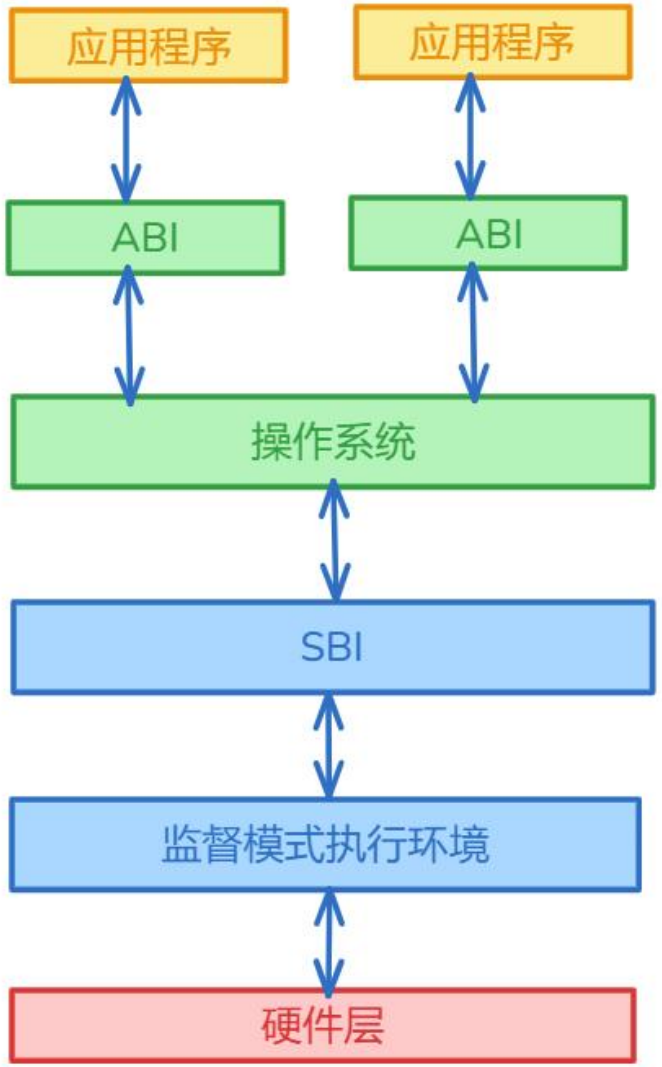
FPGA与MCU/SoC的协同设计

- 异构系统架构
  - 结合FPGA与嵌入式处理器（如ARM Cortex-M/A系列），形成“软硬协同”方案：
    - SoC+FPGA（如Xilinx Zynq、Intel Cyclone V）：
      - FPGA处理实时任务（如视频编解码）。
      - RISC-V处理器运行操作系统（Linux/FreeRTOS）管理复杂逻辑。
    - 独立FPGA+MCU：
      - FPGA作为协处理器，通过SPI/UART与主控通信。

嵌入式系统开发模式

- 裸机模式（baremetal）
- 平台模式（arduino、platformio、micropython、luatOS
- 操作系统模式（os/rtos：linux、freeRTOS、zephyr）





三种嵌入式系统开发模式

维度	裸机开发	Arduino等框架	操作系统（RTOS/OS）
资源需求	极低（KB级内存）	中等（依赖库大小）	高（MB级内存+存储）
实时性	最高（直接硬件控制）	中等（库函数延迟）	RTOS高，通用OS低
开发效率	低（需手动管理硬件）	高（快速调用库）	中等（需学习OS API）
适用复杂度	简单逻辑、单任务	中等复杂度、多外设交互	高复杂度、多任务协同
典型硬件	STM32F103、PIC16	Arduino Uno、ESP32	STM32H7（RTOS）、树莓派

■ 开发模式示例：ESP32-C3

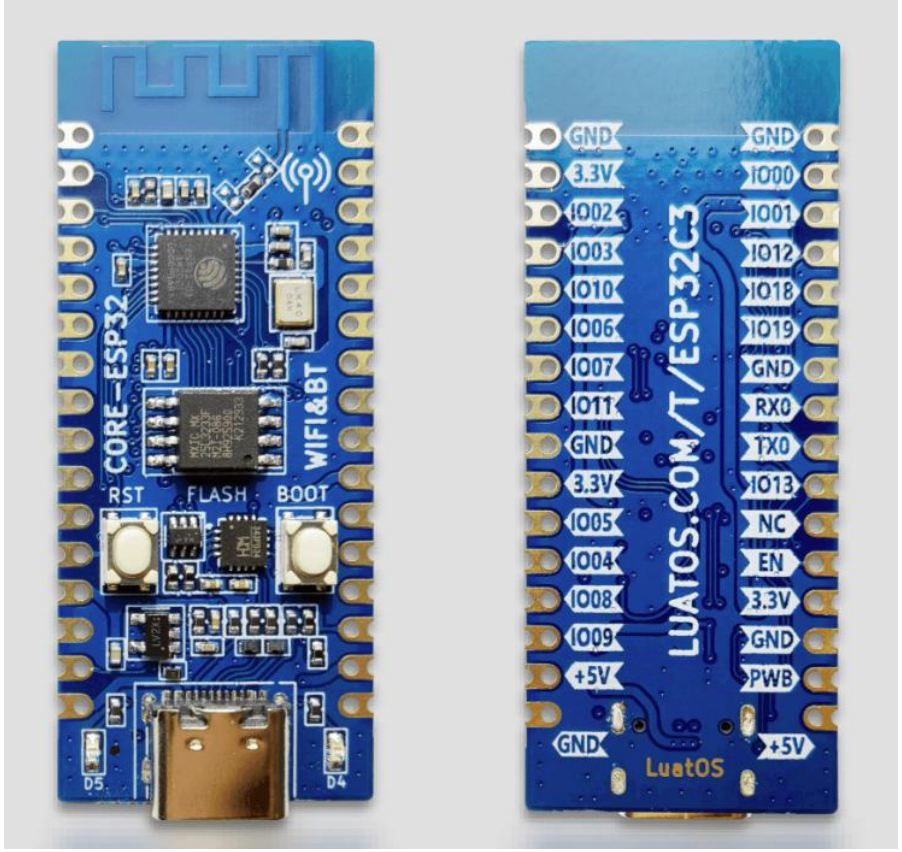
ESP32-C3 是一款安全稳定、低功耗、低成本的物联网芯片，搭载 RISC-V 32 位单核处理器，支持 2.4 GHz Wi-Fi 和 Bluetooth 5 (LE)。为物联网产品提供行业领先的射频性能、完善的安全机制和丰富的内存资源。ESP32-C3 对 Wi-Fi 和 Bluetooth 5 (LE) 的双重支持降低了设备配网难度，适用于广泛的物联网应用场景。

搭载 RISC-V 处理器

ESP32-C3 搭载 RISC-V 32 位单核处理器，时钟频率高达 160 MHz。具有 22 个可编程 GPIO 管脚、内置 400 KB SRAM，支持通过 SPI、Dual SPI、Quad SPI 和 QPI 接口外接多个 flash，满足各类物联网产品功能需求。此外，ESP32-C3 的耐高温特性也使其成为照明和工控领域的理想选择。

行业领先的射频性能

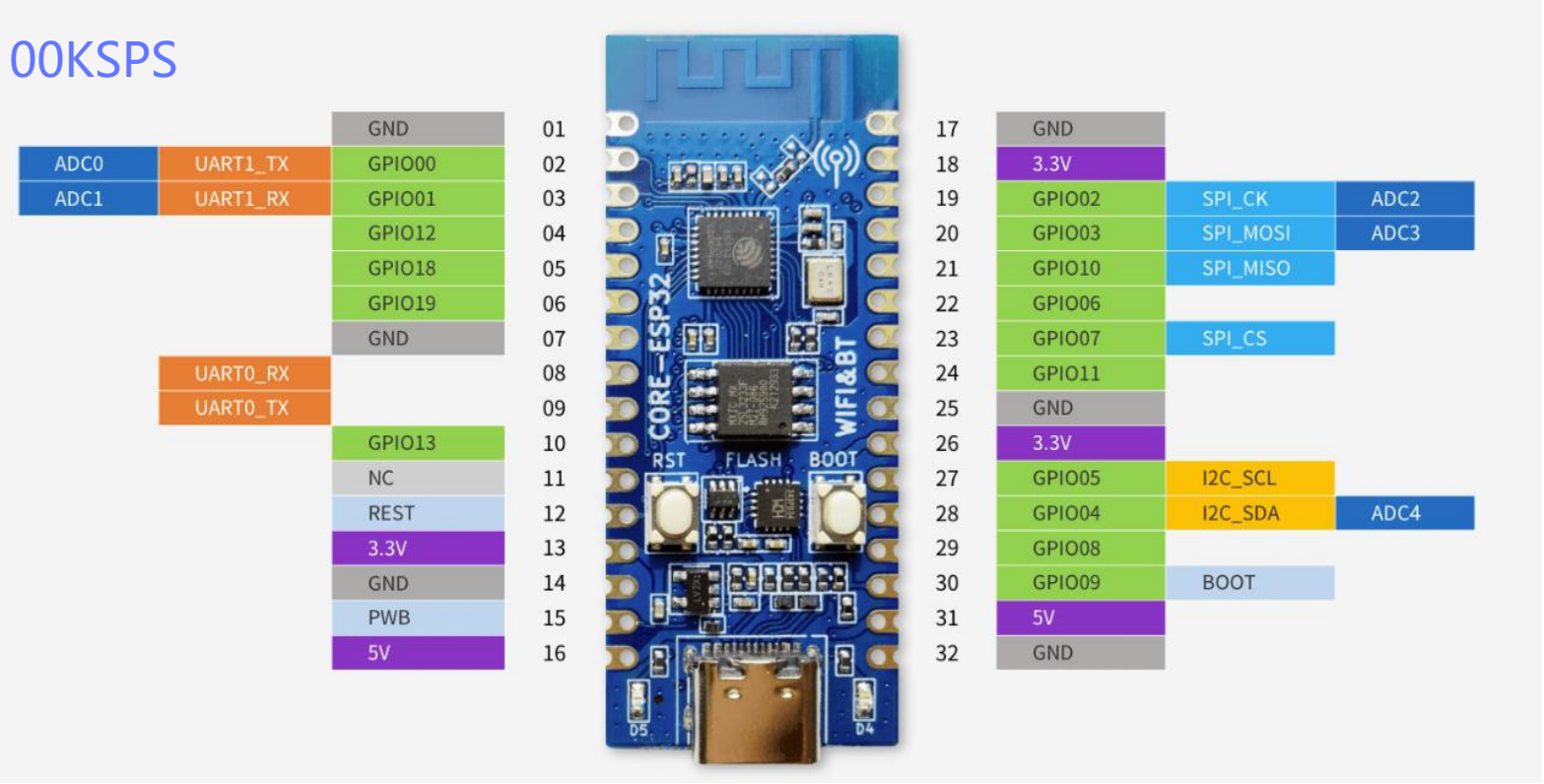
ESP32-C3 集成 2.4 GHz Wi-Fi 和支持长距离的 Bluetooth 5 (LE)，有助于构建覆盖范围更广、射频性能更强的物联网设备。它还支持蓝牙 Mesh (Bluetooth Mesh) 协议和乐鑫 Wi-Fi Mesh，在较高的工作温度下仍能保持卓越的射频性能。



开发板文档：https://wiki.luatos.com/chips/esp32c3/

■ 开发模式示例：ESP32-C3（Arduino、MicroPython、乐鑫ESP-IDF）

- 1路SPI FLASH，板载4MB，支持最高 16MB
- 2路UART接口，UART0~UART1,其中下载口为UART0
- 5 路 12 比特 ADC，最高采样率 100KSPS
- 1路低速SPI接口，支持主模式
- 1路IIC控制器
- 4路PWM接口,可使用任意GPIO
- GPIO外部管脚15路，可复用
- 2路贴片LED指示灯
- 1路复位按键+1路BOOT按键
- 1路USB转TTL下载调试口
- 2.4G PCB板载天线



Arduino

- Arduino 是一个开源电子原型平台，旨在简化电子项目的开发，让开发者（包括非专业人士）能够快速构建交互式硬件项目。
- 它结合了易于使用的硬件和软件工具，广泛应用于教育、艺术、DIY（自己动手）项目、原型设计以及物联网（IoT）等领域。

- 为什么选择 Arduino
  - 降低门槛：无需精通底层硬件知识。
  - 快速迭代：从想法到原型只需几小时。
  - 灵活性：兼容数千种传感器和执行器。

- 如果你对编程（基础 C/C++）和电子电路（如电阻、电压）有基本了解，Arduino 能让你轻松实现创意！

Arduino

- 硬件
  - Arduino 开发板：核心是一块基于微控制器（如 ATmega328P）的电路板，具备数字和模拟输入/输出引脚，可连接传感器、电机、LED 等外设。
  - 扩展模块（Shield）：可通过堆叠扩展板（如 Wi-Fi、蓝牙、电机驱动模块）增强功能。
- 软件
  - Arduino IDE（集成开发环境）：基于 C/C++ 的简化编程语言，通过编写“草图”（Sketch）控制硬件。
  - 丰富的库和社区资源：提供大量预置代码库（如控制舵机、读取传感器），降低开发难度。

型号	特点
Arduino Uno	最经典型号，适合入门
Arduino Nano	小巧，适合紧凑型项目
Arduino Mega	更多 I/O 引脚，适合复杂项目
ESP32/ESP8266	支持 Wi-Fi/蓝牙，常用于物联网项目

Arduino

- 主要特点
  - 易用性：无需专业电子知识，适合初学者快速入门。
  - 跨平台：支持 Windows、macOS、Linux 系统。
  - 开源：硬件设计文件和软件代码完全公开，可自由修改和分享。
  - 低成本：基础开发板价格亲民（如 Arduino Uno 约 ¥50–100）。
  - 社区支持：全球开发者社区提供大量教程、项目和问题解答。
- 典型应用场景
  - 教育与创客：学习电子和编程基础，制作互动装置（如温度监测、智能小车）。
  - 智能家居：控制灯光、温湿度传感器、自动浇花系统。
  - 艺术与设计：结合传感器和灯光/声音模块创作交互式艺术。
  - 工业原型：快速验证产品功能（如机器人控制、自动化测试）。

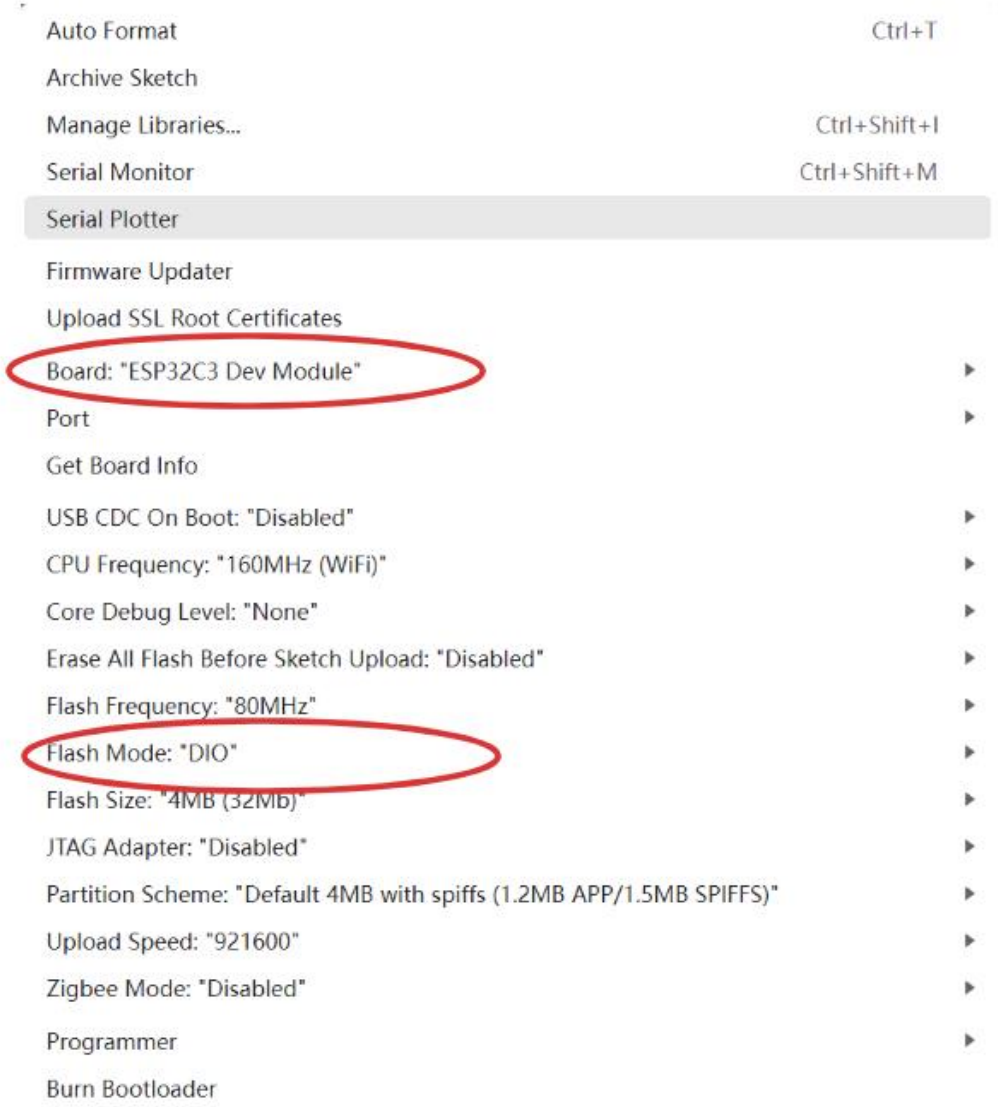
安装Arduino，添加ESP32库

- 确保已安装最新版 Arduino IDE
  - Arduino 官网下载：https://www.arduino.cc/en/software
- 添加 ESP32 开发板 URL
  - 打开 Arduino IDE，进入 文件 > 首选项。
  - 在 附加开发板管理器网址 中粘贴以下 URL
    - https://espressif.github.io/arduino-esp32/package_esp32_index.json
- 安装 ESP32 开发板包
  - 进入 工具 > 开发板 > 开发板管理器。
  - 搜索 esp32，找到 ESP32 by Espressif Systems。
  - 选择最新版本（3.11），点击 安装（需联网，时间较长）。

```
esp32:mkspiffs@0.2.3 installed
Installing esp32:openocd-esp32@v0.12.0-esp32-20241016
Configuring tool.
esp32:openocd-esp32@v0.12.0-esp32-20241016 installed
Installing esp32:riscv32-esp-elf-gdb@14.2_20240403
Configuring tool.
esp32:riscv32-esp-elf-gdb@14.2_20240403 installed
Installing esp32:xtensa-esp-elf-gdb@14.2_20240403
Configuring tool.
esp32:xtensa-esp-elf-gdb@14.2_20240403 installed
Installing platform esp32:esp32@3.1.1
Configuring platform.
Platform esp32:esp32@3.1.1 installed
```

设置ESP32开发板

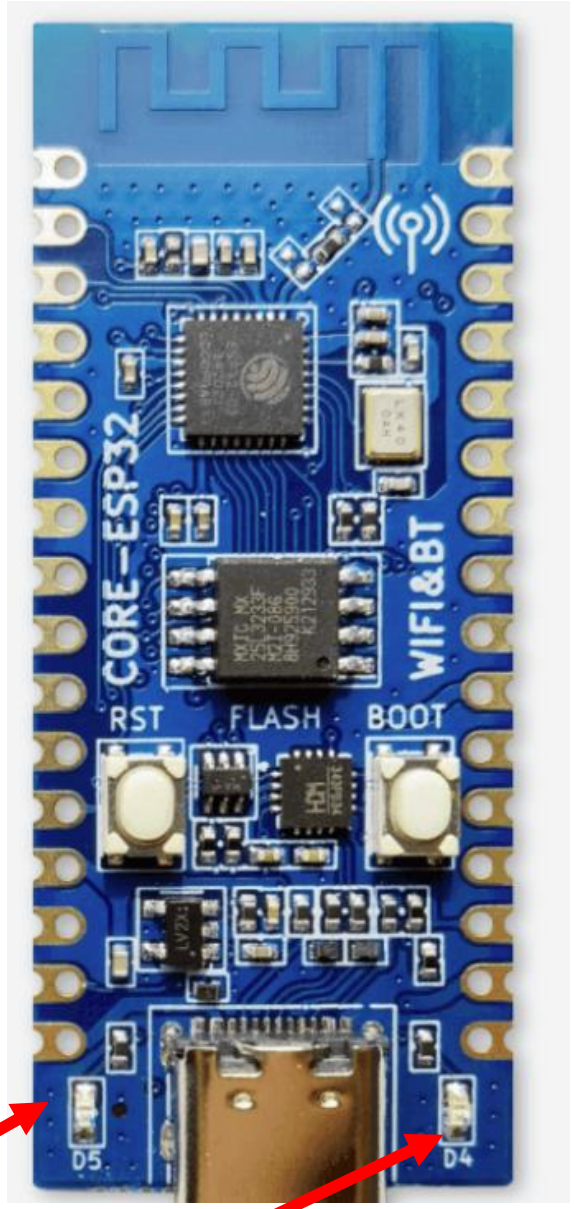
- 连接 ESP32 到电脑（需 USB 转串口驱动，如 CP210x 或 CH340）。
- 进入 工具 > 开发板，选择你的 ESP32 型号（如 ESP32 Dev Module）。
- 设置以下参数（根据具体型号调整）：
  - Upload Speed: 921600
  - Flash Frequency: 80MHz
  - Partition Scheme: Default
  - Port: 选择对应的串口号（如 COM3 或 /dev/cu.usbserial-*）。



Led灯闪烁

ESP32C3核心板板载2颗LED

LED编号	对应GPIO	管脚功能	描述
D4	IO12	GPIO12配置	高电平有效
D5	IO13	GPIO13配置	高电平有效



Led灯闪烁

LED编号	对应GPIO	管脚功能	描述
D4	IO12	GPIO12配置	高电平有效
D5	IO13	GPIO13配置	高电平有效

```
int led = 13;

// the setup function runs once when you press reset or power the board
void setup() {
  // initialize digital pin LED_BUILTIN as an output.
  pinMode(led, OUTPUT);
}

// the loop function runs over and over again forever
void loop() {
  digitalWrite(led, HIGH);  // turn the LED on (HIGH is the voltage level)
  delay(1000);                      // wait for a second
  digitalWrite(led, LOW);   // turn the LED off by making the voltage LOW
  delay(1000);                      // wait for a second
}
```

Arduino及ESP32开发文档

- Arduino 语言基础
  - https://www.arduino.cc/reference/en/
  - Arduino 编程语法、内置函数和标准库的文档
- ESP32快速入门指南
  - https://docs.espressif.com/projects/arduino-esp32/en/latest/
  - 在 Arduino IDE 中配置 ESP32 开发环境的步骤。
  - 支持的库、API 参考和示例代码。
  - 常见问题解答（如驱动安装、固件更新）。

MicroPython

- MicroPython 是一种专为微控制器（小型、低功耗的嵌入式设备）设计的精简版 Python 3 编程语言实现。它允许开发者用 Python 语法直接控制硬件（如传感器、LED、电机等），简化了嵌入式开发流程，特别适合快速原型开发和物联网（IoT）项目。

- 为什么选择 MicroPython?
  - 降低嵌入式开发门槛：Python 开发者无需学习新语言即可操控硬件。
  - 快速迭代：REPL 实时调试 + 脚本直接运行，无需编译上传。
  - 生态丰富：兼容 Python 库（部分）和硬件社区资源。

- 如果你熟悉 Python 或希望快速实现硬件交互，MicroPython 是比传统嵌入式开发更高效的选择!

MicroPython特点

- Python 语法兼容性
  - 支持大部分 Python 3 语法（如类、异常处理、列表推导），无需学习复杂的底层语言（如 C/C++）。
  - 代码简洁易读，适合编程新手或习惯 Python 的开发者。
- 硬件直接控制
  - 提供硬件级 API（如 GPIO、PWM、I2C、SPI），可直接操作引脚、读取传感器数据或驱动外设。
- 轻量高效
  - 针对资源有限的微控制器优化，内存占用小（最低需约 256KB ROM 和 16KB RAM）。
  - 支持实时操作（如中断处理）。
- 交互式开发
  - 通过 REPL（交互式解释器） 实时调试代码，类似 Python 的 Shell，方便快速测试硬件功能。
- 开源生态
  - 社区提供丰富的库（如网络通信、文件系统、传感器驱动），可快速扩展功能。

MicroPython特点

- Python 语法兼容性
  - 支持大部分 Python 3 语法（如类、异常处理、列表推导），无需学习复杂的底层语言（如 C/C++）。
  - 代码简洁易读，适合编程新手或习惯 Python 的开发者。
- 硬件直接控制
  - 提供硬件级 API（如 GPIO、PWM、I2C、SPI），可直接操作引脚、读取传感器数据或驱动外设。
- 轻量高效
  - 针对资源有限的微控制器优化，内存占用小（最低需约 256KB ROM 和 16KB RAM）。
  - 支持实时操作（如中断处理）。
- 交互式开发
  - 通过 REPL（交互式解释器） 实时调试代码，类似 Python 的 Shell，方便快速测试硬件功能。
- 开源生态
  - 社区提供丰富的库（如网络通信、文件系统、传感器驱动），可快速扩展功能。

MicroPython与Arduino

特性	MicroPython	Arduino (C/C++)
编程语言	Python 3	C/C++（类 C 语法）
学习门槛	低（语法简单，适合新手）	中（需理解指针、内存管理等概念）
开发效率	高（代码简洁，交互式调试）	中（需编译上传，调试较繁琐）
性能	较低（解释型语言，运行效率稍逊）	高（编译型语言，直接操作硬件）
适用场景	快速原型、IoT、教育	高性能需求、复杂逻辑、工业控制
典型硬件	ESP32/ESP8266、Pyboard、Raspberry Pi Pico	Arduino Uno、Mega、Nano

MicroPython安装

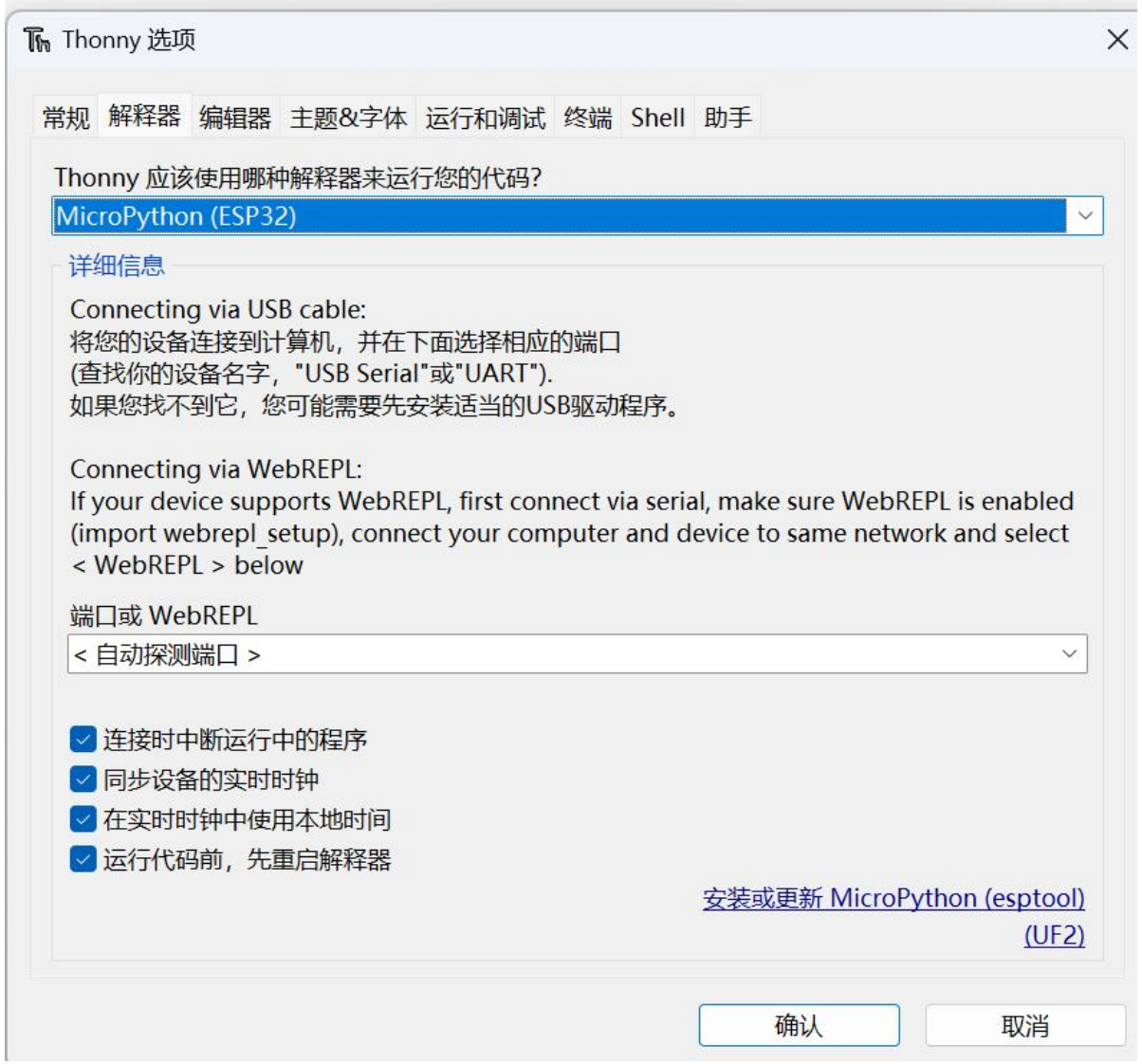
- 安装烧录工具
  - pip install esptool
  - 使用 esptool.py 烧录 MicroPython 固件
- 下载 MicroPython 固件
  - 从 MicroPython 官网下载 ESP32-C3 的预编译固件（.bin 文件）：
  - MicroPython ESP32-C3 固件下载：https://micropython.org/download/ESP32_GENERIC_C3/
  - 选择最新版本（如 esp32c3-20240105-v1.22.0.bin）
- 擦除原有固件
  - 进入下载模式：
  - 断开开发板电源。
  - 按住 BOOT 按钮（部分板子标记为 IO0）。
  - 插入 USB 线，保持按住 BOOT 按钮约 2 秒后松开。
  - 擦除 Flash：esptool.py --chip esp32c3 --port COMx erase_flash #替换 COMx 为实际串口号

MicroPython安装

- 烧录 MicroPython 固件
  - esptool.py --chip esp32c3 --port COMx --baud 460800 write_flash -z 0x0 firmware.bin
  - 替换 COMx 为串口号，firmware.bin 为固件文件名。
- 连接 MicroPython REPL
  - 使用串口工具（如 PuTTY、Thonny 或 screen）连接开发板：
  - 波特率: 115200
  - 数据格式: 8N1
- 编写并上传代码
  - 安装 Thonny，打开后选择 工具 >配置解释器。
  - 选择 MicroPython (ESP32) 和对应串口。
  - 编写代码并点击 运行 或 保存到设备。

MicroPython安装

- 使用 Thonny IDE
  - 安装thonny：https://thonny.org/
  - 安装 Thonny，打开后选择 运行 > 配置解释器。
  - 选择 MicroPython (ESP32) 和对应串口。
  - 安装烧录工具



示例代码：led闪烁

```
from machine import Pin
import time

led = Pin(13, Pin.OUT) #d5

while True:
    led.on()
    time.sleep(1)
    led.off()
    time.sleep(1)
```

LED编号	对应GPIO	管脚功能	描述
D4	IO12	GPIO12配置	高电平有效
D5	IO13	GPIO13配置	高电平有效

MicroPython及ESP32开发文档

- MicroPython 官方文档（通用）
  - https://docs.micropython.org
  - 包含 MicroPython 的通用语法、库和核心功能说明。
- ESP32 快速参考指南
  - https://docs.micropython.org/en/latest/esp32/quickref.html
  - ESP32 的引脚图、外设使用（GPIO、PWM、ADC、I2C、SPI 等）、网络功能（WiFi）等快速指南。
- ESP32 教程
  - https://docs.micropython.org/en/latest/esp32/tutorial/intro.html
  - 包含 ESP32 的固件安装、基础开发流程和示例代码。
- 图书
  - 邵子扬，MicroPython入门指南，电子工业出版社，2018

乐鑫ESP-IDF

- ESP-IDF（Espressif IoT Development Framework）是乐鑫科技（Espressif Systems）为 ESP32、ESP32-S、ESP32-C 系列芯片 官方推出的物联网开发框架，专为构建高性能、低功耗的物联网（IoT）应用而设计。它提供了一套完整的工具链和软件库，帮助开发者高效开发基于ESP芯片的嵌入式系统。

- 完全开源（Apache 2.0协议），代码托管在 GitHub。
- 活跃的开发者社区和丰富的文档

乐鑫ESP-IDF

- 硬件支持
  - 支持 ESP32、ESP32-S2/S3、ESP32-C3/C6 等主流芯片，覆盖Wi-Fi、蓝牙、低功耗、多核等不同场景需求。
- 软件架构
  - FreeRTOS 实时操作系统：支持多任务调度、内存管理等，提升系统效率。
  - 硬件抽象层（HAL）：简化外设（GPIO、SPI、I2C、ADC等）操作。
  - 网络协议栈：集成 Wi-Fi、蓝牙（BLE）、TCP/IP、HTTP、MQTT、CoAP 等协议。
  - 安全机制：支持加密算法（AES/SHA）、安全启动、Flash加密等。
- 开发工具链
  - 基于 CMake 构建系统，支持跨平台开发（Windows/Linux/macOS）。
  - 集成 编译器（GCC）、调试工具（OpenOCD、GDB），支持JTAG调试。
  - 提供 VS Code插件、Eclipse插件等，简化开发流程。

与其他开发方式的对比

- Arduino：适合快速原型开发，但功能较简单。
- MicroPython：适合脚本语言爱好者，但性能低于原生C/C++。
- ESP-IDF：提供底层控制和高性能，适合复杂项目。硬件支持
  - 官方支持：持续更新，兼容最新硬件
  - 模块化设计：按需裁剪功能，节省资源。
  - 生态丰富：兼容第三方组件（如LVGL、TensorFlow Lite Micro）。

开发环境搭建

- 安装依赖工具
  - Windows/Mac/Linux 均支持，需安装以下工具：
    - Git: https://git-scm.com
    - Python 3.7+（ESP-IDF 依赖 Python 环境）
    - 串口驱动（如 CP210x 或 CH340，根据 ESP32-C3 开发板型号选择）。
- 安装 ESP-IDF
  - 乐鑫提供了 一键式安装工具，简化环境配置：
  - 下载 ESP-IDF 安装工具：https://dl.espressif.com/dl/esp-idf/
    - 选择对应操作系统的安装包（如 esp-idf-tools-setup-offline-x.x.x.exe）。
  - 运行安装程序
    - 选择安装路径（默认即可）。
    - 勾选 ESP32-C3 作为目标芯片。
    - 安装完成后，会自动配置环境变量。

测试：idf.py --version

创建项目

- 步骤 1：获取示例代码
  - ESP-IDF 内置丰富的示例项目：cd $IDF_PATH/examples/get-started/hello_world
- 步骤 2：配置项目
  - 连接 ESP32-C3 开发板到电脑，确认串口号（如 Windows 的 COM3，Linux 的 /dev/ttyUSB0）。
  - 配置项目目标芯片和串口：
    - idf.py set-target esp32c3
    - idf.py menuconfig
- 步骤 3：编译并烧录
  - idf.py build　　# 编译项目
  - idf.py -p PORT flash # 烧录固件（PORT 替换为实际串口号，如 COM3）
- 步骤 4：监视输出
  - idf.py -p PORT monitor # 查看串口日志

Led闪烁示例

LED编号	对应GPIO	管脚功能	描述
D4	IO12	GPIO12配置	高电平有效
D5	IO13	GPIO13配置	高电平有效

```c
#include "driver/gpio.h"

#define LED_PIN 13  // D5

void app_main() {
    gpio_reset_pin(LED_PIN);
    gpio_set_direction(LED_PIN, GPIO_MODE_OUTPUT);
    while (1) {
        gpio_set_level(LED_PIN, 1);  // 点亮 LED
        vTaskDelay(1000 / portTICK_PERIOD_MS);
        gpio_set_level(LED_PIN, 0);  // 熄灭 LED
        vTaskDelay(1000 / portTICK_PERIOD_MS);
    }
}
```

乐鑫ESP-IDF官方文档与资源

- ESP-IDF 编程指南：
  - https://docs.espressif.com/projects/esp-idf/zh_CN/latest/esp32c3/index.html
- ESP32-C3 技术规格书：
  - https://www.espressif.com/sites/default/files/documentation/esp32-c3_datasheet_cn.pdf
- 乐鑫官方论坛：
  - https://www.esp32.com/
- 图书：
  - 乐鑫科技，ESP32-C3物联网工程开发实战，电子工业出版社，2022

嵌入式系统开发

- 嵌入式系统
- RISC-V在嵌入式系统中的应用
- FPGA与嵌入式系统开发
- 嵌入式系统开发模式